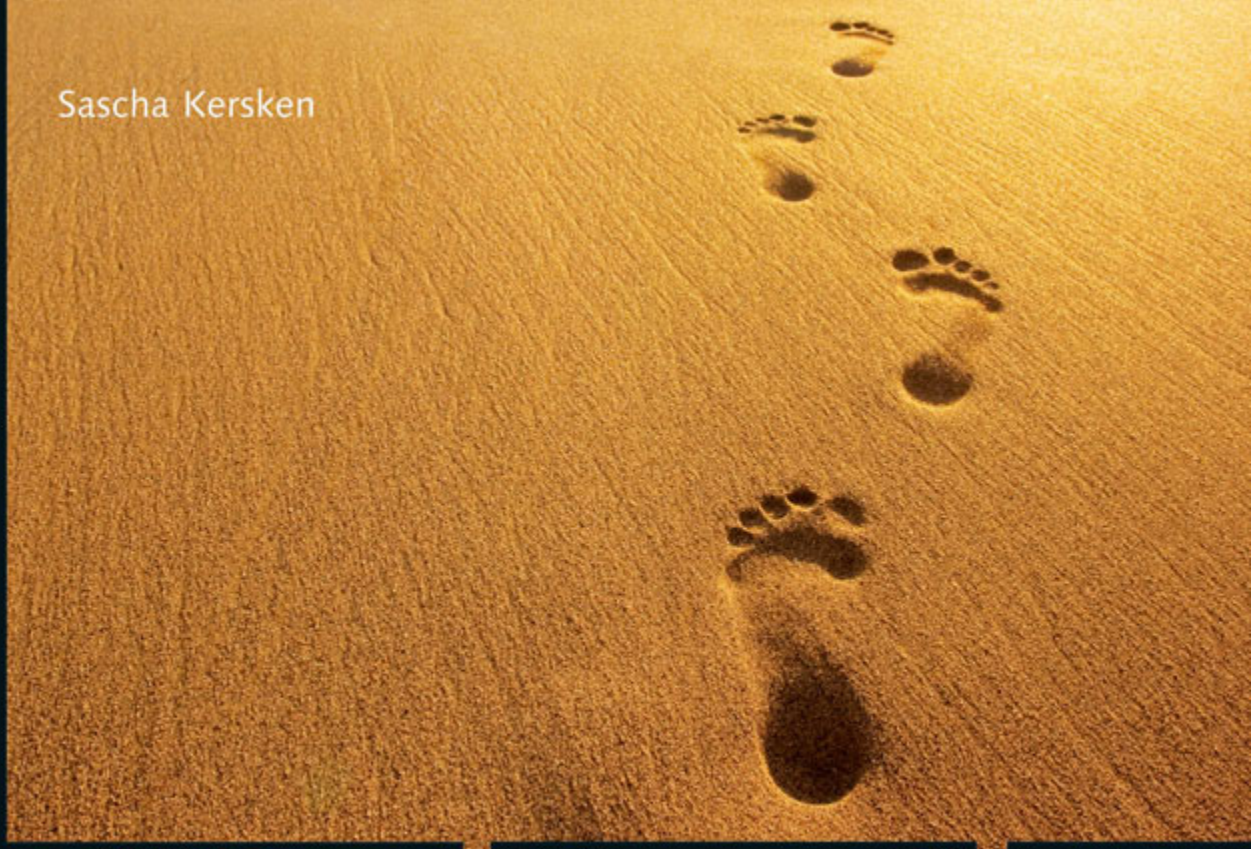
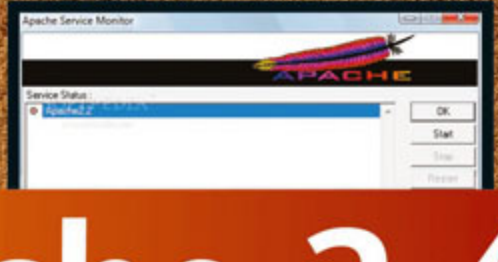
Sascha Kersken





# Apache 2.4

## Das umfassende Handbuch

Inkl. Apache 2.2

- Installation, Konfiguration, Administration
- Inkl. kommentierter Beispielkonfigurationen
- Praxiseinsatz, Sicherheit, Performance, Logging, Erweiterungen u. v. m.

DVD ROM Programme, Tools, Beispielkonfigurationen

4., aktualisierte und erweiterte Auflage

Galileo Computing

Sascha Kersken

# Apache 2.4

Galileo Press

# Liebe Leserin, lieber Leser,

Apache ist zweifellos der weltweit am häufigsten eingesetzte Webserver und mit diesem Buch haben Sie die richtige Wahl getroffen, wenn Sie ein umfassendes Lehr- und Nachschlagewerk dazu suchen. Sascha Kersken verfügt über langjährige Erfahrung im EDV-Schulungsbereich und in der Webentwicklung, ist anerkannter Fachbuchautor und kennt »den Apache« wie seine Westentasche. Sein umfassendes Handbuch hat sich seit vielen Auflagen bewährt und ist »das« Apache-Standardwerk.

Wenn Sie bisher noch keine Erfahrung mit Webservern haben, ist dies kein Hindernis. Sascha Kersken zeigt Ihnen alles, was Sie wissen müssen, um Apache sicher in Betrieb zu nehmen. Seine ganze Stärke wird dieses Buch aber als ständiger Begleiter im laufenden Betrieb Ihres Webservers ausspielen. Denn genau dies werden Sie als Anwender suchen: ein Handbuch, das Ihnen Hilfestellung bei allen Direktiven gibt, das Ihnen umfassend Auskunft gibt bei der sicheren Konfiguration und Sie auch bei professionellen Themen wie Skalierung und Load-Balancing nicht alleine lässt. Als Experte profitieren Sie von der Erfahrung des Autors: Alle wichtigen Themen rund um Apache werden lückenlos behandelt, aktuell auch zur Version 2.4 und allen Neuerungen wie z.B. mod_lua.

Ihre Meinung ist uns wichtig. Kritik oder Zuspruch hilft uns bei der Arbeit an weiteren Auflagen. Ich freue mich deshalb, wenn Sie sich mit Ihren kritischen Anmerkungen an mich wenden oder den Kontakt zum Autor in seinem Forum unter *http://buecher.lingoworld.de/apache2/* suchen.

Wir freuen uns auf den Dialog mit Ihnen!

**Ihr Stephan Mattescheck**
Lektorat Galileo Computing

stephan.mattescheck@galileo-press.de
www.galileocomputing.de
Galileo Press · Rheinwerkallee 4 · 53227 Bonn

# Auf einen Blick


| | | |
|---|---|---|
| 1 | **TCP/IP-Netzwerke, Internet und WWW** | 21 |
| 2 | **Funktionsweise von Webservern** | 49 |
| 3 | **Apache 2 im Überblick** | 105 |
| 4 | **Apache kompilieren und installieren** | 149 |
| 5 | **Apache in Betrieb nehmen** | 195 |
| 6 | **Grundkonfiguration** | 225 |
| 7 | **Header und MIME-Einstellungen** | 301 |
| 8 | **Weiterleitungen und Indizes** | 345 |
| 9 | **Authentifizierung, Autorisierung und Zugriffskontrolle** | 399 |
| 10 | **Gesicherte Verbindungen** | 475 |
| 11 | **Logging** | 515 |
| 12 | **Skalierung und Performance-Tuning** | 555 |
| 13 | **Proxy- und Cache-Funktionen** | 579 |
| 14 | **CGI** | 629 |
| 15 | **Technologien zur Webprogrammierung** | 689 |
| 16 | **SSI und Filter** | 753 |
| 17 | **Apache erweitern** | 795 |
| 18 | **Sicherheit** | 825 |
| 19 | **Kommentierte Konfigurationsdateien** | 841 |
| | **Anhang** | 889 |

Der Name Galileo Press geht auf den italienischen Mathematiker und Philosophen Galileo Galilei (1564–1642) zurück. Er gilt als Gründungsfigur der neuzeitlichen Wissenschaft und wurde berühmt als Verfechter des modernen, heliozentrischen Weltbilds. Legendär ist sein Ausspruch *Eppur si muove* (Und sie bewegt sich doch). Das Emblem von Galileo Press ist der Jupiter, umkreist von den vier Galileischen Monden. Galilei entdeckte die nach ihm benannten Monde 1610.

**Lektorat** Stephan Mattescheck
**Korrektorat** Angelika Glock und Alexandra Müller
**Einbandgestaltung** Barbara Thoben
**Coverfotos** Hauptbild: iStockphoto (DNY59); Kabel: Fotolia.com (MWiner); Mann: Fotolia.com (nyul)
**Typografie und Layout** Vera Brauner
**Herstellung** Lissy Hamann
**Satz** III-satz, Husby
**Druck und Bindung** Bercker Graphischer Betrieb, Kevelaer

**Gerne stehen wir Ihnen mit Rat und Tat zur Seite:**
stephan.mattescheck@galileo-press.de bei Fragen und Anmerkungen zum Inhalt des Buches
service@galileo-press.de für versandkostenfreie Bestellungen und Reklamationen
britta.behrens@galileo-press.de für Rezensionsexemplare

Dieses Buch wurde gesetzt aus der Linotype Syntax Serif (9,25/13,25 pt) in FrameMaker. Gedruckt wurde es auf chlorfrei gebleichtem Offsetpapier.

**Bibliografische Information der Deutschen Nationalbibliothek**
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

**ISBN 978-3-8362-1777-4**

© Galileo Press, Bonn 2012
4., aktualisierte und erweiterte Auflage 2012

Das vorliegende Werk ist in all seinen Teilen urheberrechtlich geschützt. Alle Rechte vorbehalten, insbesondere das Recht der Übersetzung, des Vortrags, der Reproduktion, der Vervielfältigung auf fotomechanischem oder anderen Wegen und der Speicherung in elektronischen Medien. Ungeachtet der Sorgfalt, die auf die Erstellung von Text, Abbildungen und Programmen verwendet wurde, können weder Verlag noch Autor, Herausgeber oder Übersetzer für mögliche Fehler und deren Folgen eine juristische Verantwortung oder irgendeine Haftung übernehmen. Die in diesem Werk wiedergegebenen Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. können auch ohne besondere Kennzeichnung Marken sein und als solche den gesetzlichen Bestimmungen unterliegen.

# Inhalt

Vorwort ..... 15

**1 TCP/IP-Netzwerke, Internet und WWW ..... 21**

1.1 TCP/IP ..... 21
1.1.1 Das Internet-Schichtenmodell ..... 22
1.1.2 Das Internet Protocol (IP) ..... 24
1.1.3 Transportprotokolle ..... 30
1.2 Das Domain Name System (DNS) ..... 32
1.2.1 Das DNS-Konzept ..... 33
1.2.2 Der DNS-Server BIND ..... 36
1.3 TCP/IP-Diagnose und -Fehlersuche ..... 42
1.3.1 ping ..... 42
1.3.2 traceroute ..... 43
1.3.3 netstat ..... 44
1.3.4 nslookup ..... 45
1.3.5 telnet ..... 46
1.4 Zusammenfassung ..... 48

**2 Funktionsweise von Webservern ..... 49**

2.1 Das HTTP ..... 49
2.1.1 Die HTTP-Client-Anfrage ..... 51
2.1.2 HTTP-Statuscodes ..... 62
2.1.3 HTTP-Header ..... 68
2.2 Einstieg für Programmierer: ein selbst geschriebener Webserver ..... 86
2.2.1 Projektanforderungen ..... 87
2.2.2 Implementierungsdetails ..... 87
2.2.3 Der komplette Quellcode ..... 97
2.2.4 Benutzerdokumentation ..... 102
2.3 Zusammenfassung ..... 104

**3 Apache 2 im Überblick ..... 105**

3.1 Einführung ..... 105
3.1.1 Entstehungsgeschichte des Apache-Webservers ..... 106
3.1.2 Die Apache Software Foundation ..... 108

3.1.3 Die Apache-Softwarelizenz ... 111
3.1.4 Sonstige Webserver ... 113
3.2 Funktionen von Apache 2 ... 116
3.2.1 Technischer Überblick ... 118
3.2.2 Apache-Module ... 133
3.3 Zusammenfassung ... 147

**4 Apache kompilieren und installieren ... 149**

4.1 Apache 2 kompilieren ... 150
4.1.1 Den Quellcode besorgen und auspacken ... 150
4.1.2 Apache 2 unter UNIX kompilieren ... 152
4.1.3 Apache 2 unter Windows kompilieren ... 182
4.2 Die binäre Apache-Distribution für Windows installieren ... 187
4.3 Module nachträglich installieren ... 192
4.4 Zusammenfassung ... 194

**5 Apache in Betrieb nehmen ... 195**

5.1 Apache 2 starten und beenden ... 195
5.1.1 Apache unter UNIX steuern ... 195
5.1.2 Apache unter Windows steuern ... 206
5.1.3 Apache-Hilfsprogramme ... 215
5.2 Apache testen ... 216
5.2.1 Die automatische Startseite ... 216
5.2.2 Die erste Website ... 217
5.3 Zusammenfassung ... 224

**6 Grundkonfiguration ... 225**

6.1 Aufbau der Apache-Konfigurationsdateien ... 225
6.1.1 Namen, Pfad und Aufgaben der Konfigurationsdateien ... 226
6.1.2 Grundlegendes zur Syntax ... 229
6.1.3 Syntaxschema ... 232
6.2 Kontexte und Container ... 234
6.2.1 Der Server-Kontext ... 234
6.2.2 Virtuelle Hosts ... 234
6.2.3 Verzeichnis- und Datei-Container ... 235
6.2.4 Spezial-Container ... 240

6.2.5 .htaccess-Dateien ... 253
6.2.6 Einfügen externer Konfigurationsdateien ... 255
6.3 Allgemeine Konfigurationsdirektiven ... 256
6.3.1 Einrichten der Server-Umgebung ... 256
6.3.2 Plattformspezifische Server-Einstellungen ... 266
6.3.3 Konfiguration des »Hauptservers« ... 279
6.3.4 Wichtige Verzeichniseinstellungen ... 288
6.4 Zusammenfassung ... 298

**7 Header und MIME-Einstellungen ... 301**

7.1 HTTP-Header manipulieren ... 301
7.1.1 MD5-Digest und ETag ... 301
7.1.2 mod_headers ... 303
7.1.3 mod_expires ... 309
7.1.4 mod_asis ... 312
7.1.5 mod_cern_meta ... 313
7.2 MIME-Konfiguration ... 315
7.2.1 MIME-Type-Einstellungen ... 316
7.2.2 Zeichensatzeinstellungen ... 323
7.2.3 MIME-Codierung ... 326
7.2.4 Spracheinstellungen ... 327
7.2.5 Handler festlegen ... 329
7.3 Content-Negotiation ... 332
7.3.1 Servergesteuerte Content-Negotiation ... 333
7.3.2 Transparente Content-Negotiation ... 339
7.3.3 Konfigurationseinstellungen für Content-Negotiation ... 341
7.4 Zusammenfassung ... 344

**8 Weiterleitungen und Indizes ... 345**

8.1 Aliase und Weiterleitungen ... 345
8.1.1 mod_alias ... 346
8.1.2 mod_rewrite ... 352
8.1.3 Benutzerverzeichnisse veröffentlichen ... 372
8.1.4 Fehlerbehandlung ... 375
8.1.5 Rechtschreibkorrektur in URLs mit mod_speling ... 376
8.1.6 Status- und Konfigurationsinformationen über den Server ... 378

8.2 Indizes ..... 381
8.2.1 mod_autoindex ..... 382
8.2.2 Serverseitige Image Maps mit mod_imagemap ..... 393
8.3 Zusammenfassung ..... 398

**9 Authentifizierung, Autorisierung und Zugriffskontrolle ..... 399**

9.1 Grundlagen ..... 399
9.1.1 Die Organisation der Authentifizierungsmodule seit Apache 2.2 ..... 401
9.1.2 Ein erstes Beispiel ..... 403
9.1.3 Grunddirektiven zur Authentifizierung ..... 405
9.2 Basic-Authentifizierung ..... 414
9.2.1 Das Programm htpasswd ..... 414
9.2.2 Direktiven zur textdateibasierten Basic-Authentifizierung ..... 416
9.3 Digest-Authentifizierung ..... 419
9.3.1 Das Tool htdigest ..... 421
9.3.2 Direktiven zur Digest-Authentifizierung ..... 422
9.4 Benutzer- und Passwortverwaltung in DBM-Dateien ..... 426
9.4.1 Das Tool dbmmanage ..... 427
9.4.2 Das Programm htdbm ..... 430
9.4.3 DBM-Direktiven ..... 431
9.5 LDAP-Authentifizierung ..... 434
9.5.1 LDAP-Grundwissen ..... 434
9.5.2 OpenLDAP einrichten und verwalten ..... 437
9.5.3 LDAP-Authentifizierungs-Direktiven ..... 441
9.5.4 LDAP-Performance-Verbesserung mit mod_ldap ..... 448
9.6 Anonymous-Authentifizierung ..... 454
9.7 Datenbankbasierte Authentifizierung mit mod_authn_dbd ..... 457
9.7.1 Datenbankverbindungen mit mod_dbd ..... 457
9.7.2 mod_authn_dbd-Direktiven ..... 461
9.8 Formularbasierte Authentifizierung mit mod_auth_form ..... 462
9.8.1 Formular- und Konfigurations-Beispiel ..... 462
9.8.2 mod_auth_form-Direktiven ..... 463
9.9 Weitere Authentifizierungseinstellungen ..... 469
9.9.1 Provider-Aliase ..... 469
9.9.2 mod_authz_owner ..... 471
9.9.3 mod_authn_default und mod_authz_default ..... 471
9.10 Zusammenfassung ..... 472

**10 Gesicherte Verbindungen ..... 475**

10.1 SSL-Grundlagen ..... 476
10.1.1 SSL einrichten ..... 478
10.1.2 SSL-Grundkonfiguration ..... 482
10.1.3 mod_ssl-Umgebungsvariablen ..... 484
10.2 mod_ssl-Direktiven ..... 485
10.2.1 Standard-Direktiven ..... 486
10.2.2 mod_ssl-Proxy-Direktiven ..... 510
10.2.3 mod_nw_ssl für NetWare ..... 513
10.3 Zusammenfassung ..... 514

**11 Logging ..... 515**

11.1 Logging-Direktiven und -Module ..... 516
11.1.1 Core-Direktiven ..... 516
11.1.2 mod_log_config ..... 524
11.1.3 mod_log_forensic ..... 532
11.1.4 mod_dumpio ..... 533
11.1.5 mod_usertrack ..... 534
11.1.6 Logging-Direktiven in mod_rewrite ..... 537
11.1.7 Benutzerdefinierte Debug-Log-Meldungen ..... 538
11.2 Auswertung von Log-Dateien ..... 539
11.2.1 Apache-Hilfsprogramme ..... 540
11.2.2 Log-Datei-Auswertung durch eigene Skripte ..... 542
11.2.3 Externe Tools ..... 552
11.3 Zusammenfassung ..... 553

**12 Skalierung und Performance-Tuning ..... 555**

12.1 Virtuelle Hosts ..... 555
12.1.1 Konfigurationsbeispiele ..... 556
12.1.2 Core-Direktiven für virtuelle Hosts ..... 560
12.1.3 mod_vhost_alias ..... 563
12.2 Performance-Tuning ..... 566
12.2.1 Allgemeines ..... 567
12.2.2 Benchmarks mit ab ..... 568
12.2.3 Performance-bezogene Core-Direktiven ..... 571
12.2.4 mod_file_cache: häufig genutzte Dateien vorausladen ..... 572

12.3 Load-Balancing ..... 574
12.3.1 Load-Balancing mit mod_rewrite ..... 575
12.3.2 Open-Source-Lösungen für Load-Balancing ..... 577
12.4 Zusammenfassung ..... 578

**13 Proxy- und Cache-Funktionen ..... 579**

13.1 Apache als Proxy-Server ..... 579
13.1.1 Proxy-Grundkonfiguration ..... 581
13.1.2 Referenz der Proxy-Direktiven ..... 584
13.1.3 mod_proxy_html ..... 603
13.1.4 Massen-Reverse-Proxys mit mod_proxy_express ..... 610
13.2 Cache-Funktionen ..... 611
13.2.1 Cache-Grundkonfiguration ..... 612
13.2.2 Cache-Direktiven ..... 614
13.2.3 htcacheclean ..... 625
13.3 Zusammenfassung ..... 626

**14 CGI ..... 629**

14.1 Die CGI-Schnittstelle ..... 629
14.1.1 FastCGI ..... 632
14.2 Apache für CGI-Skripte konfigurieren ..... 633
14.2.1 CGI-Verzeichnisse ..... 633
14.2.2 CGI in normalen Verzeichnissen aktivieren ..... 636
14.2.3 Konfigurationsanweisungen für mod_cgi und mod_cgid ..... 638
14.2.4 Plattformspezifische Einstellungen ..... 641
14.2.5 Das Modul mod_actions ..... 643
14.3 Umgebungsvariablen ..... 645
14.3.1 Die Umgebungsvariablen im Überblick ..... 646
14.3.2 Umgebungsvariablen in der Apache-Konfiguration setzen ..... 648
14.4 Session-Verwaltung mit mod_session ..... 654
14.4.1 Ein einfaches Konfigurationsbeispiel ..... 655
14.4.2 Session-Direktiven ..... 656
14.5 Grundlagen der CGI-Programmierung ..... 665
14.5.1 Das erste Beispiel ..... 666
14.5.2 Formulardaten einlesen ..... 667
14.6 Das Perl-Modul CGI.pm ..... 668
14.6.1 CGI.pm im Überblick ..... 669

14.6.2 Beispiel: Ein kleiner Taschenrechner ..... 675
14.6.3 CGI.pm-Kurzreferenz ..... 679
14.7 Zusammenfassung ..... 687

**15 Technologien zur Webprogrammierung ..... 689**

15.1 PHP ..... 690
15.1.1 MySQL installieren ..... 691
15.1.2 PHP installieren ..... 697
15.1.3 Die PHP-Konfigurationsdatei php.ini ..... 703
15.1.4 phpMyAdmin einrichten ..... 706
15.1.5 PHP-Programmierung ..... 708
15.2 mod_perl ..... 719
15.2.1 Installation von mod_perl ..... 720
15.2.2 Perl-Zugriff auf MySQL-Datenbanken ..... 726
15.2.3 Perl in der Apache-Konfigurationsdatei ..... 727
15.3 Tomcat ..... 729
15.3.1 Tomcat installieren ..... 729
15.3.2 Tomcat per Proxy einbinden ..... 734
15.3.3 Java-Webprogrammierung ..... 735
15.4 mod_lua ..... 740
15.4.1 Ein einfaches Beispiel ..... 740
15.4.2 mod_lua-Direktiven ..... 742
15.5 Weitere Programmierschnittstellen ..... 746
15.5.1 ISAPI-Anwendungen mit mod_isapi ..... 746
15.5.2 Sonstige Technologien ..... 749
15.6 Zusammenfassung ..... 750

**16 SSI und Filter ..... 753**

16.1 Server Side Includes (SSI) ..... 753
16.1.1 SSI aktivieren ..... 754
16.1.2 SSI-Elemente ..... 754
16.1.3 mod_include-Direktiven ..... 761
16.2 Filterkonfiguration ..... 765
16.2.1 Grundlegende Filterdirektiven ..... 765
16.2.2 Freie Modifikation der Filter Chain mit mod_filter ..... 770
16.2.3 Der Komprimierungsfilter mod_deflate ..... 775
16.2.4 mod_charset_lite ..... 778
16.2.5 mod_xml2enc ..... 780
16.2.6 Inhalt ersetzen mit mod_substitute ..... 781

16.3 Externe Filter programmieren ..... 786
16.3.1 mod_ext_filter ..... 786
16.3.2 Beispiele für externe Filter ..... 789
16.4 Zusammenfassung ..... 794

**17 Apache erweitern ..... 795**

17.1 WebDAV ..... 795
17.1.1 Konfigurationsbeispiel ..... 796
17.1.2 DAV-Direktiven ..... 796
17.2 Weitere Module ..... 798
17.2.1 Multiprotokoll-Unterstützung ..... 799
17.2.2 Weitere Drittanbieter-Module ..... 800
17.3 Programmierung eigener Module ..... 806
17.3.1 mod_example – Erforschen der Modul-API ..... 806
17.3.2 Arbeitsweise von Modulen ..... 807
17.3.3 Die Modulentwicklung ..... 808
17.3.4 mod_daytime – ein Beispiel zur Multiprotokoll-Unterstützung ..... 820
17.4 Zusammenfassung ..... 824

**18 Sicherheit ..... 825**

18.1 Sicherheit der Server-Umgebung ..... 825
18.2 Apache-Sicherheit ..... 827
18.2.1 Allgemeine Sicherheitshinweise ..... 827
18.2.2 Sicherheitsrelevante Direktiven ..... 829
18.2.3 SuEXEC ..... 834
18.3 mod_security ..... 837
18.4 Zusammenfassung ..... 838

**19 Kommentierte Konfigurationsdateien ..... 841**

19.1 Die Originalkonfigurationsdateien ..... 841
19.1.1 Die Grundkonfigurationsdatei httpd.conf ..... 842
19.1.2 Server-Pool-Verwaltung (httpd-mpm.conf) ..... 854
19.1.3 Mehrsprachige Fehlermeldungen (httpd-multilang-errordoc.conf) ..... 857
19.1.4 Fancy-Index (httpd-autoindex.conf) ..... 858
19.1.5 Sprach- und Zeichensatzeinstellungen (httpd-languages.conf) ..... 861

19.1.6 Benutzerverzeichnisse (httpd-userdir.conf) ..... 864
19.1.7 Status- und Anfrageinformationen (httpd-info.conf) ..... 865
19.1.8 Virtuelle Hosts (httpd-vhosts.conf) ..... 867
19.1.9 Einbinden der Apache-Dokumentation (httpd-manual.conf) ..... 868
19.1.10 WebDAV-Konfiguration (httpd-dav.conf) ..... 869
19.1.11 Erweiterte Standardeinstellungen (httpd-default.conf) ..... 870
19.1.12 Gesicherte Verbindungen (httpd-ssl.conf) ..... 872
19.2 Zusätzliche Konfigurationsdateien ..... 879
19.2.1 Authentifizierung ..... 879
19.2.2 Forward-Proxy mit Caching ..... 881
19.2.3 PHP ..... 883
19.2.4 mod_rewrite – papaya CMS ..... 884
19.3 Zusammenfassung ..... 887

**Anhang ..... 889**

A Besonderheiten von Apache 2.0 ..... 891
A.1 Zusätzliche Multiprocessing-Module ..... 891
A.2 Unterschiede bei Modulen ..... 892
A.3 Besonderheiten bei der Authentifizierung ..... 892
A.4 Weitere exklusive Direktiven ..... 899
B Kurzreferenz der Konfigurationsdirektiven ..... 903
C Sonstige Tabellen ..... 935
C.1 MIME-Types ..... 935
C.2 Sprachcodes nach ISO ..... 954
C.3 Zeichensätze ..... 958
C.4 Top-Level-Domains ..... 966
D Die Apache-Lizenz 2.0 ..... 977
E Reguläre Ausdrücke ..... 983
F VMware Workstation ..... 985
F.1 Einrichtung einer virtuellen Maschine ..... 986
F.2 Die virtuelle Maschine im Betrieb ..... 987
F.3 Einstellungen der virtuellen Maschine ändern ..... 988
F.4 VMware Tools installieren ..... 989
G Rechtliche Aspekte ..... 991
H Literatur ..... 995

Index ..... 997

*»Aller Anfang ist heiter, die Schwelle ist der Platz der Erwartung.«*
*– Johann Wolfgang von Goethe*

# Vorwort

Der Apache-HTTP-Server, Apache-Webserver oder einfach »Apache« ist die populärste Webserver-Software der Welt: Weit über die Hälfte aller Websites wird von Apache serviert. Zudem handelt es sich um eines der erfolgreichsten Open-Source-Softwareprojekte. Die Apache Software Foundation, der »Dachverband« der Apache-Entwickler, betreut Dutzende freier Softwareprojekte; neben dem HTTP-Server gehören dazu so bekannte und beliebte Produkte wie Tomcat, Spam-Assassin und Xalan. Diese Projekte beeinflussen und ergänzen einander und bestimmen so maßgeblich die Weiterentwicklung von Internet und World Wide Web mit.

Dieses Buch ist ein umfassendes Handbuch zum Apache-Webserver in der brandneuen Version 2.4, mitsamt einer ausführlichen Dokumentation der noch vorherrschenden Version 2.2; sämtliche Bestandteile, die zum »Lieferumfang« gehören, werden ausführlich beschrieben.

Die Schwerpunktthemen dieses Buches sind Installation, Administration und Programmierung. Sie erfahren zunächst einmal, wie Sie den Server unter verschiedenen Betriebssystemen kompilieren und/oder installieren können. Im weiteren Verlauf des Buches geht es vor allem um die unzähligen Konfigurationsanweisungen (Direktiven), die in den Konfigurationsdateien von Apache zur Verfügung stehen. Anders als viele andere Server-Produkte ist Apache nämlich von Hause aus nicht mit einer grafischen Konfigurationsoberfläche ausgestattet. Dies macht seine Administration zwar schwieriger, bietet aber dafür die größtmögliche Flexibilität.

Apache ist für zahlreiche verschiedene Plattformen und Betriebssysteme verfügbar. In Version 2 wurde insbesondere die Unterstützung für Nicht-UNIX-Systeme verbessert: Als Basis der eigentlichen Server-Implementierung wurde eine Bibliothek namens *Apache Portable Runtime* (APR) eingeführt, die anstelle der früher eingesetzten POSIX-Emulation die jeweiligen Stärken der einzelnen Systeme abstrahiert. Auch das Laufzeitverhalten wurde verbessert: Sie können nun aus meh-

reren sogenannten *Multiprocessing-Modulen* (MPMs) das passende für Ihre Plattform auswählen.

Ausführlich wird hier die Apache-Konfiguration für die Betriebssysteme UNIX (alle Varianten) und Windows (NT und seine Nachfolger) behandelt. Besonderheiten für andere Plattformen werden gegebenenfalls angemerkt, aber nicht weiter vertieft.

Einen großen Raum nehmen in diesem Buch die zahlreichen Module ein, die mit Apache 2 geliefert werden und die für beinahe jeden Verwendungszweck eine praktische Lösung bieten. Auf diese Weise brauchen Sie bestimmte Aspekte der Funktionalität nur dann in Ihren Webserver zu integrieren, wenn Sie sie wirklich benötigen. Dies kann Ihnen helfen, den Überblick zu behalten, und schont obendrein die Ressourcen des Server-Rechners.

**Das Komplettpaket**

Die beiliegende DVD-ROM enthält so gut wie alle Programme, Listings und Dokumente, die in diesem Buch angesprochen werden. Unter anderem finden Sie darauf Apache-Distributionen für die verschiedensten Betriebssysteme, externe Module, Zusatzprogramme, Skripte und RFC-Dokumente. Die Tatsache, dass die Dateien auf diese Weise weiterverbreitet werden dürfen, ist einer der großen Vorteile freier Software[1].

Dennoch sollten Sie, bevor Sie ein bestimmtes Programm von der DVD installieren, überprüfen, ob nicht bereits eine aktuellere Version verfügbar ist – bei Open-Source-Produkten bedeutet die Bereitstellung eines neuen Releases oft, dass wichtige Sicherheitsprobleme aus vorangegangenen Versionen behoben wurden. Eine jeweils aktuelle Liste mit Download-Links und zahlreiche Zusatzinformationen finden Sie auf der Website zum Buch. Die Adresse lautet *http://buecher.lingoworld.de/apache2*.

### Das Buch im Überblick

In den einzelnen Kapiteln dieses Buches werden folgende Themen behandelt:

- In **Kapitel 1, »TCP/IP-Netzwerke, Internet und WWW«**, finden Sie eine kurze Übersicht über die Umgebung, in der Apache ausgeführt wird: In knapper Form wird die Technik der TCP/IP-Protokollfamilie erläutert. Darüber hinaus gibt es hier auch eine Einführung in die Einrichtung eines Nameservers und Informationen über einige Hilfsprogramme.
- **Kapitel 2, »Funktionsweise von Webservern«**, behandelt das Anwendungsprotokoll HTTP, das die Grundlage des World Wide Web bildet. Neben der Besprechung sämtlicher HTTP-Methoden, -Header und -Statuscodes wird hier

1 Einige wenige Tools dürfen aus rechtlichen Gründen nicht auf der DVD verbreitet werden. In diesen Fällen finden Sie im Buch entsprechende Download-Links.

zur Veranschaulichung die Programmierung eines kleinen Webservers in Perl gezeigt.

- In **Kapitel 3, »Apache 2 im Überblick«**, wird in allgemeiner Form der Funktionsumfang des Webservers beschrieben. Dazu gehören auch Themen wie die Geschichte von Apache, ein Vergleich mit anderen Webservern sowie eine Liste der verfügbaren Module.
- Wie Sie Apache auf Ihrem System installieren können, wird ausführlich in **Kapitel 4, »Apache kompilieren und installieren«**, beschrieben. Sie erhalten Anleitungen zur Kompilierung der Quellcode-Pakete unter UNIX und Windows sowie zur Installation diverser Binärdistributionen.
- **Kapitel 5, »Apache in Betrieb nehmen«**, behandelt die Steuerung von Apache. Sie erfahren alles über das Starten, Stoppen und Neustarten des Servers sowie über Möglichkeiten, ihn beim Hochfahren des Systems automatisch zu starten.
- In **Kapitel 6, »Grundkonfiguration«**, wird zunächst der allgemeine Aufbau der Konfigurationsdatei *httpd.conf* erläutert. Anschließend werden alle Konfigurationsdirektiven besprochen, die für den Betrieb einer einfachen statischen Website wichtig sind.
- **Kapitel 7, »Header und MIME-Einstellungen«**, bietet weitere wichtige Informationen für die Administration von Websites: Apache 2 enthält Module und Konfigurationseinstellungen zur Manipulation von HTTP-Headern und MIME-Informationen. Dazu gehört auch das Thema Content-Negotiation, also die Belieferung von Clients mit deren jeweils bevorzugter Darstellungsform eines Dokuments.
- Das Thema von **Kapitel 8, »Weiterleitung und Indizes«**, sind Situationen, in denen sich unter der angeforderten URL kein Dokument befindet: URLs lassen sich auf Dokumente außerhalb des Website-Verzeichnisses oder sogar auf externe URLs umleiten; Apache kann zudem selbstständig Verzeichnisindizes generieren.
- In **Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«**, wird die Absicherung von Websites behandelt: die persönliche Anmeldung einzelner User und Gruppen und die Speicherung der entsprechenden Anmeldedaten in Quellen wie Textdateien, Datenbanken oder LDAP-Verzeichnissen.
- **Kapitel 10, »Gesicherte Verbindungen«**, behandelt die Bereitstellung SSL/TLS-gesicherter Verbindungen, die durch Verschlüsselung und andere Maßnahmen vor Mitlese- oder gar Manipulationsversuchen geschützt werden.
- **Kapitel 11, »Logging«**, beschäftigt sich mit der Einrichtung und Verarbeitung von Log-Dateien. Diese wichtigen Helfer geben über alle Zugriffe auf Ihre Websites sowie über mögliche Fehler oder Angriffsversuche Aufschluss.

- **Kapitel 12, »Skalierung und Performance-Tuning«**, behandelt die wichtigsten Themen, die für den Betrieb besonders großer Websites relevant sind: Sie erfahren alles über virtuelle Hosts, Performance-Tuning und über Load-Balancing-Verfahren.
- **Kapitel 13, »Proxy- und Cache-Funktionen«**, beschreibt den Betrieb von Apache als Proxy-Server für verschiedene Protokolle (HTTP, FTP und andere). Nicht nur zu diesem Zweck ist der Einsatz des Web-Cachings interessant, das ebenfalls hier behandelt wird.
- In **Kapitel 14, »CGI«**, wird die klassische Schnittstelle zur Entwicklung von Webanwendungen vorgestellt, das Common Gateway Interface (CGI). Die Beispiele sind in Perl geschrieben; in diesem Zusammenhang lernen Sie das Perl-Modul `CGI.pm` kennen, das die Entwicklung von CGI-Skripten erheblich einfacher und komfortabler macht.
- **Kapitel 15, »Technologien zur Webprogrammierung«**, versammelt die beliebtesten Schnittstellen für die Entwicklung von Webanwendungen: PHP, `mod_perl` und Tomcat. Der Schwerpunkt ist die Integration der Module in Apache; darüber hinaus gibt es einige Programmierbeispiele und -tipps. Neu in diesem Kapitel ist ein kurzer Überblick über `mod_lua`, ein Modul für Webanwendungen und -skripte in der Programmiersprache Lua, das seit Version 2.4 zum Lieferumfang von Apache gehört.
- In K**apitel 16, »SSI und Filter«**, wird das interessante Konzept der Filter vorgestellt, das in Apache 2 neu eingeführt wurde: Eingehende Anfrage-Daten lassen sich ebenso leicht modifizieren wie die eigentlich schon fertige Antwort an die Clients. Das klassische SSI-Verfahren (Server Side Includes) wurde in das Filterkonzept integriert und wird deshalb ebenfalls in diesem Kapitel behandelt.
- **Kapitel 17, »Apache erweitern«**, beschreibt zunächst den Betrieb von Apache als WebDAV-Server, anschließend geht es um diverse Drittanbieter-Module. Zum Schluss gibt es ein Tutorial über die Programmierung eigener Module.
- **In Kapitel 18, »Sicherheit«**, werden einige wichtige Aspekte der Apache-Sicherheit behandelt: die Absicherung der Systemumgebung und verschiedene Sicherheitsaspekte des Webservers selbst. Abschließend erhalten Sie einen Überblick über das Drittanbieter-Modul `mod_security`, das zusätzliche Sicherheitsmaßnahmen in den Server einfügt.
- **Kapitel 19, »Kommentierte Konfigurationsdateien«**, enthält schließlich einige Komplettkonfigurationen mit deutschsprachigen Kommentaren. Los geht es mit den Standardkonfigurationsdateien, wie sie mit Apache geliefert werden, danach folgen einige Konfigurationen für Spezialanwendungen wie

Authentifizierung, PHP und URL-Rewriting. Einige dieser Konfigurationen finden Sie in Form einer virtuellen Maschine auf der Buch-DVD. Sie können sie über den mitgelieferten VMware Player in (Test-)Betrieb nehmen.

In den sich daran anschließenden Anhängen finden Sie zusätzliches interessantes Material: eine kurze Übersicht über die Besonderheiten der Vorgängerversion Apache 2.0, eine Kurzreferenz sämtlicher Konfigurationsdirektiven, einige Tabellen mit MIME-Types, Sprachkürzeln und Zeichensätzen, eine Zusammenfassung rechtlicher Aspekte und andere Themen.

**Danksagungen**

Dies ist die vierte Auflage eines Buches, die nicht erschienen wäre, wenn die drei vorangegangenen keinen Erfolg gehabt hätten. Insofern danke ich allen Käufern und Lesern der ersten bis dritten Auflage dieses Buches, die mitgeholfen haben, dieses neue Projekt zu ermöglichen. Einige Leser und Besucher meiner Website haben mir zudem ausführliches Feedback gegeben; viele ihrer Verbesserungsvorschläge haben ihren Weg in diese Neuauflage gefunden. Auch dafür möchte ich mich bedanken.

Wie immer auch vielen, vielen Dank an Stephan Matteschеck und den Rest des Teams von Galileo Press – diesmal war es eine besondere logistische Meisterleistung, weil die Version 2.4 in letzter Minute vor Drucklegung fertig geworden ist – die Originaldokumentation wurde entsprechend spät vervollständigt, und ich habe einige Neuerungen erst dann lesen, ausprobieren und ergänzen können. Danke auch an die beiden Korrektorinnen Angelika Glock und Alexandra Müller.

Weiterer Dank gebührt natürlich den Entwicklern des Apache-Webservers. Ohne die zahllosen Stunden, die diese Enthusiasten freiwillig in die Entwicklung und Verbesserung dieses großartigen Produkts gesteckt haben, gäbe es das Thema dieses Buches gar nicht. Wenn Sie sich erkenntlich zeigen möchten, sollten Sie die Apache Software Foundation mit Spenden bedenken, vor allem aber mithelfen, etwas gegen die Einführung von Softwarepatenten in der Europäischen Union zu tun. Ein guter Ausgangspunkt dafür ist die Website *http://www.ffii.org*.

Danke auch an meine Kollegen bei der papaya Software GmbH fürs Probelesen, für diverse Tipps und ihr Verständnis dafür, wenn ich »buchbedingt« manchmal sehr pünktlich Feierabend machen musste.

Zu guter Letzt ist es meine Familie, die mich am meisten bei der Arbeit an diesem Buch unterstützt hat – vor allem durch ihre Toleranz für meinen zeitaufwendigen Zweitjob. Ich danke meiner Frau Tülay und meinem Sohn Leon für all ihre Geduld und Unterstützung.

*»Auch zehntausend Zhang hohe Türme*
*nehmen auf der Erde ihren Anfang.«*
*– Chinesisches Sprichwort*

# 1 TCP/IP-Netzwerke, Internet und WWW

Der *Apache-HTTP-Server* (Kurzform *Apache*) ist ein Webserver für das Internet und das Intranet. Er dient vor allem dazu, bestehende oder dynamisch generierte Dokumente über ein Netzwerk an die Webbrowser von Benutzern auszuliefern. Internet und Intranet sind Bezeichnungen für globale beziehungsweise lokale Computernetzwerke, die die Netzwerkprotokollfamilie TCP/IP verwenden. Falls Sie mit all diesen Begriffen bereits vertraut sind, können Sie im nächsten Kapitel weiterlesen. Ansonsten finden Sie hier eine kurze Übersicht über TCP/IP-Netzwerke und das Internet. Eine viel ausführlichere Darstellung dieser Themen finden Sie beispielsweise in meinem Buch »IT-Handbuch für Fachinformatiker« (5. Auflage, Bonn 2011, Galileo Computing).

An entsprechender Stelle finden Sie in diesem Kapitel noch eine Einführung in das *Domain Name System* (*DNS*) und in die Konfiguration des Nameservers *BIND*. Diese Kenntnisse benötigen Sie, um einen öffentlich verfügbaren Webserver zu betreiben. Abgerundet wird dieses Kapitel schließlich durch die kurze Vorstellung einiger praktischer TCP/IP-Tools.

## 1.1 TCP/IP

Der Vorläufer des Internets, das sogenannte *ARPANet*, wurde 1969 in Betrieb genommen. Aber erst knapp zehn Jahre später wurden die heutigen Netzwerkprotokolle eingeführt. Ein Netzwerkprotokoll ist ein Standard, der einen bestimmten Aspekt der Datenübertragung über ein Netzwerk regelt. *TCP/IP* ist der Name für eine ziemlich große Familie solcher Protokolle, benannt nach zwei ihrer wichtigsten Mitglieder: dem *Transmission Control Protocol* (*TCP*) und dem *Internet Protocol* (*IP*).

### 1.1.1 Das Internet-Schichtenmodell

Wie bei allen Computernetzen lassen sich auch beim TCP/IP-Netzwerk verschiedene Funktionsebenen oder Schichten unterscheiden. Diese Schichten betreffen jeweils eine andere Facette des Netzwerks.

Betrachten Sie für einen anschaulichen Vergleich das vorliegende Buch und seine »Schichten«:

1. Das Buch besteht aus bedrucktem Papier als »Trägermedium«.
2. Es enthält Linien, Kurven und Punkte, die von jemandem, der die lateinische Schrift lesen kann, als Buchstaben interpretiert werden.
3. Die Buchstaben ergeben Wörter und diese wiederum Sätze, die jeder versteht, der der deutschen Sprache mächtig ist.
4. Die Sätze verbinden sich schließlich für diejenigen zu sinnvollen Informationen, die grundlegende IT-Kenntnisse haben und etwas über den Apache-Webserver wissen möchten.

Dass diese Ebenen voneinander getrennt betrachtet werden können, zeigt sich daran, dass jede für sich austauschbar ist, ohne die anderen zu beeinflussen:

1. Der Text muss nicht auf Papier gedruckt sein: Ich schreibe ihn beispielsweise an einem Rechner und sehe ihn auf dem Monitor; daneben könnte er auch mit Zuckerguss auf eine (hinreichend große) Torte gespritzt oder in Stein gemeißelt werden.
2. Zur lateinischen Schrift gibt es bekanntermaßen zahlreiche Alternativen. Natürlich ist nicht jede Schrift gleich gut zur Wiedergabe jeder Sprache geeignet, aber prinzipiell kann man die Schrift wechseln und die Sprache beibehalten. Zum Beispiel wurde Türkisch früher mit arabischen Buchstaben geschrieben, heute dagegen mit leicht angepassten lateinischen (was übrigens besser funktioniert).
3. Wenn dieses Buch (lizenziert) in eine andere Sprache übersetzt würde, hätte ich überhaupt nichts dagegen.
4. Nicht alle Leser benötigen ein Apache-Buch. Vielleicht möchten Sie zuvor ein allgemeines UNIX-Buch wie z. B. [WILL2008][1] lesen, oder Sie interessieren sich für ganz andere Themen und lesen lieber Harry Potter, einen Krimi oder ein Pflanzenbestimmungsbuch.

Mit Netzwerken verhält es sich so ähnlich. Das bekannte OSI-Referenzmodell – auf das hier nicht näher eingegangen wird – besitzt sieben solcher Schichten; zur

1 Literaturangaben wie [WILL2008] (»UNIX – Das umfassende Handbuch« von Arnold Willemer, Bonn 2008, Galileo Computing) werden in Anhang I aufgelöst.

Beschreibung des Internets und sonstiger TCP/IP-Netze reichen dagegen vier aus. Nach dem DDN Standard Protocol Handbook sind diese vier Schichten folgende:

1. Die *Netzzugangsschicht* (*network access layer*[2]) beschreibt, auf welche Weise die angeschlossenen Rechner auf das Übertragungsmedium zugreifen, sich über die Sendereihenfolge einigen und mit Datenkollisionen umgehen.
2. Die *Internetschicht* (*internet layer*) beschreibt die eindeutige Adressierung der Rechner und die Weiterleitung von Daten über mehrere miteinander verbundene Netze hinweg, das sogenannte *Routing*. Dieser Name für das globale Datennetz ist übrigens älter als die Bezeichnung Internet; er weist auf die Vermittlung zwischen (lateinisch *inter*) mehreren Netzwerken hin.
3. Auf der *Host-zu-Host-Transportschicht* (*host-to-host transport layer*) werden die Daten in kleine Einheiten unterteilt, die Datenpakete, und von einer Anwendung auf dem einen Rechner (Host) zu einer Anwendung auf dem anderen geschickt.
4. Die *Anwendungsschicht* (*application layer*) schließlich definiert, wie sich verschiedene Anwendungsprogramme über das Netzwerk »unterhalten«.

Der Ablauf der Netzwerkkommunikation ist immer gleich: Eine Anwendung auf einem Rechner – die sich natürlich auf der Anwendungsschicht befindet – übergibt Daten an den passenden Dienst der Host-zu-Host-Transportschicht. Dieser teilt die Daten in kleinere Einheiten ein, die sogenannten *Datenpakete*, und versieht jedes Paket mit einer speziellen Transportinformation, dem Transport-Header. Die so vorbereiteten Pakete werden wiederum an die Internetschicht hinuntergereicht, die einen weiteren Header hinzufügt. Dieser enthält unter anderem die Absender- und die Empfängeradresse. Die Pakete, die mit all diesen zusätzlichen Informationen versehen wurden, heißen Datagramme. Erst in dieser Form werden sie tatsächlich der Netzwerkhardware anvertraut.

Der Empfängerrechner verfährt nun umgekehrt: Nachdem eine Funktion der Internetschicht festgestellt hat, dass die Daten tatsächlich für diesen Host bestimmt sind, reicht sie die Datagramme an den jeweils angesprochenen Transportdienst weiter. Dieser erkennt aus den Header-Informationen, für welche Anwendung die Daten bestimmt sind, und übergibt sie an diese.

Netzzugangsarten für TCP/IP-Netzwerke gibt es wie Sand am Meer; da sich die Internetprotokolle als der einzige ernst zu nehmende Standard für größere heterogene Netze durchgesetzt haben, kann TCP/IP mit beinahe jeder Art von Netzwerk- und DFÜ-Hardware unter so gut wie allen Betriebssystemen verwendet

2 Es gibt in der Literatur mehrere englische Bezeichnungen für die Schichten des TCP/IP-Protokollstapels; hier wird jeweils die gängigste verwendet.

werden. Aus diesem Grund würde es den Rahmen dieser kurzen Übersicht bei Weitem sprengen, sich hier mit einzelnen Zugangsformen zu befassen. Zu den wichtigsten gehören Ethernet und WLAN für lokale Netze; ein Großteil der Internetverbindungen findet dagegen bei den Endusern über verschiedene Arten von DSL-Leitungen statt, während die Backbone-Netze über schnelle Standleitungen miteinander verbunden sind.

Ähnlich sieht es übrigens mit der obersten Schicht aus, der Anwendungsschicht. In diesem Buch geht es verständlicherweise vor allem um das Anwendungsprotokoll *HTTP*, das *Hypertext Transfer Protocol*, über das sich Webserver wie Apache und Webbrowser untereinander verständigen. Da Apache sich mithilfe optionaler Module auch für andere Server-Dienste wie FTP oder Mail verwenden lässt, werden Sie auch einige andere Protokolle der Anwendungsschicht kennenlernen. Da Anwendungsprotokolle jeweils sehr speziell sind und sich nicht verallgemeinern lassen, gehören auch sie nicht hierher; HTTP wird in Kapitel 2, »Funktionsweisen von Webservern«, ausführlich behandelt.

Die folgenden Abschnitte konzentrieren sich daher auf die zwei mittleren Schichten, auf denen die beiden Protokolle arbeiten, die der ganzen Protokollfamilie den Namen gegeben haben: die Internetschicht mit dem Internet Protocol (IP) und die Host-zu-Host-Transportschicht mit dem Transmission Control Protocol (TCP) und seinem Konkurrenten, dem User Datagram Protocol (UDP).

### 1.1.2 Das Internet Protocol (IP)

Das Internet Protocol, die »Monopollösung« für die Internetschicht, löst vor allem zwei wichtige Aufgaben:

1. eindeutige Adressierung der Rechner und Netze
2. Weiterleitung der Datenpakete vom Absenderrechner an den Empfänger, auch über mehrere Netze hinweg. Letzteres wird als Routing bezeichnet.

Zur Adressierung besitzt jeder Rechner im TCP/IP-Netz eine sogenannte *IP-Adresse*. Es gibt zwei verschiedene Varianten des IP, die sich vor allem durch die Länge (und damit die verfügbare Anzahl) der Adressen unterscheiden. Das klassische IPv4-Protokoll verwendet 32 Bit lange Adressen, während die neuen IPv6-Adressen eine Länge von 128 Bit besitzen. IPv6 beginnt erst allmählich, sich durchzusetzen; der weit verbreitete Standard ist nach wie vor IPv4. Zwar wird dessen Adressraum im Internet allmählich knapp, aber der flächendeckende Umstieg auf die neue Version scheitert bisher an der Inkompatibilität der beiden Varianten miteinander sowie an der mangelnden IPv6-Unterstützung durch manche Netzwerkhard- und -software. Dennoch werden in diesem Buch auch Besonderheiten der Konfiguration von Apache 2 für IPv6 erwähnt.

Wie alle Internetprotokolle und -standards ist auch das IP öffentlich verfügbar in sogenannten *RFC-Dokumenten* (*Request For Comments*) beschrieben. Alle Personen, Institutionen und Firmen, die an der Weiterentwicklung des Internets beteiligt sind, stellen ihre Vorschläge und Lösungen seit über 30 Jahren in solchen freien Dokumenten zur Verfügung; es gibt inzwischen knapp 5.300 davon. Sie können sie z. B. unter *http://www.faqs.org/rfcs* nachlesen. IPv4 ist in RFC 791 dokumentiert; die neue Variante IPv6 in RFC 2460.

### IPv4-Adressierung

Die 32 Bit langen IPv4-Adressen werden üblicherweise dezimal als einzelne, durch Punkte getrennte Bytes geschrieben, z. B. *137.51.8.41*. Ein Teil dieser Adresse bezeichnet das Netzwerk, in dem sich der Rechner befindet, der Rest den Rechner innerhalb dieses Netzes selbst. Da es verschieden große Netzwerke gibt, haben die Entwickler von Anfang an verschiedene Arten – sogenannte *Klassen* – von IPv4-Netzen geplant. Bei der – inzwischen überholten – klassenbasierten Adressierung geben allein die Bits am Anfang der Adresse darüber Auskunft, zu welcher Klasse eine IP-Adresse gehört, und damit, wie viele Bits der Adresse das Netz und wie viele den Rechner (Host) kennzeichnen:

- Wenn das erste Bit 0 ist, gehört die Adresse zur *Klasse A*. Das erste der 4 Bytes liegt also im Bereich 0 bis 127. In dieser Klasse kennzeichnen die ersten 8 Bit das Netz und die restlichen 24 den Host. Ein Klasse-A-Netz bietet jeweils etwa 16,7 Millionen Hostadressen.
- Wenn die ersten 2 Bit 10 sind, handelt es sich um die *Klasse B*; das erste Byte hat einen Wert zwischen 128 und 191. Hier stehen die ersten 16 Bit für das Netz und die anderen 16 für den Host. Ein Klasse-B-Netz enthält demnach 65.536 Hostadressen.
- Sind die ersten 3 Bit 110, dann gehört das Netz zur *Klasse C*. Das erste Byte hat also einen Wert zwischen 192 und 223. 24 Bit adressieren das Netz, und nur 8 Bit sind für den Host zuständig, sodass es nur 256 Hostadressen in einem Netz gibt.
- Adressen, bei denen die ersten 4 Bit 1.110 sind und deren erstes Byte damit einen Wert zwischen 224 und 239 hat, gehören zur *Klasse D*. Sie dienen nicht der Adressierung einzelner Rechner, sondern sind sogenannte *Multicast-Adressen*. Sie ermöglichen die Verteilung von Daten an mehrere Rechner, denen dieselbe Multicast-Adresse zugewiesen wird. Dies ist z. B. ideal für Videokonferenzen, Video-on-Demand oder andere Streaming-Dienste. Dies gilt übrigens trotz Aufhebung der Klassenlogik im Wesentlichen noch heute.

In Tabelle 1.1 sehen Sie eine genaue Übersicht über die IP-Adressklassen. Die Anzahl der Netzwerk-Bits ist doppelt angegeben: Der erste Wert gibt die Gesamtzahl an, während der Wert in Klammern die relevante Anzahl von Netzwerk-Bits abzüglich der Klassenkennzeichnung angibt – bei der Klasse A beispielsweise 8 (7), da das erste Bit 0 lauten *muss*. Die Anzahl der pro Netz verfügbaren Hosts ist um zwei geringer, als sich rechnerisch ergeben. Das liegt daran, dass die niedrigste Adresse eines Netzes das Netzwerk selbst kennzeichnet und die höchste – die Broadcast-Adresse – dazu dient, innerhalb des Netzes Daten an alle angeschlossenen Rechner zu versenden.

| Klasse | Start-Byte | Netzwerk-Bits | Host-Bits | Anzahl Netze | Anzahl Hosts |
|---|---|---|---|---|---|
| A | 0–127 | 8 (7) | 24 | 128 | 16.777.214 |
| B | 128–191 | 16 (14) | 16 | 16.384 | 65.534 |
| C | 192–223 | 24 (21) | 8 | 2.097.152 | 254 |
| D | 224–239 | Multicast-Adressen | | | |

**Tabelle 1.1** Die IP-Adressklassen und ihre Eigenschaften

Als das Internet Protocol entwickelt wurde, schien die Anzahl der verfügbaren Netze, die sich aus der Klassenlogik ergibt, locker auszureichen. Da die Anzahl der Rechner und Netze im Internet jedoch schon seit Langem exponentiell zunimmt, erweist sich das Klassensystem allmählich als zu starr und verschwenderisch: Allein die Hälfte aller rechnerisch verfügbaren Adressen wird für die Klasse A benutzt, obwohl selbst die größten Netze der Welt wohl kaum aus den genannten etwa 16,7 Millionen Hosts bestehen. Aus diesem Grund wurde ein flexibleres System namens *CIDR* (*Classless Inter-Domain Routing*) eingeführt. Es ermöglicht das Setzen der Grenze zwischen Netz- und Hostteil der Adresse bei einem beliebigen Bit. Bei einer CIDR-Adresse muss diese Grenze deshalb angegeben werden. Dafür gibt es zwei verschiedene Schreibweisen:

- Die Anzahl der Netzwerk-Bits wird durch einen Slash getrennt hinter die Adresse geschrieben: 192.78.16.97/24 ist beispielsweise die CIDR-Angabe für die Klasse-C-Adresse 192.78.16.97.
- Alternativ kann die Grenze zwischen Netz- und Host-Bits separat durch eine sogenannte *Subnet Mask* (deutsch *Teilnetzmaske*) angegeben werden: Die Maske wird wie die IP-Adresse selbst dezimal in 4 Einzel-Bytes geschrieben; Bits der Netzwerkadresse werden auf den Wert 1 gesetzt, Bits der Hostadresse auf 0. Die Maske für ein Klasse-C-Netz ist demnach 255.255.255.0.

Die Besonderheit von CIDR wird deutlicher, wenn man eine Adresse betrachtet, die nicht zu einer der alten Klassen gehört, sondern eine individuelle Bit-Grenze besitzt: 160.76.153.12/19 gehört nicht zu dem Klasse-B-Netz 160.76.0.0 (CIDR-

Adresse 160.76.0.0/16), sondern zu dem Netz 160.76.128.0/19. Es handelt sich um eines der Netze, die durch die Aufteilung des Netzes 160.76.0.0/16 in acht Teilnetze entstehen. Diese Netze werden in Tabelle 1.2 gezeigt. Mit einem Hostteil von 13 Bit enthält jedes dieser Netze 8.190 Hosts (8.192 Einzeladressen inklusive Netzwerk- und Broadcast-Adresse). Die Subnet Mask lautet 255.255.224.0.

| Netzwerk | Erster Host | Letzter Host | Broadcast-Adresse |
|---|---|---|---|
| 160.76.0.0/19 | 160.76.0.1 | 160.76.31.254 | 160.76.31.255 |
| 160.76.32.0/19 | 160.76.32.1 | 160.76.63.254 | 160.76.63.255 |
| 160.76.64.0/19 | 160.76.64.1 | 160.76.95.254 | 160.76.95.255 |
| 160.76.96.0/19 | 160.76.96.1 | 160.76.127.254 | 160.76.127.255 |
| 160.76.128.0/19 | 160.76.128.1 | 160.76.159.254 | 160.76.159.255 |
| 160.76.160.0/19 | 160.76.160.1 | 160.76.191.254 | 160.76.191.255 |
| 160.76.192.0/19 | 160.76.192.1 | 160.76.223.254 | 160.76.223.255 |
| 160.76.224.0/19 | 160.76.224.1 | 160.76.255.254 | 160.76.255.255 |

**Tabelle 1.2** Alle Netze, die durch CIDR-Aufteilung (Subnetting) des Netzes 160.76.0.0/16 entstehen

Noch flexibler als CIDR ist die sogenannte *VLSM-Adressierung* (Variable Length Subnet Mask). Durch dieses Adressierungsschema lässt sich ein Netzwerk in Teilnetze unterschiedlicher Größe unterteilen, was beispielsweise in einer Firma mit unterschiedlich großen Abteilungen sehr nützlich ist.

Ein weiterer nützlicher IP-Dienst, der übrigens die Knappheit der IPv4-Adressen um einige Jahre verzögern konnte, ist NAT (Network Address Translation). Ein NAT-Gateway ersetzt die privaten IP-Adressen (siehe nächsten Absatz) ausgehender Datenpakete durch öffentliche Adressen und verfährt bei eingehenden Datenpaketen umgekehrt. Auf diese Weise kann das Netzwerk mit den privaten Adressen versehen werden und dennoch mit dem Internet verbunden sein. Zudem erhöht dieses Verfahren die Sicherheit, weil die eigentliche Infrastruktur des Netzes vor potenziellen Eindringlingen verborgen wird.

Jede IP-Adresse muss im Internet einmalig sein. Allerdings hat die IANA (Internet Assigned Numbers Authority) einige Adressen zur Verwendung in privaten, nicht direkt mit dem Internet verbundenen Netzwerken freigegeben:

- das Netz 10.0.0.0/8
- die 16 Netze 172.16.0.0/16 bis 172.31.0.0/16
- die 256 Netze 192.168.0.0/8 bis 192.168.255.0/8

Eine besondere Bedeutung haben darüber hinaus die folgenden IPv4-Adressen:

- 127.0.0.1 ist die sogenannte *Loopback-Adresse*, über die ein Host mit sich selbst TCP/IP-Kommunikation betreiben kann. Beim Testen der Apache-Konfiguration und serverseitiger Anwendungen werden Sie diese Adresse oft verwenden, um Client und Server auf demselben Rechner zu betreiben.
- 255.255.255.255 ist die allgemeine Broadcast-Adresse. Sie kann z. B. von einem Host verwendet werden, der beim Booten seine eigene IP-Adresse im Netz erfragt, weil sie über den Dienst DHCP dynamisch zugewiesen wird.
- Das Netz 169.254.0.0/16 wird für APIPA (Automatic Private IP Addressing) oder »link local« verwendet – über diesen Dienst kann sich ein DHCP-Client selbst eine Adresse zuweisen, wenn aus irgendeinem Grund kein DHCP-Server verfügbar ist.

#### IPv6-Adressierung

IPv6-Adressen sind, wie erwähnt, 128 Bit lang. Dies ergibt einen rechnerischen Adressraum von unvorstellbaren $3{,}4 \times 10^{38}$ Adressen. Auch wenn diese Anzahl von Hosts wohl niemals erreicht werden wird, besitzt eine so große Adresszahl andere Vorteile: Erstens sind Unmengen von Netzen mit zahlreichen verschiedenen Größen möglich, und zweitens passt die IPv6-Adressierung zu den zahlreichen Geräten, die inzwischen neben den klassischen Allround-Computern ins Internet drängen: Handys, PDAs, MP3-Player, Armbanduhren und sogar Kühlschränke und Mikrowellengeräte.

Da es etwas unhandlich wäre, die IPv6-Adressen in 16 dezimalen Byte-Gruppen zu schreiben, werden sie stattdessen in acht vierstelligen Hexadezimalblöcken dargestellt. Eine IPv6-Adresse lautet also beispielsweise so: 09AC:7B03:CC3D:BD88:0000:0037:6293:CCF1. Sie lässt sich abkürzen, indem die führenden Nullen jeder Gruppe weggelassen werden und indem eine zusammenhängende Gruppe aus reinen Nullenblöcken durch :: dargestellt wird. Die Beispieladresse lautet in dieser Schreibweise dann also folgendermaßen: 9AC:7B03:CC3D:BD88::37:6293:CCF1.

Die Grenze zwischen Netzwerk- und Hostteil der IPv6-Adressen wurde von Anfang an flexibel gesetzt; wie bei IPv4-Adressen wird die Bit-Anzahl des Netzwerkteils durch einen / getrennt hinter die Adresse geschrieben; dabei kann der Rest der Adresse weggelassen werden, wenn das Netzwerk als solches gemeint ist, z. B. 9AC:7B03/32.

Während die automatische Zuweisung der Adressen sich bei IPv4 nur durch nachträglich erfundene Dienste wie DHCP oder APIPA realisieren lässt, ist sie bei IPv6 die Regel. Aus diesem Grund wird der Hostteil der Adresse bei einem Ether-

net-LAN beispielsweise automatisch aus der 48 Bit langen MAC-Adresse (Media Access Control; eindeutige Hardwareadresse) der Netzwerkschnittstelle – ergänzt um einen 16 Bit langen Kennzeichnungsblock – gebildet.

In der Praxis wird IPv6 heute vornehmlich durch *Tunnelung* benutzt: Die IPv6-Datenpakete werden in IPv4-Datenpakete »verpackt« (erhalten also einen IPv4-Paket-Header) und über das IPv4-adressierte Internet zum Ziel geleitet – einem Netzwerk, das intern die entsprechende IPv6-Adressierung versteht. Es gibt zahlreiche kommerzielle und freie Anbieter solcher IPv6-Tunneldienste.

### IP-Routing

Das Wichtigste am Konzept des IPs ist die Fähigkeit, Daten über mehrere miteinander verbundene Netze weiterzuleiten. Diese Aufgabe übernehmen spezialisierte Rechner, die *Router*. Sie sind jeweils mit mindestens zwei verschiedenen Netzwerken verbunden und verfügen über Informationen darüber, welche Datenpakete sie wohin weiterleiten müssen, damit diese letztlich ihr Ziel erreichen. Während diese Routing-Informationen innerhalb kleinerer Firmennetze statisch konfiguriert werden, führen die Router der Internet-Provider und anderer Betreiber großer Netze sogenannte *Routing-Protokolle* aus, mit deren Hilfe die Routing-Konfiguration automatisch veröffentlicht und verbreitet wird.

Auch jeder Host im TCP/IP-Netzwerk benötigt grundlegende Routing-Informationen, um Daten jeweils über die richtige Netzwerkschnittstelle an den korrekten Router weiterzuleiten. Aus der Sicht eines Hosts gibt es zwei Sorten von Routern: Einen »normalen« Router, an den der Host Daten weitergibt, deren Ziel ein bestimmtes Netzwerk ist, und das Default Gateway (der Standard-Router), dem alle Datenpakete übergeben werden, für deren Ziel eben kein spezieller Router konfiguriert wurde.

Angenommen, das Netzwerk 196.23.17.0/24 einer kleinen Firma mit der Domain *mynet.de* ist über den Router *intern* (196.23.17.2) mit dem internen Nachbarteilnetz 196.23.18.0/24 verbunden, während *extern* (196.23.17.1) das Default Gateway ist. Im Teilnetz 196.23.18.0/24 besitzt *intern* die IP-Adresse 196.23.18.1 und ist dessen einziger Router und damit Default Gateway.

Aus diesem Szenario ergibt sich für einen Host im Netz 196.23.17.0/24 die folgende Routing-Situation:

- Datenpakete für Rechner im Netzwerk 196.23.17.0/24 werden ohne Umweg über einen Router direkt an den Empfänger versendet.
- Daten, die für Hosts im Netz 196.23.18.0/24 bestimmt sind, werden dem Router *intern* (196.23.17.2) zur Weiterleitung übergeben.

- Daten für alle anderen Netze, also für das gesamte Internet, werden an den Router *extern* (196.23.17.1) weitergegeben.

Ein Host im Netzwerk 196.23.18.0/24 besitzt dagegen eine einfachere Routing-Konfiguration; da es in diesem Teilnetz nur einen Router gibt, brauchen nur zwei Fälle unterschieden zu werden:

- Daten für das Netz 196.23.18.0/24 werden ohne Routing unmittelbar zugestellt.
- Die restlichen Datenpakete werden dem Router *intern* (hier: 196.23.18.1) übergeben; dieser weiß natürlich, dass er Daten für das Netz 196.23.17.0/24 über seine Schnittstelle 196.23.17.2 unmittelbar zustellen kann, während er alle anderen an den nächsten Router *extern* (196.23.17.1) weiterleiten muss.

In der Praxis führen Router eigene, komplexe Protokolle aus, um Informationen über die jeweils erreichbaren Netze miteinander auszutauschen. Man unterscheidet zwei Arten: Interne Routing-Protokolle wie das Routing Information Protocol (RIP) arbeiten innerhalb der Netzwerke eines einzelnen Anbieters (autonomes System), während externe Routing-Protokolle – etwa das Border Gateway Protocol (BGP) – verschiedene autonome Systeme miteinander verbinden.

Damit ein IP-Datagramm nicht endlos im Netz kreist, wenn sein Bestimmungsort nicht gefunden werden kann, enthält sein Header ein 8 Bit breites Feld namens TTL (Time To Live). Jeder einzelne Router, den das Paket passiert, zieht von diesem Wert 1 ab. Sobald er 0 ist, verwirft der Router das Paket. Ein Datagramm kommt also entweder innerhalb von 256 Hops (Routing-Schritten) an oder gar nicht. In der Regel sind allerdings erheblich weniger Hops erforderlich.

### 1.1.3 Transportprotokolle

Aufgabe der Host-zu-Host-Transportschicht ist die direkte Lieferung der Datenpakete von einer Anwendung auf einem Host zu einer Anwendung auf einem anderen Host. Ein Transportprotokoll nimmt die Daten von einem Anwendungsprogramm entgegen, das über das Netzwerk kommunizieren möchte. Es teilt die Daten in Datenpakete auf, versieht sie mit einer eindeutigen Nummer, die die entsprechende Anwendung kennzeichnet, und reicht sie an das IP weiter, das die tatsächliche Datenweiterleitung über das Netzwerk durchführt.

Im Internet-Protokollstapel gibt es zwei verschiedene Transportprotokolle. Jede Netzwerkanwendung kann sich dasjenige aussuchen, das besser zu ihr passt: Das TCP garantiert die Auslieferung jedes einzelnen Datenpakets und sorgt durch eine Datenpaketnummerierung dafür, dass die Pakete am Ziel wieder in der rich-

tigen Reihenfolge zusammengesetzt werden können. Das UDP ist dagegen erheblich schneller als TCP, kann aber nur einzelne Pakete ohne Reihenfolge übertragen und überprüft auch nicht, ob sie tatsächlich ankommen. Während TCP also datenstromorientiert ist, ist UDP nachrichtenorientiert.

#### TCP

Das TCP (RFC 793) sorgt für die Übertragung eines kontinuierlichen Datenstroms. Obwohl netzwerktypisch noch immer jedes einzelne Paket einen beliebigen Routing-Weg zum Ziel wählen kann, wird durch die eingebaute Flusskontrolle eine virtuelle Punkt-zu-Punkt-Verbindung zwischen den beiden Hosts eingerichtet.

Die TCP-Verbindung beginnt mit dem sogenannten *Drei-Wege-Handshake*:

- Der Host, der die Verbindung initiiert (üblicherweise der Client, denn er »will etwas« vom Server), sendet dem Empfänger ein Datenpaket, in dessen TCP-Header ein spezielles Bit namens SYN gesetzt ist.
- Der Empfänger antwortet mit einem Paket, in dem die beiden Flags SYN und ACK gesetzt sind.
- Der ursprüngliche Absender beantwortet dieses Paket noch einmal mit einem Datenpaket, in dem nur ACK gesetzt ist.

Diese Vorgehensweise garantiert, dass beide Seiten sowohl senden als auch empfangen können. Die Flusskontrolle kommt dadurch zustande, dass der Header jedes Pakets eine Sequenznummer enthält. Sie gibt den Offset des Datenbytes für den Teil des Gesamtdatenstroms an, der mit diesem Paket versendet wird. Der Absender erwartet dann für jedes einzelne Paket eine Bestätigung, in der das ACK-Bit gesetzt ist und die eine Bestätigungsnummer enthält: den Byte-Offset des nächsten erwarteten Pakets. Falls eine Bestätigung innerhalb einer bestimmten Zeit ausbleibt, sendet der Absender das Paket einfach erneut.

Die Identifikation der beteiligten Anwendungen erfolgt durch ein Paar 16 Bit langer Portnummern. Da eine bestimmte Verbindung durch beide Portnummern gekennzeichnet wird, ist es möglich, dass ein TCP-Server jeweils eine feste Portnummer hat, während ein Client eine zufällige Nummer (»Ephemeral Ports«) erhält. Die wichtigsten Server-Dienste haben festgelegte Ports (»Well-Known-Ports«) zwischen 0 und 1023 – ein Webserver hat z. B. standardmäßig die Portnummer 80. Auf UNIX-Systemen dürfen diese Ports lokal nur durch root verwendet werden. Daneben gibt es auch registrierte Server-Ports über 1024, z. B. 3306 für MySQL. Alle Zuweisungen finden Sie im Internet unter *http://www.iana.org/assignments/port-numbers*.

### UDP

Wenn Einzeldaten ihr Ziel so schnell wie möglich erreichen sollen, es aber nicht auf Vollständigkeit oder auf die richtige Reihenfolge einer größeren Datenmenge ankommt, kann eine Anwendung UDP statt TCP verwenden. Dieses Protokoll ist in RFC 768 spezifiziert. Der Name hat damit zu tun, dass die IP-Datagramme der Anwendung ohne großen zusätzlichen Aufwand zur Verfügung stehen.

Der Hauptgrund, warum UDP schneller ist als TCP, liegt am wesentlich kleineren Paket-Header. Das Verhältnis zwischen Konfigurations- und Nutzdaten ist also erheblich günstiger als bei TCP. Der Header enthält nämlich nichts weiter als die beiden bei TCP beschriebenen Portnummern (hier Service-Nummern genannt) sowie Paketlänge und Prüfsumme.

Ein UDP-Datenpaket ist seiner Natur gemäß eine einzelne Nachricht, die so schnell wie möglich ans Ziel befördert wird. Der Absender kümmert sich nicht darum, ob sie ankommt oder nicht, und erwartet auch keine Empfangsbestätigung. Man könnte TCP gewissermaßen als »Einschreiben mit Rückschein« bezeichnen, während UDP eher einer Postkarte entspricht.

## 1.2 Das Domain Name System (DNS)

Für die Netzwerkkommunikation zwischen Computern sind IP-Adressen eine hervorragende Einrichtung, und zwar sowohl IPv4- als auch die längeren IPv6-Adressen. Menschen dagegen bevorzugen den Umgang mit Namen gegenüber der Verwendung von Nummern. Aus diesem Grund ist es nützlich, wenn die Hosts im Internet neben der numerischen Adresse auch einen für Menschen leichter zu merkenden Namen besitzen. Die Schwierigkeit dabei ist nur, eine Möglichkeit zu schaffen, mit der dieser Name auf effiziente und wenig störanfällige Weise in die IP-Adresse umgewandelt werden kann.

Als das ARPANet in Betrieb genommen wurde, bestand es aus sehr wenigen großen Rechnern an verschiedenen amerikanischen Universitäten. Zwar wuchs es von Anfang an recht schnell, aber selbst mit einigen Hundert beteiligten Knoten war es noch kein allzu großes Problem, eine Liste der Hostnamen mit den entsprechenden Adressen als Textdatei zu pflegen, weil sich Änderungen eher in Grenzen hielten. Diese Datei hieß `hosts.txt` und wurde regelmäßig an die beteiligten Institutionen verteilt.

Noch heute enthält jeder Computer mit einem Betriebssystem aus der UNIX-Familie eine Datei namens `/etc/hosts`, die zur lokalen Namensauflösung dient.

Unter Windows befindet sie sich dagegen unter *%Systemroot%\ System32\drivers\etc\hosts*[3]. Diese Datei besitzt folgende Struktur:

```
127.0.0.1       localhost
196.23.17.1     defaultgw   defaultgw.mynet.de
196.23.17.2     mailserver  mailserver.mynet.de
```

In jeder Zeile steht also eine IP-Adresse, gefolgt von einem oder mehreren Hostnamen, die in diese Adresse aufgelöst werden sollen. In den meisten Systemen lässt sich konfigurieren, ob Einträge in dieser Datei beim Nachschlagen von Namen Vorrang vor externen Nameservern haben sollen oder nicht.

Im Laufe der Jahre wuchs das Internet immer weiter, und irgendwann war der Austausch der stets aktualisierten *hosts.txt*-Datei nicht mehr effektiv genug. Aus diesem Grund machte man sich an die Arbeit, ein neues System mit einer verteilten Datenbank aufzubauen. Dieses System heißt *DNS* (*Domain Name System*); das Grundkonzept wird in RFC 1034 und 1035 beschrieben.

### 1.2.1 Das DNS-Konzept

DNS besteht im Wesentlichen aus zwei wichtigen Elementen: zum einen aus den Nameservern, die eine DNS-Server-Software ausführen und auf Anfrage Namensinformationen herausgeben, und zum anderen aus dem hierarchischen System der Domain-Namen selbst.

Das System der Domain-Namen ist in etwa mit einem Dateisystem vergleichbar, das Verzeichnisse, Unterverzeichnisse und Dateien enthält. Angenommen, ein UNIX-Dateisystem enthält die folgende Datei:

*/home/sascha/books/apache2/kap01.txt*

Dies bedeutet, dass sich die Datei *kap01.txt* im Verzeichnis *apache2* befindet. Dieses Verzeichnis ist ein Unterverzeichnis von *books*; dies wiederum ist ein Unterverzeichnis von *sascha* im Verzeichnis *home*. Das Verzeichnis *home* schließlich befindet sich unterhalb des Wurzelverzeichnisses */*.

Bei einem DNS-Domain-Namen verhält es sich ähnlich, allerdings ist die Reihenfolge der Hierarchie umgekehrt. Betrachten Sie etwa den folgenden Domain-Namen:

*www.buecher.lingoworld.de*

3 `%Systemroot%` ist eine Windows-Umgebungsvariable, die für das Systemverzeichnis steht – häufig ist es *C:\Windows*.

Es handelt sich um den Host/Dienst `www` in der Subdomain `buecher`, die sich in der Second-Level-Domain `lingoworld` unterhalb der Top-Level-Domain (TLD) `de` – ISO-Länderkürzel für Deutschland – befindet. Die eigentliche Wurzel des DNS ist an dieser Stelle nicht sichtbar; ihr Name ist der leere String `""`.

Wer seine Rechner im Internet unter einem Domain-Namen betreiben möchte, muss zunächst eine Second-Level-Domain unter der passenden Top-Level-Domain registrieren. Je nach TLD ist jeweils eine andere Registrierungsstelle zuständig. Um die Top-Level-Domain `de` kümmert sich beispielsweise die DENIC (*www.denic.de*). In der Praxis ist es jedoch einfacher und meist billiger, einen Domain-Namen über einen Provider zu registrieren.

Für kleine bis mittlere Unternehmen, die nichts weiter als eine Website und vielleicht ein paar E-Mail-Adressen benötigen, kommt hier am ehesten ein virtueller Host (siehe Kapitel 12, »Skalierung und Performance-Tuning«) infrage: Ein Domain-Name wie *www.mynet.de* verweist auf eine Website, die sich in Wirklichkeit zusammen mit Hunderten anderer Sites auf demselben Server-Rechner bei dem Provider befindet. Darüber hinaus lassen sich natürlich E-Mail-Adressen nach dem Schema *user@mynet.de* anlegen.

Wenn Sie dieses Buch aus praktischen Gründen lesen, ist ein virtueller Host bei einem großen Hosting-Dienst natürlich nicht das Richtige für Sie. Sie benötigen Ihren eigenen Server-Rechner (oder auch mehrere), auf dem Sie Apache selbst nach Ihren Wünschen konfigurieren können. Möglicherweise wollen Sie auch selbst Hosting-Dienstleistungen für Ihre Kunden erbringen. Im ersten Fall wird sich ebenfalls der Provider um die DNS-Konfiguration kümmern; *mynet.de* verweist nun eben nicht mehr auf einen virtuellen, sondern auf einen tatsächlichen Host. Wenn Sie dagegen selbst Hoster sind, müssen Sie die virtuellen oder echten Hosts für Ihre Kunden einrichten, und dazu gehören natürlich auch die korrekten DNS-Einstellungen.

Eine DNS-Domain, die in der Obhut eines bestimmten Netzbetreibers steht, wird als *DNS-Zone* bezeichnet. Im Beispiel *www.buecher.lingoworld.de* ist *lingoworld.de* eine solche Zone. Der Administrator von *lingoworld.de* kann Hosts innerhalb dieser Zone einrichten, indem er DNS-Daten mit ihren Hostnamen und IP-Adressen erstellt. Sollte er eine Subdomain wie *buecher.lingoworld.de* anlegen, kann er die Hosts und sonstigen Informationen darin entweder selbst konfigurieren oder aber die Administration dieser Subdomain delegieren und somit zur neuen, unabhängigen Zone machen. Dieses System der optionalen Delegation garantiert größtmögliche Flexibilität.

Interessant ist schließlich, wie eine DNS-Anfrage funktioniert. Ein Host, der Daten an einen bestimmten Empfänger im Internet senden möchte, aber nur des-

sen Hostnamen kennt, muss einen Nameserver befragen, um die zugehörige IP-Adresse zu erhalten.

Jeder Host, der direkt mit dem Internet verbunden ist, kennt mindestens einen Nameserver – entweder wurde dieser bei der Netzwerkkonfiguration manuell angegeben, oder er wurde bei der automatischen Schnittstellenkonfiguration über DHCP (Dynamic Host Configuration Protocol; im LAN) oder PPP (Point-to-Point Protocol; bei Modem, ISDN, DSL) vom entsprechenden Anmelde-Server übermittelt.

Der Host, der eine Namensauskunft wünscht, befragt nun die ihm bekannten Nameserver der Reihe nach. Angenommen, es wird die IP-Adresse zu *www.othernet.com* gesucht. Ein Nameserver, der diese Anfrage erhält, versucht nun, diese Information zu ermitteln und an den Anfragenden zurückzugeben. Damit er dieselbe Antwort beim nächsten Mal schneller geben kann, speichert er die ermittelten Daten in einem Cache (Zwischenspeicher). Wenn der befragte Nameserver bereits den Nameserver kennt, der für die Zone *othernet.com* zuständig ist, wendet er sich direkt an diesen, nimmt die Antwort in Empfang und gibt sie aus. Andernfalls muss er »eine Etage höher« nachfragen: Die root-Nameserver für die Top-Level-Domain `de` kennen auf jeden Fall mindestens einen Nameserver für jede unterhalb dieser TLD registrierte Domain. Mit dieser Information kann der ursprünglich beauftragte Nameserver also weiterforschen und erhält schließlich die passende Antwort.

Es gibt im Grunde drei Arten von Nameservern:

- primäre Master-Nameserver
- Slave-Nameserver
- Caching-Only-Nameserver

Auf einem primären Master-Nameserver werden die maßgeblichen – die sogenannten *autoritativen* – Informationen für eine DNS-Zone konfiguriert. Dieser Server ist »der Weisheit letzter Schluss« für die entsprechende Zone.

Ein Slave-Nameserver dient der *Replikation*, enthält also eine Kopie der Daten eines primären Master-Nameservers. Dies dient sowohl der Ausfallsicherheit als auch der Entlastung des Master-Servers. Aus Sicherheitsgründen ist es ratsam, die DNS-Daten des eigenen primären Master-Nameservers auf mindestens einem Slave-Nameserver zu speichern, der sich nicht innerhalb der eigenen Netzwerkinfrastruktur befindet. Größere Unternehmen und Internet-Provider stellen einander diese Dienstleistung oft gegenseitig zur Verfügung.

Caching-Only-Nameserver werden oft in kleineren Unternehmen eingesetzt, die ihre öffentlichen Server nicht innerhalb ihrer Geschäftsräume betreiben. Diese Server enthalten keinerlei autoritative Informationen und sind auch keine direkten

Replikationspartner primärer Master-Nameserver, sondern beschleunigen durch die Zwischenspeicherung lediglich wiederholte Zugriffe auf externe DNS-Daten, oder sie dienen dem Zugriff auf Namensinformationen über Firewalls hinweg.

Im Sommer 2008 entdeckte der Sicherheitsexperte Dan Kaminsky ein ernsthaftes Sicherheitsproblem, das die meisten DNS-Implementierungen betrifft: Durch sogenanntes *Cache-Poisoning* ist es möglich, die Caches beliebiger Nameserver mit falschen Informationen zu füttern und Anwender, die auf beliebte Domains zugreifen, so auf Seiten mit anderen IP-Adressen umzuleiten, die dann beispielsweise versuchen, als rechtmäßige Downloads getarnte Viren oder Trojaner zu installieren. Die eigentliche Schwachstelle besteht darin, dass die Transaction-ID, die zur Identifizierung der Cache-Updates dient, bisher fortlaufend vergeben wurde und so leicht erraten werden kann. Für alle wichtigen Nameserver gibt es inzwischen Sicherheitsupdates, die Sie dringend einspielen sollten. Weitere Informationen zum Thema finden Sie beispielsweise unter *http://www.heise.de/security/news/meldung/110641*.

### 1.2.2 Der DNS-Server BIND

Es gibt verschiedene Implementierungen von Server-Software, die den DNS-Dienst versieht. Die wichtigste von ihnen ist *BIND* (*Berkeley Internet Naming Domain*). Es handelt sich um Open-Source-Software des Internet Systems Consortiums (ISC). Sie können BIND und entsprechende Informationen darüber von der Website *http://www.isc.org/products/BIND* und zahlreichen dort verzeichneten Mirror-Sites herunterladen. Die zurzeit aktuellen Versionen sind BIND 9.7.4 und 9.8.1; diese Versionen sind bereits gegen das oben erwähnte Sicherheitsproblem geschützt. Wenn Sie ausführlichere Informationen benötigen, finden Sie eine praxisorientierte Anleitung in [LIU2003].

Eine andere weitverbreitete Nameserver-Software ist der Microsoft DNS Server. Er ist in die Windows-Server-Betriebssysteme Windows Server 2003 und Windows Server 2008 integriert. Selbstverständlich erfüllt er nach außen dieselbe Funktion wie BIND, da er mit dem globalen DNS kompatibel ist. Da die Nutzung eines Windows-Servers in aller Regel mit der Verwendung der Microsoft Internet Information Services (IIS) als Webserver einhergeht, braucht die Konfiguration dieses Dienstes in einem Buch über den Apache nicht beschrieben zu werden.

#### BIND installieren

In aktuellen UNIX- oder Linux-Distributionen ist BIND normalerweise bereits enthalten. Sie können den Server also problemlos über den Paketmanager Ihres Systems installieren. Wenn Sie nicht genau wissen, wie das funktioniert, konsultieren Sie die Dokumentation Ihres Systems oder gehen Sie analog zu dem Ver-

fahren vor, das für verschiedene Betriebssysteme in Kapitel 4, »Apache kompilieren und installieren«, für den Apache-Webserver beschrieben wird.

Wenn BIND nicht mit Ihrem System geliefert wird oder wenn Sie die neueste Version installieren möchten, sollten Sie BIND im Sourcecode herunterladen und selbst kompilieren. Das funktioniert, wie bei *autoconf/automake*-Software üblich, nach dem folgenden bewährten Schema:

- Entpacken Sie BIND in ein beliebiges Verzeichnis, und wechseln Sie in das neue Unterverzeichnis:

  ```
  # tar xzvf bind-9.8.1.tar.gz
  ```

  (es folgen zahlreiche Ausgabezeilen)

  ```
  # cd bind-9.8.1
  ```

- Rufen Sie das Konfigurationsprogramm und anschließend `make` auf, um BIND mit den passenden Optionen für Ihr System zu kompilieren:

  ```
  # ./configure
  # make
  ```

- Rufen Sie zu guter Letzt den Installationsbefehl auf:

  ```
  # make install
  ```

Normalerweise wollen Sie anschließend dafür sorgen, dass der Nameserver – das Programm `named` – beim Booten automatisch gestartet wird. Da dies unter verschiedenen UNIX-Systemen (und selbst bei den verschiedenen Linux-Distributionen) unterschiedlich funktioniert, können Sie sich auch hier an die Anleitung halten, die in Kapitel 5, »Apache in Betrieb nehmen«, für den automatischen Start von Apache für Ihr Betriebssystem zur Verfügung gestellt wird.

#### BIND konfigurieren

Die Konfigurationsdaten von BIND bestehen aus zwei verschiedenen Dateien: Die Datei */etc/named.conf* enthält die Konfigurationsinformationen über den Nameserver selbst, während sogenannte *Zonendaten-Dateien* die eigentlichen Namensdaten in Form von *Resource Records* enthalten. Beide Dateiarten sind einfache Textdateien, die Sie mit Ihrem bevorzugten Editor bearbeiten können.

Die erste wichtige Angabe in der Datei *named.conf* beschreibt das Arbeitsverzeichnis, in dem sich die Zonendaten-Dateien befinden. Da diese Anweisung global ist und keine bestimmte Zone betrifft, wird sie in einen `options`-Block gesetzt:

```
options {
   directory "/var/named";
};
```

Anschließend folgen ein oder mehrere `zone`-Blöcke. Sie enthalten die Konfiguration der Zonen, für die der Nameserver zuständig sein soll.

Falls Ihr Nameserver beispielsweise primärer Master-Nameserver für die Zone *mynet.de* sein soll, sieht der entsprechende *named.conf*-Eintrag so aus:

```
zone "mynet.de" {
   type master;
   file "db.mynet.de";
};
```

Der Dateiname *db.mynet.de* ist nicht vorgeschrieben. Sie können jeden beliebigen Namen wählen, aber *db.name-der-zone* ist üblich. Natürlich müssen Sie die entsprechende Datei auch anlegen.

Für jede Zone, für die Ihr Nameserver als primärer Master dient, benötigen Sie zusätzlich eine Reverse-Lookup-Zone. Dies ist eine Zone, die den umgekehrten Dienst leistet: die Umwandlung einer gegebenen IP-Adresse in einen Domain-Namen. Der Name dieser Zone hängt von der Größe des Subnets ab, in dem sie sich befindet: Es handelt sich um den byteweise umgekehrten Netzwerkteil der IP-Adresse mit dem reservierten Domain-Namen *in-addr.arpa*. Angenommen *mynet.de* verwendet das Netz 196.23.17.0/24, dann wird die folgende Reverse-Lookup-Zone konfiguriert:

```
zone "17.23.196.in-addr-arpa" {
   type master;
   file "db.196.23.17";
};
```

Wenn Ihr Nameserver als Slave dienen soll, beispielsweise für die Zone *somenet.de*, sieht die entsprechende `zone`-Anweisung so aus:

```
zone "somenet.de" {
   type slave;
   masters { 138.19.47.3; };
   file "bak.somenet.de";
};
```

Unter `masters` wird eine Liste von Nameservern angegeben, von denen der Slave die Replikationsdaten erhält. Sie brauchen keine primären Master-Nameserver zu sein; der Begriff »masters« bezieht sich auf übergeordnete Nameserver aus der Perspektive des Slaves.

Schließlich benötigen Sie auf jeden Fall noch einen Eintrag für die root-Hints-Datei. Diese spezielle Datei enthält Informationen über die Server, die den Stamm des DNS bilden und auf die Root-Nameserver der verschiedenen TLDs verweisen. Die entsprechende Datei (z. B. *named.root*) ist entweder in Ihrer BIND-Distribution enthalten, oder Sie müssen sie per FTP unter der Adresse *ftp.rs.internic.net/domain/named.root* herunterladen. Der Eintrag in *named.conf* sieht so aus:

```
zone "." {
   type hint;
   file "named.root";
};
```

Bitte beachten Sie, dass die Syntax der Datei *named.conf* ganz exakt eingehalten werden muss – aus eigener Erfahrung weiß ich, dass Programmierer besonders gern das Semikolon hinter der schließenden geschweiften Klammer vergessen.

#### Zonendaten

Falls Ihr Nameserver als primärer Master für eine Zone dient, müssen Sie nun die entsprechende Zonendaten-Datei anlegen. Sie enthält Resource Records für die einzelnen Hosts, Server-Dienste und andere Elemente der Zone. Die erste Zeile einer Zonendaten-Datei ist eine `$TTL`-Anweisung (Time To Live) – sie gibt an, wie lange andere Nameserver die aus dieser Datei enthaltenen Informationen maximal im Cache halten dürfen. Sie können den Wert entweder komplett in Sekunden angeben oder mit den folgenden Maßeinheiten arbeiten: `w` (Wochen), `d` (Tage), `h` (Stunden), `m` (Minuten) oder `s` (Sekunden). Wenn der Wert 28 Stunden betragen soll, gibt es beispielsweise die folgenden drei Möglichkeiten:

```
$TTL 100800
$TTL 28h
$TTL 1d4h
```

Die nächste Information ist ein SOA-Record (Start of Authority). Er enthält die folgenden Konfigurationsinformationen über die Zone selbst:

- **MNAME**
  Der Hostname des primären Master-Nameservers
- **RNAME**
  Die E-Mail-Adresse des Verantwortlichen; das @ wird durch einen Punkt ersetzt.
- **Die Seriennummer der Zone**
  Sie sollte bei jeder Aktualisierung erhöht werden; ein praktisches Format für manuell gepflegte Zonendaten ist JJJJMMTTVV (die letzten beiden Stellen bil-

den eine Versionsnummer, die bei 00 beginnt und die wichtig ist, falls einmal mehrere Änderungen an ein und demselben Tag stattfinden).

- **Der Refresh-Wert**
  Er gibt das Intervall an, in dem die Slave-Nameserver anfragen sollen, ob die Zonendaten aktualisiert wurden.
- **Der Retry-Wert**
  Er bestimmt, wie lange ein Slave warten soll, bevor er nach einem Verbindungsfehler erneut nach Aktualisierungen fragt.
- **Der Expire-Wert**
  Er gibt an, wie lange die Slaves antworten sollen, wenn der primäre Master nicht erreichbar ist. Dieser Wert sollte recht hoch sein, weil er für Ausfallschutz sorgt.
- **Der Negative-Caching-Wert**
  Er gibt an, wie lange die Slaves negative Antworten (»nicht gefunden« usw.) im Cache speichern dürfen. Der Wert sollte recht klein sein, da es sich um einen vorübergehenden Ausfall halten könnte.

Ein vollständiger SOA-Record für *mynet.de* sieht z. B. so aus:

```
mynet.de.     IN  SOA  ns1.mynet.de.  (
              hostmaster.mynet.de.
              2008082501
              30m
              10m
              30d
              30m )
```

Die (runden!) Klammern sind nur dann erforderlich, wenn Daten der Übersicht halber auf mehrere Zeilen verteilt werden.

Wenn Sie einen Host zu einer Zonendaten-Datei hinzufügen möchten, benötigen Sie einen A-Record (Address). Für *pc1.mynet.de* mit der IP-Adresse 196.23.17.24 sieht ein solcher Eintrag so aus:

```
pc1.mynet.de.   IN   A   196.23.17.24
```

Außerdem benötigen Sie einen PTR-Record (Pointer) in der Zonendaten-Datei der Reverse-Lookup-Zone (in diesem Beispiel in `db.196.23.17`). Für *pc1.mynet.de* sieht dieser Eintrag so aus:

```
24.17.23.196.in-addr.arpa.   IN   PTR   pc1.mynet.de.
```

Auf diese Weise müssen Sie alle Rechner in Ihrer Domain, die öffentlich über Hostnamen verfügbar sein sollen, angeben – natürlich auch Ihre eigenen Nameserver, Webserver, Mailserver usw.

Für Webserver-Betreiber ist es oft nützlich, wenn der reine Domain-Name (*mynet.de*) ebenso auf den Webserver verweist wie *www.mynet.de*; viele Benutzer versuchen den Zugriff ohne das Präfix `www`, weil sie das von großen Sites wie *google.com* gewohnt sind. Zu diesem Zweck brauchen Sie nur einen zusätzlichen A-Record einzurichten, der auf die IP-Adresse des Webservers verweist. Beide Möglichkeiten zusammen sehen also so aus:

```
www.mynet.de.    IN   A   196.23.17.3
mynet.de.        IN   A   196.23.17.3
```

Einen CNAME-Record (Canonical Name; ein Alias, der auf einen anderen Hostnamen verweist) dürfen Sie in diesem Zusammenhang nicht verwenden. Nützlich ist der CNAME-Record aber beispielsweise dann, wenn Ihr Webserver intern anders heißt als `www`:

```
www.mynet.de.    IN   CNAME   winnetou.mynet.de
```

Wenn das Ziel des Alias-Records (der Hostname auf der rechten Seite) sich in einer anderen Zone befindet als der Alias-Name, muss dieser Record in der Zone stehen, in die der Alias-Name gehört.

Sie können sogar über die Zonendaten bereits eine triviale Form von Load-Balancing vornehmen. Wenn Sie mehrere Webserver betreiben, können Sie für jeden von ihnen einen A-Record schreiben:

```
www.mynet.de.    IN   A   196.23.17.3
www.mynet.de.    IN   A   196.23.17.4
www.mynet.de.    IN   A   196.23.17.5
```

Die IP-Adressen der Webserver werden dann im sogenannten *Round-Robin-Verfahren* (reihum) ausgegeben. Professionelles Load-Balancing (siehe Kapitel 12, »Skalierung und Performance-Tuning«) benötigt allerdings eine komplexere Konfiguration, die nicht nur DNS betrifft, da eine gerechte Verteilung der Webserver-Ressourcen beispielsweise auch von der Dateigröße der ausgelieferten Dokumente abhängt und nicht nur von der reinen Anzahl der Namensanfragen.

Ein weiterer wichtiger Typ von Resource Records sind die NS-Records (Nameserver). Sie geben sämtliche autoritativen Nameserver für die Zone an – wie bereits erwähnt, sollte mindestens einer dabei sein, der sich nicht innerhalb des eigenen Netzwerks befindet. Für die Domain *mynet.de* könnten die NS-Records also beispielsweise so aussehen:

```
mynet.de.    IN   NS   ns1.mynet.de.
mynet.de.    IN   NS   ns2.mynet.de.
mynet.de.    IN   NS   ns1.provider.de.
```

Mithilfe von NS-Records wird übrigens auch die Delegation untergeordneter Zonen durchgeführt: Für eine Subdomain wird einfach ein anderer Nameserver angegeben. Sofern dieser Nameserver sich innerhalb der Zone befindet, die in der aktuellen Zonendaten-Datei konfiguriert wird, muss zusätzlich ein A-Record für diesen Nameserver angegeben werden:

```
accounting.mynet.de.      IN   NS   ns.accounting.mynet.de.
accounting.mynet.de.      IN   NS   ns2.provider.de.
ns.accounting.mynet.de.   IN   A    196.23.17.9
```

Zu guter Letzt benötigen Sie noch MX-Records (Mail Exchange) für die Angabe von Mailzielen. Schließlich sollen Benutzer E-Mails an Adressen wie *user@mynet.de* schicken können statt an *user@mail.mynet.de*. MX-Records enthalten eine Prioritätsangabe, die festlegt, welcher Mailserver bevorzugt werden soll. Ein Server mit einem höheren Wert wird nur gewählt, wenn derjenige mit dem nächstkleineren offline oder aus anderen Gründen nicht verfügbar ist:

```
mynet.de.    IN   MX    0 mail.mynet.de.
mynet.de.    IN   MX   10 snailmail.mynet.de.
mynet.de.    IN   MX   20 mail.provider.de.
```

Es gibt noch zahlreiche andere Typen von Resource Records. Beispielsweise wurde hier nicht auf die recht komplexe DNS-Konfiguration für IPv6 eingegangen. Wenn Sie selbst für die Verwaltung Ihrer DNS-Zonen verantwortlich sind und öffentliche Nameserver betreiben müssen, kommen Sie nicht umhin, sich entsprechende Literatur zu beschaffen.

## 1.3 TCP/IP-Diagnose und -Fehlersuche

Beim Arbeiten mit TCP/IP-Netzwerken kommt es mitunter zu Ausfällen und sonstigen Problemen. Einige einfache Programme, die zur TCP/IP-Implementierung fast aller Betriebssysteme gehören, ermöglichen die Diagnose solcher Schwierigkeiten. In diesem Abschnitt werden die wichtigsten von ihnen kurz vorgestellt.

### 1.3.1 ping

Das einfachste, aber eines der wichtigsten TCP/IP-Dienstprogramme ist `ping`. Der Name ist kein Akronym, sondern ahmt das Geräusch eines Echolots nach. Genau diese Aufgabe erfüllt `ping` auch: An einen beliebigen Host gesendete Kontrolldatenpakete werden von diesem zurückgesendet. Wenn jedes Paket ordnungsgemäß zurückgesendet wird, ist die Verbindung gewährleistet. Zusätzlich gibt das Programm die Antwortgeschwindigkeit in Millisekunden aus.

Die Syntax von `ping` ist sehr einfach: Sie brauchen nur `ping hostname` beziehungsweise `ping ip-adresse` einzugeben. Das Programm sendet ein Datenpaket nach dem anderen an den Empfänger, bis Sie es mittels Strg + C abbrechen. Anschließend wird eine Statistik mit mittlerer Dauer und dem Prozentsatz verloren gegangener Pakete angezeigt.

Die Windows-Version von `ping` sendet standardmäßig nur vier Pakete. Hier müssen Sie die Option `-t` verwenden, um beliebig viele Datenpakete zu senden. Die `ping`-Ausgabe sieht beispielsweise so aus:

```
$ ping www.heise.de
PING www.heise.de (193.99.144.71): 56 data bytes
64 bytes from 193.99.144.71: icmp_seq=0 ttl=255 time=0.172 ms
64 bytes from 193.99.144.71: icmp_seq=1 ttl=255 time=0.099 ms
64 bytes from 193.99.144.71: icmp_seq=2 ttl=255 time=0.095 ms
64 bytes from 193.99.144.71: icmp_seq=3 ttl=255 time=0.093 ms
64 bytes from 193.99.144.71: icmp_seq=4 ttl=255 time=0.094 ms
64 bytes from 193.99.144.71: icmp_seq=5 ttl=255 time=0.093 ms
64 bytes from 193.99.144.71: icmp_seq=6 ttl=255 time=0.093 ms
64 bytes from 193.99.144.71: icmp_seq=7 ttl=255 time=0.098 ms
-- www.heise.de ping statistics --
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 0.093/0.104/0.172 ms
```

`ping` verwendet übrigens kein TCP oder UDP, sondern ein eigenes, noch leichtgewichtigeres Protokoll namens ICMP (Internet Control Message Protocol). Es wird in RFC 792 beschrieben.

Beachten Sie, dass die Erreichbarkeit eines Hosts per `ping` nichts weiter bedeutet, als dass er prinzipiell mit dem Netzwerk beziehungsweise Internet verbunden ist. Dies sagt nichts über seine Aufgabe im Netzwerk oder über die Funktionsbereitschaft von Server-Diensten aus.

Dennoch ist `ping` ein sehr praktisches Hilfsmittel. Beispielsweise können Sie durch »Anpingen« verschiedener Hosts unterscheiden, welche Netzwerkschnittstelle oder welcher Router unter Umständen ausgefallen ist.

### 1.3.2 traceroute

Das Programm `traceroute` verfolgt automatisch die Route zum angegebenen Host über das Internet. Dazu setzt es den TTL-Wert seiner Testpakete nacheinander auf 1, 2, 3 usw. Ein Router, bei dem das aktuelle Paket den Wert 0 erreicht, verwirft es aber nicht wie üblich automatisch, sondern sendet eine Meldung zurück. `traceroute` sortiert die Antworten nach dem ursprünglichen TTL-Wert

und gibt so nacheinander die Router aus, die bis zum Erreichen des Ziels passiert werden. Das Ganze sieht z. B. so aus:

```
$ traceroute www.heise.de
traceroute to www.heise.de (193.99.144.71), 30 hops max, 40 byte packets
1 erx-maw1.netcologne.de (195.14.247.95) 71.634 ms
2 swrt-maw1-g34.netcologne.de (213.196.239.169) 60.253 ms
3 cat6509-pg1-vl200.netcologne.de (195.14.195.145) 58.283 ms
4 rtint4-ge11.netcologne.de (81.173.192.2) 72.938 ms
5 rtdecix-g01.netcologne.de (81.173.192.85) 56.923 ms
6 de-cix2.ffm.plusline.net (80.81.193.132) 73.637 ms
7 c22.f.de.plusline.net (213.83.57.53) 64.342 ms
8 www.heise.de (193.99.144.71) 70.263 ms
```

`traceroute` ist beispielsweise dann nützlich, wenn `ping` keinen Erfolg gezeigt hat: Sie können erkennen, ob der Fehler in Ihrem eigenen Netzwerk oder »irgendwo da draußen« liegt.

Beachten Sie, dass die Windows-Version des Programms `tracert` heißt.

### 1.3.3 netstat

Das Programm `netstat` liefert zahlreiche Informationen über den Netzwerkzustand. Wenn Sie den Befehl ohne weitere Angaben eingeben, erhalten Sie eine Übersicht über sämtliche aktuellen Netzwerkverbindungen. Sie können diese Liste mithilfe einer Option einschränken, sinnvollerweise auf TCP. Unter Linux heißt die entsprechende Option `--tcp` oder `-t`, unter vielen anderen UNIX-Versionen sowie unter Windows `-p tcp`. Andere Angaben wie UDP sind nicht sonderlich sinnvoll, da es keine »UDP-Verbindungen« gibt – in der Praxis würden die Kanäle angezeigt, über die vor Kurzem UDP-Datagramme versendet wurden. Für TCP sieht die Ausgabe z. B. so aus:

```
$ netstat -t
Active Internet connections (w/o servers)
Proto  Recv-Q  Send-Q  Local Address   Foreign Address State
Tcp    0   0   localhost:1032  localhost:1033  ESTABLISHED
Tcp    0   0   localhost:1033  localhost:1032  ESTABLISHED
Tcp    0   0   localhost:1030  localhost:1034  ESTABLISHED
Tcp    0   0   localhost:1031  localhost:1030  ESTABLISHED
Tcp    0   0   localhost:1028  localhost:1029  ESTABLISHED
Tcp    0   0   localhost:1029  localhost:1028  ESTABLISHED
Tcp    0   0   localhost:1026  localhost:1027  ESTABLISHED
Tcp    0   0   localhost:1027  localhost:1026  ESTABLISHED
Tcp    0   0   localhost:1024  localhost:1025  ESTABLISHED
Tcp    0   0   localhost:1025  localhost:1024  ESTABLISHED
```

Die Option `-r` erfüllt eine andere Option: Sie gibt die aktuellen Routing-Tabellen für alle Schnittstellen aus. Das sieht beispielsweise folgendermaßen aus:

```
$ netstat -r
Aktive Routen:
Netzwerk  Ziel           Netzmaske       Gateway      Schnittst. Metrik
        0.0.0.0         0.0.0.0  213.196.251.226  213.196.251.226     1
      127.0.0.0       255.0.0.0       127.0.0.1       127.0.0.1      1
    192.168.0.0   255.255.255.0     192.168.0.2     192.168.0.2      2
    192.168.0.2 255.255.255.255       127.0.0.1       127.0.0.1      1
  192.168.0.255 255.255.255.255     192.168.0.2     192.168.0.2      1
```

### 1.3.4 nslookup

Das Programm `nslookup` befragt einen Nameserver nach der IP-Adresse zum angegebenen Hostnamen oder umgekehrt. Dabei muss die IP-Adresse des befragten Nameservers angegeben werden, sofern in der TCP/IP-Konfiguration des Betriebssystems keine festen Nameserver angegeben wurden.

Das Ergebnis sieht beispielsweise wie folgt aus, wenn der erste Nameserver von NetCologne (194.8.194.70) nach der IP-Adresse für *www.heise.de* befragt wird:

```
$ nslookup www.heise.de 194.8.194.70
Server:  ns1.netcologne.de
Address:  194.8.194.70

Nicht-autorisierte Antwort:
Name:    www.heise.de
Address:  193.99.144.71
```

Die umgekehrte Art der Abfrage sieht etwa so aus:

```
$ nslookup 193.99.144.71 194.8.194.70
Server:  ns1.netcologne.de
Address:  194.8.194.70

Name:    www.heise.de
Address:  193.99.144.71
```

Das Programm `dig` dient ebenfalls dem Befragen von Nameservern, gibt aber ausführlichere Antworten. Beispielsweise kann die Art von DNS-Resource-Records angegeben werden, nach denen gesucht wird. Mit `dig` lauten die oben gezeigte Suche nach der IP-Adresse (also einem A-Record) für den Host *www.heise.de* und ihre Ausgabe so:

```
# dig @194.8.194.70 a www.heise.de
; <<>> DiG 2.2 <<>> @194.8.194.70 a www.heise.de
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10
;; flags: qr rd ra; Ques: 1, Ans: 1, Auth: 3, Addit: 3
;; QUESTIONS:
;;      www.heise.de, type = A, class = IN

;; ANSWERS:
www.heise.de.   65987   A       193.99.144.85

;; AUTHORITY RECORDS:
heise.de.       65987   NS      ns2.pop-hannover.net.
heise.de.       65987   NS      ns.heise.de.
heise.de.       65987   NS      ns.pop-hannover.de.

;; ADDITIONAL RECORDS:
ns2.pop-hannover.net.   65987   A       62.48.67.66
ns.heise.de.            4714    A       193.99.145.37
ns.pop-hannover.de.     9785    A       193.98.1.200

;; Total query time: 321 msec
;; FROM: prefect to SERVER: 194.8.194.70
;; WHEN: Sat Aug 17 16:16:05 2008
;; MSG SIZE  sent: 30  rcvd: 175
```

Übrigens wurde `dig` auch für Windows portiert. Unter bind8nt.meiway.com/resources.cfm (der empfehlenswerten Website »BIND on Windows NT«) finden Sie den entsprechenden Download-Link.

### 1.3.5 telnet

Das Programm `telnet` beherrscht nicht etwa nur das gleichnamige Protokoll zur Terminalemulation, sondern ermöglicht auch die Kommunikation mit sämtlichen textbasierten Server-Diensten. Dabei können Sie die Eingaben, die normalerweise von einer Client-Software vorgenommen werden, manuell vornehmen. Auf diese Weise erhalten Sie einen direkten Einblick in Vorgänge, die normalerweise hinter den Kulissen stattfinden, und können so mögliche Kommunikationsfehler aufspüren.

Das Folgende ist beispielsweise der Mitschnitt einer Telnet-Session mit der Site *httpd.apache.org*, also der Website des Apache-Webserver-Projekts selbst. Die

manuellen Eingaben in `telnet` (die Zeilen von `GET / HTTP/1.1` bis `Connection: close` sowie eine zusätzliche Leerzeile) entsprechen dem wichtigsten Teil der Daten, die normalerweise Ihr Browser an den Webserver übermittelt. Die Ausgabe enthält nicht nur den HTML-Code, den ein Browser umsetzen würde (und der hier auf die ersten drei Zeilen beschränkt wurde), sondern davor zunächst die HTTP-Header. Die Bedeutung aller hier verwendeten Anfrage- und Antwort-Header wird in Kapitel 2, »Funktionsweise von Webservern«, genau erläutert.

```
$ telnet httpd.apache.org 80
Trying 209.237.227.195...
Connected to httpd.apache.org.
Escape character is '^]'.
GET / HTTP/1.1
Host: httpd.apache.org
Connection: close

HTTP/1.1 200 OK
Date: Sat, 17 Aug 2008 17:39:23 GMT
Server: Apache/2.0.53 (Unix)
Last-Modified: Sun, 11 Aug 2008 21:06:21 GMT
ETag: "eeba54-22f3-1ce1ed40"
Accept-Ranges: bytes
Content-Length: 8947
Connection: close
Content-Type: text/html; charset=ISO-8859-1

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
               "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html>
 <head>
[...]
Connection closed by foreign host.
```

Statt der Portnummer 80 hätten Sie auf den meisten Systemen auch einfach `http` schreiben können – die wichtigsten Well-Known-Ports haben Namen, die in der Datei */etc/services* (Windows: *%Systemroot%\System32\Drivers\ etc\services*) festgelegt sind.

Für die Terminalemulation, für die `telnet` ursprünglich entwickelt wurde, sollten Sie das Programm übrigens nach Möglichkeit nicht einsetzen. Das Problem ist nämlich, dass alle Daten – einschließlich Passwörtern – im Klartext über das Netzwerk beziehungsweise Internet übertragen werden. Eine vernünftige Alternative ist SSH (Secure Shell). Dieses Protokoll verwendet eine recht sichere Verschlüsse-

lung; allerdings können Sie nicht das Programm `telnet` dafür verwenden. Die meisten UNIX-Varianten enthalten den Kommandozeilen-Client `ssh`; für Windows ist beispielsweise PuTTY geeignet (kostenloser Download etwa unter *www.chiark.greenend.org.uk/~sgtatham/putty*).

## 1.4 Zusammenfassung

In diesem Kapitel haben Sie einen Einblick in die jahrzehntealte Technik erhalten, die der Funktionsweise des Apache-Webservers zugrunde liegt. Die Entwickler dieser Technik haben offensichtlich gute Arbeit geleistet: Als die Internetprotokolle entstanden, gab es kaum mehr als eine dreistellige Anzahl von Hosts; erstaunlicherweise funktioniert dieselbe Kerntechnologie noch heute, obwohl inzwischen mehrere Millionen Rechner am Internet teilnehmen.

Die eigentliche Basis der Verbindungen über beliebig viele Netzwerke hinweg stellt das Internet Protocol (IP) zur Verfügung: Es weist jedem Rechner eine eindeutige Adresse zu, die sowohl die Zugehörigkeit zu einem bestimmten Netzbetreiber als auch den einzelnen Host kennzeichnet. Der allmählich eintretenden Adressknappheit des IPv4 mit 32 Bit langen Adressen kann man bislang noch mit einigen intelligenten Tricks entgegenwirken; in Zukunft wird IPv6 mit einer Adresslänge von 128 Bit das Problem gründlicher lösen.

Für den Transport der Daten zwischen den verschiedenen Anwendungen sind die beiden Protokolle TCP und UDP zuständig; Ersteres stellt eine zuverlässige Punkt-zu-Punkt-Verbindung bereit, während Letzteres für die schnelle Zustellung von Einzelnachrichten zuständig ist. Je nach Anwendung sind die beiden Alternativen unterschiedlich gut geeignet. Für die Kommunikation zwischen Webserver und Browser wird das Anwendungsprotokoll HTTP verwendet, das TCP einsetzt.

Benutzerfreundliche Netzwerknamen wie *buecher.lingoworld.de* werden vom Domain Name System (DNS) bereitgestellt, dem weltweit verteilten Netz der Nameserver. Wenn Sie Ihren Webserver am eigenen Standort betreiben möchten, müssen Sie gegebenenfalls eigene Nameserver einsetzen; die beliebteste Software dafür – BIND – wurde hier bereits kurz vorgestellt.

Zu guter Letzt haben Sie noch einige häufig genutzte Standard-Tools zur TCP/IP-Diagnose und Fehlersuche kennengelernt: den Erreichbarkeitstest mit `ping`, die Routenverfolgung durch `traceroute`, Statusinformationen per `netstat`, Nameserver-Abfragen mit `nslookup` oder `dig` sowie direkte Protokollkommunikation über `telnet`. Alle diese Hilfsprogramme werden Sie sehr häufig zur Administration einsetzen.

*»Je besser der Zustand eines Dings ist,*
*desto besser ist seine Funktion.«*
*– Aristoteles*

# 2 Funktionsweise von Webservern

Bevor ab dem nächsten Kapitel konkret der Apache behandelt wird, finden Sie in diesem Kapitel einen allgemeinen Überblick über die Arbeitsweise des HTTP und von Webservern. Um speziell Programmierern das Verständnis zu erleichtern, wird ein eigener kleiner Webserver in Perl implementiert, der einige wichtige Eigenschaften eines »großen« HTTP-Servers besitzt und so dessen Funktionen besser verständlich macht.

## 2.1 Das HTTP

Das TCP/IP-Anwendungsprotokoll HTTP (Hypertext Transfer Protocol) bildet die Basis der Kommunikation zwischen Webserver und Browser. Es handelt sich um ein klartextbasiertes Protokoll mit wenigen Client-Befehlen und wohldefinierten Server-Antworten, das nach dem Vorbild von Protokollen wie dem viel älteren FTP (File Transfer Protocol) gestaltet wurde. Die aktuelle Version ist HTTP/1.1; ihre Spezifikation steht in RFC 2616.

HTTP ist ein typisches Request/Response-Verfahren (Anfrage und Antwort). Der Ablauf einer solchen HTTP-Anfrage und -Antwort ist immer derselbe:

1. Der Browser ermittelt aus der URL den Hostnamen des angesprochenen Servers und stellt eine TCP-Verbindung zu diesem her. Standardmäßig wird der TCP-Server-Port 80 gewählt, ansonsten die Portnummer, die in der URL steht.

   Angenommen, die angeforderte URL ist *http://www.mynet.de/seiten/info.html*. Dann ist der Server `www.mynet.de`. Der Browser ermittelt über DNS die IP-Adresse dieses Hosts und stellt eine Verbindung zu dessen Port 80 her.
2. Der Browser übermittelt eine Anfrage an den Server. Die erste Zeile enthält den HTTP-Befehl, den Pfad des angesprochenen Dokuments und die Protokollversion. In weiteren Zeilen folgen optionale Header-Felder nach dem Schema `Feldname: Wert`, in denen der Browser beispielsweise seine eigene

Versionsangabe, den Hostnamen des anfordernden Rechners oder die URL des Dokuments, in dem sich der angeklickte Hyperlink befand (den sogenannten *Referrer*), mitteilt. Bestimmte HTTP-Anfragen enthalten nach einer Leerzeile auch noch einen Body, in dem zusätzliche Daten geschickt werden.

Der einfachste HTTP-Befehl (und der einzige, der in der ursprünglichen Protokollversion HTTP/0.9 enthalten war!) ist `GET`. Er bedeutet ganz einfach, dass ein bestimmtes Dokument geliefert werden soll.

Gemäß dem Beispiel aus Punkt 1 wird hier der folgende `GET`-Befehl betrachtet: `GET /seiten/info.html HTTP/1.1`. Die Client-Header-Optionen werden weiter unten genauer erläutert.

3. Der Server nimmt die `GET`-Anfrage entgegen und muss je nach Dateityp des angeforderten Dokuments unterschiedlich verfahren: Wenn es sich um ein statisches Dokument handelt (was für dieses Beispiel der Einfachheit halber angenommen werden soll), kann er es einfach ausliefern. Handelt es sich dagegen um die Adresse eines serverseitigen Skripts, muss er zunächst das zuständige Programm aufrufen, dessen Ausgabe entgegennehmen und diese genau wie ein statisches Dokument an den Client liefern.

   Bevor der Server das Dokument sendet, schickt er eine Statusmeldung und HTTP-Header mit diversen Angaben, gefolgt von einer Leerzeile. Die Statusmeldung in der ersten Zeile der ausgegebenen Daten besteht aus der HTTP-Version, einer standardisierten Statuscodenummer und deren Textauflösung. Die beiden häufigsten Statusmeldungen dürften `200 OK` (Dokument ist vorhanden und wird mitgeliefert) und `404 Not Found` (Dokument existiert nicht) sein. In diesem Beispiel soll gelten, dass das Dokument gefunden wurde. Die erste Zeile der Antwort lautet also:

   ```
   HTTP/1.1 200 OK
   ```

   In der nächsten Zeile beginnen die HTTP-Header. Die Mindestanforderung für den Header einer HTTP-Antwort ist die Angabe des MIME-Types des gelieferten Dokuments. Im Fall eines HTML-Dokuments sieht die entsprechende Header-Zeile so aus:

   ```
   Content-Type: text/html
   ```

   Weitere häufig verwendete Header-Felder enthalten beispielsweise das Änderungs- und das Verfallsdatum des Dokuments, die Versionsangabe des Servers oder Cookies. Sie werden weiter unten ausführlich beschrieben.

   Der Body der HTTP-Antwort, im vorliegenden Beispiel also das eigentliche Dokument, muss durch eine Leerzeile vom Header getrennt werden.

Der Client nimmt die HTTP-Antwort des Servers entgegen. Er interpretiert zunächst die empfangenen Header-Daten und anschließend das eigentliche Dokument. Bei einem HTML-Dokument hat dies meist zur Folge, dass er zahlreiche weitere HTTP-Anfragen stellen muss, um die in das Dokument eingebetteten Bilder und Multimedia-Dateien anzufordern.

Formal sind sowohl die HTTP-Anfrage als auch die Server-Antwort RFC-822-kompatible Nachrichten. Dieses Format (aktuelle Version: RFC 2822) wurde ursprünglich für E-Mails entwickelt. Es besteht aus beliebig vielen Header-Zeilen mit dem Format `Headername: Wert`; der Body (Nachrichteninhalt) wird durch eine Leerzeile von den Headern getrennt.

Das RFC-822-Format wurde später durch MIME erweitert, um E-Mails in anderen Formaten als reinen Text und mit Dateianhängen versenden zu können. Eine MIME-Nachricht benötigt das zusätzliche Header-Feld `Content-Type`, in dem der Datentyp des Body-Inhalts angegeben wird. Auch für HTTP spielt MIME eine wichtige Rolle.

### 2.1.1 Die HTTP-Client-Anfrage

Neben dem bereits erwähnten `GET` gibt es noch einige andere HTTP-Befehle oder Client-Methoden. Die meisten von ihnen wurden mit HTTP/1.0 eingeführt; einige wenige dagegen erst mit HTTP/1.1. In Tabelle 2.1 sehen Sie zunächst eine Übersicht über alle HTTP-Methoden. Anschließend werden die meisten von ihnen genauer erläutert und durch Beispiele verdeutlicht.

| HTTP-Methode | Seit Version | Bedeutung |
|---|---|---|
| `GET` | 0.9 | Dokument anfordern |
| `HEAD` | 1.0 | nur Header anfordern |
| `POST` | 1.0 | umfangreiche Formulardaten senden |
| `PUT` | 1.0 | Dokument auf dem Server ablegen |
| `DELETE` | 1.0 | Dokument vom Server löschen |
| `LINK`[1] | 1.0 | Verknüpfung erstellen |
| `UNLINK` | 1.0 | Verknüpfung löschen |
| `TRACE` | 1.1 | Proxys im Header anzeigen |
| `CONNECT` | 1.1 | Proxy-Zugriff auf gesicherte Server |
| `OPTIONS` | 1.1 | Liste von Optionen anfordern |

**Tabelle 2.1** Übersicht über die HTTP-Methoden

1 LINK und UNLINK wurden kaum je implementiert. Selbst RFC 2616 schweigt sich – bis auf je einen lapidaren Satz – über ihren Sinn aus.

Die »Vorabversion« HTTP/0.9 (die diese Versionsnummer übrigens erst nachträglich, das heißt bei der Einführung von Version 1.0, erhielt) unterstützte einzig und allein `GET`-Anfragen ohne zusätzliche Header. Die wichtigste Neuerung von HTTP/1.1 gegenüber der Version 1.0 waren nicht die drei zusätzlichen Anfragemethoden (die im Übrigen gar nicht so häufig verwendet werden). Viel bedeutender ist die neu geschaffene Möglichkeit, die HTTP-Verbindung zu einem bestimmten Server nach Abschluss einer Dokumentanforderung offen (persistent) zu halten. Diese sogenannte *Keepalive-Option* ist beispielsweise nützlich, wenn ein HTML-Dokument zahlreiche Bilder enthält, die vom selben Server stammen und ebenfalls per HTTP-Anfrage angefordert werden müssen. Wenn jeweils die Zeit für den Verbindungsaufbau wegfällt, wird die Seite schneller geladen.

Das Syntaxschema jeder HTTP-Anfrage sieht folgendermaßen aus:

```
METHODE URL HTTP/Version
[Headername: Wert
 ...]

[Body]
```

Die eckigen Klammern bedeuten – wie in Syntaxschemata üblich – dass die jeweiligen Teile optional sind. Falls die Methode einen Body besitzt, muss er gemäß RFC 2822 auf jeden Fall durch eine Leerzeile vom Header getrennt werden. Erforderlich ist er auf jeden Fall bei `POST` und `PUT`.

### Die Methode GET

Die wichtigste HTTP-Methode überhaupt ist `GET`. Sie dient dazu, den Webserver zur Auslieferung eines Dokuments oder der Ausgabe eines serverseitigen Programms aufzufordern. Eine `GET`-Anfrage besteht nur aus dem Befehl selbst und einigen (größtenteils optionalen) Header-Feldern, sie besitzt aber keinen Body. Wenn ein Browser beispielsweise das Dokument *info.html* im Webserver-Verzeichnis *seiten* anfordern möchte, sieht die Anfrage etwa so aus:

```
GET /seiten/info.html HTTP/1.1
Accept: */*
Accept-Language: de, en-US
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.01)
Host: www.mynet.de
Connection: Keepalive
```

Die verschiedenen Header und ihre möglichen Werte werden weiter unten genauer erläutert. Kurz gesagt, enthalten die Header-Felder der Anfrage in diesem Beispiel folgende Informationen:

- `Accept: */*`
  Der Browser akzeptiert Dokumente mit beliebigem MIME-Type (`*/*` bedeutet: beliebiger Typ/beliebiger Untertyp).
- `Accept-Language: de, en-US`
  Der Browser bevorzugt Dokumente in den Sprachen Deutsch und US-Englisch (in dieser Reihenfolge).
- `Accept-Encoding: gzip, deflate`
  Neben unkomprimierten Daten unterstützt der Browser auch Dateien, die mit GZIP beziehungsweise ZIP (`deflate`) komprimiert sind.
- `User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.01)`
  Der Browser übergibt seine eigene Versionsangabe. Dies dient dem Erstellen der bekannten Browser-Statistiken und ermöglicht es Webentwicklern, an verschiedene Browser angepasste Versionen von Dokumenten anzubieten. In diesem Beispiel handelt es sich um den Internet Explorer 6.0 unter Windows XP.
- `Host: www.mynet.de`
  Diese Angabe ist zur Unterscheidung virtueller Hosts wichtig, die sich dieselbe IP-Adresse teilen. Da die IP-Datagramme nur die IP-Adresse, aber nicht den Hostnamen enthalten, weiß der Server nur durch dieses Header-Feld, welche konkrete Website gemeint ist.
- `Connection: Keepalive`
  Die HTTP-Verbindung soll nach der Beantwortung der Anfrage offen gehalten werden (nur HTTP/1.1). Der Server kann allerdings nicht dazu gezwungen werden, die Verbindung beizubehalten; es handelt sich nur um eine Art Empfehlung.

Wenn die Anfrage erfolgreich beantwortet wird, könnte die Antwort des Servers beispielsweise so aussehen:

```
HTTP/1.1 200 OK
Date: Sun, 14 Aug 2011 11:29:26 GMT
Server: Apache/2.2.15 (Unix)
Last-Modified: Fri, 12 Aug 2011 01:45:53 GMT
ETag: "2c986a-b20-fc93242e"
Accept-Ranges: bytes
Content-Length: 1936
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//Transitional//EN">
<html>
```

```
<head>
[...]
```

Die Header bedeuten Folgendes:

- `Date: Sun, 14 Aug 2011 11:29:26 GMT`
  Dieses Feld gibt Datum und Uhrzeit auf dem Server inklusive Zeitzone an.
- `Server: Apache/2.2.15 (Unix)`
  Dies ist die Versionsangabe des Servers. Sie erfahren in Kapitel 6, »Grundkonfiguration«, wie sich der ausgegebene Wert anpassen lässt.
- `Last-Modified: Fri, 12 Aug 2011 01:45:53 GMT`
  Datum und Uhrzeit der letzten Änderung. Diese Angabe ermöglicht es dem Browser beispielsweise, zu entscheiden, ob die Kopie der Seite in seinem Cache noch aktuell ist.
- `ETag: "2c986a-b20-fc93242e"`
  Das ETag (Entity-Tag) wird vom Server erzeugt, wenn er entsprechend konfiguriert ist. Es bestätigt die Identität eines Dokuments über die URL beziehungsweise den Pfad hinaus. Dies ist nützlich für Caching- und Proxy-Zwecke.
- `Accept-Ranges: bytes`
  Durch diesen Header gibt der Server an, dass er Bereichsanfragen akzeptiert: Der Client kann einen `Range`-Header übermitteln, der angibt, dass er nur einen bestimmten Byte-Bereich des Bodys anfordern möchte.
- `Content-Length: 1936`
  Dies ist die Länge des Bodys in Bytes. Der Browser kann anhand dieser Angabe überprüfen, ob die Lieferung vollständig ist.
- `Connection: close`
  Der Server schließt die Verbindung nach Abschluss dieser Anfrage.
- `Content-Type: text/html`
  Der MIME-Type des Dokuments ist HTML: Haupttyp Text, Untertyp HTML.

Der Browser kann über eine `GET`-Anfrage Formulardaten übermitteln, das heißt, die Eingabe in ein HTML-Formular wird zusammen mit der Anfrage gesendet. In der Formulardefinition im HTML-Dokument kann zu diesem Zweck festgelegt werden, ob das Formular mittels `GET` oder `POST` übertragen wird. Bei der Formulardatenübertragung mit `GET` werden die Formulardaten hinter einem `?` an die angeforderte URL angehängt.

Wenn ein HTML-Formular seine Daten per `GET` versenden soll, benötigt das zuständige `<form>`-Tag ein Attribut namens `method` mit dem Wert `"GET"`. Zum Beispiel wird das folgende Formular mittels `GET` an das PHP-Skript `test.php` versendet:

```
<form action="test.php" method="GET">
...
</form>
```

Die Befehlszeile der HTTP-Anfrage, die beim Absenden dieses Formulars generiert wird, sieht so aus:

```
GET /test.php?Formulardaten HTTP/1.1
```

Der Anhang hinter dem Fragezeichen wird als *Query-String* bezeichnet. Die einzelnen Formularfelder haben die Form `Name=Wert` und werden durch `&`-Zeichen oder Semikola (`;`) voneinander getrennt. Zusätzlich müssen die eigentlichen Formulardaten URL-codiert werden, weil zahlreiche Zeichen in URLs nicht zulässig sind: Leerzeichen werden durch `+` ersetzt; fast alle Zeichen, die keine Buchstaben oder Ziffern sind, werden durch ein `%`-Zeichen und ihren hexadezimalen Zeichencode ersetzt. Betrachten Sie beispielsweise folgende Formularfelder:

```
user=Peter Müller
age=35
place=Köln
```

Angenommen, das Formular wird an das PHP-Skript `guest.php` im Server-Verzeichnis `gb` versendet. Dann sieht die eigentliche Befehlszeile der entsprechenden GET-Anfrage so aus:

```
GET /gb/guest.php?user=Peter+M%FCller&age=35&place=K%F6ln
HTTP/1.1
```

#### Die Methode HEAD

Die Syntax der HTTP-Methode `HEAD` entspricht derjenigen von `GET`. Auch die Antwort des Servers ist identisch, mit der Besonderheit, dass nur die Header-Daten gesendet werden, aber kein Body. `HEAD` hat also für einen Browser vor allem den Nutzen, zu überprüfen, ob die bereits in seinem Cache befindliche Kopie eines Dokuments noch aktuell ist. Dies kann er anhand des Erstellungsdatums und eventuell mithilfe des `ETag`-Headers der entsprechenden Datei herausfinden.

Eine weitere Einsatzmöglichkeit für die `HEAD`-Methode ist die Ermittlung der Dateigrößen von Dokumenten – beispielsweise Bildern – zur Berechnung der Ladezeit. Auch für Programmierer ist die Methode geeignet; sie ermöglicht das gezielte Debugging von Webservern, Web-Clients und Webanwendungen.

Ein separates `HEAD`-Beispiel ist an dieser Stelle nicht erforderlich; es entspricht dem bereits gezeigten Beispiel für `GET`, wenn Sie sich den Body der Server-Antwort wegdenken.

### Die Methode POST

`POST` ist die bevorzugte Methode zur Übermittlung von Formulardaten an den Server. Anders als bei `GET` werden die Daten nicht an die URL des angesprochenen Skripts angehängt, sondern im Body der HTTP-Anfrage übertragen. Dies hat drei wichtige Vorteile:

- Die Datenmenge kann beliebig groß sein. Bei `GET` ist sie dagegen auf 2.000 Zeichen beschränkt, da dies die maximale Länge für URLs ist.
- Die Formulardaten müssen nicht zwangsläufig URL-codiert werden wie bei `GET`. Stattdessen kann wie bei einer HTTP-Antwort ein `Content-Type`-Header angegeben werden – dies erledigt der Browser automatisch, wenn die HTML-Formulardefinition ein Attribut namens `enctype` enthält, das den entsprechenden MIME-Type angibt, z. B.:

  ```
  <form action="test.php" method="POST" enctype="application/x-www-
  form-urlencoded">
  ...
  </form>
  ```

  Diese Angabe sorgt für die standardmäßige URL-Codierung. Eine interessantere Variante ist der MIME-Type `multipart/form-data`. Dabei handelt es sich um eine MIME-Multipart-Nachricht, die einer E-Mail mit Attachments (`multipart/mixed`) ähnelt. Formulardaten in diesem Format können aus mehreren Teilen mit jeweils eigenem `Content-Type`-Header bestehen. Dies ermöglicht vor allem Datei-Uploads per HTML-Formular.
- Wenn ein serverseitiges Skript per `POST`-Anfrage angefordert wird, ist garantiert, dass es bei jedem Aufruf neu ausgeführt wird. Am separaten Versand der Daten im Body der HTTP-Anfrage bemerken der Browser und die eventuell beteiligten Proxy-Server, dass diese Daten auf jeden Fall an das eigentliche serverseitige Programm ausgeliefert werden müssen.

  `GET`-Anfragen hängen die Formulardaten dagegen wie erwähnt an die URL an, sodass es sich formal gar nicht um Datenversand handelt. Aus diesem Grund werden sie (genau wie `HEAD` und die weiter unten besprochene Methode `PUT`) als *idempotent* bezeichnet. Dieser aus der Mathematik stammende Begriff besagt in diesem Zusammenhang, dass zwei aufeinanderfolgende, identische Anfragen auch exakt dasselbe bewirken. Deshalb kann ein Browser oder Proxy-Server die Ausgabe des Servers in seinem Cache zwischenspeichern und ausliefern.

  Bei serverseitigen Anwendungen ist es aber oft wichtig, dass sie auch wirklich bei jedem Aufruf ausgeführt werden, weil sie nicht nur eine Ausgabe für den Browser des Benutzers generieren, sondern auch etwas hinter den Kulissen

erledigen – z. B. Daten in einer Datenbank ergänzen. Aus diesem Grund werden bei einer GET-Anfrage, die ein serverseitiges Skript aufruft (was insbesondere notwendig ist, wenn die Anforderung nicht per Formularversand, sondern über einen normalen Hyperlink zustande kommt), oft eine durch Programmierung erzeugte Zufallszahl oder ein Datum und die Uhrzeit als Pseudoparameter angefügt: Da sich auf diese Weise unterschiedliche URLs wie `script.php?dummy=52784` oder `script.php?dummy=2011_08_13_15_45` ergeben, kommt kein Browser oder Proxy-Server auf die Idee, einen weiteren Aufruf des Skripts für ein bereits im Cache befindliches Dokument zu halten.

Nehmen Sie als Beispiel an, die drei folgenden Formularfelder sollen mittels `POST` an das serverseitige Skript `bestell.php` im Verzeichnis `pizza` versendet werden:

```
groesse=m
belag=funghi
extra=Bitte schneiden!
```

Das zugehörige HTML-Formular ist folgendermaßen definiert:

```
<form action="bestell.php" method="POST" enctype="application/x-www-
form-urlencoded">
...
</form>
```

Beim Absenden des Formulars wird aus diesen Angaben die folgende `POST`-Anfrage erzeugt (das `enctype`-Attribut sorgt auch hier für eine URL-Codierung der Formulardaten):

```
POST /pizza/bestell.php HTTP/1.1
Accept: */*
Accept-Language: de, en-US
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.01)
Referer: http://www.mynet.de/pizza/bestell.html
Host: www.mynet.de
Content-Length: 47
Content-Type: application/x-www-form-urlencoded

groesse=m&belag=funghi&extra=Bitte+schneiden%21
```

Die meisten der diversen Header-Felder wurden bereits erläutert. Bemerkenswert ist, dass diese Anfrage `Content-Type` und `Content-Length` enthält, die bei Anfragen wie `GET` oder `HEAD` nur in der Antwort vorkommen. Sie beschreiben wie bei einer HTTP-Antwort den (in diesem Fall aus der `enctype`-Angabe im `<form>`-Tag stammenden) Datentyp und die Länge des Body-Inhalts. Der Header `Referer`

gibt an, von welchem Dokument die Anfrage ausging, das heißt auf welcher Seite der Hyperlink angeklickt oder das Formular abgeschickt wurde.

Wenn der Server eine `POST`-Anfrage erhält, reicht er die erhaltenen Formulardaten an das in der URL angegebene Skript weiter. Der Server antwortet im Erfolgsfall wie bei einer `GET`-Anfrage mit einer HTTP-Antwort mit dem Statuscode 200 und einem Body, der ein Dokument enthält, das der Browser daraufhin anzeigt, z. B.:

```
HTTP/1.1 200 OK
Date: Tue, 16 Aug 2011 19:09:34 GMT
Server: Apache/2.2.15 (Unix)
Last-Modified: Tue, 16 Aug 2011 19:09:34 GMT
ETag: "394bcd-2c2-6d94b63a"
Accept-Ranges: bytes
Connection: close
Content-Length: 2782
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//Transitional//EN">
<html>
<head>
[...]
```

#### Die Methode PUT

`PUT` dient dazu, den mit der Anfrage übertragenen Body-Inhalt unter dem angegebenen Dateinamen auf dem Server zu speichern. Selbstverständlich lässt ein standardmäßig konfigurierter Webserver dies höchstens dann zu, wenn eine entsprechende Authentifizierung stattfindet. Näheres dazu finden Sie in Kapitel 9, »Authentifizierung und Zugriffskontrolle«.

Es ist bisher nicht allgemein üblich, Webseiten per HTTP-`PUT`-Anfrage auf den externen Webserver beim Hoster zu übertragen. In aller Regel wird die Übertragung per FTP durchgeführt, sowohl bei manueller Übertragung mithilfe eines separaten FTP-Clients (z. B. FileZilla) als auch unter Einsatz der Veröffentlichungsfunktion eines Website-Editors wie Adobe Dreamweaver. Die Verwendung von HTTP hat jedoch einige wichtige Vorteile:

- Wenn HTTP verwendet wird, braucht der Hoster keinen separaten FTP-Server für den Webspace seiner Kunden zu betreiben – dies bedeutet ein geringeres Sicherheitsrisiko.
- HTTP »kennt« die unterschiedlichen Dateitypen von Daten, die sich auf Webservern befinden können. Diese Information wird durch den MIME-Header

`Content-Type` übermittelt. Der Server kann die empfangenen Daten je nach Dateityp passend verarbeiten; z. B. könnte er die Zeilenumbrüche von Textdateien in das für seine Plattform passende Format umwandeln. Dies eröffnet erheblich flexiblere Möglichkeiten als bei der FTP-Übertragung, bei der die Zeilenumbrüche im ASCII-Modus stur und im Binärmodus gar nicht umgewandelt werden.

- Auf der HTTP-Übertragung basiert das WebDAV-Protokoll (Web Distributed Authoring and Versioning). Es ermöglicht die sichere Zusammenarbeit mehrerer im Netz verteilter Autoren an einer Website sowie eine CVS-ähnliche Pflege unterschiedlicher Versionen. Über das Modul `mod_dav` (siehe Kapitel 17, »Apache erweitern«) wird Apache zum WebDAV-Server; Programme wie Dreamweaver enthalten bereits die entsprechenden Client-Funktionen.

Die folgende `PUT`-Anfrage fordert den Server auf, den Body unter */info/neu.html* zu speichern:

```
PUT /info/neu.html HTTP/1.1
Host: www.mynet.de
Connection: close
User-Agent: MyWebPublisher/0.1
Accept: */*
Content-Type: text/html
Content-Length: 2982

<!DOCTYPE HTML PUBLIC "-//W3C/DTD HTML 4.0//Transitional//EN">
<html>
<head>
[...]
```

Die verwendeten Header wurden bereits erläutert. Der Server speichert den Inhalt des Anfrage-Bodys unter dem Pfad */info/neu.html* – falls die Datei bereits existiert, überschreibt er sie; ansonsten wird sie neu angelegt. Im Erfolgsfall gibt der Server eine Antwort wie diese zurück:

```
HTTP/1.1 201 Created
Date: Thu, 18 Aug 2011 13:10:34 GMT
Server: Apache/2.2.15 (Unix)
Content-type: text/plain
Content-Length: 41

/info/neu.html was successfully created.
```

Wenn ein Client eine `PUT`-Anfrage ohne vorherige Anmeldung sendet, erhält er in der Regel zunächst einmal eine Antwort mit folgender Kopfzeile:

```
HTTP/1.1 401 Unauthorized
```

Die meisten Clients fragen den Benutzer daraufhin nach Benutzername und Passwort. Angenommen, der Name ist `webuser` und das Passwort `TopSecret`. Dann senden Sie die Anfrage mit einer Header-Zeile wie der folgenden neu:

```
Authorization: BASIC d2VidXNlcjpUb3BTZWNyZXQ=
```

Hier handelt es sich um eine einfache Benutzername-Passwort-Kombination (Authentifizierungstyp `BASIC`); die Daten werden `base64`-codiert übertragen. Für den Rest der Sitzung merkt sich der Client normalerweise diese Information und sendet sie bei jeder weiteren Anfrage automatisch mit. Dies ist übrigens bei `GET`-Zugriffen auf die Inhalte geschützter Verzeichnisse nicht anders.

### Die Methode DELETE

`DELETE` fordert den Server auf, die Datei mit dem angegebenen Pfad zu löschen. Diese Methode ist neben `PUT` wichtig für Webautoren: Dokumente, die auf dem Server gespeichert wurden, müssen sich natürlich auch wieder entfernen lassen, wenn sie nicht mehr benötigt werden. `DELETE`-Anfragen benötigen lediglich den in der Anweisungszeile selbst angegebenen Pfad und einen `Host`-Header. Eine `DELETE`-Anfrage sieht also beispielsweise folgendermaßen aus:

```
DELETE /seiten/unwichtig.html HTTP/1.1
Host: www.mynet.de
```

Wenn das Löschen erfolgreich durchgeführt wurde, antwortet der Server mit dem Statuscode 200 und einer entsprechenden Erfolgsmeldung, z. B.:

```
HTTP/1.1 200 OK
Date: Thu, 18 Aug 2011 14:17:21 GMT
Server: Apache 2.2.15 (Unix)
Content-Type: text/plain
Content-Length: 49

/seiten/unwichtig.html was successfully deleted.
```

Beachten Sie, dass für `DELETE` dieselben Ausführungen über die Authentifizierung gelten wie für `PUT`-Anfragen.

### Die Methode TRACE

Mithilfe der Methode `TRACE` kann ein Client verfolgen, über welche (und wie viele) Proxy-Server eine HTTP-Anfrage an einen bestimmten Server gesendet wird. Sie wird in der Regel von Programmierern und Netzwerkadministratoren verwendet, um beispielsweise herauszufinden, warum bestimmte HTTP-Anfragen scheitern.

Der Client – hier in der Regel kein Browser, sondern ein spezielles Tool oder einfach `telnet` – sendet eine `TRACE`-Anfrage an das gewünschte Ziel. Sie enthält keinen Dateipfad, sondern eine vollständige URL. Ein `Max-Forwards`-Header bestimmt, wie viele Proxy-Weiterleitungen maximal stattfinden sollen. Jeder Proxy-Server auf dem Weg zum Empfänger vermindert diesen Wert um 1 und hängt seine Adresse und seine HTTP-Version durch ein Komma getrennt an einen `Via`-Header an. Der Empfänger (oder der Proxy, bei dem `Max-Forwards` den Wert 0 erreicht) macht die entsprechend modifizierte `TRACE`-Anfrage zum Body der Antwort. Eine `TRACE`-Anfrage sieht beispielsweise so aus:

```
TRACE http://www.mynet.de HTTP/1.1
User-Agent: SomeScript/0.1
Host: www.mynet.de
Accept: */*
Max-Forwards: 3
```

Angenommen, die Anfrage wird über zwei Proxys zum Ziel geleitet. Dann lautet die letztendliche Antwort des Servers folgendermaßen:

```
HTTP/1.1 200 OK
Date: Thu, 18 Aug 2011 21:01 GMT
Server: Apache 2.2.15 (Unix)
Content-Type: message/http
Content-Length: 157

TRACE http://www.mynet.de HTTP/1.1
User-Agent: SomeScript/0.1
Host: www.mynet.de
Accept: */*
Max-Forwards: 1
Via: 1.1 proxy.provider.net, 1.1 proxy.mynet.de
```

Eine wichtige Funktion von `TRACE`-Anfragen besteht darin, zu überprüfen, welche HTTP-Version die diversen Proxys für eine bestimmte Verbindung verwenden. Dies ist beispielsweise wichtig, weil nur HTTP/1.1 das Funktionieren persistenter Verbindungen garantiert. Bei der Implementierung eigener Programme, die auf das Web zugreifen, ist diese Unterscheidung sehr wichtig.

#### Die Methode CONNECT

`CONNECT` ermöglicht den Aufbau von HTTPS-Verbindungen, also verschlüsselten HTTP-Verbindungen, über Proxy-Server. Der Proxy selbst kann und wird die übertragenen Daten nicht entschlüsseln, sondern leitet sie lediglich unverändert weiter. Eine `CONNECT`-Anfrage enthält in der Befehlszeile selbst den Hostnamen und Port des Servers, zu dem die Verbindung hergestellt werden soll. Der Stan-

dard-Port für HTTPS-Server ist 443. Eine solche Anfrage sieht also beispielsweise so aus:

```
CONNECT www.mynet.de:443 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.01)
```

Der Proxy-Server stellt die Verbindung her und antwortet mit einer Meldung wie der folgenden:

```
HTTP/1.1 200 Connection established
Proxy-agent: Apache/2.2.15 (Unix)
```

#### Die Methode OPTIONS

Eine `OPTIONS`-Anfrage fordert den Server auf, eine Liste der unterstützten HTTP-Methoden zurückzugeben. Anstelle einer URL wird in der Regel ein Sternchen (*) als Platzhalter für »beliebige URL« angegeben. Beispiel:

```
OPTIONS * HTTP/1.1
```

Der Server antwortet mit einer Angabe wie der folgenden:

```
HTTP/1.1 200 OK
Public: GET, HEAD, POST, PUT
```

Es gibt keine Garantie dafür, dass der Server auf die `OPTIONS`-Anfrage tatsächlich mit den Namen aller Methoden antwortet, die er unterstützt. Die im Beispiel angegebenen HTTP-Methoden `GET`, `HEAD`, `POST` und `PUT` sind der Mindeststandard, den jeder HTTP-Server seit der Protokollversion 1.0 unterstützen muss.

### 2.1.2 HTTP-Statuscodes

Im vorigen Abschnitt haben Sie bereits einige Beispiele für HTTP-Antworten mit ihren Statuscodes gesehen. Hier finden Sie eine systematische Zusammenstellung sämtlicher Statuscodes, die in Server-Antworten vorkommen können. Der Statuscode ist eine dreistellige Zahl, die die Grundbedeutung der Antwort angibt. Eine nähere Betrachtung der Header-Felder, die mit der Anfrage oder der Antwort gesendet werden, finden Sie dagegen in Abschnitt 2.1.3, »HTTP-Header«.

#### Statuscode-Typen

HTTP verwendet fünf verschiedene Grundtypen von Statuscodes. Der Typ ist an der ersten Stelle des Statuscodes erkennbar. Tabelle 2.2 zeigt eine Übersicht über die Statuscode-Typen.

| Nummernbereich | Bedeutung |
|---|---|
| 1xx | Information |
| 2xx | Erfolgsmeldungen |
| 3xx | Umleitungsmeldungen |
| 4xx | Client-Fehler |
| 5xx | Server-Fehler |

**Tabelle 2.2** Übersicht über die HTTP-Statuscode-Typen

Beachten Sie, dass die meisten HTTP-Server-Antworten einen Body enthalten. Falls es sich dabei nicht um die angeforderte Ressource handelt, ist es meist ein HTML-Dokument, das eine genauere Beschreibung des Fehlers oder Problems enthält. In der Praxis zeigen manche Browser (vor allem der Microsoft Internet Explorer) dann aber oft nicht diesen Body an, sondern ihre eigene Fehlermeldungsseite.

### 1xx-Statuscodes (Informationen)

In Tabelle 2.3 sehen Sie die beiden einzigen 1xx-Statuscodes. In der Praxis hat dieser Nummernbereich keine besonders große Bedeutung; nur selten antworten Server mit diesen Meldungen.

| Status-code | Bezeichnung | Erläuterung |
|---|---|---|
| 100 | Continue | Der Server erwartet eine Fortsetzung der Anfrage. |
| 101 | Switching Protocols | Wechsel auf die im Upgrade-Header angegebene HTTP-Version |

**Tabelle 2.3** 1xx-Statuscodes

100 Continue gibt der Server zurück, wenn die Anfrage unvollständig ist und er deshalb eine Fortsetzung erwartet. 101 Switching Protocols besagt, dass der Server mit einer anderen Version des HTTP-Protokolls antwortet, als in der Anfrage verwendet wurde. Die eingesetzte Version des Protokolls wird im Upgrade-Header angegeben.

### 2xx-Statuscodes (Erfolgsmeldungen)

Statuscodes zwischen 200 und 299 zeigen unterschiedliche Arten erfolgreicher Verarbeitung durch den Server an. Tabelle 2.4 zeigt eine entsprechende Übersicht.

| Status-code | Bezeichnung | Erläuterung |
|---|---|---|
| 200 | OK | Anfrage erfolgreich verarbeitet; angeforderte Ressource ist im Body enthalten. |
| 201 | Created | Durch PUT-Anfrage übertragene Datei wurde gespeichert. |
| 202 | Accepted | Anfrage angenommen; keine weiteren Folgen |
| 203 | Non-Authoritative Information | Antwort wurde nicht vom eigentlichen Server, sondern z. B. von einem Proxy erzeugt. |
| 204 | No Content | Anfrage in Ordnung; Antwort besitzt keinen Body. |
| 205 | Reset Content | Dieser Satuscode fordert den Client auf, das Formular zurückzusetzen, das die Anfrage erzeugt hat. |
| 206 | Partial Content | Antwort auf eine Anfrage mit Range-Header; sie enthält nur den gewünschten Bereich der Ressource und einen entsprechenden Content-Range-Header. |

**Tabelle 2.4** 2xx-Statuscodes

Die Mehrheit aller erfolgreich verarbeiteten HTTP-Anfragen liefert eine Antwort mit der Meldung 200 OK, insbesondere bei GET, POST, HEAD und DELETE. Bei einer erfolgreich abgeschlossenen PUT-Anfrage wird dagegen die Meldung 201 Created zurückgegeben. Alle anderen 2xx-Meldungen sind wesentlich exotischer.

### 3xx-Statuscodes (Umleitungsmeldungen)

In Tabelle 2.5 finden Sie sämtliche 3xx-Statuscodes, die Auskunft über diverse Arten der Um- und Weiterleitung geben.

| Status-code | Bezeichnung | Erläuterung |
|---|---|---|
| 300 | Multiple Choices | Die Pfadangabe in der Anfrage passt auf mehrere Ressourcen. Der Body enthält in der Regel eine Liste all dieser Dateien. |
| 301 | Moved Permanently | Die Ressource wurde dauerhaft verschoben und ist unter der im Location-Header angegebenen URL zu finden. |
| 302 | Found | Vorübergehend verschoben; wurde durch 307 ersetzt. |
| 303 | See Other | Die Ressource ist unter der im Location-Header angegebenen URL zu finden. |
| 304 | Not Modified | Antwort auf spezielle Anfragen mit Headern wie If-Modified-Since; ohne Body |

**Tabelle 2.5** 3xx-Statuscodes

| Statuscode | Bezeichnung | Erläuterung |
|---|---|---|
| 305 | Use Proxy | Die Anfrage muss über den im Location-Header angegebenen Proxy erfolgen. |
| 307 | Temporary Redirect | Die Ressource wurde vorübergehend verschoben und ist zurzeit unter der im Location-Header angegebenen URL zu finden. |

**Tabelle 2.5** 3xx-Statuscodes (Forts.)

In dieser Gruppe ist beispielsweise der Statuscode `301 Moved Permanently` wichtig, der zurückgegeben wird, wenn sich ein angefordertes Dokument nun unter einem neuen (in der Antwort angegebenen) Pfad befindet. `307 Temporary Redirect` (in alten HTTP-Versionen auch `302 Found`) wird dagegen zurückgegeben, wenn eine Ressource vorübergehend nicht erreichbar ist, beispielsweise wegen Server-Wartungsarbeiten. Die dritte Art der Umleitung ist `303 See Other`; in diesem Fall handelt es sich um eine absichtliche Weiterleitung auf die angegebene URL. Sie wird etwa dann verwendet, wenn ein serverseitiges Skript keine eigene Ausgabe besitzt, sondern nach getaner Arbeit eine Weiterleitung auf eine andere Seite durchführt. In all diesen Fällen enthält die Antwort eine Header-Zeile nach dem Schema `Location: Neue_URL`.

Ein anderer interessanter Statuscode ist `304 Not Modified`. Es handelt sich um die body-lose Antwort auf eine sogenannte *bedingte Anfrage*: Spezielle Header-Felder wie `If-Modified-Since` ermöglichen es dem Client oder einem Proxy, Dokumente nur unter bestimmten Voraussetzungen zu laden – im genannten Fall beispielsweise nur dann, wenn auf dem Server eine neuere Version des Dokuments vorliegt als diejenige, die der Client oder der Proxy noch in seinem Cache gespeichert hat. Näheres zu diesem Thema finden Sie in Abschnitt 2.1.3, »HTTP-Header«, sowie in Kapitel 12, »Skalierung und Performance-Tuning«.

#### 4xx-Statuscodes (Client-Fehler)

Statuscodes zwischen 400 und 499 stehen für verschiedene Arten von Fehlern oder Mängeln in der Client-Anfrage. Tabelle 2.6 zeigt zunächst eine Übersicht über alle 4xx-Meldungen.

| Statuscode | Bezeichnung | Erläuterung |
|---|---|---|
| 400 | Bad Request | syntaktisch fehlerhafte Client-Anfrage |
| 401 | Unauthorized | Authentifizierung erforderlich |

**Tabelle 2.6** 4xx-Statuscodes

| Status-code | Bezeichnung | Erläuterung |
|---|---|---|
| 402 | Payment Required | Zahlung (Micropayment) ist erforderlich – vorgesehen, aber noch nicht implementiert. |
| 403 | Forbidden | Zugriff verweigert; auch Authentifizierung hilft nichts. |
| 404 | Not Found | Die angeforderte Ressource ist nicht vorhanden. |
| 405 | Method Not Allowed | Die verwendete Client-Zugriffsmethode ist für die angeforderte Ressource nicht zulässig; die erlaubten Methoden stehen im Allow-Header. |
| 406 | Not Acceptable | Dateityp der Ressource widerspricht einem Accept-…-Header der Anfrage. |
| 407 | Proxy Authentication Requested | Benutzeranmeldung am Proxy erforderlich |
| 408 | Request Timeout | Der Server hat innerhalb einer bestimmten Zeit nach Verbindungsaufbau keine Anfrage erhalten. |
| 409 | Conflict | Anfrage kann aufgrund eines Konflikts nicht ausgeführt werden. Beispiel: Die PUT-Anfrage versucht, eine veraltete Version zu speichern, obwohl die WebDAV-Versionskontrolle aktiviert ist. |
| 410 | Gone | Ressource nicht mehr vorhanden. Sollte bei absichtlicher Entfernung zeitlich begrenzter Angebote statt 404 Not Found zurückgegeben werden. |
| 411 | Length Required | Server weist die Anfrage (mit Body) zurück, weil sie keinen Content-Length-Header enthält. |
| 412 | Precondition Failed | Im Header der Anfrage stand eine Bedingung, die nicht zutrifft. |
| 413 | Request Entity Too Long | Body der Anfrage ist zu groß. |
| 414 | Request-URI Too Long | Die Bezeichnung der angeforderten Ressource ist zu lang – kann sehr selten bei GET-Anfragen mit CGI-Daten passieren. |
| 415 | Unsupported Media Type | Body der Anfrage hat ein unzulässiges Format. |
| 416 | Request Range Not Satisfiable | Der im Range-Header der Anfrage angeforderte Bereich existiert nicht. |
| 417 | Expectation Failed | Die im Expect-Header der Anfrage angegebenen Erwartungen konnten vom Server nicht erfüllt werden. |

**Tabelle 2.6** 4xx-Statuscodes (Forts.)

Wie Sie in der Tabelle sehen können, gibt es weit mehr unterschiedliche 4xx-Statuscodes als Meldungen der anderen Kategorien. Da die »Schuld« für das Scheitern der Anfrage bei dem Client liegt, der sie formuliert hat, ist es nützlich, dass ganz genau mitgeteilt wird, worin das Problem besteht. Auf diese Weise kann der Benutzer (oder manchmal sogar der Browser selbst) den Fehler einfacher beheben.

Der bekannteste Statuscode aus dieser Gruppe dürfte `404 Not Found` sein – er wird zurückgegeben, wenn das (meist per `GET`-Anfrage) angeforderte Dokument nicht existiert – beispielsweise, weil Sie die Adresse im Browser falsch geschrieben haben. Auch `401 Unauthorized` oder gar `403 Forbidden` kommen recht häufig vor.

Viele andere 4xx-Statuscodes interessieren eher Programmierer als Benutzer: Wenn Sie Software programmieren, die HTTP-Anfragen an entfernte Webserver sendet, können Sie an den entsprechenden Meldungen sehen, dass Ihre HTTP-Header unvollständig oder fehlerhaft sind. Sollten Sie dagegen eine solche Meldung erhalten, wenn Sie einen normalen Browser benutzen, hat entweder Ihr Browser einen Bug oder der angesprochene Server ist nicht standardkonform. In beiden Fällen können Sie nicht viel daran ändern.

#### 5xx-Statuscodes (Server-Fehler)

Statuscodes ab 500 weisen darauf hin, dass der Server trotz korrekter Client-Anfrage nicht antworten konnte. In Tabelle 2.7 finden Sie eine Übersicht über diese Codes.

| Statuscode | Bezeichnung | Erläuterung |
|---|---|---|
| 500 | Internal Server Error | fehlerhafte Client-Anfrage |
| 501 | Not Implemented | Der Server unterstützt die angeforderte Methode oder Funktionalität nicht. |
| 502 | Bad Gateway | fehlerhafte Antwort von einem übergeordneten Server oder Proxy |
| 503 | Service Unavailable | Dienst zurzeit nicht erreichbar |
| 504 | Gateway Timeout | Timeout in einer per Proxy weitergeleiteten Anfrage |
| 505 | HTTP Version Not Supported | Die in der Anfrage verwendete HTTP-Version wird vom Server nicht unterstützt. |

**Tabelle 2.7** 5xx-Statuscodes

Die Fehlermeldung `500 Internal Server Error` bekommen Sie wahrscheinlich dann häufig zu Gesicht, wenn Sie eigene serverseitige Programme mit CGI, PHP oder ähnlichen Sprachen schreiben: Dieser Statuscode weist fast immer auf einen schwerwiegenden Fehler in einem Webserver-Skript hin. In den Kapiteln 14, »CGI«, und 15, »Technologien zur Webprogrammierung«, in denen es um verschiedene serverseitige Programmiertechniken geht, finden Sie Hinweise, wie Sie Ihre Skripte derart gestalten, dass sie aussagekräftigere Fehlermeldungen liefern.

### 2.1.3 HTTP-Header

In den beiden vorangegangenen Abschnitten haben Sie bereits zahlreiche HTTP-Header kennengelernt. Hier werden sie nun systematisch vorgestellt. Jeder Header steht in einer eigenen Zeile und hat das folgende grundlegende Format:

```
Header-Name: Wert1[, Wert2, Wert3 ...]
```

Bei den Namen der Header-Felder spielt Groß- und Kleinschreibung keine Rolle. Falls für einen Header mehrere Werte existieren, werden diese durch Kommata getrennt.

Zunächst finden Sie in Tabelle 2.8 eine Übersicht über sämtliche Header. Anschließend wird jeder von ihnen näher erläutert. In der Tabelle gibt es drei Felder mit den Kontextinformationen »Anfrage«, »Antwort« und »Entity«. Sie bedeuten Folgendes:

- **Anfrage**
  Wenn dieses Feld angekreuzt ist, kann der Header in der HTTP-Anfrage vorkommen.
- **Antwort**
  Ist dieses Feld angekreuzt, dann kann der Header in einer HTTP-Antwort stehen.
- **Entity**
  Ein Header, bei dem dieses Feld angekreuzt ist, enthält Informationen über den Body der HTTP-Meldung – bekannte Beispiele sind `Content-Type` oder `Content-Length`.

| Header | Seit Version | Anfrage | Antwort | Entity |
|---|---|---|---|---|
| `Accept` | 1.0 | X | | |
| `Accept-Charset` | 1.0 | X | | |
| `Accept-Encoding` | 1.0 | X | | |
| `Accept-Language` | 1.0 | X | | |

**Tabelle 2.8** Übersicht über sämtliche HTTP-Header

| Header | Seit Version | Anfrage | Antwort | Entity |
|---|---|---|---|---|
| Accept-Ranges | 1.1 | | X | |
| Age | 1.1 | | X | |
| Allow | 1.0 | | X | X |
| Authorization | 1.0 | X | | |
| Cache-Control | 1.1 | X | X | |
| Connection | 1.0 | X | X | |
| Content-Encoding | 1.0 | X | X | X |
| Content-Language | 1.0 | X | X | X |
| Content-Length | 1.0 | X | X | X |
| Content-Location | 1.1 | | X | X |
| Content-MD5 | 1.1 | X | X | X |
| Content-Range | 1.1 | X | X | X |
| Content-Type | 1.0 | X | X | X |
| Cookie | 1.0 | X | | |
| Date | 1.0 | X | X | |
| ETag | 1.1 | | X | |
| Expect | 1.1 | X | | |
| Expires | 1.0 | | X | X |
| From | 1.0 | X | | |
| Host | 1.1 | X | | |
| If-Match | 1.1 | X | | |
| If-Modified-Since | 1.0 | X | | |
| If-None-Match | 1.1 | X | | |
| If-Range | 1.1 | X | | |
| If-Unmodified-Since | 1.1 | X | | |
| Last-Modified | 1.0 | | X | X |
| Location | 1.0 | | X | |
| Max-Forwards | 1.1 | X | | |
| Negotiate | 1.1 | X | | |
| Pragma | 1.0 | X | X | |
| Proxy-Authenticate | 1.1 | | X | |
| Proxy-Authorization | 1.0 | X | | |
| Range | 1.1 | X | | |
| Referer | 1.0 | X | | |
| Retry-After | 1.0 | | X | |

**Tabelle 2.8** Übersicht über sämtliche HTTP-Header (Forts.)

| Header | Seit Version | Anfrage | Antwort | Entity |
|---|---|---|---|---|
| Server | 1.0 | | X | |
| Set-Cookie | 1.0 | | X | |
| TE | 1.1 | X | | |
| Trailer | 1.1 | X | X | |
| Transfer-Encoding | 1.1 | X | X | |
| Upgrade | 1.1 | X | X | |
| User-Agent | 1.0 | X | | |
| Vary | 1.1 | | X | |
| Via | 1.1 | X | X | |
| Warning | 1.1 | X | X | |
| WWW-Authenticate | 1.0 | | X | |

**Tabelle 2.8** Übersicht über sämtliche HTTP-Header (Forts.)

Mithilfe des Apache-Webservers können Sie die meisten Header gezielt modifizieren. Wie dies funktioniert, steht detailliert in Kapitel 7, »Header und MIME-Einstellungen«. Hier folgt zunächst eine kurze Beschreibung sämtlicher HTTP-Header.

### Accept

Ein `Accept-Header` gibt eine Liste sämtlicher MIME-Types an, die ein Client als Antwort auf seine Anfrage akzeptiert. Hinter der Angabe eines Typs kann ein Semikolon stehen, gefolgt von Optionen für den jeweiligen Medientyp. Die einzige in RFC 2616 definierte allgemeine Standardoption ist der »Qualitätsfaktor«: `q=Wert` gibt auf einer Skala von 1 (am liebsten) bis 0 (äußerst ungern) an, wie sehr der Client einen bestimmten Typ bevorzugt. Der Standardwert ist für alle explizit angegebenen Medientypen `q=1` und für alle nicht angegebenen `q=0`. Ein entsprechend konfigurierter Server kann sich nach diesen Werten richten und dem Browser die jeweils bevorzugte Alternative liefern.

Hier sehen Sie ein Beispiel:

```
Accept: image/png, image/jpeg; q=0.8, image/gif; q=0.5,
        image/bmp; q=0.3, image/*; q=0.2
```

Der Client bevorzugt laut dieser Angabe Bilder vom Typ PNG, danach in absteigender Beliebtheitsreihenfolge JPEG, GIF, BMP und beliebige andere Bildformate. In der Praxis verwenden Browser meist nur eine Liste von MIME-Types, aber keine Qualitätsfaktoren. Eine Tabelle aller wichtigen MIME-Types finden Sie in Anhang E.

Wenn ein Webserver auf eine Anfrage nicht mit einem im `Accept`-Header angegebenen Medientyp antworten kann, sollte er eine Meldung mit dem Statuscode `406 Not Acceptable` zurückgeben. Üblicherweise enthält die Liste, die fast alle Browser senden, unter anderem den Platzhalter `*/*` (beliebiger Typ, beliebiger Untertyp), sodass dies in der Praxis fast nie vorkommt. Entsprechendes gilt auch für die drei Client-Header `Accept-Charset`, `Accept-Encoding` und `Accept-Language`, die als Nächste besprochen werden.

Die mehr oder weniger automatische Behandlung von `Accept*`-Headern wird als *Content-Negotiation* bezeichnet. Da Apache als HTTP/1.1-Server diese Fähigkeit besitzt, wird sie in Kapitel 7, »Header und MIME-Einstellungen«, ausführlich beschrieben.

#### Accept-Charset

Dieser Header gibt eine Liste der vom Client bevorzugten Zeichensätze an. Auch hier können Sie nach demselben Schema wie beim `Accept`-Header einen Qualitätsfaktor angeben. Zeichensätze, die Sie gar nicht angeben, haben automatisch `q=0`. Eine Ausnahme bildet iso-8859-1 (ISO-Latin-1), der Standardzeichensatz für USA und Westeuropa; sein Standardwert ist automatisch `q=1`.

Das folgende Beispiel bevorzugt iso-8859-9 (Spezialvariante von ISO-Latin-1 mit türkischen Sonderzeichen), an zweiter Stelle utf-8 (Unicode mit 8-Bit-Codierung aller normalen ASCII-Zeichen) und erst danach alle anderen Zeichensätze. In diesem speziellen Fall gilt der Platzhalter `*` – und damit der entsprechende Qualitätsfaktor – auch für iso-8859-1:

```
Accept-Charset iso-8859-9, utf-8; q=0.8, *; q=0.5
```

#### Accept-Encoding

Der Header `Accept-Encoding` gibt Kompressionsverfahren an, die der Client für den Body der Server-Antwort zulässt. Grundsätzlich sind die folgenden Werte möglich:

- `gzip` – das GNU-ZIP-Kompressionsverfahren
- `compress` – eine LZW-Variante
- `deflate` – der von WinZip® oder PKZIP verwendete ZIP-Algorithmus
- `identity` – die Originaldaten ohne Komprimierung (Standard)

Wie bei den anderen `Accept`-Client-Headern kann auch hier ein Qualitätsfaktor angegeben werden. Das folgende Beispiel bevorzugt `gzip` vor `deflate` und `identity`:

```
Accept-Encoding gzip, deflate; q=0.8, identity; q=0.5
```

### Accept-Language

Dieser Anfrage-Header enthält eine Liste der Sprachen, in denen der Client Inhalte bevorzugt. Ein entsprechend konfigurierter Server liefert dann jeweils die Version, die dem Client am ehesten zusagt.

Auch bei diesem Header lässt sich der schon beschriebene Qualitätsfaktor einsetzen. Im folgenden Beispiel wird Deutsch vor Englisch und Französisch bevorzugt:

```
Accept-Language: de, en; q=0.8, fr; q=0.5
```

Eine Liste der verfügbaren Sprachkürzel finden Sie in Anhang E.

### Accept-Ranges

Auch wenn er dem Namen nach einem Anfrage-Header ähnelt, handelt es sich hier um einen Antwort-Header. Er besitzt nur zwei mögliche Werte: `none` (Standard) besagt, dass der Server Ressourcen nur komplett ausliefern kann. `bytes` bedeutet dagegen, dass er Anfragen mit einem `Range`-Header verarbeiten und Dateien in kleineren Portionen ausliefern kann. Dies macht sich beispielsweise der Adobe (Acrobat) Reader beim Online-Zugriff auf größere PDF-Dateien zunutze. Beispiel:

```
Accept-Ranges: bytes
```

### Age

Dieser bei Server-Antworten verwendete Entity-Header gibt das Alter des Dokuments in Sekunden an. Dieser Header ist besonders dann wichtig, wenn das Dokument aus einem Proxy-Cache geliefert wird. Beispiel:

```
Age: 7265
```

### Allow

Dieser Header wird in einer Antwort mit dem Statuscode `405 Method Not Allowed` geliefert. Er enthält eine Liste der Anfragemethoden, die für die angeforderte URL zulässig sind. Beispiel:

```
Allow: GET, POST
```

### Authorization

Bei manchen HTTP-Anfragen ist eine Benutzeranmeldung erforderlich. Wenn ein Zugriff ohne diese Authentifizierung erfolgt, liefert der Server eine Meldung mit

dem Statuscode `401 Unauthorized`. Der `Authorization`-Header enthält das geforderte Authentifizierungsschema und die zugehörigen konkreten Anmeldedaten. Eine einfache Anmeldung mit Benutzername und Passwort besteht beispielsweise aus dem Schema `BASIC` und der Base64-codierten `Benutzername:Passwort`-Angabe. Das folgende Beispiel verwendet den Benutzernamen `webuser` und das Passwort `TopSecret`:

```
Authorization: BASIC d2VidXNlcjpUb3BTZWNyZXQ=
```

Weitere Informationen über Autorisierungsverfahren finden Sie in Kapitel 9, »Authentifizierung«.

#### Cache-Control

Dieser Header kann sowohl bei der Anfrage als auch bei der Antwort stehen und enthält detaillierte Angaben über die Behandlung der angeforderten beziehungsweise gelieferten Ressource durch den Proxy- oder Browser-Cache. Er betrifft stets sämtliche Proxys zwischen Client und Server. Die möglichen Werte für Client-Anfrage und Server-Antwort sind teilweise unterschiedlich. Tabelle 2.9 zeigt eine Übersicht über die Werte für Anfragen; in Tabelle 2.10 sehen Sie dagegen die möglichen Einstellungen bei Server-Antworten. Bei Bedarf kann dieser Header mehrere durch Komma getrennte Werte enthalten.

Näheres zum Thema Web-Caching finden Sie in Kapitel 13, »Proxy- und Cache-Funktionen«.

| Cache-Control-Wert | Bedeutung |
|---|---|
| `no-cache` | Aktualität der Kopie im Cache muss vor der Auslieferung überprüft werden. |
| `no-store` | Es soll keine Kopie im Cache gespeichert werden. |
| `max-age=Sekunden` | Der Cache darf seine Kopie ausliefern, wenn sie maximal so alt ist wie angegeben. |
| `max-stale[=Sekunden]` | Der Cache darf Dokumente ausliefern, deren Verfallsdatum überschritten ist (falls eine Sekundenzahl angegeben wurde, höchstens um diese Anzahl von Sekunden). |
| `min-fresh=Sekunden` | Der Cache darf seine Kopien nur ausliefern, wenn sie mindestens um die angegebene Sekundenzahl aktueller sind als ihr Verfallsdatum. |
| `only-if-cached` | Der Cache soll lediglich seine Kopie ausliefern, falls vorhanden, aber keine neue Version des Dokuments anfordern. |

**Tabelle 2.9** Mögliche Werte für Cache-Control bei Client-Anfragen

| Cache-Control-Wert | Bedeutung |
|---|---|
| `public` | Jeder Proxy oder Browser darf eine Kopie des Dokuments im Cache ablegen. |
| `private` | Nur der Empfänger darf eine Kopie im Cache ablegen; ein öffentlicher Proxy nicht. |
| `no-cache` | Aktualität der Kopie im Cache muss vor der Auslieferung überprüft werden. |
| `no-store` | Es soll keine Kopie im Cache gespeichert werden. |
| `no-transform` | Der Cache darf keine Änderungen am Body vornehmen. |
| `must-revalidate` | Der Cache muss das Verfallsdatum beachten. |
| `proxy-revalidate` | Der Client muss das Verfallsdatum von Dokumenten beachten, die von einem öffentlichen Proxy geliefert wurden. |
| `max-age=Sekunden` | Das Dokument soll nach der angegebenen Anzahl von Sekunden als veraltet betrachtet werden. |
| `s-maxage=Sekunden` | Öffentliche Proxy-Server sollen das Dokument nach der Anzahl von Sekunden als veraltet betrachten; Clients brauchen diese Direktive nicht zu beachten. |

**Tabelle 2.10** Mögliche Werte für Cache-Control bei Server-Antworten

### Connection

Dieser Header bestimmt, ob die verwendete Verbindung persistent sein soll (Wert `keepalive`) oder nach Bearbeitung der aktuellen Anfrage wieder geschlossen wird (`close`). Bei HTTP/1.0 war der Standardwert `close`. HTTP/1.0-Server waren nicht verpflichtet, persistente Verbindungen zu unterstützen (und taten dies oft auch nicht). Der Standard bei HTTP/1.1-Clients und -Servern ist `keepalive`. Beachten Sie, dass jeder Proxy auf dem Übertragungsweg HTTP/1.1 beziehungsweise `keepalive` unterstützen muss, damit eine Verbindung offen bleiben kann. Beispiel:

```
Connection: keepalive
```

### Content-Encoding

Dieser Entity-Header gibt das Verfahren an, mit dem der Body komprimiert wurde. Die möglichen Werte entsprechen denjenigen bei `Accept-Encoding`. Falls eine Anfrage einen Body mit einer Kompression enthält, die der Server nicht unterstützt, liefert dieser `415 Unsupported Media Type` zurück. Das folgende Beispiel zeigt einen `Content-Encoding`-Header für einen GZIP-komprimierten Body:

```
Content-Encoding: gzip
```

### Content-Language

Dieser Header gibt eine Liste der Sprachen an, in denen der Inhalt des Bodys verfasst ist. Die zweistelligen Sprachkürzel nach ISO, die auch für `Accept-Language` verwendet werden, finden Sie in Anhang E. Für ein deutschsprachiges Dokument sieht der Header beispielsweise so aus:

```
Content-Language: de
```

### Content-Length

Dieser Header gibt die Länge des Bodys in Byte an. Der Empfänger (Client oder Server) kann anhand dieser Angabe überprüfen, ob der Body vollständig übertragen wurde. Beispiel:

```
Content-Length: 52637
```

### Content-Location

Dieser Entity-Header gibt eine zusätzliche oder alternative URL an, unter der das gelieferte Dokument zu finden ist. Beispiel:

```
Content-Location: http://www.mynet.de/info.html
```

### Content-MD5

Der optionale Entity-Header `Content-MD5` enthält einen MD5-Digest des Bodys. Dieser nach einem komplexen Verfahren berechnete Wert kennzeichnet einen bestimmten Inhalt eindeutig. Eine Abweichung bedeutet, dass der gelieferte Inhalt nicht mit dem erwarteten übereinstimmt. Dieses Verfahren wird besonders häufig für Binärdateien verwendet, die zum Download angeboten werden: Durch Vergleich mit einem vom Anbieter veröffentlichten MD5-Digest lässt sich verhindern, dass eine defekte oder gar virenbehaftete Variante des gewünschten Programms heruntergeladen wird. Am häufigsten findet diese Methode in der Open-Source-Szene Anwendung, natürlich auch bei Apache – näheres dazu in Kapitel 4, »Apache kompilieren und installieren«. Beispiel:

```
Content-MD5: OIhoWmXLpVrESYHsjIrSbw==
```

### Content-Range

Dieser Entity-Header wird angegeben, wenn nicht das vollständige Dokument geliefert wird, sondern nur ein bestimmter Teil. Dies geschieht nur auf ausdrückliche Anforderung des Clients. Der Wert dieses Headers ist der gelieferte Bereich (Startbyte-Endbyte), gefolgt von einem Slash und der Gesamtgröße des Bodys in Byte. Beispiel:

```
Content-Range: 35425-68726/928379
```

### Content-Type

Bei `Content-Type` handelt es sich um den wichtigsten aller Entity-Header. Er gibt den MIME-Type des gelieferten Bodys an. Dies ist sowohl bei Client-Anfragen der Typen `POST` oder `PUT` als auch bei Server-Antworten von großer Bedeutung. Der Server ermittelt den zu liefernden Typ zumeist aus der Dateierweiterung der ausgelieferten Datei, ansonsten aus expliziten Einstellungen oder sogar anhand des Dateiinhalts – Genaues erfahren Sie in Kapitel 7, »Header und MIME-Einstellungen«. Bei `POST`-Anfragen wird der Wert aus der `enctype`-Angabe im HTML-Formular verwendet; standardmäßig übermitteln Browser den Typ `application/x-www-form-urlencoded` (siehe Diskussion in Abschnitt 2.1.1, »Die HTTP-Client-Anfrage«). Eine ausführliche Liste der MIME-Types befindet sich in Anhang E. Das folgende Beispiel zeigt den `Content-Type`-Header für ein HTML-Dokument:

```
Content-Type: text/html
```

Bei Textdatentypen wie HTML oder einfachem ASCII-Text (`text/plain`) wird üblicherweise der verwendete Zeichensatz als Zusatzoption angegeben. Der folgende Header steht für ein HTML-Dokument, dessen Zeichensatz ISO-Latin-1 (USA und Westeuropa) ist:

```
Content-Type: text/html; charset=iso-8859-1
```

### Cookie

Mit diesem Header liefert der Client ein oder mehrere Cookies zurück, die der Server zuvor gesetzt hat. Ein Cookie ist eine kleine Textinformation im Format `Name=Wert`, die der Browser im Auftrag eines Servers speichert und bei der nächsten Anfrage an diesen Server zurückgibt. Dieses Verfahren dient beispielsweise der Personalisierung von Websites (bekanntes Beispiel: Amazon). Näheres über den sinnvollen Einsatz von Cookies erfahren Sie in Kapitel 15, »Technologien zur Webprogrammierung«. Hier ein Beispiel für ein Cookie, das der Client zurücksendet:

```
Cookie: user=peter_schmitz
```

### Date

Fast alle Server und manche Clients senden HTTP-Nachrichten mit einem `Date`-Header, der Datum und Uhrzeit zum Zeitpunkt des Versands angibt. Aktuelle Software verwendet das Datumsformat nach RFC 1123, das beispielsweise so aussieht:

```
Sun, 21 Aug 2011 23:24:57 GMT
```

Die Wochentage werden bei diesem Format als `Mon`, `Tue`, `Wed`, `Thu`, `Fri`, `Sat` und `Sun` angegeben; die Monate sind `Jan`, `Feb`, `Mar`, `Apr`, `May`, `Jun`, `Jul`, `Aug`, `Sep`, `Oct`,

Nov und Dec. Die Uhrzeit wird immer als GMT und nicht als lokale Zeit angegeben.

Einige ältere Clients und Server verwenden andere Formate, nämlich RFC 1036 beziehungsweise das Format, das die ANSI-C-Funktion asctime() generiert. Die Zeitangabe aus dem gerade genannten Beispiel sieht als RFC-1036-Wert so aus:

```
Sunday, 21-Aug-11 23:24:57 GMT
```

Im asctime()-Format lautet sie dagegen folgendermaßen:

```
Sun Aug 21 23:24:57 2011
```

Wenn Sie eigene HTTP-kompatible Anwendungen programmieren, sollten diese RFC-1123-Zeitangaben erzeugen, aber den Empfang der beiden anderen Formate akzeptieren. Ein solches Vorgehen entspricht dem »Robustheitsprinzip« (Robustness Principle), das der verstorbene Internetveteran Jon Postel formuliert hat, beispielsweise in RFC 791: »... an implementation must be conservative in its sending behavior, and liberal in its receiving behavior.«[2]

#### ETag

Dieser Header, das Entity-Tag, wird vom Server generiert und besteht aus einer hexadezimalen Zahl, die die Identität eines bestimmten Inhalts bestätigt. Dies ist besonders für Cache-Überprüfungen interessant. Beispiel:

```
ETag: "2c986a-b20-fc93242e"
```

Ein Client kann in einer Anfrage einen If-Match-Header verwenden, um die Übereinstimmung der Antwort mit einem bestimmten ETag zu überprüfen. Beachten Sie, dass das Entity-Tag nicht wie beispielsweise ein MD5-Hash aus dem Datenbestand des Inhalts selbst, sondern aus Informationen wie Änderungsdatum und Dateigröße berechnet wird.

#### Expect

Der Anfrage-Header Expect hat allgemein die Aufgabe, dem Server bestimmte Erwartungen des Clients an dessen Konfiguration mitzuteilen. Zurzeit gibt es eigentlich nur eine konkrete Anwendung: Manche Clients erwarten vor dem Senden eines Bodys mit POST- oder PUT-Anfragen, dass der Server ihnen eine Antwort mit dem Statuscode 100 Continue schickt. Dies kann ein Client in der Anfrage folgendermaßen ausdrücken:

2 Eine bekanntere Fassung ist: »Be conservative in what you do, be liberal in what you accept from others.«

```
Expect: 100-Continue
```

Wenn der Server diese Erwartung nicht erfüllen kann, gibt er `417 Expectation Failed` zurück.

#### Expires

Dieser Entity-Header gibt an, bis wann die gelieferte Ressource gültig ist. Wie bei `Date` handelt es sich normalerweise um einen RFC-1123-Zeitstempel. Wenn ein Proxy oder Browser das Dokument in seinem Cache speichert, kann er es bis zum angegebenen Zeitpunkt ohne Rückfrage an den Benutzer ausliefern (es sei denn, die `Cache-Control`-Header bestimmen etwas anderes). Beispiel:

```
Expires: Thu, 25 Aug 2011 00:00:00 GMT
```

#### From

Der Anfrage-Header `From` gibt die E-Mail-Adresse des anfragenden Users an. Da Spam heutzutage einen größeren wirtschaftlichen Schaden anrichtet als Viren und Cracker zusammen, können Sie sich leicht vorstellen, dass diese ursprünglich gute Idee in der Praxis nicht mehr sehr beliebt ist. Insofern ist es nicht weiter verwunderlich, dass so gut wie kein Browser auch nur die Option bietet, diese Funktionalität zu aktivieren, geschweige denn automatisch diesen Header sendet. Beispiel:

```
From: klaus@provider.de
```

#### Host

Der Anfrage-Header `Host` bestimmt, an welchen konkreten Hostnamen die Anfrage gesendet werden soll. Da die meisten Webhoster namensbasierte virtuelle Hosts verwenden (siehe Kapitel 12, »Skalierung und Performance-Tuning«), das heißt mehrere unterschiedliche Domains auf Server-Rechnern mit derselben IP-Adresse betreiben, ist er sehr wichtig; HTTP/1.1-Anfragen sind ohne diesen Header sogar ungültig und werden mit `400 Bad Request` beantwortet. Beispiel:

```
Host: www.mynet.de
```

#### If-Match

Eine Anfrage, die einen `If-Match`-Header enthält, wird nur dann erfolgreich beantwortet, wenn das Entity-Tag (`ETag`-Header) der Antwort mit dem hier genannten übereinstimmt. Andernfalls gibt der Server `412 Precondition Failed` zurück. Beispiel:

```
If-Match: "2c986a-b20-fc93242e"
```

Eine andere Variante ist die folgende allgemeine Formulierung:

```
If-Match: *
```

Dieser Header liefert die angeforderte Ressource, wenn die Anfrage erfolgreich verarbeitet werden konnte. Andernfalls liefert er wiederum den Statuscode 412.

Eine `If-Match`-Anfrage ist beispielsweise nützlich, wenn mehrere verteilt arbeitende Personen Schreibzugriff auf eine Website haben: Eine `PUT`-Anfrage sollte nur dann ausgeführt werden, wenn das `ETag` der entsprechenden Datei dem zuletzt bekannten Stand entspricht – dies garantiert, dass keine Datei überschrieben wird, die jemand anders geändert hat.

#### If-Modified-Since

Der Anfrage-Header `If-Modified-Since` benötigt ein HTTP-Datum (siehe `Date`-Header). Falls die gewünschte Ressource seit dem angegebenen Zeitpunkt geändert wurde, wird sie geliefert. Andernfalls antwortet der Server mit `304 Not Modified`. Diese Technik wird von Browsern und Proxys verwendet, die bereits eine Kopie der gewünschten Datei im Cache gespeichert haben; sie erspart den erneuten Download, falls die Kopie noch aktuell ist. Beispiel:

```
If-Modified-Since: Sat, 20 Aug 2011 11:08:37 GMT
```

#### If-None-Match

Dieser Anfrage-Header ist das Gegenteil von `If-Match`: Die Anfrage wird nur dann erfolgreich verarbeitet, wenn die angeforderte Ressource nicht dem angegebenen Entity-Tag entspricht. Genau wie `If-Match` wird er vornehmlich für `PUT`-Anfragen verwendet: Die Datei soll nur dann aktualisiert werden, wenn sie sich geändert hat, das heißt, wenn ihr `ETag` nicht dem zuletzt bekannten Stand entspricht. Beispiel:

```
If-None-Match: "2c986a-b20-fc93242e"
```

Um eine Datei nur dann hochzuladen, falls sie noch gar nicht existiert, wird die folgende verallgemeinerte Variante eingesetzt:

```
If-None-Match: *
```

#### If-Range

Falls ein Client einen bestimmten Byte-Bereich einer Ressource bereits im Cache gespeichert hat, kann er eine Anfrage mit dem `Range`-Header, der diesen Bereich anfordert, mit einem `If-Range`-Header kombinieren. Dieser kann zwei unterschiedliche Informationen überprüfen: Wenn sein Wert ein HTTP-Datum ist,

wird der Bereich erneut geliefert, falls er nach dem angegebenen Zeitpunkt geändert wurde. Handelt es sich dagegen um ein `ETag`, wird der Bereich nur dann übermittelt, wenn er diesem `ETag` nicht entspricht. Andernfalls wird `304 Not Modified` zurückgegeben. Sollte der Bereich gar nicht existieren, wird stattdessen das gesamte Dokument übertragen. Beispiele:

```
If-Range: "2c986a-b20-fc93242e"
If-Range: Fri, 19 Aug 2011 11:08:37 GMT
```

#### If-Unmodified-Since

Dieser Anfrage-Header ist das Gegenteil von `If-Modified-Since`. Die Anfrage wird nur dann erfolgreich ausgeführt, wenn das Dokument seit dem angegebenen Zeitpunkt *nicht* geändert wurde. Beispiel:

```
If-Unmodified-Since: Sat, 20 Aug 2011 12:23:42 GMT
```

#### Last-Modified

Der Entity-Header `Last-Modified` enthält eine Datumsangabe, die angibt, wann das im Body enthaltene Dokument zuletzt geändert wurde. Auf dieses Feld kommt es besonders bei `HEAD`-Anfragen an, da es die Frage beantwortet, ob das Dokument neu geladen werden muss oder nicht. Beispiel:

```
Last-Modified: Thu, 18 Aug 2011 09:32:21 GMT
```

#### Location

Dieser Antwort-Header ist für eine Umleitung zuständig: Die gewünschte Ressource befindet sich unter der angegebenen URL. Der Statuscode der Antwort gibt Aufschluss über die näheren Umstände der Umleitung, was Einfluss auf das Caching der Antwort hat: `301 Moved Permanently` bedeutet, dass die Ressource dauerhaft unter der angegebenen neuen URL zu finden ist. `307 Moved Temporarily` (in älteren HTTP-Versionen auch `302 Found`) besagt dagegen, dass die Umleitung nur vorübergehend gilt, beispielsweise wegen einer Aktualisierung des Servers. `303 See Other` heißt schließlich, dass keine Informationen über die Dauer der Umleitung verfügbar sind – diese Meldung wird meistens von einem serverseitigen Skript ohne eigene Dokumentausgabe erzeugt. Beispiel:

```
Location: thanks.html
```

#### Max-Forwards

Dieser Anfrage-Header gibt die maximale Anzahl von Proxy-Weiterleitungen an, die der Client bis zum Erreichen des Ziels duldet. Er wird meist bei `TRACE`-Anfragen verwendet, die ermitteln, über welche Proxys eine HTTP-Verbindung abge-

wickelt wird. Jeder Proxy zieht von diesem Wert 1 ab. Derjenige Proxy, bei dem der Wert 0 erreicht wird, leitet die Anfrage nicht mehr weiter, sondern sendet eine Antwort, deren Body die ursprüngliche Anfrage mit einem `Via`-Header ist. Dieser enthält die verwendeten HTTP-Versionen und die Adressen aller Proxys, die auf dem Weg liegen. Das Verfahren ähnelt dem im vorigen Kapitel beschriebenen TTL-Wert bei IP-Datagrammen. Beispiel:

```
Max-Forwards: 4
```

#### Negotiate

Die Definition dieses Anfrage-Headers stammt nicht aus der eigentlichen HTTP/1.1-Spezifikation in RFC 2616, sondern aus RFC 2295, *Transparent Content Negotiation*. Wenn ein Client diesen Header an einen entsprechend ausgestatteten Server sendet, liefert dieser kein einzelnes Dokument zurück, sondern eine Liste mit Alternativen. Der Statuscode dieser Server-Antwort ist `300 Multiple Choices`. Das Verfahren wird in Kapitel 7, »Header und MIME-Einstellungen«, näher beschrieben.

#### Pragma

In der Spezifikation der HTTP-Version 1.0 war der `Pragma`-Header für Konfigurationsanweisungen reserviert. Die einzige konkrete Anwendung war zu dieser Zeit der folgende Wert:

```
Pragma: no-cache
```

Diese Einstellung bedeutet, dass ein Proxy oder Browser die Datei nicht in seinem Cache speichern soll. Wie Sie bereits erfahren haben, ist `no-cache` in HTTP/1.1 einer der vielen möglichen Werte des Headers `Cache-Control`. Letzteres ist heute die bevorzugte Variante; `Pragma` wird nur noch aus Gründen der Abwärtskompatibilität unterstützt.

#### Proxy-Authenticate

Wenn ein Proxy-Server eine Authentifizierung benötigt, liefert er die Meldung `407 Proxy-Authentication Required` zurück. Der Header `Proxy-Authenticate` gibt an, welches Authentifizierungsschema erwartet wird; außerdem kann ein Authentifizierungsbereich (»Realm«) übermittelt werden, der als Anmeldeaufforderung angezeigt wird und an dem der Client eventuell erkennen kann, ob er die Anmeldedaten für die entsprechende Ressource bereits kennt oder nicht. Authentifizierungsschemata sind gemäß RFC 2617 beispielsweise `BASIC` (einfache Benutzername-Passwort-Kombination) und `Digest` (verschlüsseltes Passwort). Beispiel:

```
Proxy-Authenticate: BASIC realm="shop"
```

### Proxy-Authorization

Der Anfrage-Header `Proxy-Authorization` enthält die Anmeldedaten für einen Proxy-Server nach dem Schema, das dieser über einen `Proxy-Authenticate`-Header bekannt gegeben hat. Sein Format entspricht dem weiter oben beschriebenen Header `Authorization`, der zur Anmeldung am angesprochenen Server dient. Beispiel (Benutzername `webuser` und Passwort `TopSecret` mit dem Schema `BASIC`, Base64-codiert):

```
Proxy-Authorization: BASIC d2VidXNlcjpUb3BTZWNyZXQ=
```

### Range

Dieser Anfrage-Header fordert kein vollständiges Dokument an, sondern einen bestimmten Byte-Bereich. Dieses Verfahren wird beispielsweise vom Adobe Acrobat Reader verwendet, der lange PDF-Dokumente nicht im Ganzen, sondern stückweise anfordert. Beispiel:

```
Range: 2876-3162
```

### Referer

Der Anfrage-Header `Referer` (das englische Wort heißt korrekt eigentlich »Referrer«) gibt die URL des Dokuments an, das zuvor im anfordernden Browser betrachtet wurde. In der Regel handelt es sich also um das Dokument, in dem ein Hyperlink angeklickt wurde, der danach die aktuelle Ressource angefordert hat. Anhand des Referrers, der üblicherweise auch in die Log-Datei eines Webservers wie Apache übernommen wird, lässt sich genau untersuchen, woher die verschiedenen Besucher auf die eigene Site gekommen sind.

Beispiel:

```
Referer: http://www.google.de/search?hl=de&ie=ISO-8859-1&q=apache
&btnG=Google+Suche
```

Dieses Beispiel zeigt, dass jemand über die Eingabe des Begriffs »apache« in die Suchmaschine Google auf die aktuelle Seite gelangt ist.

### Retry-After

Der Antwort-Header `Retry-After` wird mit dem Statuscode `503 Service Unavailable` oder mit `307 Moved Temporarily` geliefert. Er besagt, wann der angeforderte Dienst oder die Ressource frühestens wieder verfügbar sein wird. Der Wert ist entweder ein HTTP-Datum (siehe `Date`) oder eine reine Zahl, die der Anzahl der Sekunden ab dem aktuellen Zeitpunkt entspricht. Beispiele:

```
Retry-After: Tue, 23 Aug 2011 23:00:00 GMT
Retry-After: 7200
```

Das zweite Beispiel entspricht einer Mindestwartezeit von zwei Stunden.

#### Server

Dieser Antwort-Header liefert die Identität des Servers. Wie sich diese Angabe bei Apache 2 anpassen lässt, erfahren Sie in Kapitel 6, »Grundkonfiguration«. Beispiel:

```
Server: Apache 2.2.15 (Unix)
```

#### Set-Cookie

Der Antwort-Header `Set-Cookie` wird verwendet, um den Client zu bitten, für ihn ein Cookie zu speichern. Bei der nächsten Anfrage an denselben Host liefert der Browser diese Information automatisch in einem `Cookie`-Header zurück. In den meisten Browsern können Sie einstellen, ob Sie Cookies automatisch annehmen, ablehnen oder jeweils individuell behandeln möchten. Neben dem `Name=Wert`-Paar, das die eigentliche Cookie-Information bildet, kann der Server einige Optionen angeben:

- `expires=Datum`
  Das Cookie soll über das angegebene Datum (RFC 1123) hinaus nicht gespeichert werden. Cookies, die diese Option nicht enthalten, gelten nur für die aktuelle Browser-Sitzung (Session-Cookies).
- `path=Pfad`
  Ein Cookie mit dieser Option gilt nicht für den gesamten Host oder die gesamte Domain, sondern nur für den angegebenen Datei- oder Verzeichnispfad.
- `domain=Domainname`
  Das Cookie gilt für die angegebene Domain.
- `Secure`
  Dieses Cookie soll nur über eine gesicherte Verbindung übertragen werden.

Beispiele:

```
Set-Cookie: bgcolor=blue
Set-Cookie: agb_akzeptiert=yes expires=Sun, 24 Aug 2008 00:00:00 GMT
```

Näheres über Cookies erfahren Sie in Kapitel 15, »Technologien zur Webprogrammierung«.

#### TE

Der Name dieses Anfrage-Headers müsste eigentlich `Accept-Transfer-Encoding` lauten, analog zu weiter oben beschriebenen Headern wie `Accept-Charset` oder `Accept-Language`. `Transfer-Encoding` hat nichts mit der Codierung beziehungsweise Komprimierung der Body-Daten (`Content-Encoding`) zu tun. Es handelt sich vielmehr um Optionen zur Übertragung selbst. Der einzige standardisierte Wert in HTTP/1.1 ist `trailers`. Er besagt, dass der Client Server-Antworten akzeptiert, die aus einzelnen Blöcken (Chunks) mit jeweils eigenen Headern bestehen. Beispiel:

```
TE: trailers
```

#### Trailer

Der Header `Trailer` listet diejenigen Header auf, die in einem späteren Block (Chunk) des Dokuments folgen. Beispiel:

```
Trailer: Content-Length
```

#### Transfer-Encoding

Dieser Entity-Header beschreibt Optionen der Übertragung. Der einzige HTTP/1.1-Standardwert ist `chunked`: Eine HTTP-Nachricht mit diesem Header besteht aus mehreren Blöcken (Chunks), die jeweils eigene Header enthalten. Beispiel:

```
Transfer-Encoding: chunked
```

#### Upgrade

Ein Client kann in diesem Header die von ihm unterstützten HTTP-Versionen angeben. Möchte ein Server auf eine angegebene Version wechseln, gibt er eine Meldung mit dem Statuscode `101 Switching Protocols` zurück. Diese enthält ebenfalls einen `Upgrade`-Header mit der entsprechenden Protokollversion. Beispiel:

```
Upgrade: HTTP/1.1
```

#### User-Agent

Mit diesem Anfrage-Header liefert der Client Informationen über sich selbst. In der Regel werden ein Name, eine Versionsnummer und einige weitere Details (z. B. das Betriebssystem) angegeben. Mithilfe dieser Angabe lassen sich angepasste Dokumente liefern, die unterschiedlichen Code für die verschiedenen Browser oder Suchmaschinen enthalten. Außerdem wird der `User-Agent`-Header in die erweiterten Log-Dateien der meisten Webserver aufgenommen. Auf diese

Weise entstehen die bekannten Statistiken, aus denen hervorgeht, wie die Nutzung der verschiedenen Browser verteilt ist.

Neben Mozilla und Firefox geben unter anderem auch Microsoft Internet Explorer und Netscape als ihren Hauptnamen `Mozilla` an. Das liegt daran, dass der Drache »Mozilla« das Maskottchen der Entwickler des ersten grafischen Browsers NCSA Mosaic war, von dem sehr viele Browser abstammen. Die drei folgenden Beispiele sind der Microsoft Internet Explorer 7.0 unter Windows Vista, Mozilla Firefox unter Linux und Apple Safari unter Mac OS X:

```
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0;
SLCC1; .NET CLR 2.0.50727;
Media Center PC 5.0; .NET CLR 3.0.04506)
User-Agent: Mozilla/5.0 (X11; U; Linux i586; en-US; rv:1.6) Gecko/
2008092417 Firefox/3.0.3
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X; de-de)
AppleWebKit/103u (KHTML, like Gecko) Safari/100.1
```

#### Vary

Dieser Entity-Header wird vom Server verwendet, wenn der Body-Inhalt mehrfach verfügbar ist. Er listet die Anfrage-Header auf, für die mehrere Variationen verfügbar sind. Ein `*` wird verwendet, wenn auch andere Faktoren als die `Accept-...`-Header der Anfrage das gelieferte Dokument beeinflussen können. Der Header spielt eine wichtige Rolle für Content-Negotiation. Beispiel:

```
Vary: Accept-Charset, Accept-Language
```

Näheres zu diesem Thema erfahren Sie in Abschnitt 7.3, »Content-Negotiation« in Kapitel 7, »Header und MIME-Einstellungen«.

#### Via

Jeder Proxy-Server auf dem Weg zum Ziel hängt die von ihm verwendete Versionsnummer des HTTP und seinen DNS-Namen an diesen Header an. Besonders wichtig ist der `Via`-Header bei `TRACE`-Anfragen. Beispiel:

```
Via: 1.1 proxy.provider.net, 1.1 proxy.mynet.de
```

#### Warning

Der `Warning`-Header wird von Caching-Proxys versendet und enthält zusätzliche Informationen zum Status des Bodys. Die verfügbaren Meldungen mit ihren Codenummern und Erläuterungen finden Sie in Tabelle 2.11. Beispiel:

```
Warning: 111
```

| Statuscode | Bezeichnung | Erläuterung |
|---|---|---|
| 110 | Response Stale | Antwort ist veraltet. |
| 111 | Revalidation failed | Aktualitätsprüfung ist nicht möglich. |
| 112 | Disconnected Operation | Der Cache hat keine Netzwerkverbindung. |
| 113 | Heuristic Expiration | Proxy hat Verfallsdatum von über 24 Stunden aufgrund einer Schätzung gesetzt. |
| 199 | Miscellaneous Warning | sonstige Warnung |
| 214 | Transformation Applied | Proxy hat Content-Encoding oder Content-Type geändert. |
| 299 | Miscellaneous Persistent Warning | sonstige dauerhafte Warnung |

**Tabelle 2.11** Mögliche Meldungen des »Warning«-Headers

#### WWW-Authenticate

Ein Webserver, der bestimmte Operationen nicht für anonyme Benutzer durchführt, sondern eine Authentifizierung dafür benötigt, liefert die Meldung `401 Unauthorized` zurück. Durch den Header `WWW-Authenticate` gibt der Server an, welches Authentifizierungsschema er erwartet, außerdem kann er einen Authentifizierungsbereich (»Realm«) hinzufügen – dieser wird im Browser als Anmeldeaufforderung angezeigt. Er ermöglicht es dem Client, zu unterscheiden, ob er die Authentifizierungsdaten für diese Ressource bereits kennt oder noch vom Benutzer abfragen muss. Gemäß RFC 2617 gibt es unterschiedliche Authentifizierungsschemata, z. B. `BASIC` (einfache Kombination aus Benutzername und Passwort) oder `Digest` (verschlüsseltes Passwort). Beispiel:

```
WWW-Authenticate: Digest realm="customers_only"
```

## 2.2 Einstieg für Programmierer: ein selbst geschriebener Webserver

In diesem Abschnitt wird die selbst geschriebene Minimalversion eines Webservers vorgestellt. Das Programm ist in Perl geschrieben; diese Sprache ist ideal zum Programmieren von Netzwerkanwendungen geeignet. Als Erstes werden die Projektanforderungen formuliert. Es folgen die Implementierungsdetails, anschließend der vollständige, kommentierte Quellcode. Zum Schluss erhalten Sie noch eine Benutzerdokumentation – zu dieser können Sie vorblättern, wenn Sie den Server ausprobieren möchten, aber wenig Programmiererfahrung besitzen.

### 2.2.1 Projektanforderungen

Schematisch betrachtet soll der Arbeitsablauf des Webservers folgendermaßen aussehen:

1. Einlesen der Konfigurationsdaten von der Kommandozeile: TCP-Port (Standardwert 8000), Website-Verzeichnis (Standard ist `htdocs` unterhalb des Verzeichnisses, in dem sich das Server-Skript befindet) und Name der Indexdatei (Vorgabewert: `index.html`)
2. Am eingestellten TCP-Server-Port auf eingehende Verbindungen lauschen.
3. Verarbeiten einer eintreffenden Verbindung; gleichzeitiges Warten auf weitere Verbindungsanfragen. Dies geschieht durch Forking: Ein neu erzeugter Child-Prozess kümmert sich um die Verbindung, der Parent-Prozess lauscht weiter.
4. Auswerten der HTTP-Anfrage und Umwandeln des URL-Pfades in einen Pfad im lokalen Dateisystem
5. Ermitteln, ob die Anfrage beantwortet werden kann:
   - Wurde die HTTP-Methode `GET` verwendet (ansonsten den Status `501 Not Implemented` zurückgeben)?
   - Existiert die angeforderte Datei (andernfalls `404 Not Found` ausgeben)?
   - Wenn beide Kriterien zutreffen, den Status `200 OK` festlegen und die gewünschte Datei an den Client ausliefern.

Einen verständlicheren Überblick über die Funktionsweise des Skripts gibt Ihnen das Flussdiagramm in Abbildung 2.1.

### 2.2.2 Implementierungsdetails

Wie bei Perl-Skripten üblich, stehen am Anfang einige `use`-Anweisungen zum Import diverser Module. Als Erstes benötigt der Server natürlich Zugriff auf das TCP/IP-Netzwerk. Unter UNIX wird dieser Zugriff seit jeher über die Berkeley-Socket-API bereitgestellt. Das Perl-Standardmodul `IO::Socket` stellt eine praktische, objektorientierte Schnittstelle zu den Funktionen dieser API zur Verfügung – und das nicht nur auf UNIX-Systemen, sondern auch unter Windows, wo die Socket-Programmierung eigentlich etwas anders funktioniert. Darüber hinaus erbt das Modul die Eigenschaften und Methoden von `IO::Handle` und stellt so für Sockets dieselben I/O-Funktionen zur Verfügung wie für gewöhnliche Dateien.

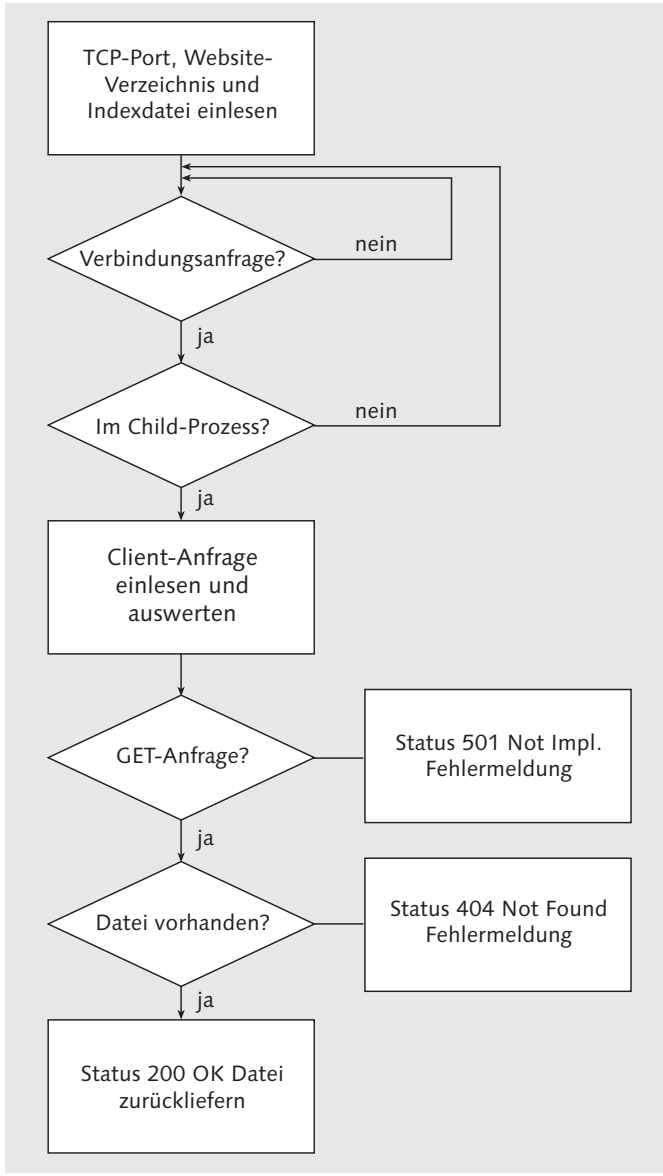



**Abbildung 2.1** Flussdiagramm des einfachen Perl-Webservers

Für den Webserver sollte `IO::Socket` nicht nur mit seinem Standardfunktionsumfang (über das Tag `:DEFAULT`) importiert werden, sondern Sie sollten auch das Tag `:crlf` hinzufügen. Es ermöglicht einen plattformübergreifenden Umgang mit Zeilenumbrüchen. Wie Sie wahrscheinlich wissen, verwenden unterschiedliche Betriebssysteme verschiedene Markierungen für das Zeilenende:

- UNIX benutzt ein einzelnes `LF` (ASCII-Code 10),
- das klassische Mac OS (bis Version 9) verwendete nur `CR` (ASCII-Code 13), während Mac OS X als BSD-Variante der UNIX-Konvention folgt,
- Windows sowie textbasierte TCP/IP-Anwendungsprotokolle (ein solches ist HTTP bekanntermaßen) setzen dagegen die Kombination `CR/LF` ein. `IO::Socket` stellt über das Import-Tag `:crlf` standardisierte Konstanten wie `CRLF` für Zeilenumbrüche zur Verfügung.

Der Importbefehl sieht damit so aus:

```
use IO::Socket qw(:DEFAULT :crlf);
```

Ein weiteres für einen Webserver nützliches Modul ist `HTTP::Date`; es stellt das spezielle Datumsformat für den HTTP-Antwort-Header `Date` zur Verfügung. Der Import erfolgt über den Befehl

```
use HTTP::Date;
```

Schließlich wird die Funktion `uname` des Moduls `POSIX` importiert. Sie gibt wie `uname(1)` Informationen über das Betriebssystem aus – hier wird sie verwendet, um im Server-Header den Namen des Systems auszugeben. Es wird folgende Importanweisung verwendet:

```
use POSIX qw(uname);
```

Nach dem Modulimport müssen einige grundlegende Variablen definiert werden. Viele von ihnen besitzen festgelegte Werte, andere sollen über Kommandozeilenparameter konfigurierbar sein. Die festen Variablen sind folgende:

```
my $os = (uname)[0];               # Betriebssystem
my $sw = "SimpleWebServer";        # Servername
my $ver = "0.11";                  # Version
my $server = "$sw/$ver ($os)";
my %mime_types = (htm => 'text/html', html => 'text/html',
                  css => 'text/css', js => 'text/javascript',
                  txt => 'text/plain', gif => 'image/gif',
                  jpg => 'image/jpeg', png => 'image/png',
                  swf => 'application/x-shockwave-flash');
```

Den Hash mit der Liste der Dateiendungen und MIME-Types können Sie natürlich nach Belieben erweitern, wenn Sie noch andere Arten von Dateien servieren möchten. Zunächst einmal wurden nur die für den normalen Webserver-Betrieb wichtigsten Dateitypen verwendet.

Um das Problem mit dem Zeilenumbruch ein für allemal zu beheben, wird die Perl-Standardvariable `$/` (der *Input Record Separator*) auf `CR/LF` gesetzt:

```
my $crlf = CRLF;
$/ = $crlf;
```

Nun werden die Kommandozeilenparameter eingelesen; die entsprechenden Variablen werden entweder auf die angegebenen Werte gesetzt oder erhalten Standardwerte. Wird als erster Parameter `-h` oder `-v` angegeben, dann wird die Hilfe beziehungsweise Versionsinformation ausgegeben, und das Programm wird sofort beendet. Andernfalls werden Port, Stammverzeichnis und Indexdatei aus der ersten, zweiten beziehungsweise dritten Option gelesen oder auf Standardwerte gesetzt:

```
my $port = $ARGV[0] || 8000;
my $serverdir = $ARGV[1] || "htdocs";
my $dirindex = $ARGV[2] || "index.html";
```

An dieser Stelle wird die Log-Datei mit dem Zugriffstyp `>>` zum Anhängen geöffnet:

```
open LOG, ">>access.log";
```

Die Child-Prozesse, die sich um die einzelnen Client-Verbindungen kümmern, übernehmen dieses offene Datei-Handle automatisch und können ihre jeweiligen Informationen hineinschreiben.

Einer der wichtigsten Schritte besteht darin, ein Socket zu erzeugen, das am gewünschten Server-Port auf eingehende Verbindungen lauscht. Mit `IO::Socket` lässt sich ein lauschendes Socket auf folgende einfache Weise erstellen:

```
my $listen = IO::Socket::INET->new
     (Proto     => 'tcp',
      LocalPort => $port,
      Listen    => SOMAXCONN);
```

`IO::Socket` besitzt die beiden Unterklassen `IO::Socket::INET` beziehungsweise `IO::Socket::Unix` für die beiden grundlegenden Socket-Typen `AF_INET` (Netzwerk-Socket) und `AF_UNIX` (UNIX-Domain-Socket). Selbstverständlich benötigt ein Netzwerkserver Ersteres. Die Konstante `SOMAXCONN` steht für die maximale Anzahl von Verbindungsanfragen, die vom Betriebssystem aus in einer Warteschlange gepuffert werden dürfen.

Nun folgt die Hauptschleife des Parent-Prozesses, die sogenannte *Accept-Schleife* (nach der gleichnamigen Socket-Funktion). Sie wird als Endlosschleife nach dem Schema `while (1) { ... }` realisiert. Falls im aktuellen Durchgang keine Verbindung eintrifft, wird mit `next()` der nächste Durchlauf eingeleitet. Andernfalls wird zunächst die Funktion `accept()` aufgerufen, die ein Socket-Paar zwischen dem Server und dem entfernten Client erzeugt. Danach erzeugt `fork()` einen

Child-Prozess für die Verarbeitung der neuen Verbindung. Ist der Rückgabewert von `fork()` in der anschließenden Fallentscheidung 0, dann handelt es sich um den Child-Prozess. In diesem Fall wird deshalb die Subroutine `handle_conn()` aufgerufen, die sich um die Client-Verbindung kümmert. Die Accept-Schleife sieht demnach so aus:

```
while (1) {
   # Weiter, falls keine Anfrage eintrifft
   next unless my $conn = $listen->accept();

   # Anfrage eingetroffen - Child-Prozess erzeugen
   my $child = fork;
   if ($child == 0) {
      # Im Child-Prozess Verbindung aufbauen
      # und Anfrage beantworten
      handle_conn ($conn);
   }
}
```

Die Subroutine `handle_conn()`, die in den Child-Prozessen Verbindungen verarbeitet, nimmt als Argument eine Referenz auf das Verbindungs-Socket entgegen – sie wird mithilfe von `shift()` aus der Argumentliste der Subroutine (`@_`) ausgelesen. Damit es auch auf einem Windows-Rechner keine Probleme mit der automatischen Umwandlung von Zeilenumbrüchen gibt, wird das Socket mittels `binmode()` in den Binärmodus versetzt, der Zeilenumbrüche in Ruhe lässt. Anschließend wird die `IO::Socket`-Methode `peerhost()` verwendet, um die Adresse des entfernten Hosts für die Log-Datei zu ermitteln. Die entsprechenden Zeilen des Skripts sehen so aus:

```
sub handle_conn
{
   my $sock = shift;
   binmode $sock;
   my $peer_addr = $sock->peerhost;
```

Nun muss die Client-HTTP-Anfrage entgegengenommen werden. Sie wird mittels `<>` aus dem Socket ausgelesen. Für diesen einfachen Server genügt die erste Zeile; weitere HTTP-Header der Anfrage werden nicht beachtet. Zunächst werden Zeilenumbrüche und sonstiger Whitespace am Ende der Zeile entfernt – aus Sicherheitsgründen wird statt eines einfachen `chomp()` der Ersetzungsbefehl `s/\s*$//` (jeglichen Whitespace am Zeilenende entfernen) verwendet. Anschließend wird die HTTP-Befehlszeile mittels `split()` an den trennenden Leerzeichen in ihre drei Bestandteile `$method`, `$uri` und `$http` (HTTP-Methode, Pfad der angeforderten Ressource und HTTP-Version) zerlegt:

```
my $request = <$sock>;
$request =~ s/\s*$//;
my ($method, $uri, $http) = split (/\s+/, $request);
```

Da die URI (*Uniform Resource Identifier*) aus der Anfrage relativ zum Stammverzeichnis der Website formuliert ist, muss ihr nun der Pfad dieses Verzeichnisses vorangestellt werden, damit der Server die angeforderte Datei in seinem lokalen Dateisystem finden kann. Dazu wird am Anfang der URI (RegExp-Kürzel ^) der Wert von `$serverdir` mit anschließendem / hinzugefügt. Falls das erste Zeichen der Anfrage bereits ein / ist (in der Regel dürfte dies der Fall sein), wird er dabei ersetzt:

```
$uri =~ s|^/?|$serverdir/|;
```

Mithilfe der Datei-Testoperation `-d` wird nun geprüft, ob es sich bei der URI um ein Verzeichnis handelt. Ist dies der Fall, wird der festgelegte Startseitenname angehängt:

```
if (-d $uri) {
   $uri =~ s|/?$|/$dirindex|;
}
```

Für die Server-Antwort und die Log-Datei sind zwei unterschiedliche Datums- und Uhrzeitformate nötig. Während sich das HTTP-Datumsformat für den `Date`-Header komfortabel über das Modul `HTTP::Date` erstellen lässt, ist für die spezielle Schreibweise in der Log-Datei ein wenig Handarbeit erforderlich, wie Sie im Folgenden sehen können.

1. Zunächst einmal werden die Kürzel der Monatsnamen in einer Liste gespeichert:

   ```
   my @months = qw(Jan Feb Mar Apr May Jun Jul Aug
        Sep Oct Nov Dec);
   ```

2. Nun wird (als Grundlage für beide Datumsformate) die Systemzeit ausgelesen; die Funktion `time()` liefert die Sekunden seit EPOCH[3]:

   ```
   my $tm = time();
   ```

3. Da die Zeit in der Log-Datei als GMT mit Differenz zur lokalen Zeitzone angegeben wird, müssen nun beide Zeitangaben ermittelt werden. Dazu dienen die Funktionen `localtime()` und `gmtime()`, die jeweils eine Liste mit Sekunden, Minuten, Stunden, Tag, Monat, Jahr (und weiteren Angaben, die hier nicht benötigt werden) in Ortszeit beziehungsweise GMT zurückgeben:

3 Idealisiertes UNIX-Einführungsdatum: 01.01.1970, 00:00 Uhr UTC

```
my ($lsec, $lmin, $lhour, $lday, $lmon, $lyear) =
     localtime ($tm);
my ($gsec, $gmin, $ghour, $gday, $gmon, $gyear) =
     gmtime ($tm);
```

4. Die Differenz zwischen den beiden Stundenangaben wird als Nächstes gespeichert:

```
my $diff = $lhour - $ghour;
```

5. Nun kann der einfache Teil von Datum und Uhrzeit für die Log-Datei zusammengebaut werden (Schema: `22/08/2008:09:55:07`):

```
my $logdate = "$gday/";
$logdate .= $months[$gmon];
$logdate .= "/";
$logdate .= $gyear + 1900;
$logdate .= ":";
$logdate .= ($ghour < 10) ? "0$ghour:" : "$ghour:";
$logdate .= ($gmin < 10) ? "0$gmin:" : "$gmin:";
$logdate .= ($gsec < 10) ? "0$gsec" : "$gsec";
```

6. Je nachdem, ob die Zeitdifferenz positiv oder negativ ist, wird ein `-` oder `+` angefügt:

```
$logdate .= ($diff < 0) ? " -" : " +";
```

7. Von der eigentlichen Differenz kann daraufhin mittels `abs()` der Absolutwert ermittelt werden; anschließend wird diese Differenz mit zwei weiteren Nullen an die Zeitangabe angehängt:

```
$diff = abs $diff;
$logdate .= ($diff < 10) ? "0${diff}00" : "${diff}00";
```

Im Gegensatz zu diesen recht komplexen Arbeitsschritten lassen sich Datum und Uhrzeit für das HTTP-Datum in einer einzigen Zeile formatieren:

```
my $pagedate = time2str ($tm);
```

Der Rest der Subroutine dient dazu, die Antwort für den Client zu erstellen und über das Socket zurückzusenden. Der Server unterscheidet dabei die folgenden drei Fälle:

- Wurde in der Anfrage eine andere HTTP-Methode als `GET` verwendet, dann antwortet der Server, dass er diese nicht versteht. Da dies seine eigene »Schuld« ist, muss er einen 5xx-Statuscode verwenden – in diesem Fall `501 Not Implemented`. Der Einfachheit halber geschieht dies auch dann, wenn der Cli-

ent eine Fantasiemethode verwenden sollte, die es im HTTP-Standard gar nicht gibt.

- Wenn die angeforderte URI nicht vorhanden ist, gibt der Server eine Meldung mit dem Statuscode `404 Not Found` zurück.
- Falls sich die beiden vorgenannten Fehlerzustände ausschließen ließen, ist alles in Ordnung. Der Server antwortet mit `200 OK` und liefert im Body der Antwort die angeforderte Datei mit.

Schematisch wird zur Unterscheidung dieser drei Antwortzustände die folgende verschachtelte Fallentscheidung verwendet:

```
if ($method ne 'GET') {
   # 501 Not Implemented ...
} elsif (!(-e $uri)) {
   # 404 Not Found ...
} else {
   # 200 OK ...
}
```

Interessant ist hier die Datei-Testoperation `-e`, die `true` zurückgibt, wenn die überprüfte Datei existiert.

Da die drei Server-Antworten schematisch identisch funktionieren, genügt es im Folgenden, die komplexeste von ihnen – die Erfolgsmeldung `200 OK` mit Ausgabe der angeforderten Datei – zu betrachten.

Zunächst einmal wird anhand des Hashs `%mime_types` der Typ der Datei aus ihrer Dateiendung ermittelt. Wenn die Endung der angeforderten Datei nicht mit einem MIME-Type verknüpft ist, wird als Standardtyp `text/plain` zurückgegeben:

```
my $mime_type = "text/plain";
```

Falls über diesen Server auch Binärdateien zum Download angeboten werden sollen, empfiehlt es sich, stattdessen `application/octet-stream` als Standard zu benutzen. Andernfalls würde der Browser nämlich einfach den Inhalt der Binärdateien als ASCII-Zeichensalat anzeigen.

Nun werden die Schlüssel von `%mime_types` in einer Schleife nacheinander mit der Endung der angeforderten Datei verglichen. Wenn die Endung in der Liste gefunden wird, setzt das Programm den MIME-Type auf den entsprechenden Wert:

```
foreach (keys %mime_types) {
   if ($uri =~ /\.$_$/) {
```

```
      $mime_type = $mime_types{$_};
   }
}
```

Für den HTTP-Header `Content-Length` wird jetzt die Dateigröße ermittelt. Dies geschieht über den Datei-Testoperator `-s`:

```
$size = -s $uri;
```

Als Nächstes wird die angeforderte Datei zum Lesen geöffnet und aus den oben erwähnten Gründen ebenfalls in den Binärmodus versetzt:

```
open (IN, "<$uri") ||
     warn ("Kann Datei $uri nicht oeffnen!");
binmode IN;
```

Mithilfe der Funktion `read()` wird die Datei nun eingelesen; jeder einzelne Block wird an `$body` angehängt. Mit der Blockgröße von 16.384 Byte können Sie experimentieren, wenn Sie Lust haben – möglicherweise lässt sich die Performance des Servers durch eine Änderung dieser Größe steigern. Anschließend wird die Datei wieder geschlossen:

```
my $buffer;
while (read (IN, $buffer, 16384)) {
   $body .= $buffer;
}
close (IN);
```

Dann werden Statuscode und -meldung festgelegt:

```
$statuscode = 200;
$statusmsg = "OK";
```

Das Array `@headers` enthält nun sämtliche Header-Felder:

```
push (@headers, "Date: $pagedate");
push (@headers, "Server: $server");
push (@headers, "Connection: close");
push (@headers, "Content-Type: $mime_type");
push (@headers, "Content-Length: $size");
```

Die restlichen Befehle stehen hinter der schließenden Klammer der verschachtelten `if`-Fallentscheidung. Sie werden also für alle drei Arten von Server-Antworten ausgeführt.

Die Ausgabe in die Log-Datei wird im Common Log Format (CLF) durchgeführt – dieses Format kann auch von Apache verwendet werden (siehe Kapitel 11, »Logging«):

```
print LOG "$peer_addr - - [$logdate] \"$request\"
     $statuscode $size\n";
```

Die einzelnen Werte in einer Zeile sind (getrennt durch Leerzeichen) folgende:

- IP-Adresse des entfernten Hosts
- RFC-1413-Identität (hier ein »–«, da dieser Server keine Identitätsprüfung unterstützt)
- Benutzername (ebenfalls nur ein »–«, weil Authentifizierungsfunktionen ebenfalls nicht implementiert sind)
- Datum und Uhrzeit im weiter oben hergestellten komplexen Format; komplett in eckigen Klammern
- die komplette Client-Anfrage in Anführungszeichen
- der Statuscode der Server-Antwort
- die Größe des Antwort-Bodys

Nun folgt die wichtigste Aufgabe des Servers: die Ausgabe der Antwort an den Client. Zunächst wird die Einleitungszeile generiert, bestehend aus der HTTP-Version (unverändert aus der Anfrage übernommen), dem Statuscode und der Statusmeldung:

```
print $sock "$http $statuscode $statusmsg$crlf";
```

Als Nächstes werden in einer Schleife die einzelnen Header-Felder zeilenweise ausgegeben:

```
foreach my $header (@headers) {
   print $sock "$header$crlf";
}
```

Eine Leerzeile trennt die Header vom Body der Antwort:

```
print $sock "$crlf";
```

Zu guter Letzt wird der Body selbst ausgegeben:

```
print $sock "$body";
```

Der Ordnung halber wird jetzt das Socket geschlossen; danach wird der Child-Prozess sauber beendet:

```
close $sock;
exit (0);
```

### 2.2.3 Der komplette Quellcode

In Listing 2.1 sehen Sie den Quellcode des Servers. Sie brauchen ihn aber nicht selbst abzutippen. Sie finden ihn im Verzeichnis `listings` auf der DVD zum Buch oder können ihn – zusammen mit allen anderen Listings und Beispielen zu diesem Buch – aus dem Web herunterladen. Die Adresse lautet: *buecher.lingoworld.de/apache2/listings.html*.

```
#!/usr/bin/perl -w

use strict;
use IO::Socket qw(:DEFAULT :crlf); # TCP/IP-Sockets
use HTTP::Date;                    # HTTP-Header-Datum
use POSIX qw(uname);               # Betriebssystem ermitteln

# --------------------------------

# Wichtige Konfigurationsvariablen; nicht änderbar
my $os = (uname)[0];               # Betriebssystem
my $sw = "SimpleWebServer";        # Servername
my $ver = "0.11";                  # Version
my $server = "$sw/$ver ($os)";
my %mime_types = (htm => 'text/html', html => 'text/html',
                  css => 'text/css', js => 'text/javascript',
                  txt => 'text/plain', gif => 'image/gif',
                  jpg => 'image/jpeg', png => 'image/png',
                  swf => 'application/x-shockwave-flash');

# Zeilenende-Kompatibilität herstellen
my $crlf = CRLF;
$/ = $crlf;                        # Zeilenende auf CRLF

# --------------------------------

# Konfiguration von der Kommandozeile lesen
# oder auf Standardwerte setzen

if ($ARGV[0] && $ARGV[0] eq "-h") {
   # Nur Hilfe abgefragt
   print << "ENDHELP";
$sw/$ver ($os)

Usage: $0 -h | -v | [PORT [SERVER_DIRECTORY [DIRECTORY_INDEX]]]

-h                Print this help message and exit
-v                Print version information and exit
```

```
PORT              Listening Port (Default: 8000)
SERVER_DIRECTORY  Public Web Server Directory (Default: ./htdocs)
DIRECTORY_INDEX   Name of Index File (Default: index.html)
ENDHELP
   exit (0);
} elsif ($ARGV[0] && $ARGV[0] eq "-v") {
   # Nur Version abgefragt
   print << "ENDVERSION";
$sw/$ver ($os)

Written by Sascha Kersken (sk\@lingoworld.de)
For the Galileo Press book 'Apache 2', 4th Ed. (2011)
http://buecher.lingoworld.de/apache2

This program may be used, distributed and modified
under terms of the GNU General Public License, Version 2.0
(See file COPYING in this directory)
ENDVERSION
   exit (0);
}

# Normalfall: Server soll ausgeführt werden

# $port: Server-TCP-Port
# 1. Kommandozeilenargument
# Standard: 8000
my $port = $ARGV[0] || 8000;

# $serverdir: Server-Wurzelverzeichnis
# 2. Kommandozeilenargument
# Standard: Unterverzeichnis "htdocs" des aktuellen
# Verzeichnisses
my $serverdir = $ARGV[1] || "htdocs";

# $dirindex: Name der Indexdatei in jedem Verzeichnis
# 3. Kommandozeilenargument
# Standard: index.html
my $dirindex = $ARGV[2] || "index.html";

# --------------------------------

# Logdatei initialisieren
open LOG, ">>access.log";

# Lauschendes Socket erzeugen
```

```
my $listen = IO::Socket::INET->new
     (Proto     => 'tcp',
      LocalPort => $port,
      Listen    => SOMAXCONN);

print "$server laeuft auf Port $port ...\n\n";
print "Server-Verzeichnis: $serverdir\n";
print "Indexdokument:     $dirindex\n";

# Accept-Schleife
while (1) {
   # Weiter, falls keine Anfrage eintrifft
   next unless my $conn = $listen->accept();

   # Anfrage eingetroffen - Child-Prozess erzeugen
   my $child = fork;
   if ($child == 0) {
      # Im Child-Prozess Verbindung aufbauen
      # und Anfrage beantworten
      handle_conn ($conn);
   }
}

sub handle_conn
{
   my $sock = shift;
   binmode $sock;
   my $peer_addr = $sock->peerhost;

   # Client-Anfrage lesen
   my $request = <$sock>;
   $request =~ s/\s*$//;
   my ($method, $uri, $http) = split (/\s+/, $request);

   # URI auf das Server-Verzeichnis umsetzen
   $uri =~ s|^/?|$serverdir/|;

   # Wenn URI ein Verzeichnis ist, Indexdatei verwenden
   if (-d $uri) {
      $uri =~ s|/?$|/$dirindex|;
   }

   # Die verschiedenen Datumsformate für Logdatei
   # und Server-Antwort erzeugen
   my @months = qw(Jan Feb Mar Apr May Jun Jul Aug
```

```
         Sep Oct Nov Dec);
   my $tm = time();
   my ($lsec, $lmin, $lhour, $lday, $lmon, $lyear) =
         localtime ($tm);
   my ($gsec, $gmin, $ghour, $gday, $gmon, $gyear) =
         gmtime ($tm);
   my $diff = $lhour - $ghour;
   my $logdate = "$gday/";
   $logdate .= $months[$gmon];
   $logdate .= "/";
   $logdate .= $gyear + 1900;
   $logdate .= ":";
   $logdate .= ($ghour < 10) ? "0$ghour:" : "$ghour:";
   $logdate .= ($gmin < 10) ? "0$gmin:" : "$gmin:";
   $logdate .= ($gsec < 10) ? "0$gsec" : "$gsec";
   $logdate .= ($diff < 0) ? " -" : " +";
   $diff = abs $diff;
   $logdate .= ($diff < 10) ? "0${diff}00" : "${diff}00";
   my $pagedate = time2str ($tm);

   # Weitere Elemente der Server-Antwort
   my ($statuscode, $statusmsg, @headers, $body, $size);

   if ($method ne 'GET') {
      # Es werden NUR GET-Anfragen beantwortet
      $statuscode = 501;
      $statusmsg = "Not Implemented";
      $body = << "END_NI_BODY";
<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//Final//EN\">
<html>
<head>
<title>501 Not Implemented</title>
</head>
<body>
The request method $method is not supported by this server.<br>
Server: $server<br>
$pagedate
</body>
</html>
END_NI_BODY
$size = length ($body);
      push (@headers, "Allow: GET");
      push (@headers, "Date: $pagedate");
      push (@headers, "Server: $server");
      push (@headers, "Connection: close");
```

```
      push (@headers, "Content-Type: text/html");
      push (@headers, "Content-Length: $size");
   } elsif (!(-e $uri)) {
      # Angeforderte Datei existiert nicht
      $statuscode = 404;
      $statusmsg = "Not Found";
      $body = << "END_NF_BODY";
<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//Final//EN\">
<html>
<head>
<title>404 Not Found</title>
</head>
<body>
The requested resource $uri could not be found on this
server.<br>
Server: $server<br>
$pagedate
</body>
</html>
END_NF_BODY
$size = length ($body);
      push (@headers, "Date: $pagedate");
      push (@headers, "Server: $server");
      push (@headers, "Connection: close");
      push (@headers, "Content-Type: text/html");
      push (@headers, "Content-Length: $size");
   } else {
      # Datei existiert => lesen und zurückgeben
      # Wird der MIME-Type der angeforderten Datei unterstützt?
      # Standardtyp: text/plain
      my $mime_type = "text/plain";
      foreach (keys %mime_types) {
         if ($uri =~ /\.$_$/) {
            $mime_type = $mime_types{$_};
         }
      }
      # Dateigröße für Content-length ermitteln
      $size = -s $uri;
      # Angeforderte Datei zeilenweise in @body einlesen
      open (IN, "<$uri") ||
           warn ("Kann Datei $uri nicht oeffnen!");
      binmode IN;
      my $buffer;
      while (read (IN, $buffer, 16384)) {
         $body .= $buffer;
      }
```

```
        close (IN);
        $statuscode = 200;
        $statusmsg = "OK";
        push (@headers, "Date: $pagedate");
        push (@headers, "Server: $server");
        push (@headers, "Connection: close");
        push (@headers, "Content-Type: $mime_type");
        push (@headers, "Content-Length: $size");
    }

    # Ausgabe in die Logdatei
    print LOG "$peer_addr - - [$logdate] \"$request\"
         $statuscode $size\n";

    # Ausgabe der Daten an den Client
    print $sock "$http $statuscode $statusmsg$crlf";
    foreach my $header (@headers) {
        print $sock "$header$crlf";
    }
    print $sock "$crlf";
    print $sock "$body";
    close $sock;
    exit (0);
}
```

**Listing 2.1** sws.pl – der eigene Webserver

### 2.2.4 Benutzerdokumentation

Dieses kleine Skript soll (und kann) selbstverständlich keine Konkurrenz zu einem ausgewachsenen Webserver wie Apache 2 sein, sondern soll vielmehr die Grundfunktionalität eines Webservers verdeutlichen. Der Perl-Webserver besitzt die folgenden grundlegenden Eigenschaften:

- Es handelt sich um einen *Forking*-Webserver: Erst wenn eine Anfrage eintrifft, wird ein Child-Prozess erzeugt, der diese behandelt. Der Forking-Server ist die klassische und einfachste Form eines Servers, der mehrere Clients zur gleichen Zeit bedienen kann. Sie werden im nächsten Kapitel andere Modelle kennenlernen, die besonders zur Behandlung sehr vieler Client-Anfragen besser geeignet sind.
- Der Server unterstützt nur eine einzige Methode, nämlich `GET`. Das bedeutet, dass er lediglich statische Dokumente ausliefert, deren Adresse Sie in das Adressfeld eines Browsers eingeben. Dies ist die Grundfunktion eines Webservers, sodass Sie dessen Arbeitsprinzip an diesem einfachen Beispiel nach-

vollziehen können. Es gibt natürlich auch keinerlei Unterstützung für serverseitige Programmierung wie CGI, PHP oder Ähnliches.

- Es werden nur wenige Dateitypen unterschieden, nämlich die wichtigsten, die im Zusammenhang mit Websites geliefert werden: HTML-Dokumente, Bilder der Typen GIF, JPEG und PNG sowie Flash-Filme. Für alle anderen Dateitypen gibt der Server den MIME-Type `text/plain` zurück.

Der Server wird üblicherweise über die Kommandozeile gestartet. Dabei können Sie drei Parameter angeben, die ansonsten Standardwerte erhalten. Für einen »normalen« Start ohne Sonderkonfiguration müssen Sie unter UNIX Folgendes eingeben:

```
# ./sws.pl
```

Unter Windows ist dagegen folgende Eingabe erforderlich:

```
> perl sws.pl
```

Wie Sie Perl für Windows installieren können, ist in Kapitel 15, »Technologien zur Webprogrammierung«, beschrieben.

Die drei Parameter sind folgende:

- **Server-Port** (Standard 8000): Beachten Sie, dass Sie unter UNIX root-Rechte benötigen, um einen Port unter 1024 zu verwenden – einschließlich dem Standard-HTTP-Port 80. Allerdings sollten Sie diesen Server aus Sicherheitsgründen *niemals* einsetzen, wenn Ihre Internetverbindung offen ist.
- **Wurzelverzeichnis** (Standard `htdocs` unterhalb des Verzeichnisses, in dem sich der Server befindet): Dieser Pfad gibt das Verzeichnis an, das die freigegebenen Dateien enthält. Wenn Sie der Angabe keinen `/` voranstellen, gilt sie relativ zum aktuellen Verzeichnis. Beachten Sie, dass Sie auch unter Windows den `/` (und keinen `\`) als Trennzeichen einsetzen müssen.
- **Indexdokument** (Standard `index.html`): Das ist der Name eines Dokuments, das standardmäßig ausgeliefert wird, wenn in der Anfrage ein Pfad ohne expliziten Dateinamen angegeben wurde.

Der Server legt in seinem Verzeichnis eine einfache Log-Datei namens `access.log` an, deren Format dem auch von Apache unterstützten CLF (Common Log Format) entspricht.

Wenn Sie den Server also auf Port 8888 im Verzeichnis `/home/user/mydocs/ html` mit dem Indexdokument `default.htm` starten möchten, müssen Sie Folgendes eingeben:

```
# ./sws.pl 8888 /home/user/mydocs/html default.htm
```

Da die Reihenfolge der drei Parameter verbindlich ist, müssen Sie zur Änderung des zweiten oder dritten auch die vorigen explizit angeben – selbst dann, wenn Sie für diese den Standardwert beibehalten möchten.

Nachdem Sie den Server gestartet haben, können Sie einen beliebigen Browser auf Ihrem Rechner (oder auf einem anderen Rechner in Ihrem LAN) öffnen und ein Dokument anfordern, das sich im freigegebenen Verzeichnis des Webservers befindet. Angenommen, der Server läuft unter seinem Standard-Port 8000 auf demselben Rechner wie der Browser. Wenn Sie in diesem Fall die Startseite (standardmäßig `index.html`) im Stammordner des Servers (normalerweise `htdocs` innerhalb des Ordners, in dem das Server-Skript liegt) aufrufen möchten, müssen Sie Folgendes in die Adressleiste des Browsers eingeben:

```
http://127.0.0.1:8000
```

Beachten Sie, dass die Angabe `http://`, die bei »normalen« Internetadressen weggelassen werden kann, in vielen Browsern eingegeben werden muss, wenn ein Server nicht auf dem TCP-Port 80 läuft.

## 2.3 Zusammenfassung

In diesem Kapitel wurde Ihnen zunächst das TCP/IP-Anwendungsprotokoll HTTP entsprechend ausführlich vorgestellt. Sie haben damit hinter die Kulissen geschaut und gesehen, wie die Nachrichten aussehen, die Apache (oder andere Webserver), Browser und Proxys miteinander austauschen. Das Verständnis dieser Informationen ermöglicht Ihnen eine genaue Analyse von Kommunikationsfehlern und anderen Problemen mit Ihrem Apache-Webserver. Im Verlauf dieses Buches wird noch oft auf Informationen aus diesem Kapitel zurückgegriffen werden; unter anderem werden Sie erfahren, wie Sie Apache 2 konfigurieren können, um seine Antwort-Header-Felder mit bestimmten Werten auszufüllen oder auf spezielle Anfrage-Header zu reagieren.

Der kleine, selbst geschriebene Webserver, der Ihnen im zweiten Abschnitt präsentiert wurde, kann Ihnen den Einstieg in die Materie erleichtern, wenn Sie mehr Erfahrung mit dem Programmieren als mit der System- und Netzwerkadministration haben. Wenn Sie Perl-Grundkenntnisse besitzen, lesen Sie den Sourcecode und probieren Sie andernfalls aus, wie der Server auf verschiedene Testfälle reagiert. Er ist so geschrieben, dass er den HTTP/1.1-Standard einhält (wenn auch nur einen winzigen Ausschnitt davon), und sollte daher mit jedem beliebigen Browser problemlos zusammenarbeiten.

*»Gut Werkzeug, gute Arbeit.«*
*– Deutsches Sprichwort*

# 3 Apache 2 im Überblick

Nach den diversen Vorabinformationen in den ersten beiden Kapiteln erhalten Sie hier einen Überblick über den Webserver Apache 2. Der erste Abschnitt enthält einen kurzen historischen Abriss über Apache und seine Vorläufer, über die Apache Software Foundation (ASF), die seine Weiterentwicklung koordiniert, sowie einen Vergleich mit einigen bekannten Konkurrenzprodukten. Im zweiten Abschnitt geht es dagegen technischer zu: Sie erhalten in kurzer Form die wichtigsten Informationen über die Funktionen und Eigenschaften des Servers.

## 3.1 Einführung

Der Apache-HTTP-Server oder Apache-Webserver ist ein Gemeinschaftsprodukt von zahlreichen Softwareentwicklern, die ihre Arbeit über öffentliche Mailinglisten und verteilte Subversion-Repositorys koordinieren. Es handelt sich um ein – wenn nicht sogar um das – Prestigeprodukt aus der Welt der freien Software beziehungsweise Open-Source-Software: Kaum ein anderes Programm aus dieser Szene ist so beliebt und verbreitet wie Apache. Selbst Microsoft-Gründer Bill Gates konnte bereits vor Jahren nicht anders, als es zuzugeben: »Apache, free software, has been used by large companies for a long time.« (c't-Interview, Ausgabe 4/1999).

Da die Quellcodes des Programms öffentlich verfügbar sind, kann jeder (ausreichende C-Kenntnisse vorausgesetzt) den Server genau auf seine eigenen Bedürfnisse zuschneiden. Besonders geglückte Lösungen für spezifische Probleme können Sie natürlich auch der Apache-Nutzergemeinschaft zukommen lassen.

Wie Sie im nächsten Kapitel sehen werden, können Sie aber sogar ohne Programmierkenntnisse von dieser Flexibilität profitieren: Wenn Sie den Server selbst kompilieren, können Sie zuvor zahlreiche Einstellungen vornehmen, um ihn an Ihre Betriebssystemumgebung und an Ihre speziellen Wünsche anzupassen. Und mit dem laufenden Versionsfortschritt steigt sogar der Anteil von Optionen, die sich durch Konfiguration des bereits installierten Servers modifizieren lassen.

### 3.1.1 Entstehungsgeschichte des Apache-Webservers

Der erste Webserver überhaupt wurde als Referenzimplementierung der WWW-Idee von Tim Berners-Lee und seinem Team am europäischen Kernforschungszentrum geschrieben. Aus diesem Grund wurde das Programm später zur Unterscheidung von anderen Webservern als CERN httpd bezeichnet – »httpd« steht für *Hypertext Transfer Protocol Daemon*.

Sehr bald erhielt der CERN-Webserver allerdings Konkurrenz aus den USA: Am National Center for Supercomputing Applications (NCSA) der University of Illinois wurde neben Mosaic – dem Urahn aller modernen GUI-basierten Webbrowser – auch ein erweiterter HTTP-Server programmiert, eben der NCSA HTTPd. Als der Leiter dieses Projekts, Rob McCool, das NCSA 1994 verließ, wurde dieser Webserver nicht mehr weiter gepflegt. Aber einige der Programmierer, die daran mitgearbeitet hatten, blieben per E-Mail in Kontakt. Sie bildeten die ursprüngliche Apache Group:

- Brian Behlendorf
- Roy T. Fielding
- Rob Hartill
- David Robinson
- Cliff Skolnick
- Randy Terbush
- Robert S. Thau
- Andrew Wilson

Weitere Beiträge zur ursprünglichen Version von Apache haben folgende Entwickler geleistet:

- Eric Hagberg
- Frank Peters
- Nicolas Pioch

Im Laufe der Jahre haben Hunderte von Personen an der Weiterentwicklung des Servers mitgearbeitet. Eine aktuelle Liste der Mitglieder des HTTP Server Projects finden Sie im Web unter *http://httpd.apache.org/contributors/*.

Auf diese Weise entwickelte sich aus der Version 1.3 des NCSA HTTPd und zahlreichen von diesen Entwicklern geschriebenen Patches (»Flicken«) »a PATCHy web server« (ein Webserver mit Patches). Laut Apache-FAQ (*http://httpd.apache.org/docs/misc/FAQ.html*) ist dies allerdings nicht der eigentliche Grund für die Namenswahl (auch wenn er dort verdächtigerweise erwähnt wird): »The

name ›Apache‹ was chosen from respect for the Native American Indian tribe of Apache (Indé), well-known for their superior skills in warfare strategy and their inexhaustible endurance.«[1]

Das erste offizielle Release des Apache-Webservers (1.0) wurde im Dezember 1995 veröffentlicht. Die ursprüngliche Codebasis (1.x) des Webservers wurde bis zur Version 1.3 weitergeführt. Inzwischen wird sie nicht mehr weiterentwickelt; die letzte Unterversion 1.3.42 erschien 2010. Daher geht es in diesem Buch ausschließlich um die von Grund auf weiterentwickelte Version 2 mit ihrem derzeit stabilen Zweig 2.2 und den kurz vor der endgültigen Veröffentlichung stehenden neuen Zweig 2.4. Dennoch gelten viele Themen auch für 1.3; falls Sie für ein Altsystem noch Informationen darüber benötigen, können Sie unter *http://httpd.apache.org/docs/1.3/* die (nicht mehr gepflegte) Dokumentation lesen. In Anhang A des vorliegenden Buches finden Sie Informationen zu Apache 2.0, dem ältesten Zweig von Version 2, der ebenfalls nicht mehr aktiv weiterentwickelt wird.

Bereits ein Jahr nach seiner Einführung überholte Apache den NCSA HTTPd als meistgenutzten Server im Internet. Dies ist bis heute unverändert geblieben: Im Oktober 2005 erreichte der Einsatz des Apache-Webservers mit einem Anteil von über 69 Prozent aller weltweit erreichbaren Websites (genauer gesagt Domains) seinen Höhepunkt.

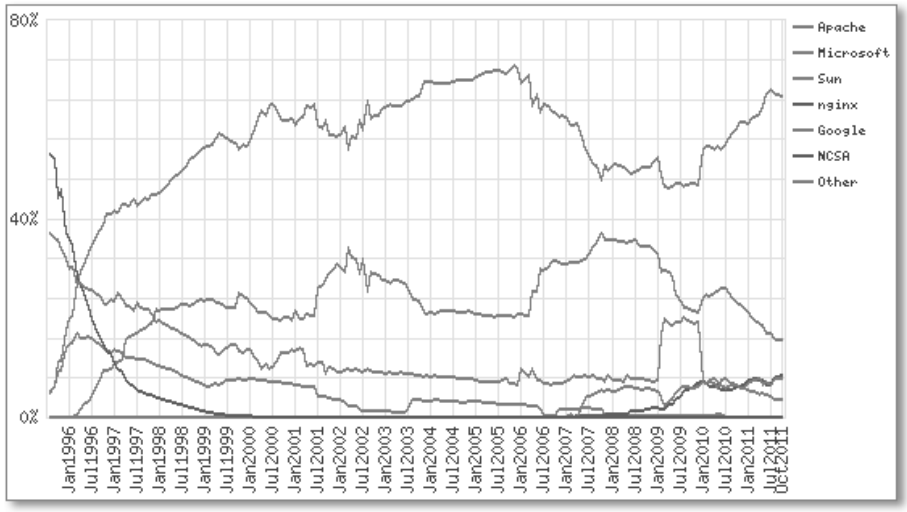



**Abbildung 3.1** Entwicklung der Webserver-Marktanteile seit 1995. Quelle: Netcraft

1 Übersetzung: »Der Name ›Apache‹ wurde aus Respekt vor dem amerikanischen Indianerstamm der Apachen (Indé) gewählt, die für ihre außergewöhnlichen Fähigkeiten in der Strategie der Kriegsführung und ihre unerschöpfliche Ausdauer berühmt waren.«

Im Laufe der Jahre 2006 und 2007 entwickelte sich der Marktanteil von Apache zugunsten des Webservers Microsoft Internet Information Services (IIS) rückläufig. Dies geschah insbesondere, weil einige große Domain-Parking-Dienste wie *godaddy.com* auf IIS umstiegen, wobei aber Millionen geparkter Domains keine aktiv genutzten Websites sind. Apache hat sich dennoch stets über 50 Prozent gehalten, und in letzter Zeit ist wieder ein leicht steigender Trend auszumachen. IIS nimmt mit etwa 16 Prozent den zweiten Platz ein, während leichtgewichtige Webserver-Alternativen wie lighttpd oder nginx an Verbreitung gewinnen. Abbildung 3.1 zeigt die Entwicklung der Webserver-Anteile, seit Netcraft (*http://www.netcraft.com*) im Jahr 1995 mit deren Erhebung begonnen hat.

### 3.1.2 Die Apache Software Foundation

Um die Entwicklung des Apache-Webservers auf eine solide Basis zu stellen, wurde 1999 die Apache Software Foundation (ASF) als »Dachorganisation« der Apache Group gegründet. Sie kümmert sich seitdem um zahlreiche Projekte freier Software; die wichtigsten von ihnen werden kurz vorgestellt. Eine weitere Aufgabe der ASF besteht darin, die Markenrechte an dem Namen »Apache« und den sonstigen Projekten zu wahren.

Die Foundation definiert sich selbst als »Meritocracy« – Mitglied wird, wer reichlich an Projekten mitgearbeitet hat, daraufhin von einem bestehenden Mitglied vorgeschlagen und schließlich von einer Mehrheit gewählt wird.

Hier eine kurze Übersicht wichtiger Projekte der Apache Software Foundation außer dem HTTP-Server. Alle sind sowohl auf der Website der ASF (*http://www.apache.org*) verlinkt als auch über die jeweils in der Liste genannten URLs direkt erreichbar:

- **Ant**
  Ein Compiler-Fernsteuerungs- und Software-Projektmanagement-Tool für Java auf XML-Basis, also ein moderner Java-Ersatz für `make`. Die Website des Projekts finden Sie unter *http://ant.apache.org*.
- **APR**
  Die Apache Portable Runtime wird noch genauer erläutert – diese API liegt dem Apache-Webserver zugrunde und abstrahiert einige System- und I/O-Funktionen. Website: *http://apr.apache.org*.
- **Cocoon**
  Framework für XML- und XSLT-basierte Server-Anwendungen, das unter anderem mit Tomcat zusammenarbeitet. Alles über das Projekt erfahren Sie unter *http://cocoon.apache.org*.

- **DB**
  Dieses Projekt kümmert sich um verschiedene Lösungen zur Datenbankprogrammierung und -integration. Bemerkenswert ist z. B. die ObjectRelationalBridge, ein Programm zur automatischen, persistenten Abbildung relationaler Datenbanken in Java-Objekten. Downloads und weitere Informationen gibt es unter *http://db.apache.org*.
- **Directory**
  Der Apache Directory Server ist ein Vielzweck-Verzeichnisserver, der Dienste wie LDAP, Kerberos, DHCP und DNS miteinander kombiniert. Genaueres erfahren Sie unter *http://directory.apache.org*.
- **Geronimo**
  Dies ist eine Open-Source-Implementierung des J2EE-Server-Standards, der die vollständige J2EE-1.4-Spezifikation von Sun erfüllt. Siehe *http://geronimo.apache.org*.
- **Incubator**
  Der »Brutkasten« ist kein Softwareprojekt, sondern eine Gruppe innerhalb der Apache Software Foundation, die sich um die Aufnahme neuer Projekte unter das Dach der Foundation kümmert. Vielleicht möchten Sie sich ja eines Tages mit einem vielversprechenden Open-Source-Projekt selbst an die Apache Software Foundation wenden? Dann erfahren Sie alles Nötige unter *http://incubator.apache.org*.
- **Jakarta**
  Jakarta ist ein gemeinsames Dach für zahlreiche Java-Softwareprojekte in der ASF. Einige Beispiele sind der JSP- und Servlet-Server Tomcat, das Webanwendungs-Framework Struts oder die Dokumentationsverwaltung Alexandria. Die Jakarta-Website mit Links zu den zahlreichen Unterprojekten finden Sie unter *http://jakarta.apache.org*.
- **James**
  Der Java Apache Mail Enterprise Server ist ein vollständig in Java geschriebener SMTP-, POP3- und NNTP-News-Server. Das Projekt finden Sie unter *http://james.apache.org*.
- **Lenya**
  Dieses Projekt ist ein Java- und XML-basiertes Content-Management-System. Es setzt auf Cocoon auf und bietet über ein Webinterface benutzerfreundliche Administrations- und Bearbeitungswerkzeuge. Näheres unter *http://lenya.apache.org*.
- **Logging**
  Da Log-Dateien nicht nur Systemadministratoren, sondern auch Programmierern wertvolle Informationen zur Fehlersuche liefern können, ist es wün-

schenswert, eine einheitliche Logging-Schnittstelle zur Verfügung zu haben. Das Projekt `log4j` bietet eine solche für Java. Seit es als Logging-Projekt in die ASF aufgenommen wurde, begann man dort mit der Entwicklung ähnlicher Hilfsmittel für C, C++, .NET, PHP und Perl. Die Website ist *http://logging.apache.org*.

- **Lucene**
  Das Projekt Lucene kümmert sich um Such-Tools: Lucene Java stellt grundlegende Such- und Indexfunktionalität zur Verfügung; darauf bauen unter anderem der Index-Server Solr und der Crawler Nutch für die Websuche auf. Ein weiteres Unterprojekt ist die C-Implementierung Lucene4c, die auf der Apache Portable Runtime basiert. Siehe *http://lucene.apache.org*.
- **Maven**
  Dies ist ein Java-Projektmanagement-Paket. Es enthält ein CVS-ähnliches Repository, Ant-artige Build-Werkzeuge und weitere Features und basiert auf einem Project Object Model (POM), das bisher im XML-Format gespeichert wird, zukünftig aber beispielsweise auch in einer relationalen Datenbank abgelegt werden soll. Näheres erfahren Sie unter *http://maven.apache.org*.
- **Perl**
  Hinter dem schlichten Projektnamen »Perl« verbirgt sich das mächtige Webserver-Modul `mod_perl`. Es zählt zu den beliebtesten Apache-Modulen überhaupt, weil es nicht nur das Schreiben von Webanwendungen in Perl, sondern auch die Erweiterung des Servers selbst durch Perl-Module ermöglicht. Das Wichtigste über `mod_perl` lesen Sie in Kapitel 15, »Technologien zur Webprogrammierung«, dieses Buches; noch genauere Informationen gibt es auf der Site *http://perl.apache.org*.
- **SpamAssassin**
  Dieser mächtige, modular aufgebaute Spamfilter kann mit beinahe beliebigen Mailservern und -Clients zusammenarbeiten und beherrscht diverse flexible Formen der Spamerkennung und -kennzeichnung. Weitere Informationen und Download unter *http://spamassassin.apache.org*.
- **TCL**
  Diese klassische Skriptsprache besitzt eine recht einfache Syntax. Das Apache-Projekt bietet Schnittstellen zur Integration der Sprache für Webserver-Anwendungen über CGI und das Modul `mod_tcl`. Die Projektwebsite ist *http://tcl.apache.org*.
- **Web Services**
  Mithilfe von Web Services können Anwendungen unter Verwendung von HTTP über das Web miteinander kommunizieren. Diese Anwendungen können in den unterschiedlichsten Sprachen programmiert werden – Web-Ser-

vice-APIs gibt es unter anderem für Java, PHP, Perl oder das Microsoft .NET Framework. Das Apache-Web-Services-Projekt bietet freie Implementierungen dieser APIs für verschiedene Programmiersprachen; Website: *http://ws.apache.org*.

- **XML**
  Die *eXtensible Markup Language* hat sich seit ihrer Einführung 1998 zum weithin anerkannten Standard für die klartextbasierte Strukturierung von Daten aller Art entwickelt. Das Apache-XML-Projekt bietet Parser, XSLT-Prozessoren und XML-APIs für verschiedene Programmiersprachen. Das Projekt befindet sich unter *http://xml.apache.org*.

### 3.1.3 Die Apache-Softwarelizenz

Die Begriffe Open Source beziehungsweise freie Software bedeuten nicht nur, dass der Quellcode frei verfügbar ist, sondern sie bezeichnen allgemein Computerprogramme, mit denen Anwender »alles« machen dürfen. Dies betont den Gegensatz zu kommerzieller Software, die meist unter einer sehr restriktiven Lizenz steht: Ein von Juristen formulierter Lizenzvertrag (meist »EULA« oder »End User License Agreement« genannt) regelt detailliert, was Sie mit einem solchen Programm anstellen dürfen. Neben dem Kopieren über eine persönliche Sicherheitskopie hinaus ist es meist vor allem untersagt, die Software selbst näher zu untersuchen oder gar durch Reverse Engineering ihre Funktionsweise zu ermitteln und dann genau nachzuahmen.

Solchen Einschränkungen ist freie Software nicht unterworfen. Das bedeutet aber keineswegs, dass sie vollkommen rechtsfrei veröffentlicht wird. Zunächst einmal gilt natürlich das Urheberrecht der eigentlichen Autoren, wobei die meisten freien Softwareprojekte von zahlreichen, über die ganze Welt verteilten Entwicklern geschrieben werden. Abgesehen davon stehen auch die meisten Open-Source-Projekte unter Softwarelizenzen. Diese werden natürlich ebenfalls von Juristen formuliert, dienen aber einem anderen Zweck: Es muss sichergestellt werden, dass sich kein kommerzieller Softwarehersteller das Programm exklusiv zu eigen macht oder gar zum Patent anmeldet. Es gibt unterschiedliche Lizenzen für freie Software; die Lizenz der Apache Software Foundation ist nur eine von ihnen.[2] Andere bekannte Open-Source-Lizenzen sind folgende:

2 Das denkwürdige Editorial von Ausgabe 10/2005 der Zeitschrift iX (online unter *http://www.heise.de/ix/artikel/2005/10/003/*) kommt sogar zu dem Schluss, dass es inzwischen zu viele unterschiedliche Open-Source-Lizenzen gibt, was zu Unübersichtlichkeit und Rechtsunsicherheit führt.

- Die *GNU General Public License* (GPL) der Free Software Foundation ist die älteste bekannte Lizenz für freie Software. Sie gilt für sämtliche Software des GNU-Projekts und beispielsweise auch für den Linux-Kernel. Natürlich erlaubt sie die beliebige Weitergabe und Modifikation der Software, untersagt aber grundsätzlich deren Verwendung in kommerziellen Produkten. Da dies im Falle von Compilern oder Bibliotheken zu Problemen führen kann, existiert daneben die LGPL (Library oder Lesser GPL), die diesen Einsatz in engen Grenzen gestattet.
- Die *BSD-Lizenz*, unter der vor allem die freien BSD-UNIX-Varianten FreeBSD, OpenBSD und NetBSD verbreitet werden, kann als Gegenteil der restriktiven GPL betrachtet werden: Sie erlaubt den Einsatz der Software unter fast beliebigen Bedingungen.

Die Apache-Softwarelizenz liegt etwa in der Mitte zwischen diesen beiden Extremen.[3] Sie gestattet die Verwendung, Weitergabe und Veränderung von Software der ASF unter einigen Bedingungen; einige wesentliche Punkte dieser Lizenzvereinbarung besagen beispielsweise Folgendes:

- Bei einer Weitergabe des Quellcodes der Software muss die vollständige Lizenz einschließlich Copyright und Haftungsausschluss mitgeliefert werden.
- Auch bei der Weitergabe in binärer Form muss die Lizenzvereinbarung enthalten sein, beispielsweise in der mitgelieferten Dokumentation.
- Bei Weiterverwendung in eigenen Projekten muss die Dokumentation oder die Bildschirmausgabe den Hinweis enthalten, dass das Programm Software der Apache Software Foundation enthält.
- Ohne Erlaubnis der ASF darf der Name »Apache« oder »Apache Software Foundation« nicht verwendet werden, um für eigene, davon abgeleitete Produkte zu werben.
- Projekte, die aus der Weiterentwicklung eines ASF-Programms entstanden sind, dürfen nicht Apache heißen oder das Wort »Apache« im Namen enthalten.

Ein weiterer wesentlicher Bestandteil der Lizenz ist ein Haftungsausschluss (Disclaimer), der verhindert, dass die Entwickler der Software oder die Apache Software Foundation Schadenersatz leisten müssen, wenn das Programm nicht richtig funktionieren sollte oder im Zusammenhang mit der Verwendung des Programms Datenverluste oder andere Schäden auftreten sollten.

3 Genauer gesagt ähnelt die veraltete Fassung 1.1 eher der BSD-Lizenz, während die neue Version 2.0 (siehe Anhang E) sich stärker der GPL angenähert hat.

Im Januar 2004 wurde die neue Version 2.0 der Apache-Lizenz eingeführt. Sie präzisiert viele Formulierungen und ergänzt einige Punkte, etwa um den schädlichen Einfluss der ärgerlichen Softwarepatente zu minimieren. Den vollständigen Wortlaut finden Sie in Anhang E.

### 3.1.4 Sonstige Webserver

Neben Apache gibt es natürlich noch einige andere Webserver – sowohl freie als auch kommerzielle. Einige von ihnen sollen an dieser Stelle kurz vorgestellt werden, um einen Vergleich mit dem Apache-Webserver zu bieten.

#### Microsoft Internet Information Services

Dies ist der Webserver von Microsoft. Er wird nicht als eigenständiges Programmpaket angeboten, sondern ist seit Windows NT Server 4.0 aus dem Jahr 1996 Teil der Microsoft-Server-Betriebssysteme. Er läuft also nur unter Windows und nicht unter einer Vielzahl von Plattformen wie Apache. Das aktuellste Server-Betriebssystem, Windows Server 2008 Second Edition, enthält den IIS in der neuesten Version 7.

Internet Information Services (vormals unter dem Namen »Internet Information Server« bekannt) ist übrigens nicht nur ein Webserver, sondern enthält auch einen eingebauten FTP-, einen News- und einen SMTP-Server. Alle diese Komponenten lassen sich über ein grafisches Tool konfigurieren; eine Konfigurationsdatei nach UNIX- oder Apache-Art gibt es nicht. Dies erleichtert zwar den Überblick über die verfügbaren Konfigurationsoptionen, aber es macht die Automatisierung und Übertragbarkeit der Konfigurationsdateien kompliziert.

Natürlich gibt es einige spezielle Vorteile, die nur dieser Server aufgrund seiner Integration in die Windows-Funktionalität zu bieten hat. Dazu gehören unter anderem folgende Punkte:

- **Active Server Pages**
  Dies ist Microsofts Schnittstelle für Skriptbefehle, die unmittelbar in HTML-Dokumente hineingeschrieben und vom Server ausgeführt werden, bevor das Dokument an den Client geliefert wird. Das Konzept ähnelt somit PHP oder Java ServerPages (JSP). Die ursprünglichen ASP wurden in der Sprache VBScript geschrieben, einer einfachen Skriptvariante von Microsoft Visual Basic. Die aktuelle Variante ASP.NET lässt sich dagegen in allen Sprachen programmieren, die die Common Language Runtime (CLR) des .NET Frameworks verwenden kann. Dies sind von Microsofts Seite Visual Basic .NET, Visual C++ und C#.

- **ISAPI**
  Der IIS ist mit einer speziellen Schnittstelle namens Internet Server API (ISAPI) ausgestattet. Diese ermöglicht das Schreiben von Erweiterungen, die als DLL-Dateien kompiliert werden und somit schneller laufen als ASP oder CGI. Für die Windows-Version von Apache steht übrigens ebenfalls eine ISAPI-Implementierung in Form des Moduls `mod_isapi` zur Verfügung (siehe Kapitel 15, »Technologien zur Webprogrammierung«). Die Performance dieses Moduls ist allerdings nicht mit dem Original-ISAPI vergleichbar. Außerdem ermöglicht die Apache-Variante keine ISAPI-Filter.
- **Windows- und .NET-Integration**
  Da es sich beim IIS um eine Technologie von Microsoft handelt, lässt er sich besser in die Windows-Systemumgebung integrieren als Apache. Sie haben mithilfe der erwähnten ASP.NET-Schnittstellen und ISAPI Zugriff auf das gesamte Leistungsspektrum eines Windows-Server-Systems und auf das .NET Framework: Beispielsweise können Sie über ADO.NET auf Datenbanken und XML zugreifen oder Authentifizierungen über Active Directory durchführen.

Fazit: In einer bestehenden Microsoft-Server-Umgebung bietet sich die Anwendung des Internet Information Services einfach an. In diesem speziellen Fall ist selbst das Kostenargument hinfällig: Wenn Sie bereits Server-Betriebssysteme von Microsoft betreiben, ist der Internet Information Services automatisch darin enthalten und verursacht keine Zusatzkosten. Der Nutzen, den die Integration in die vorhandene Infrastruktur bietet, übersteigt hier den möglichen Sicherheits- und Performance-Gewinn durch Apache.

### lighttpd

Der »Lighty« ist ein Open-Source-Produkt, genau wie Apache. Wie der Name vermuten lässt, handelt es sich um eine leichtgewichtige Alternative zu einem umfangreichen HTTP-Server wie Apache. Sein Entwickler Jan Kneschke schrieb ihn im Hinblick auf Performance und geringen Ressourcenverbrauch – insbesondere, um das »C10K«-Problem (*http://www.kegel.com/c10k.html*) zu lösen, das heißt 10.000 gleichzeitige Client-Verbindungen zu unterstützen. Im Gegensatz zu Apache und anderen großen Webservern arbeitet `lighttpd` nicht mit Forking oder Threading, sondern rein eventbasiert beziehungsweise asynchron (siehe die Diskussion über verschiedene Laufzeitmodelle weiter unten in diesem Kapitel, und dort besonders die `select()`-Server). `lighttpd` bietet weniger Features als Apache, ist aber ebenfalls durch Module erweiterbar.

Aufgrund dieser Eigenschaften gewinnt `lighttpd` langsam höhere Marktanteile; große und wichtige Sites wie *Youtube* setzen ihn bereits teilweise ein.

### nginx

nginx (gesprochen »engine x«) entstand in Russland und ähnelt in seiner Architektur dem lighttpd. Allerdings verwendet er ähnlich wie Apache in der Standard-Unix-Variante einen Master-Prozess und mehrere sogenannte *Worker-Prozesse*, sodass er von Hause aus multiprozessorfähig ist. Für lighttpd kann SMP-Unterstützung inzwischen optional aktiviert werden, gilt aber noch als experimentell. Nähere Informationen zu nginx finden Sie auf der Website *nginx.org*.

### Zeus

Der Zeus Web Server von Zeus Technology Ltd. ist ein kommerzieller Webserver für UNIX-Systeme, der für möglichst hohe Belastung und Performance optimiert wurde. Eine der bekanntesten Websites, die unter anderem auf Zeus setzen, ist *Ebay*.

Der Zeus Web Server wird über ein einfaches Shell-Skript installiert. Es stellt Ihnen einige Konfigurationsfragen, etwa nach Installationspfad, Administratorpasswort usw. Nachdem der Server installiert ist, können Sie seine webbasierte Konfiguration aufrufen. Öffnen Sie dazu einen beliebigen Browser, und geben Sie als URL *http://localhost:9090* (beziehungsweise Port 9090 auf dem entfernten Host, auf dem Zeus ausgeführt wird) ein. Nun müssen Sie sich mit dem Benutzernamen »admin« und dem bei der Installation festgelegten Passwort anmelden. Die eigentlichen Websites werden bei Zeus stets als virtuelle Hosts eingerichtet. Bequemerweise lässt sich ein vorhandener virtueller Host duplizieren, wenn Sie einen neuen einrichten möchten, der nur geringfügig vom vorherigen abweichen soll.

### Tomcat

Jakarta Tomcat, wie bereits erwähnt ebenfalls ein Projekt der Apache Software Foundation, wird in Kapitel 15, »Technologien zur Webprogrammierung«, genauer behandelt. Während dort vor allem auf seine Fähigkeiten als Java-Servlet- und JSP-Engine eingegangen wird, die mit einem Apache-HTTP-Server zusammenarbeitet, soll an dieser Stelle erwähnt werden, dass Tomcat selbst ein recht leistungsfähiger Webserver ist. Natürlich kommt er in Performance und Leistungsspektrum nicht an den Apache-Webserver heran, aber das ist auch gar nicht die Intention seiner Entwickler. Für Websites, die nur in geringem Umfang aus statischen HTML-Dokumenten bestehen und fast ausschließlich auf Java-Technologie setzen, dürfte eine Tomcat-Standalone-Lösung ideal sein. Für alle anderen Sites, die nur teilweise Servlets oder JSP einsetzen, existieren zahlreiche Möglichkeiten, Tomcat in Kombination mit Apache 2 zu betreiben.

Auch wenn der Schwerpunkt von Kapitel 15, »Technologien zur Webprogrammierung«, aufgrund der Thematik dieses Buches ein anderer ist, erhalten Sie dort einen kurzen Überblick über die Verwendung von Tomcat als Webserver. Anders

als der Apache-HTTP-Server, der aus Kompatibilitätsgründen das historisch gewachsene Format der NCSA-Konfigurationsdateien übernommen hat, verwaltet Tomcat seine Einstellungen konsequent in XML-Dateien.

## 3.2 Funktionen von Apache 2

Die Informationen über die Installation, Konfiguration und Programmierung des Apache-HTTP-Servers in den restlichen Kapiteln dieses Buches betreffen eher einzelne Aspekte. Deshalb erhalten Sie an dieser Stelle zunächst einen Gesamteindruck über seine Funktionsweise und seinen Leistungsumfang.

Es ist keine große Überraschung, dass eine vollständige und vorbildliche Implementierung des HTTP/1.1-Standards gemäß RFC 2616, wie sie im vorigen Kapitel beschrieben wurde, den Kern von Apache 2 bildet. Daneben gibt es unzählige Erweiterungen für Spezialaufgaben wie die Zusammenarbeit mit bestimmten Skriptsprachen, URL-Umleitungen, erweitertes Logging oder sogar die Unterstützung anderer Protokolle außer HTTP (Apache als Proxy-Server, FTP-Server oder gar Mailserver). Alle diese Erweiterungen sind in Form von Modulen realisiert, die sich entweder statisch einkompilieren oder dynamisch zur Laufzeit hinzuladen lassen. Viele dieser Module gehören zum Lieferumfang von Apache; zahlreiche weitere werden von Drittanbietern bereitgestellt.

Die ursprüngliche Fassung des Servers wurde ausschließlich für UNIX-Systeme entwickelt. Erst ab Version 1.2 wurde mit der Portierung auf andere Betriebssysteme begonnen. Durch die Schaffung der weiter unten näher behandelten Apache Portable Runtime (APR), einer virtuellen Systemschicht, konnte die Stabilität und Performance der Portierungen in Version 2 entscheidend verbessert werden.

Apache 2 wird (mindestens) von folgenden Systemen der großen UNIX-Familie unterstützt:

- Linux
- FreeBSD
- OpenBSD
- NetBSD
- Sun Solaris
- IBM AIX
- HP UX
- BeOS
- Mac OS X

Auf den meisten anderen modernen UNIX-Systemen müsste sich Apache 2.4 oder 2.2 ebenfalls problemlos kompilieren lassen.

Daneben wurde Apache erfolgreich auf folgende Nicht-UNIX-Plattformen portiert:

- Microsoft Windows NT-Familie (NT 4.0, 2000, XP, Vista, Windows 7)
- Microsoft-Windows-Server-Familie (Windows NT 4.0 Server, Windows 2000 Server, Windows Server 2003, Windows Server 2008)
- Microsoft Windows 9x-Familie (95, 98, Me)
- Novell NetWare
- VMS und OpenVMS
- IBM OS/2
- BS2000

Diese Plattformliste ist mit Sicherheit nicht vollständig – zumindest der veraltete Apache 1.3 wurde auch für seltenere Systeme wie etwa AmigaOS portiert. Apache ist freie Software, und im Prinzip kann jeder Interessierte versuchen, das Projekt auf eine neue Plattform zu portieren. Unterstützt wird dies dadurch, dass auch die GNU-Entwicklungswerkzeuge für immer mehr Nicht-UNIX-Plattformen zur Verfügung stehen.

**Welche Plattform ist zu empfehlen?**

Auf diese Frage sollte man am besten die vorsichtige Antwort des Juristen geben: »Es kommt darauf an.« Der größte Teil der Apache-Webserver im Internet ist auf UNIX-Systemen installiert, allen voran Linux und die freien BSD-Varianten.

Das Internet wurde auf UNIX-Maschinen zu dem, was es heute ist, und Apache (mitsamt seinem Vorläufer NCSA HTTPd) ist auf dieser Plattform entstanden. Die Unterstützung für andere Systeme war am Anfang eher experimentell – selbst frühe Releases von Apache 1.3 für Windows wurden noch mit dem ausdrücklichen Warnhinweis versehen, dass die Windows-Version als eine Art Beta-Version zu verstehen sei und man von ihr bei Weitem nicht den von UNIX gewohnten Standard erwarten dürfe.

Seit der Veröffentlichung von Apache 2.0 hat sich dies allerdings grundlegend geändert: Durch die Apache Portable Runtime und die unterschiedlichen Multiprocessing-Module konnte die Performance und Stabilität des Servers auch auf Nicht-UNIX-Systemen erheblich gesteigert werden. Deshalb kann die Empfehlung heutzutage im Grunde lauten: Verwenden Sie Apache einfach auf der Plattform, die zu Ihrer IT-Infrastruktur passt. Wenn Sie ein auf Windows basierendes Netzwerk betreiben, können Sie den Server genauso gut auf einem Windows 7, Vista- oder XP-Rechner betreiben wie auf einem UNIX-System. Sollten Ihre Server unter Novell NetWare arbeiten, verwenden Sie doch einfach die passende Apache-Version.

Gemäß der praktischen Verbreitung des Webservers konzentriert sich dieses Buch auf die Apache-Versionen für UNIX-Systeme – die man in diesem Fall trotz gewisser Unterschiede über einen Kamm scheren kann – und für die Windows NT-Familie.

**Vorsicht!**

Sie sollten aus Performance- und vor allem aus Sicherheitsgründen *dringend* davon absehen, einen Produktionsserver unter Windows 9x zu betreiben! Natürlich hält Sie niemand davon ab, in einem kleinen bis mittleren Unternehmen einen alten Pentium-Rechner (I-III) in die Ecke zu stellen, Windows 98 und Apache zu installieren und damit im Intranet E-Books, Dokumentationen oder MP3-Dateien für die Mittagspausenunterhaltung zu verbreiten. Aber selbst eine solche altersschwache Maschine läuft prima unter Linux, und es schadet überhaupt nichts, das entsprechende Plus an Sicherheit mitzunehmen – eine »Linux-auf-einer-CD-Distribution« wie *Knoppix (http://www.knopper.net/knoppix*) reicht dazu vollkommen aus.

### 3.2.1 Technischer Überblick

Behandelt wird in diesem Buch die aktuelle, stabile Version des Apache-HTTP-Servers. Das ist zurzeit (Herbst 2011) noch die Version 2.2, von der im September 2011 die Unterversion 2.2.21 erschienen ist. Parallel dazu wird der neue Versionszweig 2.4 beschrieben, der sich in der späten Beta-Phase befindet (aktuell ist die Version 2.3.15 von November 2011) und in Kürze als fertige Release-Version erscheinen wird. Weitere Informationen über den teilweise noch verwendeten Zweig 2.0 erhalten Sie in Anhang A.

Hier einige der überzeugendsten Gründe dafür, warum Sie Apache als Webserver verwenden sollten:

- **Verbreitung**
  Wie bereits erwähnt, ist Apache der mit Abstand am häufigsten eingesetzte Webserver der Welt. Das heißt, dass kein anderer HTTP-Server so vielen Praxistests unterworfen wurde. Apache ist somit nachgewiesenermaßen ein praxistauglicher Allzweck-Webserver. Dies gilt inzwischen auch für Apache 2, der auf immer mehr Produktionsservern die frühere Version 1.3 ersetzt. Darüber hinaus gewährleistet der hohe Verbreitungsgrad den Zugriff auf Unmengen an Dokumentationen und öffentlichem Support.
- **Performance**
  Es mag vereinzelt unsäglich teure kommerzielle Webserver geben, die ein wenig schneller sind als Apache. Durch wirklich neutrale und reproduzierbare Benchmarks wurde dies allerdings noch nicht nachgewiesen; wie bei anderen Server-Diensten, Betriebssystemen oder Hardware pflegen sich die kommerziellen Hersteller stark angepasste Umstände zu schaffen, unter denen ihr Pro-

dukt schneller läuft. Die Apache Group dagegen hat für so etwas keine Zeit und arbeitet lieber an der Leistungsfähigkeit des eigenen Produkts. Auch hier gilt der Verbreitungsgrad als Nachweis: Es gibt zahllose große Websites mit Millionen von Zugriffen am Tag, die Apache einsetzen und deren gute Performance für 99,9 % aller Anfragen gewährleistet ist.

Die Performance von Apache 2.2 beziehungsweise 2.4 wurde gegenüber früheren Versionen noch einmal erheblich gesteigert – und zwar auch für Nicht-UNIX-Plattformen. Das liegt vor allem an der Einführung der Apache Portable Runtime (APR) als solider Plattform-Abstraktions-API sowie an den verschiedenen Multiprocessing-Modulen, die die Leistung für unterschiedliche Einsatzzwecke verbessern.

- **Stabilität**
  Bei der Weiterentwicklung des Apache-Webservers wird vor allem auf stabile und robuste Funktionsfähigkeit Wert gelegt. Besonders der modulare Aufbau unterstützt dies: Ein schlankes, leistungsfähiges Grundprogramm mit wenigen Funktionen wird durch zahllose mithilfe einer sauberen API angebundene Module ergänzt.
- **Sicherheit**
  Apache ist freie Software. Anders als bei kommerziellen Programmen mit einem in sich geschlossenen Entwicklungsteam, wird der Quellcode hier von zahlreichen über die ganze Welt verteilten Programmierern mit den unterschiedlichsten Erfahrungen und Umfeldern geschrieben und begutachtet. So werden Fehler in aller Regel sehr schnell gefunden und beseitigt. Dank des verfügbaren Quellcodes kann zur Not sogar jeder mit entsprechenden C-Kenntnissen selbst einen Bug beseitigen, wenn es schnell gehen muss. Da viele Apache-Anwender dies tun und ihre Bugfixes anschließend oft per Mailingliste an die Apache Group weitergeben, steigt die Chance für schnelle Fehlerbereinigungen sogar noch an. Da Fehler im Programmcode (neben schlechter Konfiguration, die dieses Buch hoffentlich verbessern hilft) das größte Sicherheitsrisiko darstellen, kann man ein Open-Source-Projekt wie den Apache-Webserver als sehr sicher einstufen.[4]
- **Skalierbarkeit**
  Mit Apache können Sie Sites in beliebiger Größe und Komplexität betreiben: Angefangen von einem kleinen Intranet-Webserver mit wenigen Hundert Dokumenten über eine mittlere Firmenwebsite bis hin zu einem riesigen Ser-

4 Wenn Sie Sicherheits-Mailing-Listen wie BugTraq lesen (was Sie als verantwortungsbewusster Administrator natürlich tun sollten), werden Sie etwa wöchentlich Meldungen über neue Apache-Sicherheitslücken erhalten. Die Sicherheit wird nämlich nicht durch Software ohne Fehler gewährleistet (die gibt es nicht!), sondern durch ein effizientes und vernetztes Finden und Beseitigen von Fehlern.

ver-Raum voller Load-Balancing-Server-Rechner. Für den erstgenannten Zweck kann Apache 2 fast ohne erkennbaren Ressourcenverbrauch auf einem Desktop-PC im Hintergrund laufen. Sie können den Server an eine wachsende Website anpassen, indem Sie die Konfigurationsdateien, die Sie im Laufe der Zeit Ihren Bedürfnissen angepasst haben, auf eine größere Maschine mitnehmen – der wesentliche Aufbau der Datei ist über alle System- und Plattformunterschiede hinweg derselbe.

#### Apache-Versionierung

Seit Version 2.0 hat die Apache Group die Versionszählweise anderer freier Softwareprojekte wie Linux bis zur Kernel-Version 2.4 oder Perl seit Version 5.6 übernommen: Eine gerade Unterversionsnummer (2.0, 2.2, 2.4 usw.) steht für die stabilen Releases, während ungerade Nummern, also 2.1 oder 2.3, die jeweilige Entwicklerversion bezeichnen: Zurzeit ist 2.2 noch die stabile Version; an der neuen Beta-Version 2.3 wird gerade intensiv gearbeitet, sodass sehr bald die fertige Version 2.4 erscheinen wird.

Manchmal kommt es dagegen vor, dass neue Errungenschaften der Entwicklerversion nachträglich in neuere Releases der vorigen stabilen Version übernommen werden (sogenannte *Backports*) – dies geht natürlich nur dann, wenn sie grundsätzlich mit der alten Version kompatibel sind.

Dieser sparsame und vorsichtige Umgang mit Versionsnummern und Veröffentlichungen führt natürlich zu größerer Anwenderzufriedenheit: Sie können fast immer ein einigermaßen ausgereiftes, stabiles und fehlerarmes Programm erwarten, wenn Sie sich ein stabiles Release besorgen. Bei kommerziellen Softwareherstellern sieht das leider oft anders aus: Manche bringen innerhalb weniger Monate schon wieder die nächste Hauptversion heraus, die sich dann als »Bananaware« entpuppt: Sie ist im Auslieferungszustand noch »grün« und »reift erst beim Kunden«.

Betrachten Sie zum Vergleich die Versionsnummern des Linux-Kernels: Allein der Sprung von Version 2.4 (Ende 2001) bis zum Kernel 2.6 hat ganze drei Jahre gedauert, dieser wiederum war über sechs Jahre lang (mit regelmäßigen Detailverbesserungen) im Einsatz, bis im Sommer 2011 die neue Hauptversion 3.0 erschien.

Aber selbst wenn eine neue stabile Version veröffentlicht wird, steigt nicht unbedingt alle Welt von heute auf morgen darauf um. Das hat mitunter wichtige praktische Gründe: Bei Apache war beispielsweise die Modularchitektur der Vorgängerversion 1.3 inkompatibel mit dem neuen 2.x-Format. Zwar wurde 2.0 erst als stabil gekennzeichnet, als die wichtigsten Module der Apache Group selbst por-

tiert waren, aber Drittanbieter-Module wurden erst allmählich (oder gar nicht) auf die neue Version zugeschnitten. Wer also eines oder mehrere wichtige Module im Einsatz hatte, die nicht für Apache 2 verfügbar waren, musste noch länger bei 1.3 bleiben.

Dass es gute Gründe geben mag, die Haupt- oder Unterversion nicht zu wechseln, ist kein Argument dafür, dass Sie innerhalb Ihrer Version nicht jeweils das aktuellste Release installieren sollten, sobald es verfügbar ist. Dies ist aus Sicherheitsgründen unabdingbar, weil jedes neue Release alle bisher bekannt gewordenen Bugs der Vorversionen beseitigt. Da die allermeisten Angriffe auf Server durch die Ausnutzung vorhandener Sicherheitslücken stattfinden, gibt es kaum eine wichtigere Sicherungsmaßnahme, die ein verantwortungsbewusster Administrator regelmäßig durchführen sollte.

Wenn Sie dieses Buch also schon eine Weile besitzen und erst dann Apache installieren möchten, sollten Sie für einen Produktionsserver nicht einfach die Version der beiliegenden DVD-ROM verwenden. Stattdessen möchte ich Ihnen dringend empfehlen, zuallererst auf der Website *http://httpd.apache.org* nachzuschauen, ob dort nicht eine neuere Version verfügbar ist. Und selbst wenn Sie den Server nur zum Ausprobieren oder Lernen verwenden, sollten Sie diesen Rat beherzigen, denn wenn ein Angreifer erst einmal einen Zugang zu Ihrem Rechner gefunden hat, ist es in der Regel nicht der Webserver, der ihn wirklich interessiert. Näheres zu diesem ernsten und wichtigen Thema lesen Sie in Kapitel 18, »Sicherheit«.

#### Funktionsübersicht

Das HTTP-Schema mit Anfrage und Antwort wurde bereits im vorigen Kapitel genau untersucht. Als Server ist Apache 2 für den Antwort-Teil (Response) zuständig: Clients (Webbrowser, Suchmaschinen-Robots und andere Programme, die Informationen aus dem Web beschaffen) senden Anfragen an den HTTP-Server. Dieser zerlegt die Anfragen in die beiden wesentlichen Bestandteile: URL und zusätzliche Header. Anschließend führt er die Anfrage entweder aus oder generiert eine Meldung, die beschreibt, warum dies nicht möglich ist. In jedem Fall wird eine Antwort an den Client gesendet, die dem ebenfalls im vorigen Kapitel beschriebenen HTTP-Antwort-Schema entspricht.

So weit der grobe Ablauf. Das Interessante an einem ausgewachsenen, flexiblen und erweiterbaren Webserver wie Apache ist allerdings, dass zwischen Anfrage und Antwort beinahe beliebige Manipulationen möglich sind. Hier nur eine kleine Auswahl:

- **URL-Umwandlung**
  Mithilfe spezieller Module (z. B. `mod_alias` oder `mod_rewrite`) kann der Server die URL der Anfrage nach einem bestimmten Schema umwandeln. Dies ist beispielsweise für automatische Weiterleitungen nach Umstrukturierungen einer Website, zum Einbinden externer Ressourcen oder für intelligente halb automatische Links interessant (siehe Kapitel 8, »Weiterleitungen und Indizes«).
- **Filter**
  Seit der Einführung von Apache 2 können Sie Module oder externe Programme bestimmen, die die Anfrage verändern sollen (Eingabefilter) oder »Last-Minute-Änderungen« an der Antwort vornehmen (Ausgabefilter). Es kann sich dabei z. B. um die automatische Konvertierung von Zeichensätzen, die Herausfilterung potenziell gefährlicher Anfrageteile oder gar gezielte Textersetzungen handeln. Ausgabefilter sind erheblich leistungsfähiger und vor allem flexibler als die altbekannten Server Side Includes (SSI), die Apache natürlich auch unterstützt. Näheres zu diesem Thema steht in Kapitel 16, »SSI und Filter«.
- **MIME-Informationen**
  Der HTTP-Server kann auf intelligente Art und Weise entscheiden, welchen MIME-Type (Antwort-Header `Content-Type`) er für bestimmte angeforderte Ressourcen zurückliefern soll. Unter anderem kann er sich dazu an der Dateiendung oder auch am Inhalt der entsprechenden Dateien orientieren. Diese Informationen finden Sie in Kapitel 7, »Header und MIME-Einstellungen«.
- **Content Negotiation**
  Diese Technik ermöglicht es Apache, den Bedürfnissen verschiedener Clients in optimaler Weise zu entsprechen: Gemäß den verschiedenen `Accept-*`-Anfrage-Headern, die im vorigen Kapitel besprochen wurden, kann der Webserver automatisch unterschiedliche Fassungen einer Ressource liefern. Einfaches Beispiel: Wenn Sie Dateien in einer Website mit einer zusätzlichen Endung abspeichern, die einem ISO-Sprachcode entspricht, wertet Apache den Header `Accept-Language` der Anfrage aus und liefert etwa eine der Dateien *index.html.de* (Deutsch), *index.html.en* (Englisch) oder *index.html.fr* aus, obwohl der entsprechende Hyperlink nur auf *index.html* lautete. Auch dieses Thema wird in Kapitel 7, »Header und MIME-Einstellungen«, behandelt.
- **Authentifizierung und Zugangskontrolle**
  Apache kann die Berechtigungen für Client-Anfragen auf zahlreiche Arten einschränken und diese Einschränkungen für Sie verwalten. Neben einer Autorisierung auf der Grundlage von IP-Adresse oder Hostname (`mod_authz_host`;

bis Version 2.0 `mod_access`) und der einfachen Anmeldung mit Benutzername und Passwort (`mod_auth` beziehungsweise `mod_auth_basic`) wurden viele weitere interessante Authentifizierungsmodule entwickelt; beispielsweise ermöglicht `mod_authnz_ldap` die Überprüfung der Anmeldedaten anhand von LDAP-basierten Verzeichnisdiensten. Dies ist das Thema von Kapitel 9, »Authentifizierung«.

- **CGI und Webanwendungen**
  Wie bereits angesprochen, muss die vom Client angeforderte Ressource kein statisches Dokument sein, sondern es kann sich auch um ein Skript oder Programm handeln, das die Ausgabe für den Client aus einer Vorlage und aus dynamischen Daten erzeugt. Apache unterstützt neben der klassischen CGI-Schnittstelle die direkte Zusammenarbeit mit zahlreichen Sprachen, Technologien und Application-Servern. Einige Beispiele, die auch in diesem Buch behandelt werden, sind die Perl-Integration mit `mod_perl`, PHP oder Java ServerPages und Java-Servlets mit Tomcat. Lesen Sie dazu die Kapitel 14, »CGI«, und 15, »Technologien zur Webprogrammierung«.
- **Gesicherte Verbindungen**
  Sie können Apache für die abgesicherte HTTPS-Kommunikation konfigurieren: Alle zwischen Client und Server ausgetauschten Daten werden dadurch verschlüsselt und digital signiert. Dies ist für Anwendungen wie E-Commerce oder Homebanking äußerst wichtig, weil dabei private Daten über das öffentliche Internet übertragen werden, deren Integrität und Geheimhaltung gewährleistet sein muss. Näheres darüber erfahren Sie in Kapitel 10, »Gesicherte Verbindungen«.
- **Header-Manipulation**
  Es besteht die Möglichkeit, die meisten der im vorigen Kapitel angesprochenen Anfrage- und Antwort-Header beliebig zu ändern. Auch dies verbessert die Flexibilität in der Reaktion auf unterschiedliche Anfragen (siehe Kapitel 7, »Header und MIME-Einstellungen«.
- **Virtuelle Hosts**
  Apache macht es Ihnen leicht, virtuelle Hosts einzurichten – mehrere vermeintlich eigenständige Websites innerhalb einer Server-Installation. Deshalb ist er der ideale Webserver für große Unternehmen mit mehreren Sites und natürlich für Webhoster, die die Websites ihrer Kunden jeweils zu Hunderten gebündelt auf die Server-Maschinen packen. Für Letztere steht explizit das Modul `mod_vhost_alias` zur Verfügung, das Verzeichnisse im Dateisystem nach bestimmten Vorgaben automatisch als virtuelle Hosts einrichtet. Das Wichtigste darüber steht in Kapitel 12, »Skalierung und Performance-Tuning«.

- **Multiprotokoll-Support**
  Wenn Sie möchten, kann sich Apache 2 sogar um andere Server-Dienste kümmern: Es gibt Module für Apache als Proxy-Server, FTP-Server, Echo-Server und sogar Mailserver. Diese Server-Dienste erledigt Apache ehrlich gesagt nicht ganz so leistungsfähig wie entsprechende Spezialserver – aber wenn Sie sie nur gelegentlich benötigen, reicht das Leistungsspektrum völlig aus. Informationen zu diesem Thema erhalten Sie in Kapitel 17, »Apache erweitern«.

### Neuerungen in Apache 2.4

Der neueste Entwicklungszweig von Apache 2, Version 2.4, enthält eine Reihe von Neuerungen; die wichtigsten werden hier kurz aufgelistet:

- **Dynamisch ladbare Multiprocessing-Module**
  Wie bereits erwähnt, kann Apache mit verschiedenen Server-Architekturen betrieben werden, die jeweils unterschiedliche Kombinationen aus Prozessen und/oder Threads verwenden. Diese Architekturen werden durch Multiprocessing-Module (MPM) bereitgestellt. Bei früheren Versionen musste man sich bereits beim Kompilieren für ein bestimmtes MPM entscheiden. In Version 2.4 kann man dagegen mehrere kompilieren und sie wie andere Module durch eine Konfigurationsänderung dynamisch laden. Sinnvoll verwendbar ist dies nur auf UNIX-Systemen, da es für andere Plattformen nur jeweils ein optimiertes MPM gibt. Genaueres darüber steht in Kapitel 4, »Apache kompilieren und installieren«, und 6, »Grundkonfiguration«.
- **Neuordnung der Zugriffskontrolle**
  Während in Apache 2.2 die Authentifizierung neu geregelt wurde (siehe nächsten Unterabschnitt), haben die Entwickler sich diesmal die Zugriffskontrolle vorgenommen, das heißt die Entscheidung, aufgrund welcher Kriterien Benutzer auf den Server und bestimmte Ressourcen darauf zugreifen dürfen. Das Modul `mod_authz_core` integriert diese Einstellungen in die bisher für die Autorisierung verwendete Direktive `Require` und fügt zudem Logik-Container wie `<RequireAll>` oder `<RequireAny>` hinzu. Näheres dazu erfahren Sie in Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«.
- **Allgemeine Bedingungs-Direktive**
  In Konfigurationsdateien können Einstellungen in einem Container im Format `<If Bedingung>...</If>` stehen, in dem bei jeder einzelnen Anfrage ein bestimmter Ausdruck geprüft wird. Nur, wenn der Ausdruck zutrifft, werden die enthaltenen Einstellungen beachtet. Weitere Informationen erhalten Sie in Kapitel 6, »Grundkonfiguration«.
- **Neue und verbesserte Dienstprogramme**
  Als neues Dienstprogramm kommt `fcgistarter` hinzu, ein externer Daemon

für FastCGI-Prozesse. Diese Prozesse können durch das ebenfalls neue Modul `mod_proxy_fcgi` als Reverse-Proxy über Apache ausgeliefert werden (siehe Kapitel 14, »CGI«). Das Dienstprogramm `htcacheclean` (beschrieben in Kapitel 13, »Proxy- und Cache-Funktionen«) wurde stark erweitert und verbessert – beispielsweise lassen sich im Cache befindliche URLs nun individuell löschen, und es kann eine Liste der gecachten Dateien ausgegeben werden.

- **Neue Module**
  Es wurden zahlreiche neue Module hinzugefügt: `mod_allowmethods` (Beschränkung von HTTP-Anfragemethoden), `mod_buffer` (Pufferung von Filter-Stacks), `mod_data` (Umwandlung des Antwort-Bodys in eine Data-URL gemäß RFC 2397), `mod_lua` (Einbettung der Skriptsprache Lua für dynamische Konfiguration und kleinere Webanwendungen), `mod_proxy_express` (dynamische Reverse-Proxy-Konfiguration), `mod_proxy_fcgi` (FastCGI-Proxy), `mod_proxy_scgi` (SCGI-Proxy), `mod_ratelimit` (Beschränkung der Bandbreite pro Client), `mod_reflector` (Zurückgabe des Anfragebodys über den Ausgabe-Filter-Stack), `mod_remoteip` (Ersetzung der Client-IP-Adresse in den Anfrage-Headern durch die Liste eines Proxys oder Load-Balancers), `mod_reqtimeout` (Timeout und Mindestdatenrate für Anfragen), `mod_request` (Filter für den Anfrage-Body), `mod_slotmem_shm` (ein Slot-basierter Shared-Memory-Provider wie das Scoreboard) und `mod_log_debug` (Debug-Protokollierung mithilfe von Lua). Die von diesen Modulen bereitgestellten Konfigurationsdirektiven und weitere Möglichkeiten werden in den jeweils thematisch passenden Kapiteln dieses Buches besprochen.
- **Verbesserung vorhandener Module**
  In Apache 2.4 wurden unter anderem folgende Module verbessert und erweitert: `mod_ssl` (Überprüfung von Client-Zertifikaten durch einen OSCP-Server; Session-Sharing über `memcached`; siehe Kapitel 10, »Gesicherte Verbindungen«), `mod_proxy` und `mod_proxy_balancer` (unter anderem Performance-Verbesserungen; siehe Kapitel 13, »Proxy- und Cache-Funktionen«), `mod_cache` (Caching von `HEAD`-Anfragen, Konfiguration vieler Einstellungen pro Verzeichnis, `X-Cache`-Header mit Angaben über den Status einer Ressource; siehe Kapitel 13) sowie `mod_cgi`, `mod_include` und `mod_isapi` (jeweils eine strengere Überprüfung bei der Umwandlung von Anfrage-Headern in Umgebungsvariablen zur Verhinderung von Header-Injection, einer Form der Cross-Site-Scripting-Attacke).

### Neuerungen in Apache 2.2

Auch die Apache-Version 2.2 war im Vergleich zur vorigen Version 2.0 keine völlige Neuentwicklung, enthielt aber dennoch einige wichtige Verbesserungen und Erweiterungen, die in Version 2.4 größtenteils weiter existieren. Hier die wichtigsten Änderungen im Überblick:

- **Neuordnung der Authentifizierungsmodule**
  (siehe Kapitel 9, »Authentifizierung«). Die Apache-Module zur Authentifizierung und Zugriffskontrolle wurden vollständig neu organisiert. Dies ermöglicht Administratoren eine flexiblere Steuerung der Authentifizierung und erleichtert zudem die Programmierung von Erweiterungen: Es handelt sich um ein zweistufiges Modul, in dem die Module `mod_auth_basic` und `mod_auth_digest` die Anmeldedaten mit dem Browser austauschen (im Klartext beziehungsweise verschlüsselt), während sogenannte *Provider-Module* sich um die Verwaltung der Vergleichsdaten kümmern, beispielsweise in Textdateien, DBM-Files, SQL-Datenbanken oder LDAP-Verzeichnissen. Wer eine weitere Speichermethode implementieren möchte, braucht so nur noch ein neues Provider-Modul zu verfassen, ohne die stets gleiche Logik der Client-Kommunikation neu schreiben zu müssen.
- **Eingebaute Unterstützung für SQL-Datenbanken**
  Das Modul `mod_dbd` kann mit SQL-basierten Datenbanken wie MySQL, Oracle oder Microsoft SQL Server kommunizieren. Dies ist für die Entwicklung eigener Module mit Datenbankunterstützung nützlich, etwa zur Authentifizierung, für Logging-Aufgaben oder für das Content-Management. Da die erste mitgelieferte Nutzeranwendung der Authentifizierungs-Provider `mod_authn_dbd` ist, wird dieses Modul in Kapitel 9, »Authentifizierung«, behandelt.
- **Proxy- und Cache-Erweiterungen**
  (siehe Kapitel 13, »Proxy- und Cache-Funktionen«). Seit Apache 2.2 werden zwei neue Proxy-Module mitgeliefert: `mod_proxy_ajp` für das Apache JServ Protocol, das z. B. von Tomcat eingesetzt wird, sowie `mod_proxy_balancer` für Proxy-Load-Balancing. Auch die oft mit dem Proxy-Einsatz verbundenen Cache-Funktionen wurden verbessert und gelten nicht länger als experimentell.
- **Graceful Stop**
  Apache 2.2 lässt sich sauber herunterfahren, das heißt, er wickelt zunächst alle laufenden Client-Anfragen ab und beendet erst dann den Hauptprozess. Bisher gab es lediglich die Möglichkeit eines sauberen Neustarts. Dieses Feature wird in Kapitel 5, »Apache in Betrieb nehmen«, beschrieben.
- **Event-MPM**
  Die weiter unten beschriebenen Multiprocessing-Module zur Anpassung des Server-Laufzeitverhaltens wurden um eine neue Variante erweitert: Das Event-MPM beseitigt den Nachteil bisheriger Thread-basierter MPMs, die Threads bei Keepalive-Verbindungen stets dauerhaft mit ein und derselben Anfrage zu verknüpfen. In Apache 2.2 galt das Event-MPM noch als experimentell, erst in 2.4 ist es als stabil gekennzeichnet.

- **Smart Filtering**
  Das neu eingeführte Modul `mod_filter` ermöglicht es Ihnen, die Reihenfolge der Ausgabefilter (Filter Chain) per Konfiguration selbst festzulegen und von Headern oder Umgebungsvariablen abhängig zu machen. Mehr darüber lesen Sie in Kapitel 16, »SSI und Filter«.
- **Schlankere Konfiguration**
  Die Hauptkonfigurationsdatei *httpd.conf* enthält nun nur noch die wichtigsten Einstellungen; optionale Zusatzdirektiven wurden – thematisch sortiert – in das Unterverzeichnis *extra* ausgelagert und können mithilfe von Include-Direktiven aktiviert werden. Genaueres darüber steht in Kapitel 6, »Grundkonfiguration«.

### Die Apache Portable Runtime

Eine der wichtigsten Neuerungen, die mit Version 2 eingeführt wurde, ist die Verwendung einer »Plattform-Abstraktionsschicht«. Dies verbessert die Stabilität und Performance des Servers insbesondere auf Nicht-UNIX-Systemen. Alle früheren Versionen verwendeten unter UNIX für den Zugriff auf Speicher, Dateien, Netzwerkfunktionen usw. die standardisierten POSIX-Systemaufrufe. Auf Plattformen wie Windows wurde deshalb eine nicht ganz so leistungsfähige POSIX-Emulation verwendet, auf die die Apache-Funktionalität aufsetzen konnte.

Die Apache Portable Runtime verbessert die Situation erheblich: Sie setzt auf die nativen Funktionen des jeweiligen Betriebssystems auf, statt umständlich POSIX zu emulieren, und bietet so eine plattformneutrale API für den eigentlichen HTTP-Server und seine Erweiterungen. Die Aufgabe der APR ist also vergleichbar mit derjenigen der virtuellen Maschine für Java-Programmierer oder mit derjenigen der DirectX-API für die Windows-Multimedia- und Spieleprogrammierung.

Die APR ist die logische Fortsetzung einer Entwicklung, die einst bei Apache 1.3 begann: Die API dieser Version enthielt bereits zahlreiche Funktionen mit dem Präfix `ap_`, also verallgemeinerte Apache-Grundfunktionen.

Die Apache Portable Runtime abstrahiert vor allem die folgenden Funktionen von Betriebssystem und Bibliotheken:

- Dateizugriffe
- Zeichensatzkonvertierung
- Netzwerk-Sockets
- Datums- und Uhrzeitfunktionen
- Text- und Zeichenkettenbehandlung
- UNIX-artige Passwortverwaltung

- Tabellendatenverwaltung
- Erzeugung von UUIDs (weltweit einmaligen ID-Nummern)
- Angleichung von Datei- und Pfadnamen in ein vom Dateisystem unabhängiges Format
- Zufallsgenerator
- Sperrverfahren für Dateien, Prozesse usw.
- Thread- und Prozessverwaltung
- Laden dynamischer Bibliotheken
- Speicherverwaltung

Wie Sie an der Liste erkennen können, gehen die Aufgaben der Apache Portable Runtime inzwischen weit über die Bereitstellung von POSIX-Alternativen hinaus. Es bot sich einfach an, nach und nach alle verallgemeinernswerten Funktionen in diese Schicht auszulagern – es entspricht dem Paradigma des modernen Software-Engineerings (dem ein Vorzeigeprojekt wie der Apache-Webserver genügt), um das Rad niemals zweimal zu erfinden und alles so weit wie möglich zu modularisieren und zu verallgemeinern.

Die APR hat für Programmierer ebenfalls einiges zu bieten; sie dient nämlich verständlicherweise nicht nur zur Implementierung der Kernfunktionalität des HTTP-Servers, sondern ermöglicht auch die Programmierung von Modulen, die genauso plattformübergreifend einsetzbar sind wie der Server selbst. Einige Informationen darüber gibt es in Kapitel 17, »Apache erweitern«.

Inzwischen wird die Apache Portable Runtime nicht mehr nur für den Apache-HTTP-Server als plattformneutrale Grundlage eingesetzt, sondern auch für einige andere Open-Source-Projekte. Die APR-Website (*http://apr.apache. org*) listet zurzeit folgende Projekte auf:

- *Apache Flood* – ein Performance-Testprogramm für HTTP-Server, siehe *http://httpd.apache.org/test/flood*
- *JXTA-C* – eine offene Peer-to-Peer-Plattform, siehe *http://www.jxta.org*
- die *Tomcat-Module* `mod_j` und `mod_webapp`, siehe *http://tomcat. apache.org*
- *Subversion* – eine moderne Alternative zu CVS, realisiert als Apache-Modul, *siehe http://subversion.tigris.org/*
- *OPENdj* – ein Streaming-Server, siehe *http://www.opendj.org*
- *mod_spin* – ein Drittanbieter-Apache-Modul mit diversen Template- und Tracking-Fähigkeiten, siehe *http://www.rexursive.com/software/modspin*

Darüber hinaus verwendet der Softwarehersteller *Covalent* die APR in seinen kommerziellen Enterprise-Erweiterungen für Apache.

### Multiprocessing-Module

Bevor weiter unten die verschiedenen Multiprocessing-Module (MPMs) für Apache 2 vorgestellt werden, erhalten Sie hier eine kurze Information über das Problem, das diese Spezialmodule lösen. Dazu ist es wichtig, dass Sie sich die Aufgabe eines Netzwerk-Servers vor Augen führen: Der Server lauscht an einem bestimmten TCP-Port auf eingehende Verbindungen. Sobald eine Verbindungsanforderung ankommt, stellt er die Verbindung mithilfe des TCP-Drei-Wege-Handshakes her und verarbeitet die Client-Anfrage. Da lauschende TCP-Sockets so beschaffen sind, dass sie eine gewisse Anzahl wartender Client-Anfragen puffern können, würde der einfachste denkbare Fall eines TCP-Servers wie ein Supermarktkassierer einfach eine Anfrage bearbeiten, anschließend die Verbindung wieder beenden und erst dann die nächste Anforderung entgegennehmen.

In der Praxis ist dieses Verfahren allerdings absolut nicht zu empfehlen: Die Verarbeitung jeder einzelnen Anfrage kann unter Umständen so lange dauern, dass es noch vor dem TCP-Verbindungsaufbau zum Timeout kommt. Bei einem HTTP/1.0-Server wäre dies nicht das größte Problem, weil er keine persistenten Verbindungen zulassen muss. Bei vielen anderen TCP-Servern wie HTTP/1.1, FTP, Telnet oder den diversen Mailprotokollen sind länger dauernde Verbindungen dagegen üblich. Abgesehen davon würden die Ressourcen des Server-Rechners auf diese Weise fast ununterbrochen brachliegen, weil die Geschwindigkeit durch das Netzwerk-I/O bestimmt würde. Dieses ist aber selbst bei der im Breitband-Maximalbereich befindlichen Internet- oder LAN-Verbindung immer langsamer als die eigentliche Rechengeschwindigkeit des Computers.

Der langen Rede kurzer Sinn: Es muss eine Möglichkeit her, wie der Server mehrere Verbindungsanforderungen annehmen und parallel verarbeiten kann. Dies ist ein Problem der *Gleichzeitigkeit* oder *Nebenläufigkeit* (Concurrency). Dafür wurden im Laufe der Jahre zahlreiche unterschiedliche Lösungen entwickelt. Die wichtigsten sind folgende:

- **Forking-Server**
  Der Name dieses Server-Modells stammt von dem POSIX-Systemaufruf `fork()`. Dies ist unter UNIX die übliche Methode, einen neuen Prozess zu erzeugen: Vom ursprüngliche Prozess – Parent-Prozess genannt – wird eine völlig identische Kopie angefertigt (der Child-Prozess), die alle Variablen und geöffneten Dateideskriptoren des Parent-Prozesses erbt. Beachten Sie allerdings, dass es sich aus Programmierersicht zwar um Variablen mit denselben Namen und Anfangswerten handelt, dass Änderungen in einem der beiden Prozesse aber keinen Einfluss auf die Variablen des anderen Prozesses haben. Dasselbe gilt für den Fall, dass Sie in einem Prozess Dateideskriptoren schließen, die im anderen offen bleiben. Die beiden Prozesse lassen sich anhand des

Rückgabewerts von `fork()` unterscheiden: Im Parent-Prozess ist es die PID des neuen Child-Prozesses, im Child-Prozess 0.

Ein Forking-Server verwendet ein lauschendes TCP-Socket, das in einer Schleife immer wieder den Systemaufruf `accept()` für den Verbindungsaufbau ausführt. Sobald dieser Befehl Erfolg hat, sobald also tatsächlich eine Verbindungsanfrage eintrifft, ruft das Programm `fork()` auf. Während der Parent-Prozess weiter Verbindungen akzeptiert, kümmert sich der Child-Prozess um die eingegangene Verbindung.

Dieses Modell ist der Klassiker unter den nebenläufigen Server-Modellen. Aus Gründen der Stabilität und Performance verwendet keines der verfügbaren Apache-MPMs dieses Verfahren. Der kleine Perl-Webserver, der im vorigen Kapitel vorgestellt wurde, arbeitet allerdings genau nach dieser Methode, weil sie sich am einfachsten implementieren lässt.

- **Preforking-Server**
  Ein Server, der sehr viele Anfragen pro Minute zu verarbeiten hat, kann durch das Forking beim Eingang jeder Anfrage leicht in die Knie gezwungen werden. Das Erzeugen neuer Prozesse ist nämlich zeit- und ressourcenaufwendig. Aus diesem Grund besteht ein verbessertes Server-Modell darin, bereits beim Start des Servers mehrere Child-Prozesse »auf Vorrat« zu erzeugen, die sich dann um die eingehenden Verbindungsanforderungen kümmern können. Durch Fine-Tuning kann die Leistungsfähigkeit eines solchen Servers noch gesteigert werden: Die Anzahl der vorab erzeugten Prozesse und die Mindestanzahl freier Prozesse, die erreicht werden muss, bevor neue erzeugt werden, können je nach Belastung des Servers angepasst werden.

  Dieses Verfahren war bis einschließlich Version 1.3 die einzige Methode, die Apache verwendete (zumindest auf UNIX-Systemen). In Apache 2 lebt es als `mpm_prefork` weiter und ist unter UNIX noch immer der Standard.

- **Threading-Server**
  Moderne Betriebssysteme wie die Windows NT-Familie, Linux ab Kernel 2.4 oder Mac OS X unterstützen eine schlanke, leichtgewichtige Alternative zu den schwerfälligen Prozessen: Genau wie ein Betriebssystem mehrere Prozesse parallel ausführen kann, können innerhalb eines Prozesses mehrere Threads laufen. Im Gegensatz zu Prozessen teilen sich alle Threads in einem Prozess denselben Speicherbereich und somit auch dieselben Variablen. Aufgrund dieser »Einsparungen« lassen sich Threads erheblich schneller und ressourcenschonender erzeugen als Prozesse.

  Das einzige Problem besteht hier darin, dass die verschiedenen Threads nicht gegeneinander abgeschirmt sind – dies kann bei schlechter Programmierung zu seltsamen Vermischungen zwischen den einzelnen Anfragen oder schlimmstenfalls sogar zu Sicherheitsproblemen führen.

Ein gewöhnlicher Threading-Server entspricht vom Ablauf her gesehen dem Forking-Server – bei Eingang einer Anfrage wird ein neuer Thread gestartet, der diese bearbeitet. Gängiger ist allerdings ein Pre-Threading-Modell, dessen Funktionsweise dem Preforking-Server ähnelt. Apache bietet diverse Pre-Threading-MPMs; unter Windows, wo Threads seit Langem zur Kernfunktionalität des Betriebssystems gehören, ist dieses Modell sogar Standard.

- **select()-Server**
  Überraschenderweise gibt es auch noch ein Server-Modell, das *keine* Nebenläufigkeit verwendet. Benannt ist diese Server-Variante nach dem POSIX-Systemaufruf `select()`. Dieser überprüft, ob ein Dateideskriptor (in diesem Fall genauer gesagt ein Netzwerk-Socket) gerade bereit zur Ein- oder Ausgabe ist. Ein `select()`-Server verwendet diesen Aufruf, um in einer Schleife nacheinander alle offenen Verbindungen abzufragen. In jedem Socket werden dann je nach dessen Zustand einige Bytes an Daten gelesen oder geschrieben; anschließend geht es mit dem nächsten weiter. Dieses Modell erspart sich den Aufwand der Prozess- oder Thread-Erzeugung völlig.

  Moderne Betriebssysteme besitzen einen alternativen Systemaufruf namens `poll()`. Dieser überprüft nicht nur einen Dateideskriptor oder ein Socket, sondern gleich eine ganze Gruppe davon. Für jedes Socket in der Gruppe gibt er einen Status zurück. Mit `poll()` lässt sich also eine noch schnellere Variante eines solchen Servers implementieren. Die oben angesprochenen Webserver `lighttpd` und `nginx` basieren auf dieser Methode.

  Es gibt zwar kein Apache-MPM, das eine dieser beiden Varianten ohne Nebenläufigkeit verwendet, aber zur Sicherheit setzen sie innerhalb der Worker-Prozesse oder -Threads trotzdem `select()` ein. Das neue Event-MPM (siehe unten) basiert sogar im Wesentlichen auf `poll()`-Funktionalität.

Wenn Sie Interesse an Details der Implementierung solcher unterschiedlichen Server haben oder sich für andere Aspekte der TCP/IP-Programmierung begeistern, empfehle ich Ihnen das (leider nicht mehr regulär lieferbare) Buch [Stein 2000].

Apache 2 löst das Problem der Nebenläufigkeit auf flexible Art und Weise, nämlich durch den Einsatz von Multiprocessing-Modulen (MPMs). Sie können sich also (in gewissen plattformbedingten Grenzen) aussuchen, wie der Server mehrere Anfragen beantworten soll. Im nächsten Kapitel wird erklärt, wie Sie bei der Kompilierung das für Sie passende Modul auswählen können. Zum Lieferumfang gehören folgende MPMs:

- `prefork`
  Ein klassisches Preforking-Modell; Standard unter UNIX. Beim Start erzeugt der Server durch Forking eine bestimmte (wählbare) Anzahl von Prozessen.

Diese Prozesse werden nach und nach verwendet, um die eingehenden Verbindungsanforderungen zu bearbeiten. Sobald eine Mindestanzahl freier Prozesse unterschritten wird, werden wieder einige neue erzeugt.

- `worker`
  Dies ist sozusagen eine Mischform, ein Prozess-Thread-Modell. Es profitiert gleichermaßen von der Performance-Steigerung durch Threads wie von der Stabilität eines klassischen Prozessmodells: Beim Start wird durch Preforking eine bestimmte Anzahl von Prozessen erzeugt. Jeder dieser Prozesse enthält eine feste Anzahl von Threads, die für die Verarbeitung von Verbindungen zuständig sind. Der Server wird durch Forking oder durch das Beenden von Prozessen an unterschiedliche Belastungen durch Anfragen angepasst.
- `event` (seit Apache 2.2)
  Diese neuere `worker`-Variante versucht, einen Nachteil von Keepalive-HTTP-Verbindungen auszugleichen: Bei allen anderen MPM-Modulen kann sich ein Prozess oder Thread, der eine solche Verbindung verarbeitet, während der gesamten Verbindungsdauer um nichts anderes kümmern. `event`-MPM verwendet dagegen einen eigenen Thread zum Speichern aller Persistenzinformationen und weist sie bei Bedarf jeweils einem beliebigen `worker`-Thread zu.
- `mpm_winnt`
  Dies ist das Standardmodell für Windows NT und seine Nachfolger. Gemäß den Gegebenheiten der Windows-Plattform setzt es voll und ganz auf Threads: Ein Parent-Prozess, der die grundlegende Steuerung des Servers übernimmt (Einlesen der Konfigurationsdaten, Neustart, Beenden), erzeugt beim Start einen einzigen Child-Prozess. Dieser enthält wiederum eine variable Anzahl von Threads, die sich um die Client-Verbindungen kümmern.
- `mpm_netware`
  Dieses MPM ist für Apache unter Novell NetWare optimiert. Es handelt sich um ein reines Pre-Threading-Modell: Ein Parent-Thread steuert eine variable Anzahl von Child-Threads, die sich um die einzelnen Verbindungen kümmern.
- `beos` (nur bis Version 2.2)
  Ein ebenfalls rein Thread-basiertes Modell für das Betriebssystem BeOS.
- `mpmt_os2`
  Das Standardmodell für IBM OS/2 funktioniert ähnlich wie das `perchild`-Modell (siehe Anhang B): Ein Parent-Prozess startet eine festgelegte Anzahl von Child-Prozessen, die jeweils eine variable Anzahl von `worker`-Threads verwalten.

Je nachdem, welches Laufzeitmodell Sie verwenden, stehen unterschiedliche Konfigurationsoptionen zur Verfügung, die dessen genaues Verhalten bei unterschied-

lich hoher Server-Belastung bestimmen. Diese Optionen werden in Kapitel 6, »Grundkonfiguration«, behandelt.

### 3.2.2 Apache-Module

Die vielleicht wichtigste und bekannteste Eigenschaft des Apache-HTTP-Servers ist seine Erweiterbarkeit durch *Module*. Jedes Modul fügt bestimmte zusätzliche Fähigkeiten zu Apache hinzu, deren Parameter durch einige neue Anweisungen in der Apache-Konfigurationsdatei eingestellt werden können.

Einige Module werden bereits in der Grundausstattung des Servers mitgeliefert und sind standardmäßig aktiviert. Andere sind zwar ebenfalls im Grundpaket enthalten, werden aber nur auf Verlangen einkompiliert beziehungsweise (bei Binärdistributionen) eingeschaltet. Zu guter Letzt gibt es noch Unmengen von Drittanbieter-Modulen für den Apache-Webserver.

Grundsätzlich können Module auf zweierlei Art und Weise eingebunden werden: statisch einkompiliert oder als Dynamic Shared Objects (DSOs). Letzteres ist die flexiblere Lösung, weil sich dynamisch eingebundene Module nachträglich ein- und ausschalten lassen. Eine nähere Betrachtung der Vor- und Nachteile dieser beiden Varianten des Modulbetriebs finden Sie im nächsten Kapitel.

Grob betrachtet gibt es vor allem die folgenden Gruppen von Modulen:

- **Autorisierung und Authentifizierung** (Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«)
  Diese Module ermöglichen die Zugriffsbeschränkung auf den Webserver oder auf einzelne Server-Verzeichnisse.
- **Serverseitige Programmierung** (Kapitel 14, »CGI«, und 15, »Technologien zur Webprogrammierung«)
  Einige der bekanntesten Apache-Module wie `mod_cgi`, `mod_perl` oder `mod_php` sind für die Einbindung von Programmiersprachen zuständig. Sie können Programme in diesen Sprachen schreiben, deren Ausgabe als Dokument an den Client ausgeliefert wird.
- **URL-Manipulation** (Kapitel 8, »Weiterleitungen und Indizes«)
  Mithilfe dieser Module können Sie die URL einer Anfrage nach einem bestimmten Schema umwandeln. Dies ermöglicht beispielsweise automatische Umleitungen, intelligente Linklisten oder Sitemaps.
- **Antwort-Manipulation** (Kapitel 7, »Header und MIME-Einstellungen«)
  Wie Sie ausführlich in Kapitel 2, »Funktionsweise von Webservern«, erfahren haben, bestehen HTTP-Antworten unter anderem aus zahlreichen Headern. Einige Module ermöglichen Ihnen das gezielte automatische Setzen oder

Ändern solcher Header. In diesen Zusammenhang gehören natürlich auch Module, die sich um die Ermittlung des passenden MIME-Types für den Antwort-Body kümmern.

- **Logging** (Kapitel 11, »Logging«)
  Für verantwortungsvolle Webmaster sind aussagekräftige Log-Dateien sehr wichtig. Mithilfe einiger Module können Sie den Standardinhalt der Apache-Log-Dateien um zusätzliche Informationen erweitern oder durch externe Programme sammeln, sortieren und manipulieren lassen.
- **Filter** (Kapitel 16, »SSI und Filter«)
  Eines der mächtigsten neuen Features der Version 2 ist das Vor- oder Nachschalten beliebiger Filter. Diese bieten die Möglichkeit, eingehende Daten vor der Verarbeitung durch den Server zu manipulieren (Eingabefilter) beziehungsweise die Server-Antwort vor der Auslieferung an den Client zu ändern (Ausgabefilter). Zumindest Ausgabefilter müssen noch nicht einmal zwangsläufig als Module realisiert sein: Es kann sich alternativ auch um externe Programme handeln, die bestimmte Anforderungen erfüllen.
- **Multiprotokoll-Unterstützung** (Kapitel 13, »Proxy- und Cache-Funktionen«, und 17, »Apache erweitern«)
  Wie bereits erwähnt, können Sie Apache nicht nur als HTTP-Server betreiben, sondern zusätzlich auch als Web-Cache und Proxy-Server, als FTP-Server oder als Server für andere TCP-basierte Anwendungsprotokolle. Einige Module sorgen für die entsprechenden Erweiterungen.

In Tabelle 3.1 finden Sie eine Liste derjenigen Module, die mit dem Quellcode-Paket des Apache-HTTP-Servers 2.2 bzw. 2.4 geliefert werden. In der Spalte »Automatisch aktiviert« finden Sie Informationen darüber, ob das entsprechende Modul bei der Kompilierung des Servers mit Standardoptionen automatisch einkompiliert wird oder nicht. Diese Information ist für die im nächsten Kapitel ausführlich beschriebene Build-Konfiguration von Bedeutung: Wenn Sie ein Modul weglassen möchten, das normalerweise aktiviert ist, müssen Sie die entsprechende `disable`-Option verwenden. Umgekehrt muss ein Modul, das automatisch deaktiviert würde, mithilfe einer `enable`-Option ausdrücklich hinzugefügt werden.

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| `mod_access_compat` (2.4) | ja | 6, 9 | Bereitstellung der veralteten Direktiven `Order`, `Allow`, `Deny` und `Satisfy` zur hostbasierten Zugriffskontrolle |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.2 und 2.4 gehören

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_authz_host (bis 2.0 mod_access) | ja | 6, 9 | Durch dieses Modul können Sie Zugriffe auf bestimmte Verzeichnisse einer Website anhand von Hostnamen und IP-Adresse des anfragenden Clients steuern. |
| mod_actions | ja | 13 | Dieses Modul ermöglicht die automatische Auslösung von CGI-Handlern anhand der Dateiendung beziehungsweise des MIME-Types von Dateien. |
| mod_alias | ja | 8 | mod_alias ist das klassische Modul für die einfache Umwandlung von URLs. Erheblich mehr Möglichkeiten bietet das neue Modul mod_rewrite (siehe unten). |
| mod_allowmethods (2.4) | ja | 6 | Beschränkung von Anfragen auf bestimmte HTTP-Methoden |
| mod_asis | ja | 7 | Mithilfe dieses Moduls können Sie eine Datei an den Client senden, »wie sie ist«. Das bedeutet, dass Apache für eine solche Datei keine automatischen HTTP-Header generiert; diese können Sie flexibel selbst bereitstellen. |
| mod_auth_basic (bis 2.0 Teilaufgabe von mod_auth und anderen Modulen) | ja | 9 | Mithilfe der durch dieses Modul gebotenen Basic-Authentifizierung können Sie Verzeichnisse einer Website schützen: Benutzer müssen sich mit Benutzernamen und Passwort anmelden, um Zugriff zu erhalten. |
| mod_auth_form (2.4) | ja | 9 | Authentifizierung mithilfe eines HTML-Formulars, das in der Konfiguration angegeben werden kann |
| mod_authn_anon (bis 2.0 mod_auth_anon) | nein | 9 | Bietet anonymen Zugriff mit dem Benutzernamen anonymous und der eigenen E-Mail-Adresse als Passwort. Diese Funktionalität wurde dem bekannten Anonymous FTP nachempfunden, ist aber auf einem Webserver in den meisten Fällen nicht sinnvoll. |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.2 und 2.4 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_authn_core (2.4) | ja | 9 | Grundmodul für die Authentifizierung (bis Version 2.2 war die Funktionalität im Webserver-Core enthalten) |
| mod_authn_dbd | nein | 9 | Dieses Modul liest das Vergleichspasswort für einen gegebenen Benutzernamen aus einer mittels mod_dbd hergestellten SQL-Datenbankverbindung. |
| mod_authn_dbm (bis 2.0 mod_auth_dbm) | nein | 9 | Diese Variante steigert die Effizienz, wenn Ihre Site viele Verzeichnisse mit unterschiedlichen Benutzernamen-Passwort-Paaren enthält: Die Anmeldedaten werden nicht in einer einfachen Passwortdatei gespeichert, sondern in datenbankähnlichen DBM-Dateien. |
| mod_auth_digest | nein | 9 | Dieses Modul verbessert die Sicherheit der Authentifizierung, indem es statt der BASIC-Authentifizierung mit base64-codierten Klartextdaten MD5-Digests gemäß RFC 2617 verwendet. |
| mod_authnz_ldap (bis 2.0 mod_auth_ldap) | nein | 9 | Dieses Modul ermöglicht die Benutzerauthentifizierung anhand existierender Daten aus LDAP-Verzeichnissen. |
| mod_authn_alias | nein | 9 | Bietet die Möglichkeit, verschiedene Authentifizierungs-Provider zu komplexeren Modellen zu verknüpfen |
| mod_authn_default (nur 2.2) | nein | 9 | Steuert die Weitervermittlung der Authentifizierung an untergeordnete Module |
| mod_authn_file | ja | 9 | Liest die Anmeldedaten aus einfachen Textdateien (bis Version 2.0 in mod_auth enthalten) |
| mod_authn_socache (2.4) | nein | 9 | Caching von Authentifizierungsdaten |
| mod_authz_core (2.4) | ja | 9 | Grundmodul für die Autorisierung und Zugriffskontrolle (bis Version 2.2 war die Funktionalität im Webserver-Core enthalten) |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.2 und 2.4 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_authz_dbm | nein | 9 | Gruppenberechtigungen anhand von DBM-Dateien |
| mod_authz_default (nur 2.2) | nein | 9 | Steuert die Weitervermittlung der Zugriffskontrolle an untergeordnete Module. |
| mod_authz_groupfile | ja | 9 | Liest Gruppeninformationen aus einfachen Textdateien (bis Version 2.0 in mod_auth enthalten) |
| mod_authz_owner | nein | 9 | Zugriffsberechtigung auf der Basis von Dateieigentumsrechten |
| mod_authz_user | nein | 9 | Zugriffsberechtigungen anhand von Benutzernamen (bis Version 2.0 in mod_auth enthalten) |
| mod_autoindex | ja | 8 | Dieses Modul stellt den Inhalt des angeforderten Verzeichnisses auf (in Grenzen) konfigurierbare Weise dar, wenn dieses Verzeichnis keine Indexseite enthält. Ideal für umfangreiche Download-Verzeichnisse, aber ein gewisses Sicherheitsrisiko, sofern es unkontrolliert aktiviert wird: Wenn ein Website-Autor den Namen der Startseite falsch schreibt oder in einem Unterverzeichnis keine Namen verwendet, wird die Liste aller Dateien angezeigt – bei wenig sicherheitsbewussten Webmastern können dies Dateien sein, die einen externen Besucher nichts angehen! |
| mod_bucketeer | nein | – | Dieses Modul ist ein Beispiel für einen Ausgabefilter. Er führt beim Auftreten spezieller Sonderzeichen Buckets (interne Apache-Funktionen) aus. |
| mod_buffer (2.4) | ja | 16 | Pufferspeicher für Ein- und Ausgabefilter-Stacks |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.2 und 2.4 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_cache | nein | 13 | Dieses Modul konfiguriert Apache als Web-Cache – dies ist neben der Funktion als Anwendungs-Gateway (die durch mod_proxy zur Verfügung gestellt wird) die wichtigste Aufgabe eines Proxy-Servers. Provider-Module (in Version 2.4 nur noch mod_cache_disk; früher mod_disk_cache und mod_mem_cache) stellen den eigentlichen Zwischenspeicher bereit. |
| mod_cache disk (bis 2.2: mod_disk_cache) | nein | 13 | mod_cache benötigt ein Provider-Modul, das entscheidet, wo der Zwischenspeicher zur Verfügung gestellt wird. Dieses Modul realisiert ihn auf der Festplatte. |
| mod_cern_meta | nein | 7 | Dieses Modul implementiert META-Files des CERN HTTPd (nur aus Kompatibilitätsgründen). Besser mod_headers verwenden! |
| mod_cgi | ja | 14 | Das Modul mod_cgi bietet die klassische Unterstützung für CGI-Skripte: Das entsprechende Programm wird als externer Prozess gestartet. Es enthält Formulardaten als Eingabe; seine Ausgabe wird an den Client weitergeleitet. |
| mod_cgid | ja | 14 | Wenn ein Thread-basiertes MPM-Modul verwendet wird, kann mod_cgid die Performance von CGI-Skripten verbessern, da sie von einem separaten Prozess als leichtgewichtige Threads ausgeführt werden. |
| mod_charset_lite | ja | 16 | Dieses Modul bietet Unterstützung für diverse nicht lateinische Zeichensätze und kann die entsprechende Konvertierung in bestimmten Fällen automatisch vornehmen. |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.2 und 2.4 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_dav | nein | 17 | mod_dav richtet Apache 2 als WebDAV-Server ein: Mehrere Autoren können auf den Server ähnlich wie auf ein CVS-Repository zugreifen und ihre Änderungen mit Versionsinformationen versehen. |
| mod_dav_fs | nein | 17 | Bei installiertem mod_dav ermöglicht dieses Zusatzmodul den WebDAV-Zugriff auf das lokale Dateisystem. |
| mod_dav_lock | nein | 17 | Stellt Locking-Funktionalität für mod_dav zur Verfügung. |
| mod_dbd | nein | 9 | Eingebaute Unterstützung für SQL-Datenbankverbindungen; erste eingebaute Praxisanwendung ist mod_authn_dbd (siehe oben), ansonsten auch interessant für die Entwicklung eigener Module. |
| mod_deflate | nein | 16 | Dieses Modul ermöglicht die Auslieferung deflate-komprimierter Daten gemäß RFC 2616 (siehe Kapitel 2, »Funktionsweise von Webservern«), was bei modernen Browsern, die dies unterstützen, Netzwerkbandbreite spart. |
| mod_dir | ja | 6 | Dieses Modul erledigt zwei wichtige Aufgaben: Es definiert den Namen der Startseite und ermöglicht die automatische Verarbeitung von Verzeichnis-URLs, die nicht mit einem / enden. |
| mod_dumpio | nein | 11 | Dieses Modul ermöglicht die Sammlung der vollständigen Ein- und/oder Ausgabedaten in einer Log-Datei. |
| mod_echo | nein | 17 | Dieses Modul ist als Demonstration der Multiprotokoll-Fähigkeiten von Apache 2 gedacht und sollte normalerweise nicht auf einem Produktionsserver verwendet werden. Es konfiguriert Apache zusätzlich zu HTTP als Server für das echo-Protokoll, das die übergebenen Daten zurücksendet. |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.2 und 2.4 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_env | ja | 14 | Dieses Modul ermöglicht die Manipulation von Server-Umgebungsvariablen für bestimmte Anfragen und Verzeichnisse. |
| mod_example | nein | 17 | Das Modul mod_example bietet keine praktische Funktionalität, sondern demonstriert allgemein, wie Apache-Module überhaupt funktionieren. Es sollte niemals auf Produktionsservern eingesetzt werden! |
| mod_expires | nein | 7 | Dieses Modul ermöglicht das Setzen des HTTP/1.1-Headers Expires, der die Gültigkeitsdauer von Dokumenten für Browser und vor allem Proxys angibt. |
| mod_ext_filter | nein | 16 | Über dieses Modul können Sie fast beliebige externe Programme als Ausgabefilter definieren. Dies ermöglicht beispielsweise Zeichensatz- oder gar Inhaltskonvertierungen. |
| mod_filter | nein | 16 | Dieses Modul ermöglicht eine genauere Kontrolle des Filterprozesses – vor allem die Steuerung der Filterabfolge (Filter Chain). |
| mod_file_cache | nein | 13 | Mithilfe dieses Moduls kann Apache 2 bestimmte Dateien beim Start ins RAM kopieren. Dies ermöglicht den schnelleren Zugriff auf häufig verwendete Daten. |
| mod_headers | nein | 7 | Über mod_headers können Sie fast alle HTTP-Header setzen, ändern oder entfernen. Nur einige wenige Header sind davon ausgenommen, weil sie erst nach der Verarbeitung der Direktiven dieses Moduls automatisch gesetzt werden. |
| mod_ident | nein | 11 | Abfrage der RFC-1413-Identität des Client-Hosts (bis Version 2.0 teilweise im Core) |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.2 und 2.4 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_imagemap (bis 2.0 irreführenderweise mod_imap) | ja | 8 | Dieses Modul bietet interne Unterstützung für die (selten gewordenen) serverseitigen Image Maps. |
| mod_include | ja | 16 | Mithilfe von mod_include wird die grundsätzliche Unterstützung für Server Side Includes bereitgestellt. Diese relativ alte Technologie ermöglicht das schnelle Einfügen dynamischer Daten wie Dateien, Variablen oder Datum und Uhrzeit. |
| mod_info | nein | 8 | Dieses Modul ermöglicht die Einrichtung einer virtuellen URL, über die sich ausführliche Konfigurationsinformationen des Servers abfragen lassen. |
| mod_isapi | nein | 15 | Unterstützung für die vom Microsoft Internet Information Services bekannte ISAPI-Schnittstelle (nur Windows) |
| mod_ldap | nein | 9 | Das Lightweight Directory Access Protocol hat sich in den letzten Jahren zum Standard für Verzeichnisdienste entwickelt; sowohl UNIX-Implementierungen wie OpenLDAP als auch Microsoft ActiveDirectory basieren darauf. Das Modul verbessert die grundsätzliche Zusammenarbeit von Apache mit LDAP-Diensten und stellt so beispielsweise Caching- und Connection-Pooling-Dienste für LDAP bereit, insbesondere für mod_auth[nz]_ldap. |
| mod_log_config | ja | 11 | Bereits ab Werk liefert Apache 2 recht aussagekräftige Log-Dateien. Mithilfe von mod_log_config können Sie noch allerlei Erweiterungen daran vornehmen und die Dateien durch externe Programme sortieren oder weiterverarbeiten lassen. |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.2 und 2.4 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_log_forensic | nein | 11 | Stellt forensische Log-Dateien bereit, die vollständige Client-Anfragen und deren korrekte Verarbeitung protokollieren – dies vereinfacht die Suche nach Fehlern oder Angriffen. |
| mod_logio | nein | 11 | Dieses Modul protokolliert zusätzlich zu den Standardinformationen die Anzahl der tatsächlich gesendeten beziehungsweise empfangenen Bytes in Log-Dateien. |
| mod_mem_cache (bis 2.2) | nein | 13 | Ähnlich wie bei mod_disk_cache können Sie mithilfe dieses Moduls den Zwischenspeicher für mod_cache vorkonfigurieren. Hier wird er im RAM eingerichtet. |
| mod_mime | ja | 7 | Dieses Modul setzt den wichtigen Content-Type-Header der Server-Antwort automatisch anhand der Dateinamenerweiterung. |
| mod_mime_magic | nein | 7 | mod_mime_magic untersucht einige Bytes des Dateiinhalts, um den MIME-Type für die Antwort bei Bedarf noch sorgfältiger zu ermitteln als mod_mime. |
| mod_negotiation | ja | 7 | Dieses Modul ist für die verschiedenen Arten der Content-Negotiation – die automatische Lieferung unterschiedlicher Inhalte als Reaktion auf Accept*-Anfrage-Header – zuständig. |
| mod_nw_ssl | nein | 10 | gesicherte SSL-Verbindungen für die Apache-Version unter Novell Netware |
| mod_proxy | nein | 13 | Dieses Modul stellt die Grundfunktion für Apache als Proxy-Server zur Verfügung. Einige weitere Module richten spezielle Proxy-Dienste wie HTTP- oder FTP-Proxy ein. |
| mod_proxy_ajp | nein | 13 | In Zusammenarbeit mit mod_proxy stellt dieses Modul einen Proxy-Dienst für das Apache JServ Protocol (AJP) Version 1.3 bereit, das unter anderem von Tomcat verwendet wird. |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.2 und 2.4 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_proxy_balancer | nein | 13 | Dieses Modul erweitert mod_proxy um Load-Balancing-Funktionalität für HTTP, FTP und AJP. |
| mod_proxy_connect | nein | 13 | Zusammen mit mod_proxy stellt dieses Modul eine Implementierung der HTTP-Methode CONNECT zur Verfügung: Der Proxy-Server überträgt die Daten gesicherter Verbindungen (SSL), ohne ihren Inhalt – den er ohnehin nicht verstehen kann – zu modifizieren. |
| mod_proxy_fcgi (2.4) | nein | 13, 14 | Mit diesem Modul dient mod_proxy als Proxy für FastCGI-Prozesse, die z. B. durch fcgistarter bereitgestellt werden. |
| mod_proxy_fdpass (2.4) | nein | 13 | Ein Socket-Proxy übergibt das Socket einer Client-Verbindung als Unix-Domain-Socket an einen anderen Prozess. |
| mod_proxy_ftp | nein | 13 | Dieses Modul stellt einen FTP-Proxy-Server bereit, der zusammen mit mod_proxy funktioniert. |
| mod_proxy_http | nein | 13 | Wenn mod_proxy installiert ist, bietet dieses Modul die wichtigste Komponente eines Web-Proxy-Servers, nämlich einen einfachen HTTP-Proxy. |
| mod_proxy_scgi | nein | 13, 14 | Ermöglicht es mod_proxy, als SCGI-Proxy (Simple CGI) zu arbeiten. |
| mod_ratelimit (2.4) | nein | 18 | Limitiert die verfügbare Bandbreite für Clients. |
| mod_reflector (2.4) | nein | 16 | Verarbeitet den Anfrage-Body mithilfe der definierten Ausgabefilter und liefert ihn wieder an den Client zurück. |
| mod_remoteip (2.4) | nein | 12, 13 | Ersetzt die Client-IP-Adresse in einer Proxy- oder Load-Balancer-Anfrage durch die in einem speziellen Header befindliche eigentliche IP-Adresse. |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.2 und 2.4 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_reqtimeout | nein | 6 | Ermöglicht das Setzen von Timeout und Mindest-Datenrate für Client-Anfragen. |
| mod_request (2.4) | ja | 16 | Filter, um den Anfrage-Body verfügbar zu machen und zu verarbeiten (auch bei GET-Anfragen) |
| mod_rewrite | nein | 8 | Dieses Modul ermöglicht die fast beliebige Umwandlung von URLs aus HTTP-Anfragen mithilfe regulärer Ausdrücke. |
| mod_sed (2.4) | nein | 16 | Ein- und Ausgabefilter mit sed(1)-Syntax |
| mod_session (2.4) | nein | 14 | Unterstützung für Sessions (die normalerweise auf der Ebene der Webanwendungen bereitgestellt werden) |
| mod_session_cookie (2.4) | nein | 14 | Speicherung der durch mod_session verwalteten Session-Daten in einem clientseitigen Cookie |
| mod_session_crypto (2.4) | nein | 14 | Verschlüsselung der durch mod_session verwalteten Session-Daten |
| mod_session_dbd (2.4) | nein | 14 | Speicherung der durch mod_session verwalteten Session-Daten in einer serverseitigen SQL-Datenbank |
| mod_setenvif | ja | 14 | Dieses Modul ermöglicht die Manipulation von Server-Umgebungsvariablen als Reaktion auf bestimmte HTTP-Anfrage-Header. |
| mod_slotmem_plain (2.4) | nein | (17) | Slot-basierter Shared-Memory-Provider (nur für Modulentwickler interessant) |
| mod_slotmem_shm (2.4) | nein | (17) | Slot-basierter Shared-Memory-Provider, prozessübergreifend (nur für Modulentwickler interessant) |
| mod_so | ja | 6 | Liefert die Funktionsgrundlage für das dynamische Laden von Modulen. |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.2 und 2.4 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_speling | nein | 8 | Dieses Modul kümmert sich um die automatische Korrektur von Schreibfehlern in Anfrage-URLs, vor allem bezüglich Groß- und Kleinschreibung. Das Modul heißt tatsächlich mod_speling mit nur einem »l«: eben ein Rechtschreibfehler. |
| mod_ssl | nein | 10 | mod_ssl stellt die grundsätzliche Unterstützung für gesicherte HTTP-Verbindungen über SSL beziehungsweise TLS zur Verfügung. |
| mod_status | ja | 8 | Dieses Modul liefert über eine virtuelle URL oder mithilfe des Textbrowsers Lynx und einer Kommandozeilenoption für das Programm httpd Statusinformationen über den laufenden Server. |
| mod_substitute (seit 2.2.7) | nein | 16 | Textersetzung im Body von Ausgabedokumenten anhand regulärer Ausdrücke oder fester Strings |
| mod_suexec | nein | 18 | suexec ermöglicht es, CGI-Skripte unter einer bestimmten User- und Group-ID auszuführen, was bei sorgfältiger Konfiguration die Sicherheit verbessert. |
| mod_unique_id | nein | – | Dieses Modul ermöglicht es, jeder einzelnen Client-Anfrage eine eindeutige ID zuzuweisen. Dies kann für bestimmte Aspekte der serverseitigen Programmierung und für Logging-Zwecke hilfreich sein. |
| mod_unixd | ja (UNIX) | 6, 18 | Enthält einige nur für UNIX wichtige Basisdirektiven, die vormals zum Webserver-Core gehörten |
| mod_userdir | ja | 8 | Dieses Modul bildet die Home-Verzeichnisse der Benutzer (beziehungsweise ein konfigurierbares Unterverzeichnis davon) nach dem Schema *http://servername/~user* automatisch auf den Webserver ab. |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.2 und 2.4 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_usertrack | nein | 11 | Dieses Modul ermöglicht das Session-Tracking durch Cookies. Sie müssen die Genehmigung der Besucher einholen, um es zu verwenden. Außerdem ist der Nutzen fraglich, da etliche Benutzer die Annahme von Cookies verweigern. |
| mod_version | ja | 6 | Bietet die Möglichkeit, Konfigurationsanweisungen versionsabhängig zu setzen. |
| mod_vhost_alias | nein | 12 | Dies ist das wichtigste Modul für Hoster: Es ermöglicht die automatische Erzeugung von Unmengen virtueller Hosts anhand von Datei- und Verzeichnismustern. |
| mod_xml2enc (2.4; für ältere Versionen als Drittanbieter-Modul) | nein | 16 | verbesserte Zeichensatz-/Encoding-Unterstützung für libxml2-basierte Filtermodule |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.2 und 2.4 gehören (Forts.)

In diesem Buch werden noch einige weitere wichtige Module besprochen. Im Gegensatz zu denjenigen aus Tabelle 3.1 lassen sie sich aber nicht durch standardisierte Konfigurationsparameter einkompilieren, weil sie nicht zum Lieferumfang gehören. Zu dieser Kategorie gehören unter anderem folgende Module:

- mod_perl
  Wie bereits erwähnt, ist dieses mächtige Modul ein weiteres Projekt der Apache Software Foundation. Es ist nicht nur ein beschleunigter Ersatz für Perl-CGI-Skripte, sondern ermöglicht darüber hinaus die Erweiterung des Apache-Webservers durch selbst geschriebene sowie öffentlich verfügbare Perl-Module. mod_perl wird in diesem Buch in Kapitel 15, »Technologien zur Webprogrammierung«, behandelt.
- mod_php
  Dies ist laut Security Space (*http://www.securityspace.com/s_survey*) das beliebteste aller im Einsatz befindlichen Drittanbieter-Apache-Module. Die Programmiersprache PHP besticht durch eine einfach zu erlernende Syntax, eine leistungsfähige API und das Vorhandensein von Schnittstellen zu zahlreichen Datenbanken (am bekanntesten dürfte in diesem Zusammenhang MySQL sein). In Kapitel 15, »Technologien zur Webprogrammierung«, geht es unter anderem um PHP.

- `mod_ruby`
  Die Skriptsprache Ruby stammt aus Japan und ist konsequent objektorientiert. Bekannt wurde sie vor allem durch das Web-Framework Ruby on Rails (dafür gibt es inzwischen ein eigenes Modul namens Passenger oder auch `mod_rails`). `mod_ruby` ermöglicht die Einbindung serverseitiger Programme, die in Ruby geschrieben sind, ohne das Rails-Framework zu nutzen. Auch `mod_ruby` wird in Kapitel 15, »Technologien zur Webprogrammierung«, angesprochen.

Neben den wenigen hier aufgeführten Modulen finden Sie zahlreiche weitere in der zentralen Registry für Apache-Module unter *http://modules.apache.org*. Beachten Sie, dass manche Module noch immer nur für Apache 1.3 verfügbar sind und nicht mehr aktualisiert werden. Darüber hinaus genügen nicht alle dort angebotenen Module den hohen Performance- und Sicherheitsstandards, die Sie vom Apache-Webserver selbst erwarten können. Für einen Produktionsserver sollten Sie sich also nicht auf ein solches Modul verlassen, ohne zusätzliche Informationen oder Erfahrungsberichte einzuholen.

## 3.3 Zusammenfassung

Der Apache-HTTP-Server ist der mit Abstand erfolgreichste Webserver überhaupt. Kein Wunder, schließlich ist er nicht nur frei verfügbar, sondern auch stabil, sicher und höchst belastbar. Sein Einsatz auf vielen der beliebtesten Websites der Welt beweist dies auch in der Praxis.

Für die Verwendung dieses Servers spricht darüber hinaus, dass er auf zahlreichen Plattformen und Betriebssystemen funktioniert. Durch die Apache Portable Runtime (APR), die mit Version 2.0 eingeführt wurde, arbeitet der Server auch auf Nicht-UNIX-Systemen mit optimaler Performance. Hinzu kommen die verschiedenen Multiprocessing-Module (MPMs), die es ermöglichen, spezifische Leistungsvorteile einzelner Systeme zu nutzen.

Ein weiterer Pluspunkt sind die zahlreichen im Lieferumfang enthaltenen Module, die es ermöglichen, den Webserver optimal an individuelle Bedürfnisse anzupassen. Über dieselbe Schnittstelle lassen sich obendrein die verschiedensten Drittanbieter-Module integrieren, die den Funktionsumfang von Apache 2 nochmals erweitern.

*»Wie schön ist es zu säen, damit geerntet werde!«*
*– Johann Wolfgang von Goethe*

# 4 Apache kompilieren und installieren

Der erste Schritt zur Verwendung von Software ist ihre Installation. Bei kommerziellen Anwendungsprogrammen ist diese in der Regel nicht weiter der Rede wert, weil es nur eine einzige Installationsmethode mit einem offiziellen Installationsprogramm des Herstellers gibt. In der Welt der freien Software ist auch dies anders: Apache 2 wird in vielen verschiedenen »Darreichungsformen und Packungsgrößen« angeboten, und je nach Bedarf können Sie sich die passende Variante aussuchen. Im Wesentlichen gibt es drei verschiedene Arten von Apache-Installationspaketen:

- Quellcode-Pakete zum Selbstkompilieren
- binäre Distributionen für verschiedene Plattformen
- Installationspakete für bestimmte Betriebssystemdistributionen

Das eigenständige Kompilieren des Apache-Webservers ist die klassische Installationsmethode und bietet Ihnen die flexibelste Kontrolle über den Aufbau Ihrer Webserver-Software. In diesem Kapitel wird die Kompilierung mit `gcc` unter UNIX sowie mit Microsoft Visual Studio unter Windows beschrieben.

Die einzige standardisierte Binärdistribution der Apache Group ist diejenige für Windows – ein Windows-Installer-Paket (MSI), dessen Funktionsweise der üblichen Software-Installationsprozedur unter Windows entspricht. Deshalb wird die Verwendung dieses Pakets hier ebenfalls beschrieben, zumal es die häufigste Installationsvariante auf Windows-Systemen ist.

Einige Betriebssystemdistributionen besitzen spezielle Paketmanager, die die gezielte Installation und Deinstallation kompletter Softwarepakete ermöglichen. Als eines der am häufigsten installierten Programme auf UNIX-Systemen wird natürlich auch Apache in diesem Format angeboten. Für nähere Informationen sollten Sie die Dokumentation der jeweiligen Systemdistribution konsultieren. In meinem Buch »openSUSE 11.2« (3. Auflage, Galileo Computing, Bonn 2010) wird die Einrichtung und Konfiguration von Apache 2 mit den eingebauten Tools von openSUSE genau beschrieben; in Kapitel 5, »Apache in Betrieb nehmen«, fin-

den Sie einige kurze Hinweise dazu sowie zu entsprechenden Bordmitteln von Mac OS X.

Wie Sie sehen, konzentriert sich dieses Kapitel auf UNIX- und Windows-Systeme, da diese in der Praxis die gängigsten Plattformen für Apache sind. Wenn Sie den Server unter einer der zahlreichen anderen unterstützten Plattformen wie etwa OS/2, Novell NetWare oder VMS nutzen wollen, konsultieren Sie bitte die Online-Dokumentation unter *http://httpd.apache.org* und die Handbücher Ihres Betriebssystems.

## 4.1 Apache 2 kompilieren

Einer der unschätzbaren Vorteile freier Software besteht darin, dass sie sich aus den Quellcodes kompilieren und damit erheblich flexibler anpassen lässt als kommerzielle oder sonstige binär ausgelieferte Software. Der Kompiliervorgang ist dabei nicht einmal schwierig und schon gar nicht nur für Hacker zu verstehen: Dank der modernen Installerwerkzeuge GNU Autoconf und Automake besteht er im Wesentlichen aus dem Aufruf dreier festgelegter Befehle, deren erster – das Konfigurationsskript – durch zahlreiche Optionen angepasst werden kann. In diesem Abschnitt wird die Kompilierung unter UNIX sowie unter Windows beschrieben; für andere Plattformen, die Apache unterstützt, funktioniert sie anders.

### 4.1.1 Den Quellcode besorgen und auspacken

Bevor Sie Apache kompilieren können, müssen Sie sich das aktuelle Quellcode-Paket beschaffen. Auf der beiliegenden DVD finden Sie die derzeit stabile Version 2.2.21 im Verzeichnis *apache2_2* und die aktuelle Beta-Version 2.4, nämlich 2.3.14, unter *apache2_4*.

Um ein neueres Release herunterzuladen, schicken Sie Ihren Browser auf die Website *http://httpd.apache.org/download.cgi*. Diese Download-Site wurde so weit automatisiert, dass Sie zu einem passenden Mirror in Ihrer Nähe weitergeleitet werden; falls dieser einmal ausgefallen oder überlastet sein sollte, können Sie manuell einen anderen wählen. In jedem Fall ist es aus Traffic-Gründen nicht empfehlenswert, die Dateien von *apache.org* selbst zu beziehen.

Klicken Sie auf der Download-Site den Link hinter Unix Source an. Zurzeit heißt die Datei *httpd-2.2.21.tar.gz* (oder die Variante *httpd-2.2.21.tar.bz2*, die bzip2- statt GNU ZIP-komprimiert und dadurch etwas kleiner ist).

Für die 2.4-Beta-Version müssen Sie zurzeit noch zwei Dateien herunterladen; eine Hauptdatei und eine mit Dependencies (insbesondere APR). In der aktuellen

Version heißen die beiden Dateien *httpd-2.3.15-beta.tar.bz2* und *httpd-2.3.15-beta-deps.tar.bz2*; gz-komprimierte Varianten sind derzeit nicht verfügbar.

Da die Dateien von einem Mirror stammen, ist es mehr als ratsam, ihre Identität elektronisch zu überprüfen, damit Ihnen niemand eine defekte Version (oder gar einen Trojaner!) unterschieben kann. Deshalb gibt es zu jeder Datei einen MD5-Hash und eine PGP-signierte Kontrolldatei. Die technischen Grundlagen solcher Verfahren werden übrigens in Kapitel 10, »Gesicherte Verbindungen«, näher erläutert.

Zu empfehlen ist die PGP-Variante: Die digitale Signatur eines Mitglieds der Apache Group garantiert die Echtheit der Datei. Wenn Sie den Link [PGP] neben Ihrer gewünschten Download-Datei anklicken, erhalten Sie die PGP-Signatur für die Datei. Diese trägt denselben Namen wie der Download mit der zusätzlichen Endung `.asc` – Beispiel: *httpd-2.2.21.tar.gz.asc*.

Damit ein Vergleich überhaupt möglich ist, müssen Sie sich zunächst die Schlüsseldatei mit den öffentlichen PGP-Schlüsseln der Entwickler vom Server der Apache Software Foundation besorgen. Diese Datei ist unter *http://www.apache.org/dist/httpd/KEYS* verfügbar. Sie müssen sie in PGP importieren. Je nach PGP-Variante (PGP oder GnuPG) wird dazu *einer* der drei folgenden Befehle verwendet:

```
# pgpk -a KEYS
# pgp -ka KEYS
# gpg --import KEYS
```

Anschließend wird der entsprechende Befehl zur Überprüfung der Integrität eingegeben. In den verschiedenen PGP-Versionen funktioniert auch dies unterschiedlich:

```
# pgpv httpd-2.2.21.tar.gz.asc
# pgp httpd-2.2.21.tar.gz.asc
# gpg --verify httpd-2.2.21.tar.gz.asc
```

Wenn die Datei in Ordnung ist, erhalten Sie nun eine Meldung wie diese:

```
gpg: Signature made 10/10/05 03:35:15 using RSA key ID 10FDE075
gpg: Good signature from "wrowe@covalent.net"
gpg:            aka "William A. Rowe, Jr. <wrowe@rowe-clan.net>"
gpg:            aka "wrowe@lnd.com"
gpg:            aka "wrowe@apache.org"
```

Unter Windows funktioniert das Ganze noch einfacher, weil es komfortable grafische Versionen wie die Freeware Gpg4win (*https://www.gpg4win.org*) gibt. Während der Installation können Sie gleich Ihr persönliches Schlüsselpaar erstellen, mit dem Sie in Zukunft E-Mails und Dateien verschlüsseln oder digital signie-

ren können. Eine Alternative ist das bereits erwähnte Konsolenprogramm GnuPG, das auf Windows-Rechnern genauso funktioniert wie beschrieben.

Entpacken Sie die heruntergeladene Datei nach der Integritätsprüfung mithilfe der folgenden Schritte:

```
# gunzip httpd-2.2.21.tar.gz
# tar xvf httpd-2.2.21.tar
```

Aktuelle `tar`-Versionen können GNU ZIP-Dateien selbst entpacken. In diesem Fall können Sie die beiden Schritte zu folgendem Befehl zusammenfassen:

```
# tar xzvf httpd-2.2.21.tar.gz
```

Nun können Sie in das neu erstellte Verzeichnis (`httpd-2.2.21`) wechseln:

```
# cd httpd-2.2.21
```

Die bz2-Versionen werden folgendermaßen in zwei Schritten entpackt:

```
# bunzip2 httpd-2.2.21.tar.bz2
# tar xvf httpd-2.2.21.tar
```

In einem Schritt funktioniert das Ganze so:

```
# tar xjvf httpd-2.2.21.tar.bz2
```

Bei der 2.4-Beta-Version müssen Sie beide Dateien in dasselbe Verzeichnis entpacken, damit die APR-Dateien an der richtigen Stelle eingefügt werden:

```
# tar xjvf httpd-2.3.15-beta.tar.bz2
# tar xjvf httpd-2.3.15-beta-deps.tar.bz2
```

Anschließend können Sie mit dem Kompilierungsvorgang beginnen, der im nun folgenden Abschnitt beschrieben wird.

### 4.1.2 Apache 2 unter UNIX kompilieren

Unter UNIX ist die Kompilierung die bevorzugte Installationsmethode. Falls es für Ihre Distribution keine angepassten Apache-Pakete gibt, ist sie sogar beinahe die einzige. *UNIX* steht hier und im ganzen Buch – falls nicht anders angegeben – für sämtliche Unix-artigen Betriebssysteme, beispielsweise Linux, FreeBSD, Mac OS X oder Solaris.

Unter Mac OS X ist sogar bereits eine Apache-Version installiert. Sie können sie über Systemeinstellungen • Sharing • Web Sharing aktivieren. Wenn Sie gemäß den nachfolgenden Ausführungen eine neuere Version installieren möchten,

scheitert dies daran, dass Apple in aktuellen Mac OS X-Versionen keinen Compiler mehr vorinstalliert. Laden Sie sich unter *http://developer.apple.com/tools/xcode/* die Entwicklungsumgebung *Xcode* herunter; sie enthält auch den GNU-Compiler GCC.

### Die Kompilierung im Überblick

Beachten Sie, dass Sie alle Schritte dieser Anleitung – zumindest aber den letzten Schritt, die eigentliche Installation – als `root` durchführen müssen, um einen Produktionsserver zu installieren: Da nur `root` berechtigt ist, die Well-Known-TCP-Ports 0 bis 1023 einzurichten, wird Apache für Port 8080 konfiguriert, wenn ein gewöhnlicher Benutzer ihn installiert. Zudem haben Nicht-root-Benutzer in Verzeichnissen, die von den meisten Installationslayouts verwendet werden, keine Schreibrechte.

Falls Sie das Quellcode-Paket nicht soeben frisch heruntergeladen oder von der DVD zum Buch kopiert haben, sollten Sie sicherstellen, dass frühere Build-Vorgänge keine Veränderungen an den Dateien vorgenommen haben. Geben Sie dazu vor den eigentlichen Konfigurations- und Installationsbefehlen Folgendes ein:

```
# make clean
```

Wenn Sie ganz ungeduldig auf die Installation warten und für das erste Ausprobieren noch keine Lust auf Fine-Tuning haben, können Sie einfach nacheinander die folgenden Befehle eingeben, um Apache 2 auf einem Unix-artigen System zu installieren:

```
# ./configure
# make
# make install
```

Dies installiert Apache komplett unter dem Verzeichnis `/usr/local/apache2` und bindet die wichtigsten Module ein. Beachten Sie aber, dass die Ausführung dieser Befehle je nach Rechnergeschwindigkeit ziemlich lange dauert. Insofern ist es ärgerlich, sie später noch einmal mit angepassten Optionen wiederholen zu müssen. Im Übrigen gibt es keine »Uninstall«-Option; einen fehlerhaft installierten Apache-Server müssen Sie manuell wieder entfernen. Aus diesem Grund sollten Sie verantwortungsvoll entscheiden, wohin Sie den Server installieren möchten. Die Verzeichnisauswahl und viele andere Optionen sind Sache des Konfigurationsskripts `configure`, das zuerst aufgerufen wird. Die allgemeine Syntax lautet folgendermaßen:

```
# ./configure [OPTION]... [VAR=WERT]...
```

Wenn Sie keine weiteren Angaben vornehmen möchten (eine ziemlich schlechte Idee), sieht der Aufruf so aus:

```
# ./configure
```

In diesem Fall wird das Apache 2-Makefile mit Standardeinstellungen konfiguriert. Diese entsprechen in aller Regel weder den Gepflogenheiten Ihrer Systemdistribution noch Ihren Bedürfnissen für einen Produktionsserver. Aus diesem Grund gibt es zahllose Optionen, die zur Anpassung des Skripts verwendet werden können. Sie werden im Folgenden genau beschrieben. Hinter den Optionen können Sie Umgebungsvariablen vorübergehend auf spezielle Werte setzen – beispielsweise um mittels `CC` einen anderen Compiler anzugeben als den auf Ihrem System üblichen.

Um die Optionen zu sehen, die für die Installation Ihrer konkreten Version zur Verfügung stehen, können Sie zunächst Folgendes eingeben:

```
# ./configure --help |less
```

Auf einigen UNIX-Systemen steht der komfortable Pager `less` nicht zur Verfügung; dort müssen Sie `more` verwenden.

#### Konfigurationsoptionen

Die folgenden wichtigen Optionen beeinflussen unmittelbar die eigentliche Ausführung des Skripts oder sind aus anderen Gründen so wichtig, dass sie einen eigenen Kurzbefehl erhalten haben:

- `-h`, `--help`
  Diese Option gibt eine ausführliche Hilfemeldung aus und beendet dann das Skript.
- `--help=short`
  Diese Variante gibt eine Kurzfassung der Hilfe aus, nämlich nur Informationen, die das aktuelle Paket betreffen.
- `--help=recursive`
  Geht alle Pakete rekursiv durch und gibt die entsprechenden Hilfemeldungen aus.
- `-V`, `--version`
  Gibt nur Versionsinformationen aus und beendet dann das Skript.
- `-q`, `--quiet`, `--silent`
  Unterdrückt die üblichen Meldungen wie `Checking...`. Dies ist nicht zu empfehlen, weil eventuell Hinweise auf Probleme ausbleiben. Wenn Sie die Ausgabe im Terminal-Fenster stört, sollten Sie sie lieber umleiten, indem Sie hinter allen Konfigurationsoptionen `>DATEINAME` anfügen. Mithilfe des Zusatzes

`|tee DATEINAME` lässt sich die Ausgabe sogar verdoppeln, sie wird also sowohl am Bildschirm angezeigt als auch in die angegebene Datei geschrieben.

- `--cache-file=DATEI`
  Speichert die Ergebnisse von `configure` in der angegebenen Datei.
- `-C`, `--config-cache`
  Dies ist eine Kurzfassung für die Option `--cache-file` mit dem Standard-Dateinamen `config.cache`.
- `-n`, `--no-create`
  Diese Option führt alle Tests durch, nimmt aber keine Änderungen an den Makefiles vor. Dies ist nützlich, um zu prüfen, ob bei einer bestimmten Konfiguration Probleme auftreten können.
- `--srcdir=VERZEICHNIS`
  Mit dieser Option können Sie ein alternatives Verzeichnis für die Quelldateien angeben, falls sie sich nicht im selben Verzeichnis befinden wie das Skript `configure`.

#### Verzeichnisse

Die meisten UNIX-Varianten besitzen eine spezifische Konvention für die Verteilung unterschiedlicher Softwarebestandteile im Verzeichnisbaum. Es kann zu Problemen mit Startskripten und Systemaktualisierungen führen oder die Verständigung mit anderen Administratoren und dem Support-Personal erschweren, wenn Sie die Apache-Voreinstellungen akzeptieren, anstatt sich an die Gepflogenheiten Ihrer Systemdistribution zu halten. Aus diesem Grund lassen sich die Installationsverzeichnisse des Apache-Webservers extrem flexibel anpassen.

Standardmäßig wird der Webserver nach der Konfiguration und Kompilierung in das Verzeichnis */usr/local/apache2* und die entsprechenden Unterverzeichnisse installiert. Eine solche Zentralisierung widerspricht zwar den genannten Konventionen der meisten Systeme, hat aber auch wichtige Vorteile: Sie finden alle Bestandteile von Apache an einer einzigen Stelle und können ihn so z. B. mit einem einzigen `rm -rf Verzeichnis` wieder entfernen, um eine neuere Version zu installieren. Außerdem eröffnet diese Installationsvariante Ihnen die Möglichkeit, Apache in einen `chroot`-Dateisystemkäfig zu sperren (siehe Kapitel 18, »Sicherheit«).

Letztendlich hängt es also von Ihren eigenen Bedürfnissen und Vorlieben ab, ob Sie den Server lieber systemkonform oder eher praktisch in einem Stammverzeichnis installieren – im Folgenden wird beides beschrieben.

`configure` ermöglicht über die Option `--enable-layout` den Zugriff auf einige vorgefertigte Layouts für bestimmte Systeme und Plattformen, die in der Datei

`config.layout` definiert sind und Ihnen das Auswählen der Verzeichnisse erleichtern. Beachten Sie, dass diese Layouts *Präfixe* (Stammverzeichnisse) definieren und relative Pfadangaben zu diesen enthalten. Die Einstellungen der einzelnen Layouts finden Sie im nächsten Abschnitt nach der Erläuterung der verschiedenen Verzeichnisse.

Wenn es für Ihr System kein vorgefertigtes Layout gibt oder wenn Sie Anpassungen vornehmen möchten, können Sie die einzelnen Verzeichnisse auch manuell auswählen. Zunächst einmal können Sie zwei grundlegende Präfixe angeben; alle Verzeichnisse für die verschiedenen Teile des Servers werden als deren Unterverzeichnisse angelegt, sofern Sie sie nicht einzeln ändern. Die beiden Präfixe werden folgendermaßen gesetzt:

- `--prefix=PREFIX`
  Das `PREFIX` ist das Hauptverzeichnis für »architekturunabhängige« Teile von Apache 2: Konfigurationsdateien, Include-Dateien, Manpages usw. Der Standardwert ist – wie bereits erwähnt – */usr/local/apache2*.
- `--exec-prefix=EPREFIX`
  Die Verzeichnisangabe `EPREFIX` enthält die »architekturabhängigen« Teile des Webservers, insbesondere das ausführbare Programm (binary executable). Wenn Sie diese Angabe weglassen, erhält dieses Verzeichnis denselben Wert wie `PREFIX`.

Statt der Präfixe können Sie auch alle (oder einzelne) Installationsverzeichnisse manuell angeben. Alle, die Sie nicht separat erwähnen, werden als Unterverzeichnis des jeweils passenden Präfixes installiert. Im Einzelnen sind die folgenden Angaben möglich:

- `--bindir=VERZEICHNIS`
  Dies ist das Verzeichnis für die eigentlichen Programmdateien des Servers. Standardmäßig wird `EPREFIX/bin` verwendet.
- `--sbindir=VERZEICHNIS`
  In dieses Verzeichnis werden die Administrationsprogramme für Apache installiert. Sie dienen dem Start und der Steuerung des Servers. Der Standardwert ist `EPREFIX/sbin`.
- `--libexecdir=VERZEICHNIS`
  Hier werden ausführbare Programmteile abgelegt, genauer gesagt dynamisch ladbare Module. Standard: `EPREFIX/libexec`.
- `--datadir=VERZEICHNIS`
  Dieses Verzeichnis enthält allgemeine (architekturunabhängige) Eingabedaten wie Fehlerseiten, Icons und Ähnliches. Standardmäßig wird `PREFIX/share` ver-

wendet. Das Verzeichnis besitzt die folgenden (ebenfalls umkonfigurierbaren) Unterverzeichnisse:

- `errordir`: angepasste Vorlagen für Fehlermeldungsseiten wie 404 (Seite nicht gefunden). Standard: `DATADIR/error`.
- `iconsdir`: Icons, die Apache beispielsweise für automatisch generierte Indexseiten verwendet. Standard: `DATADIR/icons`.
- `htdocsdir`: Dieses Verzeichnis ist eines der wichtigsten: Es enthält die Website, die dieser Server veröffentlicht. Standard: `DATADIR/htdocs`.
- `manualdir`: Standardmäßig stellt Apache 2 seine eigene Dokumentation öffentlich unter *http://servername/manual* zur Verfügung. `manualdir` bestimmt das Verzeichnis, in dem die tatsächlichen Dateien installiert werden. Das Standardverzeichnis ist `DATADIR/manual`.
- `cgidir`: Bei der Standardkonfiguration von Apache 2 werden CGI-Skripte aus Sicherheitsgründen nicht im *htdocsdir*-Verzeichnis ausgeführt, sondern nur in dem hier angegebenen Verzeichnis, das automatisch auf *http://servername/cgi-bin* abgebildet wird. Standard: `DATADIR/cgi-bin`.

- `--sysconfdir=VERZEICHNIS`
  Hier werden rechnerabhängige Eingabedaten, das heißt die aktuellen Konfigurationsdaten wie *httpd.conf*, abgelegt. Das Standardverzeichnis ist `PREFIX/etc`.
- `--sharedstatedir=VERZEICHNIS`
  In diesem Verzeichnis werden architekturunabhängige Ausgabedaten abgelegt. Standard: `PREFIX/com`.
- `--localstatedir=VERZEICHNIS`
  Dieses Verzeichnis dient der Ausgabe rechnerabhängiger Daten, insbesondere der Log-Dateien. Standardmäßig wird `PREFIX/var` benutzt. Das Verzeichnis besitzt automatisch drei Unterverzeichnisse, die ebenfalls umkonfiguriert werden können:
  - `runtimedir`: Apache-Laufzeitdaten, insbesondere die PID-Datei. Standard: `LOCALSTATEDIR/run`.
  - `logfiledir`: Hier werden standardmäßig die Server-Log-Dateien abgelegt. Standard: `LOCALSTATEDIR/logs`.
  - `proxycachedir`: Wenn Apache 2 als Proxy-Server eingesetzt wird, enthält dieses Verzeichnis den Proxy-Cache. Standard: `LOCALSTATEDIR/cache`.
- `--libdir=VERZEICHNIS`
  In diesem Verzeichnis befinden sich die Objektcode-Bibliotheken. Der Standardwert ist `EPREFIX/lib`.

- `--includedir=VERZEICHNIS`
  Hier befinden sich die Include-Dateien, das heißt die C-Header-Dateien, für den Compiler. Standard: `PREFIX/include`.
- `--oldincludedir=VERZEICHNIS`
  Dieses Verzeichnis kann verwendet werden, um allgemeine C-Header-Dateien anzugeben, wichtig insbesondere für Nicht-`gcc`-Compiler. Das Standardverzeichnis ist `/usr/include`.
- `--infodir=VERZEICHNIS`
  In das hier angegebene Verzeichnis wird die GNU-Info-Dokumentation des Apache-Webservers installiert. Standard: `PREFIX/info`.
- `--mandir=VERZEICHNIS`
  Dieses Verzeichnis enthält die Manpages für Apache. Der Standardwert ist `PREFIX/man`.

#### Layouts

Wie bereits erwähnt, stellt die Option `--enable-layout=LAYOUT` die Verzeichnisse auf eines der Layouts aus der Datei `config.layout` ein. Im Folgenden werden die Layouts und ihre Verzeichniswerte aufgelistet. Beachten Sie, dass viele Angaben in den Layoutdefinitionen relativ sind: Einige beziehen sich auf die zuerst definierten Verzeichnisse `PREFIX` und `EPREFIX`, andere bauen wiederum aufeinander auf. Zur einfachen Unterscheidung habe ich die Namen dieser Variablen konsequent großgeschrieben.

Sie können übrigens auch bei Auswahl eines Layouts jedes Verzeichnis mithilfe der bereits beschriebenen Optionen anpassen. Sie können sich also ein Layout aussuchen, das weitgehend Ihren Vorstellungen entspricht, und das Fine-Tuning anschließend selbst vornehmen.

Einige Angaben in dieser Liste enthalten ein +-Zeichen. Das bedeutet keineswegs, dass spezielle Verzeichnisse erzeugt werden, die dieses Zeichen enthalten, sondern dass ein spezieller Apache-Verzeichnisname angehängt wird. Zurzeit ist dies `/apache2`; ein Kommentar in `config.layout` weist darauf hin, dass dieser Wert später einmal frei konfigurierbar gemacht werden könnte. Falls Sie also einen Wert wie `PREFIX/etc+` entdecken, bedeutet dies, dass `PREFIX/etc/apache2` verwendet wird; `PREFIX` wird dabei wie gehabt aufgelöst.

- `--enable-layout=Apache`
  Dies ist das Standardlayout, das alle Daten in Unterverzeichnisse von `/usr/local/apache2` kopiert und das verwendet wird, wenn `configure` ganz ohne Verzeichnisangaben aufgerufen wird:

```
PREFIX: /usr/local/apache2
EPREFIX: PREFIX
BINDIR: EPREFIX/bin
SBINDIR: EPREFIX/bin
LIBDIR: EPREFIX/lib
LIBEXECDIR: EPREFIX/modules
MANDIR: PREFIX/man
SYSCONFDIR: PREFIX/conf
DATADIR: PREFIX
INSTALLBUILDDIR: DATADIR/build
ERRORDIR: DATADIR/error
ICONSDIR: DATADIR/icons
HTDOCSDIR: DATADIR/htdocs
MANUALDIR: DATADIR/manual
CGIDIR: DATADIR/cgi-bin
INCLUDEDIR: PREFIX/include
LOCALSTATEDIR: PREFIX
RUNTIMEDIR: LOCALSTATEDIR/logs
LOGFILEDIR: LOCALSTATEDIR/logs
PROXYCACHEDIR: LOCALSTATEDIR/proxy
```

- `--enable-layout=GNU`

  Das GNU-Layout ist dann zu empfehlen, wenn Sie keine eigenen Vorstellungen umsetzen möchten und es für Ihr System kein spezielles Layout gibt:

```
PREFIX: /usr/local
EPREFIX: PREFIX
BINDIR: EPREFIX/bin
SBINDIR: EPREFIX/sbin
LIBDIR: EPREFIX/lib
LIBEXECDIR: EPREFIX/libexec
MANDIR: PREFIX/man
SYSCONFDIR: PREFIX/etc+
DATADIR: PREFIX/share+
INSTALLBUILDDIR: DATADIR/build
ERRORDIR: DATADIR/error
ICONSDIR: DATADIR/icons
HTDOCSDIR: DATADIR/htdocs
MANUALDIR: DATADIR/manual
CGIDIR: DATADIR/cgi-bin
INCLUDEDIR: PREFIX/include+
LOCALSTATEDIR: PREFIX/var+
RUNTIMEDIR: LOCALSTATEDIR/run
LOGFILEDIR: LOCALSTATEDIR/log
PROXYCACHEDIR: LOCALSTATEDIR/proxy
```

- `--enable-layout=Mac OS X Server`
  Dies ist das passende Layout für die Server-Version von Mac OS X:

```
PREFIX: /Local/Library/WebServer
EPREFIX: /usr
BINDIR: EPREFIX/bin
SBINDIR: EPREFIX/sbin
LIBDIR: EPREFIX/lib
LIBEXECDIR: /System/Library/Apache/Modules
MANDIR: EPREFIX/share/man
SYSCONFDIR: PREFIX/Configuration
DATADIR: PREFIX
INSTALLBUILDDIR: /System/Library/Apache/Build
ERRORDIR: /System/Library/Apache/Error
ICONSDIR: /System/Library/Apache/Icons
MANUALDIR: /System/Library/Apache/Manual
HTDOCSDIR: DATADIR/Documents
CGIDIR: DATADIR/CGI-Executables
INCLUDEDIR: /System/Library/Frameworks/Apache.framework/
     Versions/2.0/Headers
LOCALSTATEDIR: /var
RUNTIMEDIR: PREFIX/Logs
LOGFILEDIR: PREFIX/Logs
PROXYCACHEDIR: PREFIX/ProxyCache
```

- `--enable-layout=Darwin`
  Die folgenden Verzeichnisse sind das Layout für die »Workstation«-Version von Mac OS X (Leopard, Tiger usw.):

```
PREFIX: /usr
EPREFIX: PREFIX
BINDIR: EPREFIX/bin
SBINDIR: EPREFIX/sbin
LIBDIR: EPREFIX/lib
LIBEXECDIR: EPREFIX/libexec+
MANDIR: PREFIX/share/man
DATADIR: /Library/WebServer
SYSCONFIDR: /etc+
INSTALLBUILDDIR: PREFIX/share/httpd/build
ERRORDIR: PREFIX/share/httpd/error
ICONSDIR: PREFIX/share/httpd/icons
HTDOCSDIR: DATADIR/Documents
MANUALDIR: DATADIR/share/httpd/manual
cgidir: DATADIR/CGI-Executables
includedir: PREFIX/include+
```

```
localstatedir: /var
runtimedir: localstatedir/run
logfiledir: localstatedir/log+
proxycachedir: runtimedir/proxy
```

- `--enable-layout=RedHat`

  Das Layout für RedHat und Fedora Linux sieht so aus:

```
PREFIX: /usr
EPREFIX: PREFIX
BINDIR: PREFIX/bin
SBINDIR: PREFIX/sbin
LIBDIR: PREFIX/lib
LIBEXECDIR: PREFIX/lib/apache
MANDIR: PREFIX/man
SYSCONFDIR: /etc/httpd/conf
DATADIR: /var/www
INSTALLBUILDDIR: datadir/build
ERRORDIR: DATADIR/error
ICONSDIR: DATADIR/icons
HTDOCSDIR: DATADIR/html
MANUALDIR: DATADIR/manual
CGIDIR: DATADIR/cgi-bin
INCLUDEDIR: PREFIX/include/apache
LOCALSTATEDIR: /var
RUNTIMEDIR: LOCALSTATEDIR/run
LOGFILEDIR: LOCALSTATEDIR/log/httpd
PROXYCACHEDIR: LOCALSTATEDIR/cache/httpd
```

- `--enable-layout=opt`

  Wenn Sie Apache 2 im Verzeichnis `/opt` (optionale Software) unterbringen möchten, sieht das vorgeschlagene Layout folgendermaßen aus:

```
PREFIX: /opt/apache
EPREFIX: PREFIX
BINDIR: EPREFIX/bin
SBINDIR: EPREFIX/sbin
LIBDIR: EPREFIX/lib
LIBEXECDIR: EPREFIX/libexec
MANDIR: PREFIX/man
SYSCONFDIR: /etc/PREFIX
DATADIR: PREFIX/share
INSTALLBUILDDIR: DATADIR/build
ERRORDIR: DATADIR/error
ICONSDIR: DATADIR/icons
HTDOCSDIR: DATADIR/htdocs
```

```
MANUALDIR: DATADIR/manual
CGIDIR: DATADIR/cgi-bin
INCLUDEDIR: PREFIX/include
LOCALSTATEDIR: /var/PREFIX
RUNTIMEDIR: LOCALSTATEDIR/run
LOGFILEDIR: LOCALSTATEDIR/logs
PROXYCACHEDIR: LOCALSTATEDIR/proxy
```

- `--enable-layout=beos`

  Für das Betriebssystem BeOS, das eine Zeit lang als spezialisiertes Multimediasystem gefeiert wurde und in letzter Zeit etwas in Vergessenheit geraten ist, gibt es ein eigenes Layout (ob es für den kommerziellen BeOS-Nachfolger Zeta geeignet ist, ist fraglich):

```
PREFIX: /boot/home/apache
EPREFIX: PREFIX
BINDIR: EPREFIX/bin
SBINDIR: EPREFIX/bin
LIBDIR: EPREFIX/lib
LIBEXECDIR: EPREFIX/libexec
MANDIR: PREFIX/man
SYSCONFDIR: PREFIX/conf
DATADIR: PREFIX
INSTALLBUILDDIR: DATADIR/build
ERRORDIR: DATADIR/error
ICONSDIR: DATADIR/icons
HTDOCSDIR: DATADIR/htdocs
MANUALDIR: DATADIR/manual
CGIDIR: DATADIR/cgi-bin
INCLUDEDIR: PREFIX/include
LOCALSTATEDIR: PREFIX
RUNTIMEDIR: LOCALSTATEDIR/logs
LOGFILEDIR: LOCALSTATEDIR/logs
PROXYCACHEDIR: LOCALSTATEDIR/proxy
```

- `--enable-layout=SuSE`

  Dieses Layout ist für ältere Versionen von SUSE Linux vorgesehen (im Kommentar steht die Versionsangabe »6.x«) und definiert folgende Verzeichnisse:

```
PREFIX: /usr
EPREFIX: PREFIX
BINDIR: PREFIX/bin
SBINDIR: PREFIX/sbin
LIBDIR: PREFIX/lib
LIBEXECDIR: PREFIX/lib/apache
MANDIR: PREFIX/share/man
```

```
SYSCONFDIR: /etc/httpd
DATADIR: /usr/local/httpd
INSTALLBUILDDIR: DATADIR/build
ERRORDIR: DATADIR/error
ICONSDIR: DATADIR/icons
HTDOCSDIR: DATADIR/htdocs
MANUALDIR: DATADIR/manual
CGIDIR: DATADIR/cgi-bin
INCLUDEDIR: PREFIX/include/apache
LOCALSTATEDIR: /var/lib/httpd
RUNTIMEDIR: /var/run
LOGFILEDIR: /var/log/httpd
PROXYCACHEDIR: /var/cache/httpd
```

- `--enable-layout=BSD/OS`

  Das Layout `BSD/OS` ist für BSD UNIX-Varianten vorgesehen und auch für die freien BSD-Derivate (FreeBSD, OpenBSD und NetBSD) geeignet:

```
PREFIX: /var/www
EPREFIX: /usr/contrib
BINDIR: EPREFIX/bin
SBINDIR: EPREFIX/bin
LIBDIR: EPREFIX/lib
LIBEXECDIR: EPREFIX/libexec/apache
MANDIR: EPREFIX/man
SYSCONFDIR: PREFIX/conf
DATADIR: PREFIX
INSTALLBUILDDIR: DATADIR/build
ERRORDIR: DATADIR/error
ICONSDIR: DATADIR/icons
HTDOCSDIR: DATADIR/htdocs
MANUALDIR: DATADIR/manual
CGIDIR: DATADIR/cgi-bin
INCLUDEDIR: EPREFIX/include/apache
LOCALSTATEDIR: /var
RUNTIMEDIR: LOCALSTATEDIR/run
LOGFILEDIR: LOCALSTATEDIR/log/httpd
PROXYCACHEDIR: LOCALSTATEDIR/proxy
```

- `--enable-layout=Solaris`

  Für das Betriebssystem Sun/Oracle (Open)Solaris wurde das folgende Layout definiert:

```
PREFIX: /usr/apache
EPREFIX: PREFIX
BINDIR: EPREFIX/bin
```

```
SBINDIR: EPREFIX/bin
LIBDIR: EPREFIX/lib
LIBEXECDIR: EPREFIX/libexec
MANDIR: EPREFIX/man
SYSCONFDIR: /etc/apache
DATADIR: /var/apache
INSTALLBUILDDIR: DATADIR/build
ERRORDIR: DATADIR/error
ICONSDIR: DATADIR/icons
HTDOCSDIR: DATADIR/htdocs
MANUALDIR: DATADIR/manual
CGIDIR: DATADIR/cgi-bin
INCLUDEDIR: EPREFIX/include
LOCALSTATEDIR: PREFIX
RUNTIMEDIR: /var/run
LOGFILEDIR: DATADIR/logs
PROXYCACHEDIR: DATADIR/proxy
```

- `--enable-layout=OpenBSD`
  Speziell für OpenBSD können Sie statt `BSD/OS` auch dieses Layout wählen. Es definiert folgende Verzeichnisse:

```
PREFIX: /var/www
EPREFIX: /usr
BINDIR: EPREFIX/bin
SBINDIR: EPREFIX/sbin
LIBDIR: EPREFIX/lib
LIBEXECDIR: EPREFIX/lib/apache/modules
MANDIR: EPREFIX/share/man
SYSCONFDIR: PREFIX/conf
DATADIR: PREFIX
INSTALLBUILDDIR: PREFIX/build
ERRORDIR: PREFIX/error
ICONSDIR: PREFIX/icons
HTDOCSDIR: PREFIX/htdocs
MANUALDIR: DATADIR/manual
CGIDIR: PREFIX/cgi-bin
INCLUDEDIR: EPREFIX/lib/apache/include
LOCALSTATEDIR: PREFIX
RUNTIMEDIR: PREFIX/logs
LOGFILEDIR: PREFIX/logs
PROXYCACHEDIR: PREFIX/proxy
```

- `--enable-layout=FreeBSD`
  Auch für FreeBSD gibt es ein angepasstes Layout mit folgenden Verzeichnissen:

```
PREFIX: /usr/local
EPREFIX: PREFIX
BINDIR: EPREFIX/bin
SBINDIR: EPREFIX/sbin
LIBDIR: EPREFIX/lib
LIBEXECDIR: EPREFIX/libexec/apache2
MANDIR: PREFIX/man
SYSCONFDIR: PREFIX/etc/apache2
DATADIR: PREFIX/www
INSTALLBUILDDIR: PREFIX/share/apache2/build
ERRORDIR: DATADIR/error
ICONSDIR: DATADIR/icons
HTDOCSDIR: DATADIR/data
MANUALDIR: PREFIX/share/doc/apache2
CGIDIR: DATADIR/cgi-bin
INCLUDEDIR: PREFIX/include/apache2
LOCALSTATEDIR: /var
RUNTIMEDIR: LOCALSTATEDIR/run
LOGFILEDIR: LOCALSTATEDIR/log
PROXYCACHEDIR: DATADIR/proxy
```

- `--enable-layout=Debian`

  Zu guter Letzt wird noch ein Verzeichnislayout für Debian GNU/Linux mitgeliefert, das auch für Debian-Derivate wie Ubuntu geeignet ist. `PREFIX` hat hier absichtlich *keinen* Wert, sodass `EPREFIX` beispielsweise den Wert `/usr` besitzt:

```
PREFIX:
EPREFIX: PREFIX/usr
BINDIR: EPREFIX/bin
SBINDIR: EPREFIX/sbin
LIBDIR: EPREFIX/lib
LIBEXECDIR: EPREFIX/lib/apache2/modules
MANDIR: EPREFIX/share/man
SYSCONFDIR: PREFIX/etc/apache2
DATADIR: EPREFIX/share/apache2
ICONSDIR: DATADIR/icons
HTDOCSDIR: PREFIX/usr/share/apache2/default-site/htdocs
MANUALDIR: HTDOCSDIR/manual
CGIDIR: PREFIX/usr/lib/cgi-bin
INCLUDEDIR: EPREFIX/include/apache2
LOCALSTATEDIR: PREFIX/var/run
RUNTIMEDIR: PREFIX/var/run
LOGFILEDIR: PREFIX/var/log/apache2
PROXYCACHEDIR: PREFIX/var/cache/apache2/proxy
INFODIR: EPREFIX/share/info
```

```
   INSTALLBUILLDIR: PREFIX/etc/apache2/build
   ERRORDIR: DATADIR/error
```

Wenn Sie wissen, wie die Layouts in der Datei *config.layout* aufgebaut sind, hält Sie natürlich nichts davon ab, mit einem Texteditor Ihr eigenes Layout hinzuzufügen. Sehen Sie sich als Beispiel die Originalfassung des GNU-Layouts an:

```
#   GNU standards conforming path layout.
#   See FSF's GNU project 'make-stds' document for details.
<Layout GNU>
    prefix:        /usr/local
    exec_prefix:   ${prefix}
    bindir:        ${exec_prefix}/bin
    sbindir:       ${exec_prefix}/sbin
    libdir:        ${exec_prefix}/lib
    libexecdir:    ${exec_prefix}/libexec
    mandir:        ${prefix}/man
    sysconfdir:    ${prefix}/etc+
    datadir:       ${prefix}/share+
    installbuilddir: ${datadir}/build
    errordir:      ${datadir}/error
    iconsdir:      ${datadir}/icons
    htdocsdir:     ${datadir}/htdocs
    manualdir:     ${datadir}/manual
    cgidir:        ${datadir}/cgi-bin
    includedir:    ${prefix}/include+
    localstatedir: ${prefix}/var+
    runtimedir:    ${localstatedir}/run
    logfiledir:    ${localstatedir}/log
    proxycachedir: ${localstatedir}/proxy
</Layout>
```

Enthält ein Eintrag einen Wert in der Form `${variable}`, dann handelt es sich um eine zuvor definierte Verzeichnisangabe. Die beiden obersten Zeilen, die mit # beginnen, sind natürlich Kommentare (wie in Shell- oder Perl-Skripten).

Ihr eigenes Layout könnte nun beispielsweise so aussehen (ich habe es stark an das SUSE-Layout angelehnt, weil ich die Installation damit unter SUSE Linux getestet habe):

```
#   Angepasstes eigenes Layout
<Layout Eigen>
    prefix:        /usr
    exec_prefix:   ${prefix}
    bindir:        ${prefix}/bin
    sbindir:       ${prefix}/sbin
```

```
    libdir:         ${prefix}/lib
    libexecdir:     ${prefix}/lib/apache
    mandir:         ${prefix}/share/man
    sysconfdir:     /etc/httpd
    datadir:        /usr/local/apache2
    installbuilddir: ${datadir}/build
    errordir:       /home/user/error
    iconsdir:       ${datadir}/icons
    htdocsdir:      /home/user/htdocs
    manualdir:      ${datadir}/manual
    cgidir:         ${datadir}/cgi-bin
    includedir:     ${prefix}/include/apache
    localstatedir: /var/lib/httpd
    runtimedir:     /var/run
    logfiledir:     /var/log/httpd
    proxycachedir: /var/cache/httpd
</Layout>
```

Dieses Layout ist bequem, weil sich die häufig genutzten Verzeichnisse `htdocsdir` (die Website) und `errordir` (Fehlermeldungsvorlagen) innerhalb des eigenen Home-Verzeichnisses befinden. Sicherheitstechnisch ist dies allerdings keine besonders gute Lösung – unterschiedliche Benutzerrechte kommen einander so in die Quere. Aufgerufen wird dieses Layout mit folgender `configure`-Option:

```
--enable-layout=Eigen
```

#### Einkompilieren oder Weglassen von Modulen

Zahlreiche Optionen für `./configure` beziehen sich auf das Kompilieren oder ausdrückliche Weglassen von Modulen. Der modulare Aufbau von Apache 2 wurde bereits in Kapitel 3, »Apache 2 im Überblick«, erläutert, zusammen mit einer Übersicht über die verfügbaren Module. Schauen Sie sich diese Liste an, um zu entscheiden, welche Module Sie für Ihre Anwendung von Apache 2 benötigen. Die meisten Module werden zudem in späteren Kapiteln noch genau besprochen.

Wenn Sie keinerlei Angaben über Module machen, werden bei Apache 2.2 die folgenden Module statisch einkompiliert:

```
mod_actions
mod_alias
mod_asis
mod_auth_basic
mod_authn_default
mod_authn_file
```

```
mod_authz_default
mod_authz_groupfile
mod_authz_host
mod_authz_user
mod_autoindex
mod_cgi
mod_dir
mod_env
mod_include
mod_log_config
mod_mime
mod_negotiation
mod_setenvif
mod_so
mod_status
mod_userdir
```

Bei Apache 2.4 werden standardmäßig folgende Module einkompiliert:

```
mod_authn_file
mod_authn_core
mod_authz_host
mod_authz_groupfile
mod_authz_user
mod_authz_core
mod_access_compat
mod_auth_basic
mod_auth_form
mod_allowmethods
mod_buffer
mod_ratelimit
mod_reqtimeout
mod_request
mod_include
mod_filter
mod_charset_lite
mod_mime
mod_log_config
mod_env
mod_setenvif
mod_version
mod_status
mod_autoindex
mod_cgid
mod_cgi
```

```
mod_negotiation
mod_dir
mod_actions
mod_userdir
mod_alias
```

Bei einem installierten Apache können Sie sich diese Liste mithilfe des folgenden Befehls anzeigen lassen:

```
# httpd -l
```

In welchem Verzeichnis sich das Binary `httpd` befindet, hängt natürlich ebenfalls vom gewählten Layout ab. Beim Layout Apache liegt es beispielsweise unter */usr/local/apache2/bin*, beim GNU-Layout unter */usr/local/sbin*.

Möglicherweise möchten Sie eines der Standardmodule explizit weglassen, etwa weil eine Sicherheitslücke darin bekannt geworden ist, weil es eine leistungsfähigere Alternative gibt oder weil Sie es einfach nicht brauchen. In diesem Fall können Sie eine der unten beschriebenen `--disable`-Optionen verwenden.

Im Übrigen können Sie bei der Konfiguration entweder jedes einzelne Modul angeben, das Sie in den Webserver einkompilieren möchten, oder aber (fast) alle Module zur Installation auswählen und dann einzelne Module explizit entfernen.

Zunächst einmal gibt es drei grundsätzliche Möglichkeiten, wie die Module in den Server integriert werden sollen:

- **Statisch einkompiliert**
  Wenn Sie sich für diese Option entscheiden, werden alle ausgewählten Module zum festen und unverrückbaren Bestandteil von Apache 2. Sie müssen den Server ganz neu kompilieren, um nachträglich Änderungen vorzunehmen.

  **Vorteil**: Dies ist die schnellste und effizienteste Variante von Apache 2.

  **Nachteil**: Diese Lösung ist extrem unflexibel.

  **Empfehlung**: Dieses Verfahren ist für Sites mit besonders hoher Belastung geeignet, bei denen aber zu erwarten ist, dass der Server über längere Zeit ohne Änderungen laufen soll. Vor allem für Hoster, die die virtuellen Hosts ihrer Kunden mit Apache betreiben, ist dies die ideale Lösung.
- **Dynamic Shared Objects**
  Sie können alle Module als DSOs (Dynamic Shared Objects oder dynamische Bibliotheken) laden – dies entspricht dem Prinzip der bekannten Windows DLLs. Auf diese Weise lassen sich Module auch beim fertig kompilierten Ser-

ver beliebig hinzufügen und entfernen; die entsprechende Konfigurationsdirektive `LoadModule` wird in Kapitel 6, »Grundkonfiguration«, besprochen.

**Vorteil**: Diese Variante ist die flexibelste Apache-Installation.

**Nachteil**: Durch das dynamische Laden der Module leidet die Performance des Webservers etwas.

**Empfehlung**: Dies ist die beste Lösung für einen Entwicklungs- und Versuchsserver, weil Sie die Konfiguration immer wieder ändern können, bis Sie die beste Lösung gefunden haben. Auch für einen »normal« belasteten Produktionsserver (z. B. die Website eines mittleren Unternehmens) funktioniert diese Variante problemlos.

- **Statisch einkompiliert, DSOs möglich**
  Bei dieser Lösung werden alle Module, deren Betrieb auf dem Server vorgesehen ist, statisch einkompiliert. Zusätzlich wird mithilfe der Konfigurationsoption `--enable-so` die Möglichkeit offen gehalten, weitere Module nachträglich als Dynamic Shared Objects zu installieren.

  **Vorteil**: Diese Lösung ist flexibler als die rein statische Variante.

  **Nachteil**: Wenn später viele Module als DSOs hinzugefügt werden, wird wiederum die Performance beeinträchtigt.

  **Empfehlung**: Für die allermeisten Produktionsserver, auf denen eine einzelne Website betrieben wird, ist diese Variante die ideale Lösung.

Für die Steuerung der Kompilierung von Modulen können Sie die folgenden Konfigurationsoptionen verwenden:

- `--enable-so`
  Wie bereits erwähnt, sollten Sie diese Option fast immer wählen; erfreulicherweise ist sie inzwischen Standard. Sie ermöglicht grundsätzlich die Verwendung von Modulen als Shared Objects, die nachträglich hinzugefügt werden können. Die Auswahl dieser Option sagt noch nichts darüber aus, ob die unmittelbar bei der Kompilierung hinzugefügten Module tatsächlich als Shared Objects oder statisch hinzugefügt werden.
- `--enable-modules=MODULLISTE`
  Mithilfe dieser Option können Sie eine Liste von Modulen angeben, die statisch einkompiliert werden sollen. Dazu gibt es grundsätzlich zwei Möglichkeiten: Sie können manuell eine Liste in Anführungszeichen (Modulnamen ohne vorangestelltes `mod_`) oder den Namen einer vorgefertigten Liste angeben.

  Angenommen, Sie möchten `mod_ssl`, `mod_proxy`, `mod_proxy_http` und `mod_headers` statisch kompilieren. Dann lautet die entsprechende Option:

```
--enable-modules="ssl proxy proxy_http headers"
```

Es stehen folgende automatische Modullisten zur Auswahl: `--enable-modules=all` kompiliert alle im Quellpaket verfügbaren Module. Sollten Probleme mit einem Modul auftreten, bricht die Kompilierung ab. `--enable-modules=most` lässt dagegen einige festgelegte Module weg sowie diejenigen, die bei der Kompilierung Probleme bereiten.

- `--disable-modules=MODULLISTE`
  Die Kompilierung der in dieser Liste angegebenen Module wird ausdrücklich ausgeschlossen. Dies ist nützlich in Verbindung mit `--enable-modules=all` beziehungsweise `--enable-modules=most`, wenn Sie einzelne Module gezielt weglassen möchten. Daneben ist die Option auch dann praktisch, wenn Sie einzelne Module nicht fest einkompilieren, sondern als DSOs hinzufügen möchten.

  Das folgende Beispiel zeigt, wie Sie die Module `mod_alias` und `mod_cern_meta` weglassen können (weil `mod_rewrite` beziehungsweise `mod_headers` einen erheblich mächtigeren Ersatz bieten):[1]

  ```
  --enable-modules=all --disable-modules="alias cern_meta"
  ```

- `--enable-mods-shared=MODULLISTE`
  Die Module, die Sie bei dieser Option angeben, werden als DSOs kompiliert. Natürlich muss `--enable-so` für die grundsätzliche DSO-Fähigkeit aktiviert sein, um von dieser Option Gebrauch machen zu können; dies ist standardmäßig der Fall. Auch bei dieser Option haben Sie wieder die Wahl zwischen einer selbst definierten Modulliste oder den Vorgaben `most` und `all`, die bei `--enable-modules` beschrieben wurden.

- `--disable-mods-shared=MODULLISTE`
  Die Liste, die Sie bei dieser Option angeben, wird ausdrücklich von der Kompilierung als DSOs ausgenommen. Dies ergibt nur im Zusammenhang mit `--enable-mods-shared=all` oder `--enable-mods-shared=most` einen Sinn, wenn Sie einzelne Module gezielt weglassen (oder statisch einkompilieren) möchten.

Alle weiteren Moduloptionen beziehen sich auf Angaben zur Kompilierung einzelner Module. Die Bezeichnungen in den Optionen entsprechen meist ebenfalls den Namen der Module ohne `mod_`. Sie finden sie (zusammen mit den eigentlichen Modulnamen) im Folgenden in Tabelle 4.1. Diese Optionen können den folgenden Schemata folgen:

1 In der Praxis sollten Sie `mod_alias` nicht weglassen. Erstens müssten Sie dann einige Teile der vorgefertigten Konfigurationsdatei umschreiben, zweitens ist `mod_rewrite` recht performancehungrig und sollte nur für Aufgaben eingesetzt werden, die `mod_alias` partout nicht beherrscht. Näheres dazu in Kapitel 8, »Weiterleitungen und Indizes«.

- `--enable-MODUL`
  `--enable-MODUL=yes`
  Eine Angabe nach diesem Muster kompiliert das angegebene Modul statisch.
- `--enable-MODUL=shared`
  Auf diese Weise wird das angegebene Modul als DSO kompiliert. Damit dies grundsätzlich funktioniert, muss auch die Option `--enable-so` aktiviert sein.
- `--disable-MODUL`
  `--enable-MODUL=no`
  Diese beiden Optionen deaktivieren die Kompilierung des entsprechenden Moduls ausdrücklich.

| Option | Modul | Beschreibung |
|---|---|---|
| `--enable-auth`<br>`--disable-auth` | `mod_auth` | Aktiviert beziehungsweise deaktiviert die allgemeine Authentifizierung anhand von Benutzernamen und Passwort (bis Version 2.2). |
| `--enable-auth-digest`<br>`--disable-auth-digest` | `mod_auth_digest` | verschlüsselte MD5-Digest-Authentifizierung gemäß RFC 2617 |
| `--enable-auth-basic`<br>`--disable-auth-basic` | `mod_auth_basic` | grundsätzliche Fähigkeit zur Basic-Authentifizierung |
| `--enable-auth-form`<br>`--disable-auth-form` | `mod_auth_form` | HTML-Formular-basierte Authentifizierung (ab Version 2.3-beta) |
| `--enable-authn-alias`<br>`--disable-authn-alias` | `mod_authn_alias` | Einrichtung komplexer Authentifizierungsschemata aus vorhandenen Bausteinen |
| `--enable-authn-anon`<br>`--disable-authn-anon` | `mod_authn_anon` | Zugriff mit Benutzernamen `anonymous` und E-Mail-Adresse als Passwort (dem bekannten Anonymous FTP nachempfunden) |
| `--enable-authn-dbm`<br>`--disable-authn-dbm` | `mod_authn_dbm` | Authentifizierung anhand von DBM-Dateien (Provider-Modul für `mod_auth_basic` und `mod_auth_digest`) |
| `--enable-authn-default`<br>`--disable-authn-default` | `mod_authn_default` | Regelung der Zuständigkeit beim Scheitern anderer Authentifizierungs-Provider (nur Version 2.2) |
| `--enable-authn-file`<br>`--disable-authn-file` | `mod_authn_file` | Authentifizierung anhand von Textdateien (Provider-Modul) |
| `--enable-authnz-ldap`<br>`--disable-authnz-ldap` | `mod_authnz_ldap` | Authentifizierung und Zugriffskontrolle durch LDAP-Verzeichnisse (Provider-Modul) |

**Tabelle 4.1** Konfigurationsoptionen für Apache-Module

| Option | Modul | Beschreibung |
|---|---|---|
| `--enable-authz-dbm`<br>`--disable-authz-dbm` | `mod_authz_dbm` | Zugriffskontrolle anhand von DBM-Dateien (Provider-Modul) |
| `--enable-authz-default`<br>`--disable-authz-default` | `mod_authz_default` | Regelung der Zuständigkeit beim Scheitern anderer Autorisierungs-Provider (nur Version 2.2) |
| `--enable-authz-groupfile`<br>`--disable-authz-groupfile` | `mod_authz_groupfile` | Verwaltung von Benutzergruppen in einfachen Textdateien |
| `--enable-authz-host`<br>`--disable-authz-host` | `mod_authz_host` | Zugriffskontrolle anhand von Hostname und IP-Adresse |
| `--enable-authz-owner`<br>`--disable-authz-owner` | `mod_authz_owner` | Zugriffskontrolle anhand des Dateieigentümers |
| `--enable-authz-user`<br>`--disable-authz-user` | `mod_authz_user` | Zugriffskontrolle anhand von Benutzerrechten |
| `--enable-file-cache`<br>`--disable-file-cache` | `mod_file_cache` | Bestimmte Dateien beim Apache-Start ins RAM kopieren (schnellerer Zugriff auf häufig verwendete Daten) |
| `--enable-echo`<br>`--disable-echo` | `mod_echo` | Konfiguriert Apache zusätzlich zu HTTP als Server für das Echo-Protokoll (sendet die übergebenen Daten zurück). |
| `--enable-charset-lite`<br>`--disable-charset-lite` | `mod_charset_lite` | diverse Optionen zur automatischen Zeichensatzkonvertierung |
| `--enable-dav`<br>`--disable-dav` | `mod_dav` | WebDAV-Unterstützung |
| `--enable-dav-fs`<br>`--disable-dav-fs` | `mod_dav_fs` | Über WebDAV auf das lokale Dateisystem zugreifen |
| `--enable-dav-lock`<br>`--disable-dav-lock` | `mod_dav_lock` | Locking-Funktionalität für `mod_dav` |
| `--enable-cache`<br>`--disable-cache` | `mod_cache` | Apache als Web-Cache |
| `--enable-cache-disk` (2.4)<br>`--disable-cache-disk` (2.4)<br>`--enable-disk-cache` (2.2)<br>`--disable-disk-cache` (2.2) | `mod_cache_disk` (2.4)<br>`mod_disk_cache` (2.2) | Realisiert den Zwischenspeicher für `mod_cache` auf der Festplatte |
| `--enable-mem-cache` (nur 2.2) | `mod_mem_cache` | Realisiert den Zwischenspeicher für `mod_cache` im RAM |
| `--enable-dumpio`<br>`--disable-dumpio` | `mod_dumpio` | Ein-/Ausgabedaten komplett in Log-Datei schreiben |
| `--enable-example`<br>`--disable-example` | `mod_example` | Das Modul `mod_example` demonstriert, wie Apache-Module funktionieren. Sollte niemals auf Produktionsservern eingesetzt werden! |

**Tabelle 4.1** Konfigurationsoptionen für Apache-Module (Forts.)

| Option | Modul | Beschreibung |
|---|---|---|
| `--enable-ext-filter`<br>`--disable-ext-filter` | `mod_ext_filter` | Ermöglicht die Definition externer Programme als Ein- oder Ausgabefilter. |
| `--enable-case-filter`<br>`--disable-case-filter` | `mod_case_filter` | Beispiel für einen Ausgabefilter: wandelt die Ausgabe in Großbuchstaben um. |
| `--enable-case-filter-in`<br>`--disable-case-filter-in` | `mod_case_filter_in` | Beispiel für einen Eingabefilter: wandelt die Eingabe in Großbuchstaben um. |
| `--enable-ldap`<br>`--disable-ldap` | `mod_ldap` | Stellt Caching- und Connection-Pooling-Dienste für LDAP bereit. |
| `--enable-auth-ldap`<br>`--disable-auth-ldap` | `mod_auth_ldap` | Authentifizierung über LDAP-Verzeichnisse |
| `--enable-include`<br>`--disable-include` | `mod_include` | Unterstützung für Server Side Includes |
| `--enable-deflate`<br>`--disable-deflate` | `mod_deflate` | Ermöglicht die Auslieferung `deflate`-komprimierter Daten gemäß RFC 2616 (siehe Kapitel 2, »Funktionsweise von Webservern«). |
| `--enable-log-config`<br>`--disable-log-config` | `mod_log_config` | benutzerdefinierte Log-Dateien |
| `--enable-logio`<br>`--disable-logio` | `mod_logio` | Anzahl der gesendeten/empfangenen Bytes in Log-Dateien (benötigt auch `mod_log_config`) |
| `--enable-log-forensic`<br>`--disable-log-forensic` | `mod_log_forensic` | forensische Log-Dateien (komplette Anfrage vor und nach der Verarbeitung) |
| `--enable-env`<br>`--disable-env` | `mod_env` | Modifikation von Umgebungsvariablen |
| `--enable-mime-magic`<br>`--disable-mime-magic` | `mod_mime_magic` | automatische Erkennung des MIME-Types |
| `--enable-cern-meta`<br>`--disable-cern-meta` | `mod_cern_meta` | Unterstützung von META-Files des CERN HTTPd (nur aus Kompatibilitätsgründen). Besser `mod_headers` verwenden! |
| `--enable-expires`<br>`--disable-expires` | `mod_expires` | Setzen des `Expires`-Headers, der die Gültigkeitsdauer von Dokumenten für Browser und vor allem Proxys angibt |

**Tabelle 4.1** Konfigurationsoptionen für Apache-Module (Forts.)

| Option | Modul | Beschreibung |
|---|---|---|
| `--enable-headers`<br>`--disable-headers` | `mod_headers` | Setzen/Ändern von HTTP-Headern |
| `--enable-usertrack`<br>`--disable-usertrack` | `mod_usertrack` | Session-Tracking durch Cookies |
| `--enable-unique-id`<br>`--disable-unique-id` | `mod_unique_id` | Weist jeder Client-Anfrage eine eindeutige ID zu. |
| `--enable-setenvif`<br>`--disable-setenvif` | `mod_setenvif` | Umgebungsvariablen je nach Anfrage-Headern setzen. |
| `--enable-proxy`<br>`--disable-proxy` | `mod_proxy` | Apache als Proxy-Server; benötigt eines oder mehrere der nachfolgenden `mod_proxy_*`-Module für konkrete Dienste. |
| `--enable-proxy-ajp` | `mod_proxy_ajp` | Proxy für Apache JServ Protocol 1.3 (z. B. Tomcat) |
| `--enable-proxy-balancer`<br>`--disable-proxy-balancer` | `mod_proxy_balancer` | Load-Balancing-Funktionalität für den Apache-Proxy |
| `--enable-proxy-connect`<br>`--disable-proxy-connect` | `mod_proxy_connect` | Implementierung der HTTP-Methode `CONNECT` (siehe Kapitel 2, »Funktionsweise von Webservern«) für den Apache-Proxy |
| `--enable-proxy-ftp`<br>`--disable-proxy-ftp` | `mod_proxy_ftp` | FTP-Proxy-Server |
| `--enable-proxy-http` | `mod_proxy_http` | HTTP-Proxy-Server |
| `--enable-ssl`<br>`--disable-ssl` | `mod_ssl` | Unterstützung für verschlüsselte Übertragung mit SSL/TLS |
| `--enable-distcache`<br>`--disable-distcache` | – | zusätzliche Caching-Funktionalität für `mod_ssl` |
| `--enable-bucketeer`<br>`--disable-bucketeer` | `mod_bucketeer` | Ausgabefilter, der bei Auftreten spezieller Sonderzeichen Buckets (interne Apache-Funktionen) ausführt |
| `--enable-mime`<br>`--disable-mime` | `mod_mime` | automatische Erkennung des MIME-Types anhand der Dateierweiterung |
| `--enable-status`<br>`--disable-status` | `mod_status` | Statusinformationen über den laufenden Server |
| `--enable-autoindex`<br>`--disable-autoindex` | `mod_autoindex` | automatische Darstellung von Verzeichnisinhalten bei fehlender Startseite |
| `--enable-asis`<br>`--disable-asis` | `mod_asis` | Datei senden, »wie sie ist« (ohne HTTP-Header!). |

**Tabelle 4.1** Konfigurationsoptionen für Apache-Module (Forts.)

| Option | Modul | Beschreibung |
|---|---|---|
| `--enable-info`<br>`--disable-info` | `mod_info` | Konfigurationsinformationen des Servers |
| `--enable-suexec`<br>`--disable-suexec` | `mod_suexec` | CGI-Skripte unter bestimmtem Benutzer/bestimmter Gruppe ausführen. |
| `--enable-cgid`<br>`--disable-cgid` | `mod_cgid` | CGI-Skripte in einem Thread-MPM |
| `--enable-cgi`<br>`--disable-cgi` | `mod_cgi` | CGI-Skripte als klassische Prozesse |
| `--enable-version`<br>`--disable-version` | `mod_version` | Ermöglicht versionsabhängige Konfigurationsanweisungen. |
| `--enable-vhost-alias`<br>`--disable-vhost-alias` | `mod_vhost_alias` | automatische Erzeugung virtueller Hosts anhand von Datei- und Verzeichnismustern |
| `--enable-negotiation`<br>`--disable-negotiation` | `mod_negotiation` | Content-Negotiation – automatische Lieferung unterschiedlicher Inhalte als Reaktion auf `Accept-*`-Anfrage-Header |
| `--enable-dir`<br>`--disable-dir` | `mod_dir` | Definition der Startseite; automatische Verarbeitung von Verzeichnis-URLs, die nicht auf einen `/` enden |
| `--enable-imagemap`<br>`--disable-imagemap` | `mod_imagemap` | Unterstützung für serverseitige Image Maps |
| `--enable-actions`<br>`--disable-actions` | `mod_ actions` | automatisches Auslösen von CGI-Handlern anhand des MIME-Types |
| `--enable-speling`<br>`--disable-speling` | `mod_ speling` | automatische Korrektur von Schreibfehlern in URLs (Groß-/Kleinschreibung) |
| `--enable-userdir`<br>`--disable-userdir` | `mod_userdir` | Bildet die Home-Verzeichnisse der Benutzer automatisch auf den Webserver ab (nach dem Schema *http://servername/~username*) |
| `--enable-alias`<br>`--disable-alias` | `mod_alias` | einfache Umsetzung von URLs |
| `--enable-rewrite`<br>`--disable-rewrite` | `mod_rewrite` | komplexe Umsetzung von URLs mit regulären Ausdrücken |

**Tabelle 4.1** Konfigurationsoptionen für Apache-Module (Forts.)

Wenn Sie weitere Module kompilieren möchten, die Sie in der Übersicht in Kapitel 3, »Apache 2 im Überblick«, gesehen haben, aber nicht in Tabelle 4.1 finden, gibt es für diese keine eigene Option nach dem Schema `--enable-MODUL`. Sie müs-

sen ein solches Modul in der Liste hinter einer der Optionen `--enable-modules`, `--disable-modules`, `--enable-mods-shared` oder `--disable-mods-shared` (siehe Tabelle 4.1) angeben, um es statisch oder dynamisch einzukompilieren beziehungsweise ausdrücklich wegzulassen.

### MPM auswählen

Die im vorigen Kapitel vorgestellten Multiprocessing-Module (MPM), die das Laufzeitverhalten des Servers bestimmen, werden ebenfalls bei der Konfiguration angegeben. Bis Version 2.2 mussten Benutzer sich für ein bestimmtes MPM entscheiden, da es stets fest einkompiliert werden musste. In Version 2.4 können MPM dagegen wie alle anderen Module dynamisch geladen werden. Sinnvoll ist dies nur auf UNIX-Systemen, weil es für die anderen unterstützten Plattformen nur jeweils ein optimiertes MPM gibt.

Um ein einzelnes MPM auszuwählen, das statisch kompiliert werden soll, wird die Option `--with-mpm=Name` verwendet. Für UNIX stehen folgende Varianten zur Auswahl, die im vorigen Kapitel näher beschrieben wurden:

- `prefork` – für jede Client-Verbindung wird ein eigener Prozess verwendet; es werden stets einige Prozesse auf Vorrat gefork.
- `worker` – eine kleine Anzahl von Prozessen verwaltet für jede Client-Verbindung einen Thread; dieses Modell ist speicherschonender als `prefork`.
- `event` – eine erweiterte Variante von `worker`, bei der Keepalive-Verbindungen nicht permanent einen Thread blockieren. In Apache 2.2 gilt `event` als experimentell und sollte nicht im produktiven Einsatz verwendet werden. In Version 2.4 ist das MPM dagegen stabil.

Um MPMs in Apache 2.4 dynamisch laden zu können, müssen Sie die Option `--enable-mpms-shared=Liste` verwenden. Das folgende Beispiel kompiliert `prefork` und `worker` dynamisch:

```
--enable-mpms-shared="prefork worker"
```

Alternativ kann `--enable-mpms-shared=all` verwendet werden, um alle für die jeweilige Plattform verfügbaren MPMs dynamisch zu kompilieren.

Wenn Sie MPMs dynamisch einkompilieren, können Sie wie gehabt die Option `--with-mpm=Name` verwenden, um das entsprechende MPM als vorkonfigurierten Standard auszuwählen. Das folgende Beispiel kompiliert alle MPMs dynamisch und wählt `worker` als Standard aus:

```
--enable-mpms-shared=all --with-mpm=worker
```

### Sonstige Optionen

Es gibt noch weitere Optionen für das Skript `configure`, die weder etwas mit der Ausführung des Skripts selbst noch mit Zielverzeichnissen oder Modulen zu tun haben. Sie beschäftigen sich unter anderem mit der Unterstützung bestimmter Features, die nicht durch Module gewährleistet werden, oder mit optionalen Pfadangaben für externe Bibliotheken. Sie finden die Liste dieser Optionen in Tabelle 4.2.

| Option | Beschreibung |
|---|---|
| `--enable-http`<br>`--disable-http` | Unterstützung des HTTP. Diese eigentlich wichtigste Eigenschaft des HTTP-Servers (!) wird zwar nicht durch ein Modul bereitgestellt, lässt sich aber trotzdem deaktivieren, wenn Sie unbedingt wollen. |
| `--enable-pie` | Apache als Position Independent Executable (PIE) kompilieren – auf einigen modernen Plattformen verhält sich ein ausführbares Programm wie eine Shared Library mit veränderlichen Basisadressen. |
| `--enable-static-support`<br>`--disable-static-support` | Diese Option regelt, wie die Apache-Zusatzprogramme (support binaries) kompiliert werden sollen: `--enable-static-support` erzeugt eine statisch gelinkte Version (schneller); `--disable-static-support` eine dynamisch ladbare (flexibler).<br>Die nächsten sieben Optionen behandeln noch einmal jedes der Programme einzeln. |
| `--enable-static-htpasswd`<br>`--disable-static-htpasswd` | Statische/dynamische Version von `htpasswd` kompilieren |
| `--enable-static-htdigest`<br>`--disable-static-htdigest` | Statische/dynamische Version von `htdigest` kompilieren |
| `--enable-static-rotatelogs`<br>`--disable-static-rotatelogs` | Statische/dynamische Version von `rotatelogs` kompilieren |
| `--enable-static-logresolve`<br>`--disable-static-logresolve` | Statische/dynamische Version von `logresolve` kompilieren |
| `--enable-static-htdbm`<br>`--disable-static-htdbm` | Statische/dynamische Version von `htdbm` kompilieren |
| `--enable-static-ab`<br>`--disable-static-ab` | Statische/dynamische Version von `ab` kompilieren |
| `--enable-static-checkgid`<br>`--disable-static-checkgid` | Statische/dynamische Version von `checkgid` kompilieren |

**Tabelle 4.2** Zusätzliche Optionen für die Apache-Kompilierung

| Option | Beschreibung |
|---|---|
| `--enable-exception-hook`<br>`--disable-exception-hook` | Zusätzlichen Hook (Modul-Verknüpfungspunkt) zur Behandlung schwerwiegender Ausnahmen einrichten/deaktivieren (Entwickleroption; standardmäßig deaktiviert) |
| `--enable-maintainer-mode`<br>`--disable-maintainer-mode` | Debugging und zusätzliche Compiler-Warnungen ein-/ausschalten (Entwickleroption; standardmäßig deaktiviert) |
| `--enable-optional-hook-export`<br>`--disable-optional-hook-export` | Unterstützung für Hook-Export aktivieren/deaktivieren (Entwickleroption; standardmäßig deaktiviert) |
| `--enable-optional-hook-import`<br>`--disable-optional-hook-import` | Unterstützung für Hook-Import aktivieren/deaktivieren (Entwickleroption; standardmäßig deaktiviert) |
| `--enable-optional-fn-import`<br>`--disable-optional-fn-import` | Unterstützung für Funktionsimport aktivieren/deaktivieren (Entwickleroption; standardmäßig deaktiviert) |
| `--enable-optional-fn-export`<br>`--disable-optional-fn-export` | Unterstützung für Funktionsexport aktivieren/deaktivieren (Entwickleroption; standardmäßig deaktiviert) |
| `--enable-v4-mapped`<br>`--disable-v4-mapped` | Diese Option ermöglicht die Verarbeitung von IPv4-Verbindungen durch IPv6-Sockets. |
| `--enable-maintainer-mode`<br>`--disable-maintainer-mode` | Schaltet zusätzliche Compiler-Warnungen und Debug-Informationen ein/aus |
| `--build=BUILD` | Mit `BUILD` können Sie explizit die Plattform angeben, für die Apache kompiliert wird (für Cross Compiling; ansonsten höchstens bei einem zweiten Durchlauf erforderlich, falls Ihr System nicht automatisch erkannt wurde). |
| `--host=HOST` | Beim Cross Compiling können Sie mit dieser Option auch explizit den Rechner angeben, für den Apache 2 kompiliert wird. `BUILD` ist dabei in der Regel trotzdem erforderlich. |
| `--target=TARGET` | Auch dieser Parameter gibt die Zielplattform (vor allem für Cross Compiling) an. |
| `--with-apr=VERZEICHNIS\|DATEI` | Verzeichnis für die Apache Portable Runtime beziehungsweise Pfad der Datei `apr-config` |
| `--with-apr-util=VERZEICHNIS` | Verzeichnis für die Apache Portable Runtime Utilities |

**Tabelle 4.2** Zusätzliche Optionen für die Apache-Kompilierung (Forts.)

| Option | Beschreibung |
|---|---|
| `--with-included-apr` | Explizit die mitgelieferte APR-Bibliothek verwenden, auch wenn bereits eine installiert ist (verfügbar seit Version 2.2.3; hilfreich, wenn eine zu alte APR-Version, das heißt < 1.2, installiert ist). |
| `--with-pcre=PFAD` | Pfad einer externen Version der PCRE-Bibliothek (mit Perl kompatible reguläre Ausdrücke) |
| `--with-port=PORT` | Angabe einer speziellen TCP-Portnummer statt 80 |
| `--with-z=VERZEICHNIS` | Verzeichnis der `zlib`-Bibliothek (Komprimierungsfunktionen), falls eine spezielle Variante benutzt werden soll |
| `--with-ssl=VERZEICHNIS` | Verzeichnis des OpenSSL-Toolkits für SSL/TLS-Verschlüsselung |
| `--with-module=MODULTYP:DATEI` | Einbinden eines speziellen Drittanbieter-Moduls aus dem Verzeichnis `modules` |
| `--with-program-name=NAME` | Spezialisierter Name für das ausführbare Webserver-Programm (Standard: `httpd`). Ist nur in absoluten Ausnahmefällen zu empfehlen, da es Verwaltung und Support enorm erschwert! |
| `--with-gdbm[=PFAD]`<br>`--with-ndbm[=PFAD]`<br>`--with-berkeley-db[=PFAD]` | Wenn Sie eine dieser Optionen wählen, verwendet Apache statt DBM das angegebene alternative Format für die interne Speicherung datenbankartiger Informationen. Optional kann der Pfad zur jeweiligen Bibliothek angegeben werden. |
| `--with-suexec-bin=VERZEICHNIS` | Pfad zum Programm `suexec` (zum Ausführen von CGIs unter alternativer User- und Group-ID; diese Option und die jetzt noch folgenden werden in Kapitel 18, »Sicherheit«, näher erläutert) |
| `--with-suexec-caller=USER` | Angabe des Benutzers, der `suexec` aufrufen darf (und unter dessen UID und GID Apache dann läuft) |
| `--with-suexec-userdir=VERZEICHNIS` | Unterverzeichnis des `suexec`-Benutzers |
| `--with-suexec-docroot=VERZEICHNIS` | `suexec`-Stammverzeichnis |
| `--with-suexec-uidmin=UID` | numerische Angabe der mindestens erforderlichen UID für `suexec` |
| `--with-suexec-gidmin=UID` | numerische Angabe der mindestens erforderlichen GID für `suexec` |
| `--with-suexec-logfile=DATEI` | `suexec`-Log-Datei |
| `--with-suexec-safepath=VERZEICHNIS` | Verzeichnis der `suexec`-CGI-Skripte |

**Tabelle 4.2** Zusätzliche Optionen für die Apache-Kompilierung (Forts.)

| Option | Beschreibung |
|---|---|
| `--with-suexec-umask` | `umask` (Dateirechte) für `suexec` |
| `--with-mysql` | Den DBD-Treiber für MySQL installieren (seit einigen Releases wird er mitgeliefert, aber nicht automatisch installiert, da er unter der mit Apache inkompatiblen GPL steht; Näheres siehe Kapitel 9, »Authentifizierung«). |

**Tabelle 4.2** Zusätzliche Optionen für die Apache-Kompilierung (Forts.)

Schließlich können Sie beim `configure`-Aufruf noch einige Umgebungsvariablen setzen. Dies ist in der Regel nur dann erforderlich, wenn `configure` den jeweiligen Wert nicht automatisch findet und mit einer entsprechenden Fehlermeldung abbricht. Das Schema dafür ist `VAR=wert`. Hier die wichtigsten Variablen:

- `CC`
  Angabe des zu verwendenden C-Compilers (Standard: `gcc`)
- `CFLAGS`
  Compiler-Flags (werden dem Compiler beim Aufruf unverändert übergeben). Die möglichen Werte einiger Apache-Konfigurationsoptionen werden durch spezielle Flags limitiert. Diese werden im Buch bei der Besprechung der entsprechenden Einstellungen angegeben.
- `LDFLAGS`
  Linker-Flags wie `-L VERZEICHNIS` zur Angabe eines unüblichen Bibliotheksverzeichnisses
- `CPPFLAGS`
  Präprozessor-Flags wie `-I VERZEICHNIS` zur Angabe eines unüblichen Verzeichnisses für Header-Dateien
- `CPP`
  Angabe des zu verwendenden Präprozessors

### Endspurt

Vielleicht ist es nach dieser langen Beschreibung der `configure`-Optionen in Vergessenheit geraten: Nachdem `configure` fertig ausgeführt wurde, müssen Sie noch die beiden folgenden Befehle eingeben, um die eigentliche Kompilierung und Installation durchzuführen:

```
# make
# make install
```

Anschließend empfiehlt es sich, in den Quellcode-Verzeichnissen aufzuräumen, um diese für einen späteren erneuten Kompiliervorgang vorzubereiten. Dazu ist folgende Eingabe erforderlich:

```
# make clean
```

Von hier aus geht es dann mit dem Start des Servers weiter, der in Kapitel 5, »Apache in Betrieb nehmen«, beschrieben wird.

### 4.1.3 Apache 2 unter Windows kompilieren

Die bevorzugte Apache-Installationsmethode unter Windows ist die weiter unten beschriebene Verwendung des offiziellen MSI-Installer-Pakets der Apache Group. Allerdings kann es auch unter Windows erforderlich oder zumindest sinnvoll sein, Apache 2 selbst zu kompilieren. Dies ist allerdings ein wenig problematischer als auf UNIX-Systemen, weil Entwicklungswerkzeuge bei Windows kein so selbstverständlicher Teil des Betriebssystems sind. In diesem Abschnitt wird die Kompilierung mit Microsoft Visual Studio beschrieben. Ich habe sie mit Visual Studio .NET getestet.

Bei der Windows-Kompilierung gibt es erheblich weniger direkte Konfigurationsmöglichkeiten als auf UNIX-Systemen. Das hat vor allem drei wichtige Gründe:

- Die Windows-Version wird mit einem festgelegten Satz von Modulen kompiliert. Diese sind grundsätzlich immer als DSOs konfiguriert und können mithilfe einer einfachen Direktive in der Apache-Konfigurationsdatei geladen werden. Auch zusätzliche Module von Drittanbietern lassen sich auf diese Weise leicht hinzufügen.
- Alle Dateien liegen in festgelegten Unterverzeichnissen eines wählbaren übergeordneten Verzeichnisses; es gibt unter Windows keine frei konfigurierbaren »Layouts«.

  Der Vollständigkeit halber hier ein Überblick über die Verzeichnisse, die auf Windows-Systemen innerhalb des Apache-Stammverzeichnisses angelegt werden:

  - `bin` – ausführbare Programme
  - `cgi-bin` – CGI-Skripte
  - `conf` – Konfigurationsdateien (`httpd.conf`)
  - `error` – Vorlagen für Fehlermeldungsseiten
  - `htdocs` – veröffentlichtes Verzeichnis (Website)
  - `icons` – Icons für Autoindex-Seiten usw.
  - `include` – Header-Dateien
  - `lib` – Bibliotheksdateien
  - `logs` – Log-Dateien
  - `manual` – Dokumentation

  - `modules` – Module (DSOs)
  - `proxy` – der Proxy-Cache
- Der Apache 2-Server für Windows wird mit einem eigenen, für Win32/Win64 optimierten MPM-Modul betrieben. Eine Anpassung ist weder möglich noch wünschenswert.
- Es gibt zwei verschiedene Kompilierungsmöglichkeiten, die im Folgenden beide beschrieben werden:
- auf der Kommandozeile kompilieren
- in der Visual-Studio-IDE kompilieren

Für beide Arten der Kompilierung benötigen Sie zunächst einmal Folgendes:

- **Das Apache-Quellcode-Paket für Windows**
  Sie finden es auf der DVD zum Buch oder können es von *httpd.apache.org/download.cgi* herunterladen: Klicken Sie den Link *apache-2.2.nn-win32-src.zip* (zurzeit 2.2.21) neben dem Text »Win32 Source« an.
- **Einen C-/C++-Compiler**
  Die Dokumentation der Apache Group beschreibt die Kompilierung mit Microsoft Visual C++ (oder dem ganzen Visual Studio) ab Version 5.0. Theoretisch genügt jeder ANSI-C-Compiler, der Win32-Code erzeugen kann. In der Praxis besteht das Problem aber darin, dass die mitgelieferten Makefiles und Projektdateien nur zum Microsoft-Compiler kompatibel sind. Wenn Sie einen anderen Compiler verwenden, müssten Sie sich selbst darum kümmern, sie entsprechend zu ändern.
- **Die Textprozessorsprache AWK**
  Sie wird im Verlauf der Kompilierung verwendet, um einige Dateien zu modifizieren. Sie finden GNU AWK (GAWK) auf der DVD zum Buch oder können es unter *http://gnuwin32.sourceforge.net/ packages/gawk.htm* herunterladen. Wenn Sie eine ältere Version von GAWK haben, müssen Sie die Programmdatei *gawk.exe* möglicherweise in *awk.exe* umbenennen (oder besser kopieren); bei dem hier beschriebenen Paket ist dies nicht nötig. Falls auf Ihrem Rechner CygWin installiert ist, können Sie auch die dort vorhandene Version von GAWK verwenden. Dazu müssen Sie allerdings den Symlink *awk* löschen und wiederum *gawk* in *awk* umbenennen, weil Windows keine Unix-artigen Symlinks versteht.

  Nachdem Sie *awk.exe* entsprechend installiert beziehungsweise angepasst haben, müssen Sie dessen Verzeichnis noch zur Umgebungsvariablen `PATH` hinzufügen, damit die Anweisungen im Makefile das Programm finden können. Wenn Sie sich für die Kommandozeilenkompilierung entscheiden, genügt das Setzen des Pfades innerhalb des verwendeten Konsolenfensters.

Dazu müssen Sie den AWK-Pfad allerdings zum bisherigen Wert von PATH hinzufügen; Sie dürfen ihn nicht etwa ersetzen! Geben Sie dazu Folgendes ein:

```
> set PATH=%PATH%; C:\Programme\GNUWin32\bin
```

Statt *C:\Programme\GNUWin32\bin* müssen Sie natürlich das korrekte Verzeichnis angeben.

Falls Sie Apache 2 in der Visual-Studio-IDE kompilieren möchten, müssen Sie den Pfad zuvor permanent hinzufügen.

#### Apache 2 auf der Windows-Kommandozeile kompilieren

Microsoft Visual C++ beziehungsweise Visual Studio besteht nicht nur aus der grafischen Entwicklungsumgebung (IDE), sondern enthält zusätzlich Kommandozeilen-Tools: Sie können Komponenten wie Compiler, Linker, Debugger und Build-Tools über die Eingabeaufforderung aufrufen. Dies bietet sich besonders dann an, wenn Sie Konsolenprogramme entwickeln und darüber hinaus einen externen Editor verwenden. Zwischenzeitlich waren die C++-Kommandozeilen-Tools sogar frei verfügbar; man konnte sie unter dem Namen »Visual C++ Toolkit 2003« ohne die grafische Entwicklungsumgebung von der Microsoft-Website herunterladen. Da Microsoft diese Version nicht mehr unterstützt, ist sie dort auch nicht mehr erhältlich. Auf der Buch-DVD darf sie aus rechtlichen Gründen auch nicht mitgeliefert werden. Falls Sie Interesse haben, suchen Sie mithilfe einer Suchmaschine nach »vctoolkit«, aber untersuchen Sie den Download auf jeden Fall mit einem Virenscanner, da Sie bei inoffiziellen Drittanbietern nicht sicher sein können, dass Sie das Originalprodukt erhalten.

Die größte Herausforderung vor der Kommandozeilenkompilierung besteht darin, eine Reihe von Umgebungsvariablen auf die richtigen Werte zu setzen, damit der Compiler alle benötigten Komponenten findet. Es ist verhältnismäßig frustrierend, dies manuell zu versuchen. Deshalb gibt es zwei verschiedene Möglichkeiten, die benötigten Variablen automatisch zu setzen:

- Im Startmenü finden Sie unter dem Eintrag für Visual Studio eine eigene Verknüpfung zur Eingabeaufforderung. Es handelt sich um die gewöhnliche Windows-Eingabeaufforderung mit dem Unterschied, dass die benötigten Umgebungsvariablen eben bereits gesetzt sind. Für Visual Studio .NET befindet sich der Eintrag beispielsweise unter Microsoft Visual Studio .NET • Visual Studio .NET Tools • Visual Studio .NET-Eingabeaufforderung. Bei anderen Versionen von Visual Studio/Visual C++ gibt es ebenfalls einen entsprechenden Menüpunkt.
- Wenn Sie ein Standard-Eingabeaufforderungsfenster verwenden, bietet Visual Studio alternativ eine Batch-Datei namens *vcvars32.bat*, die die benötigten

Variablen einstellt. Je nach Visual-Studio-Version und -Variante befindet sich die Datei in unterschiedlichen Verzeichnissen, bei Visual Studio .NET beispielsweise unter *Microsoft Visual Studio .NET\Vc7\bin* im Programme-Verzeichnis (beziehungsweise dort, wo Sie Visual Studio installiert haben). Wenn Sie nicht sicher sind, wo sich das Skript befindet, können Sie das Arbeitsverzeichnis auf den Stamm der richtigen Festplatte setzen und den folgenden Suchbefehl eingeben:

```
> dir vcvars32.bat /s
```

Sie erhalten daraufhin (nach einiger Zeit) eine Ausgabe des korrekten Verzeichnisses.

Nachdem Sie das Skript ausgeführt haben, steht die benötigte Umgebung (nur in diesem Konsolenfenster!) zur Verfügung. Falls Sie noch Visual Studio 5.0 verwenden, müssen Sie zusätzlich die Umgebungsvariablen des aktualisierten Windows Platform SDK anpassen. Zu diesem Zweck steht ebenfalls eine Batch-Datei zur Verfügung, die sich im Verzeichnis des Platform SDK (in der Regel `Platform SDK` im Programme-Verzeichnis) befindet und `setenv.bat` heißt.

Nun können Sie mit der eigentlichen Kompilierung beginnen. Sie wird durch die Datei `Makefile.win` im obersten Apache-Quellcode-Verzeichnis gesteuert. Dieses Makefile können Sie durch das Build-Tool von Visual Studio, `nmake.exe`, aufrufen. Wechseln Sie in das Verzeichnis mit dem Apache-Quellcode, und geben Sie den entsprechenden `nmake`-Aufruf ein. Schematisch funktioniert dies so:

```
> nmake /f Makefile.win OPTIONEN target
```

Für Target sind die folgenden Werte definiert:

- `_apacher` – Kompilieren im Release-Modus
- `_apached` – Kompilieren im Debug-Modus
- `installr` – Kompilieren und automatisches Installieren im Release-Modus
- `installd` – Kompilieren und automatisches Installieren im Debug-Modus
- `clean` – Entfernen (fast) aller bei einem früheren Kompiliervorgang erzeugten Dateien
- `_cleanr` – Entfernen der Dateien einer Release-Kompilierung
- `_cleand` – Entfernen der Dateien einer Debug-Kompilierung
- `_browse` – Ausgabe der verfügbaren Optionen

Wenn Sie nur Kompilieren und keine automatische Installation wählen (Optionen `_apacher` beziehungsweise `_apached`), müssen Sie die fertig kompilierten

Verzeichnisse selbst in das künftige Server-root-Verzeichnis (Standard: `\Apache2` auf der aktuellen Festplatte; Anpassung siehe unten) kopieren. In der Regel ist daher eine der beiden Optionen mit automatischer Installation (`installr` beziehungsweise `installd`) zu empfehlen.

Eine Debug-Version brauchen Sie eigentlich nur dann zu kompilieren, wenn Sie sich selbst gezielt auf die Jagd nach Bugs begeben möchten: Es werden zusätzliche Debugging-Informationen einkompiliert, die Sie mit Werkzeugen wie dem Visual-Studio-Debugger auswerten können. Für einen Produktionsserver ist dagegen dringend der Release-Modus zu empfehlen.

Wenn Sie dasselbe Quellcode-Paket zum wiederholten Mal zur Kompilierung verwenden, sollten Sie zuerst die passende `clean`-Option durchlaufen lassen, um aufzuräumen. Am praktischsten ist es, dies unmittelbar nach dem Build zu erledigen. Wenn Sie noch wissen, welche Variante (Debug oder Release) Sie beim letzten Mal kompiliert haben, können Sie gezielt die spezialisierte Variante `_cleand` beziehungsweise `_cleanr` aufrufen, andernfalls müssen Sie sich mit dem nicht ganz so gründlichen allgemeinen `clean` begnügen. Sie können die folgenden zusätzlichen Optionen für die Kompilierung angeben:

- `INSTDIR=Pfad`
  Dieses Verzeichnis wird die Server-root des fertig kompilierten Apache 2; bei den `install`-Builds werden die fertigen Dateien automatisch dorthin kopiert. Standard: `\Apache2`.
- `SERVERNAME=DNS-Name`
  Diese Option gibt den Namen an, unter dem der künftige Server angesprochen werden soll. Es handelt sich hier lediglich um einen entsprechenden Eintrag in der Apache-Konfigurationsdatei; die tatsächliche Erreichbarkeit unter diesem Namen kann nur eine passende Konfiguration Ihrer Nameserver (siehe Kapitel 1, »IP-Netzwerke, Internet und WWW«) garantieren. Standard: `localhost`.
- `PORT=TCP-Port`
  Wenn Ihr Webserver an einem anderen TCP-Port als dem Standard-HTTP-Port 80 lauschen soll, können Sie hier einen entsprechenden Wert eingeben. Dies ist nützlich für einen Zweit-Webserver oder einen zu reinen Testzwecken installierten Apache 2.

Angenommen, Sie möchten eine Release-Version mit dem Server-Namen *www.mynet.de* kompilieren und automatisch nach *D:\Programm Files\Apache2* installieren. Der passende `nmake`-Aufruf sieht dann so aus:

```
> nmake /f Makefile.win INSTDIR="D:\Program Files\Apache2"
SERVERNAME=www.mynet.de installr
```

Nachdem Sie den für Sie passenden Befehl eingegeben haben, wird die Kompilierung durchgeführt, die recht lange dauern kann. Beobachten Sie die Ausgaben von `nmake`, um Hinweise auf eventuelle Probleme zu erhalten. Wenn Sie nicht die ganze Zeit vor dem Rechner sitzen und auf die Kompilierung warten möchten, sollten Sie die Ausgabe für eine spätere Untersuchung in eine Datei umleiten: Hängen Sie einfach `>DATEINAME` an Ihren `nmake`-Befehl an.

#### Apache 2 in der Visual-Studio-IDE kompilieren

Ein wenig bequemer (aber auch etwas langsamer) geht die Kompilierung in der grafischen Entwicklungsumgebung von Visual Studio vonstatten. Folgende Schritte sind dafür erforderlich:

1. Starten Sie Visual Studio, und wählen Sie Datei • Projektmappe öffnen. Da die Apache 2-Projektdatei im älteren DSW-Format vorliegt, müssen Sie den Dateityp unter Visual Studio .NET auf Alle Projektdateien einstellen. Wählen Sie die Datei `Apache.dsw` aus dem Stammverzeichnis des Quellcode-Pakets, und klicken Sie auf Öffnen.

   Falls Sie Visual Studio .NET verwenden, erhalten Sie nun eine Meldung, dass die Projektdateien aktualisiert werden müssen. Wählen Sie Ja und anschließend alle, um die Projektmappe mit allen Unterprojekten zu aktualisieren.

2. Klicken Sie nun im Fenster Projektmappen-Explorer den Eintrag InstallBin an. Dies ist das Projekt für die Apache-Installationsversion. Wenn Sie zu Testzwecken nur kompilieren, aber nicht installieren möchten, können Sie stattdessen BuildBin wählen.

   Wenn Sie die Optionen ändern möchten, die im Abschnitt »Apache 2 auf der Windows-Kommandozeile installieren« beschrieben wurden, können Sie InstallBin mit der rechten Maustaste anklicken und Eigenschaften aus dem Kontextmenü auswählen. Unter NMake finden Sie den Eintrag Build-Befehlszeile. Hier lässt sich der aufzurufende `nmake`-Befehl gemäß den Anweisungen im gerade zitierten Abschnitt anpassen.

3. Wählen Sie im Menü Erstellen • InstallBin erstellen, um mit der Kompilierung zu beginnen. Falls Sie das Projekt schon einmal kompiliert haben (oder gerade mit der Kompilierung fertig geworden sind), sollten Sie Erstellen • InstallBin bereinigen wählen, um aufzuräumen.

## 4.2 Die binäre Apache-Distribution für Windows installieren

Unter Windows empfiehlt sich in den meisten Fällen die Installation von Apache 2 aus dem offiziellen Windows-Installer-Paket (MSI). Sie finden die

Dateien auf der DVD zum Buch oder können sie von der Apache-Website herunterladen. Auf der Haupt-Download-Site für den Apache-Webserver (*http://httpd.apache.org/download.cgi*) finden Sie unmittelbar einen Link auf diese Datei. Klicken Sie dazu den Link neben Win32 Binary (MSI Installer) an. Die Datei gibt es in zwei Versionen: mit und ohne OpenSSL (siehe Kapitel 10, »Gesicherte Verbindungen«). Sie heißt zurzeit entsprechend *apache_2.2.21-win32-x86-no_ssl.msi* beziehungsweise *apache_2.2.21-win32-x86-openssl-0.9.8r.msi*.

Offizielle Binaries für 64-Bit-Windows bietet die Apache Software Foundation noch nicht an. Sie finden allerdings bei Drittanbietern vorkompilierte 64-Bit-Binaries mit Installer – beispielsweise auf der Website von *Jorge Schrauwen* unter *http://www.blackdot.be/?inc=apache/binaries*. Dort finden Sie zudem Links auf inoffizielle Windows-Binaries von Apache 2.3/2.4.

Unter Windows 7, Vista[2], XP, 2000 und Me können Sie die Installation anschließend einfach per Doppelklick starten, weil der Windows Installer im System vorinstalliert ist. Windows NT 4.0 und 9x waren dagegen ab Werk nicht damit ausgestattet. Suchen Sie auf der Microsoft-Website danach.

Die Nummerierung der folgenden Schritt-für-Schritt-Anleitung bezieht sich auf die verschiedenen Dialoginhalte, die Sie nach dem Start der Installation sehen werden:

1. Der erste Bildschirm informiert kurz darüber, dass das Programm Apache installiert wird. Klicken Sie Next an.
2. Auf dem nächsten Bildschirm wird die Apache-Lizenz angezeigt. Informationen über diese Lizenz, die bekannten Open-Source-Lizenzen wie der GNU GPL ähnelt, finden Sie in Kapitel 3, »Apache 2 im Überblick«, den Wortlaut in Anhang F. Wählen Sie I accept the terms in the license agreement (andernfalls geht es nicht weiter), und klicken Sie Next an.
3. Der dritte Dialog zeigt eine allgemeine Übersicht über den Apache-Webserver an und nennt weitere Informationsquellen. Wenn Sie den Text später lesen möchten, können Sie ihn markieren und über die Zwischenablage in einen Texteditor kopieren. Klicken Sie anschließend auf Next.
4. Im folgenden Dialog (siehe Abbildung 4.1) müssen Sie einige grundlegende Daten eingeben, die in der automatisch erstellten Version der Konfigurationsdatei *httpd.conf* verwendet werden:

2 Frühere Releases von Apache 2.2 ließen sich nicht ohne Weiteres unter Vista installieren, auch der in Kapitel 5, »Apache in Betrieb nehmen«, beschriebene Apache-Monitor funktionierte dort nicht. Verwenden Sie also unbedingt ein aktuelles Release; bei älteren Versionen müssen Sie unter anderem das Kommandozeilen-Tool `msiexec` als Administrator starten, um die notwendigen Rechte für die Installation zu erhalten.

- Network Domain – Geben Sie den Domain-Namen der Domain ein, in der der Server arbeiten soll. Beispiel: *mynet.de*.
- Server Name – Geben Sie den Host- beziehungsweise Dienstnamen an, unter dem der Webserver erreichbar sein soll. In der Regel wird hier derselbe Domain-Name wie oben mit vorangestelltem *www.* verwendet, also etwa *www.mynet.de*.
- Administrator's E-Mail Address – Auf automatisch generierten Fehlermeldungsseiten zeigt Apache einen Link auf die hier eingegebene E-Mail-Adresse an, damit Besucher Ihrer Website Ihnen einen Hinweis über deren Fehlfunktion geben können. Beispiel: *webmaster@mynet.de*.
- Die letzte Frage ist die wichtigste: Soll Apache 2 als Produktionsserver oder als Testprogramm installiert werden? Wählen Sie For All Users, on Port 80, as a Service, um Apache als automatisch startenden Dienst auf dem Standard-TCP-Port für HTTP zu installieren; dies dürfte der Normalfall sein. Wollen Sie den Server dagegen nur zum Ausprobieren oder für die Anwendungsentwicklung verwenden, können Sie only for the Current User, on Port 8080, when started Manually auswählen.

**Abbildung 4.1** Eintrag der Server-Informationen bei der Apache-Installation unter Windows

Nachdem Sie alle Einträge vorgenommen haben, können Sie Next anklicken.

Denken Sie bei den Domain- und Server-Namen daran, dass die Eintragung bei der Installation diese Werte nur in der Webserver-Konfigurationsdatei festlegt. Dies garantiert nicht, dass Ihr Webserver nach außen unter diesem

Namen erreichbar ist – damit das der Fall ist, müssen Sie Ihre Nameserver entsprechend einrichten. Einen kleinen Einstieg in dieses Thema finden Sie in Kapitel 1, »IP-Netzwerke, Internet und WWW«.

**Vorsicht!**

Wenn zum Zeitpunkt der Installation eine Internetverbindung über einen Provider besteht, kann es vorkommen, dass der Installer automatisch die Werte übernimmt, die der ISP Ihnen zugewiesen hat. Lesen Sie also sorgfältig, was dort eingetragen ist – eine Domain wie `t-online.de` oder ein Server-Name wie `xdsl-47-11.freenet.de` sind weder für einen Produktionsserver noch für eine lokale Testinstallation geeignet, zumal Ihnen beim nächsten Internet-Verbindungsaufbau fast immer ein anderer Hostname zugeordnet wird.

Andererseits bedeutet das Ändern dieser Einstellungen auch für einen Testserver noch nicht, dass er nicht von außen erreichbar wäre. Wie Sie Hosts aussperren können, die nicht zum lokalen Netz gehören, erfahren Sie in Kapitel 6, »Grundkonfiguration«.

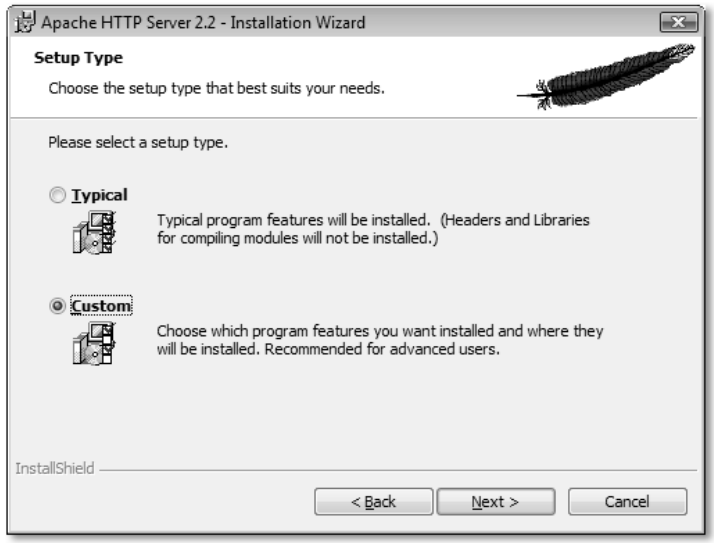



**Abbildung 4.2** Bei der Installation unter Windows können Sie sich zwischen einer Standardinstallation und auswählbaren Komponenten entscheiden.

5. Das nächste Dialogfeld sehen Sie in Abbildung 4.2. Hier können Sie auswählen, ob Sie eine Standardinstallation (TYPICAL) oder eine angepasste Installation (CUSTOM) durchführen möchten. Die Standardinstallation enthält keine Header und Bibliotheken, die zum Kompilieren von Modulen benötigt werden. Wählen Sie daher CUSTOM, und klicken Sie auf NEXT.

6. Den folgenden Bildschirm (siehe Abbildung 4.3) sehen Sie nur, wenn Sie im vorigen Schritt Custom gewählt haben. Hier können Sie in einer Baumansicht auswählen, welche Teile von Apache 2 installiert werden sollen. Wenn Sie genügend Festplattenplatz (bei Version 2.2.21 sind es gut 30 Megabyte) zur Verfügung haben, können Sie einfach für den Punkt Apache HTTP Server 2.2.21 die Option This Feature, and all subfeatures, will be installed on local hard drive wählen. Andernfalls sollten Sie vorrangig die 11 Megabyte große Dokumentation weglassen, da sie auch online verfügbar ist. Die Unterpunkte SSL Binaries und OpenSSL Runtime sind natürlich nur in der SSL-Version verfügbar.

   Unten im Fenster wird das Verzeichnis angezeigt, in das Apache installiert werden soll. Vorausgewählt ist das Verzeichnis *Apache Software Foundation\Apache2.2* im Systemverzeichnis für Anwendungsprogramme – z. B. *C:\Program Files* in Windows Vista. Es empfiehlt sich allerdings, Apache nicht so tief in den Verzeichnisbaum zu verschachteln, da Sie wahrscheinlich häufig auf das Verzeichnis zugreifen werden, um die Konfiguration zu ändern oder die Websites selbst zu bearbeiten. Wenn Sie auf die Schaltfläche Change... klicken, lässt sich das Verzeichnis ändern. Zu guter Letzt können Sie wieder Next anklicken.

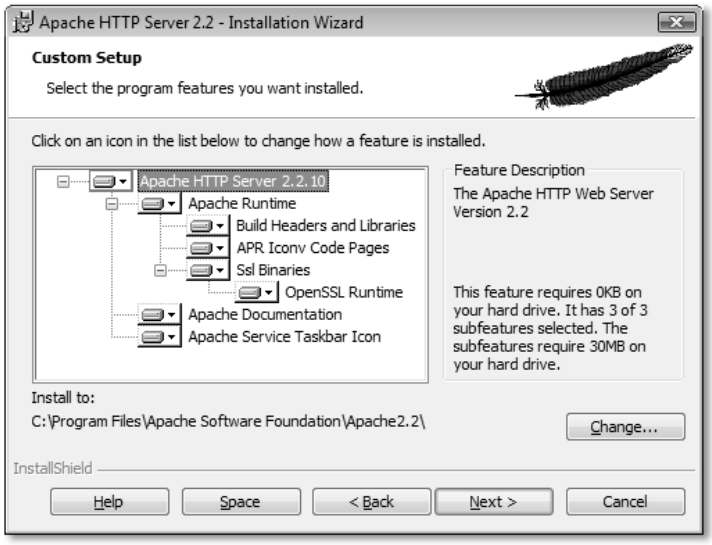



**Abbildung 4.3** Auswahl der Programmfeatures bei der angepassten Installation

7. Der nun folgende Bildschirm ist die letzte Chance, die Installationseinstellungen zu ändern (auf Back klicken). Wenn Sie sicher sind, dass alle Einstellungen in Ordnung sind, klicken Sie dagegen Install an. Daraufhin wird die Installation durchgeführt; das dauert etliche Minuten.

**Diese Version deinstallieren**

Die Standarddistribution im MSI-Format bietet den zusätzlichen Vorteil, dass sie sich wieder leicht und (fast) vollständig deinstallieren lässt. Bevor Sie eine neuere Version von Apache installieren, müssen Sie dies auch tun. Wählen Sie dazu in der Systemsteuerung das Applet Programme und Funktionen (Vista) beziehungsweise Software (ältere Windows-Versionen). Suchen Sie im Bereich Programme deinstallieren oder ändern nach Apache HTTP Server 2.2.x, und klicken Sie je nach Betriebssystemversion auf Deinstallieren, Entfernen beziehungsweise Ändern/Entfernen. Folgen Sie anschließend den Anweisungen des Deinstallationsprogramms.

Beachten Sie, dass mindestens die folgenden Verzeichnisse innerhalb Ihres Apache-Installationsverzeichnisses aus gutem Grund bestehen bleiben (je nach Konfiguration und installierten Modulen auch noch mehr):

- *conf* – Konfigurationsdateien
- *logs* – Log-Dateien
- *htdocs* – das Standard-Website-Verzeichnis
- *cgi-bin* – CGI-Skripte

Auf diese Weise wird gewährleistet, dass Sie diese wichtigen Daten behalten und in eine neu installierte Apache-Version kopieren können.

**XAMPP**

Für Entwicklerrechner gibt es eine Alternative zu den bisher beschriebenen Installationsvarianten: Unter `http://www.apachefriends.org` können Sie sich das Komplettpaket XAMPP herunterladen. Darin sind neben dem Apache-Webserver auch PHP, MySQL, phpMyAdmin und andere Serverkomponenten enthalten. XAMPP ist für Linux, Windows und Mac OS X verfügbar; es handelt sich um eine Archivdatei, die Sie lediglich zu entpacken brauchen. Danach können Sie alle enthaltenen Server gemeinsam oder einzeln über einfache Startskripte in Betrieb nehmen. Bitte beachten Sie, dass XAMPP nicht als Server im realen Produktiveinsatz geeignet ist, da die Konfiguration erheblich zu unsicher ist.

## 4.3 Module nachträglich installieren

Der große Vorteil der bereits erläuterten DSO-Unterstützung für Module besteht darin, dass Sie nachträglich Module hinzufügen (oder wieder entfernen) können, ohne den Server neu installieren zu müssen. Das Verfahren wird in diesem Abschnitt für UNIX-Systeme erläutert und ist im Prinzip für beliebige Module geeignet. Allerdings sind für einige komplexe Module wie `mod_perl` oder `mod_ssl`

zusätzliche Schritte erforderlich, die nichts mit der eigentlichen Modulinstallation zu tun haben – diese werden zum Teil erst in den entsprechenden Kapiteln näher erläutert.

Für die Installation von Modulen ist grundsätzlich das mit Apache gelieferte Perl-Skript `apxs` zuständig. Es befindet sich im `SBINDIR` Ihrer Installation. Beachten Sie, dass das hier beschriebene Verfahren nur auf UNIX-Systemen funktioniert. Zwar ist `apxs` auch Bestandteil des Windows-Apache, sein tatsächliches Funktionieren hängt aber von der Verfügbarkeit der üblichen UNIX-Entwicklungswerkzeuge ab. Wenn Sie ein Modul herunterladen, das keine Installationsanleitung für Windows enthält, stehen die Chancen mit anderen Worten sehr schlecht. Wie Sie wichtige externe Module wie `mod_perl`, `mod_php` oder `mod_ssl` unter Windows zum Laufen bekommen, wird deshalb in den entsprechenden Kapiteln separat erläutert.

Je nachdem, wie komplex das zu installierende Modul ist, gibt es zwei grundsätzliche Vorgehensweisen für die Verwendung des Skripts:

- Ein einfaches Modul, das nur aus einer einzigen C-Quelldatei besteht, kann durch einen direkten Aufruf von `apxs` in einem Schritt kompiliert und installiert werden. Betrachten Sie als Beispiel das Modul `mod_acronym`. Sie können es von der DVD zum Buch (Verzeichnis *modules/mod_acronym*) kopieren, auf *modules.apache.org* finden oder direkt von der Website seines Autors (*http://www.apache.org/~dirkx/oscon2002/mod_acronym*) herunterladen. Dieses Modul ist ein Beispiel für einen Ausgabefilter, der Akronyme in HTML-Dokumenten automatisch durch die Langfassung ersetzen kann. Der folgende Befehl wird zur Installation verwendet:

  ```
  # apxs -cia mod_acronym.c
  ```

  Die Optionen `-c`, `-i` und `-a` stehen für Kompilieren, Installieren und Aktivieren. Das Modul wird also zunächst zur `.so`-Datei kompiliert, anschließend in das richtige Verzeichnis kopiert und zuletzt aktiviert, indem automatisch die korrekte `LoadModule`-Direktive zur Datei *httpd.conf* hinzugefügt wird.
- Komplexe Module, die sich aus mehreren C- und Header-Dateien zusammensetzen, müssen dagegen in mehreren Schritten kompiliert und installiert werden. Ein interessantes Beispiel ist `mod_ftpd`, ein Modul, das die Multiprotokoll-Fähigkeit von Apache 2.2 demonstriert, indem es einen FTP-Server hinzufügt. Sie finden es auf der DVD zum Buch im Verzeichnis *modules/mod_ftpd* oder können es unter *http://www.outoforder.cc/projects/apache/mod_ftpd* herunterladen. Das Paket ist ein bzip2-komprimierter Tarball, der sich folgendermaßen entpacken lässt (die Versionsnummer hat sich inzwischen möglicherweise geändert, wenn Sie dieses Buch lesen):

```
# bunzip2 mod_ftpd-0.14.tar.bz2
# tar xvf mod_ftpd-0.14.tar
```

Anschließend können Sie mittels `cd` in das neu angelegte Verzeichnis `mod_ftpd-0.14` wechseln und folgende Befehle eingeben, um das Modul zu installieren:

```
# ./configure --with-apxs=/Pfad/von/apxs
# make
# make install
```

Der `/Pfad/von/apxs` richtet sich, wie immer in solchen Fällen, nach Ihrem Installationslayout. Beim `GNU`-Layout ist es beispielsweise */usr/local/sbin*, beim `Apache`-Layout */usr/local/apache2/bin*.

Auch wenn eines dieser beiden Standardverfahren bei den meisten Modulen funktionieren sollte, ist es empfehlenswert, nach einer Installationsanleitung für das jeweilige Modul Ausschau zu halten, um Informationen über Besonderheiten zu erhalten.

## 4.4 Zusammenfassung

Die Vielzahl von Installationsvarianten für den Apache-Webserver könnte auf Sie verwirrend wirken. Lesen Sie sich die Anleitungen in diesem Kapitel in Ruhe durch, und überprüfen Sie, wie sich Ihr konkretes Ziel am besten erreichen lässt.

Grundsätzlich gibt es zwei verschiedene Methoden, den Server zu installieren: Sie können ein Quellcode-Paket selbst kompilieren oder eine binäre Distribution verwenden. Die Kompilierung bietet den Vorteil der größeren Flexibilität, während die Binärinstallation schneller und einfacher funktioniert.

Um Apache 2 selbst zu kompilieren, müssen Sie sich zunächst das Paket mit den Quellcodes besorgen. Sie finden es auf der DVD zum Buch, sollten sich aber vergewissern, dass noch keine neuere Version erschienen ist. Ansonsten können Sie die jeweils neueste Fassung aus dem Web herunterladen. Anschließend sind je nach Plattform unterschiedliche Arbeitsschritte erforderlich, um die Kompilierung durchzuführen.

Die binäre Installation empfiehlt sich vor allem unter Windows: Von jeder neuen Apache-Version wird relativ schnell ein standardisiertes MSI-Paket erstellt, das der üblichen Softwareinstallation unter Windows entspricht und nicht nur den Server, sondern auch die meisten wichtigen Module enthält. Auf UNIX-Systemen ist die Binärinstallation dagegen eigentlich nur sinnvoll, wenn es ein angepasstes Installationspaket für den Paketmanager Ihrer Systemdistribution gibt.

*»Im Grunde bewegen nur zwei Fragen die Menschheit:*
*Wie hat alles angefangen, und wie wird alles enden?«*
*– Stephen Hawking*

# 5 Apache in Betrieb nehmen

In diesem Kapitel wird die komplette Steuerung des Apache-HTTP-Servers behandelt: Sie erfahren zunächst, welche Optionen das ausführbare Programm bietet, um den Server zu starten, zu beenden oder neu zu starten. Anschließend erfahren Sie, wie sich der Start von Apache auf verschiedenen Plattformen beim Booten automatisieren lässt. Zudem lernen Sie einige Apache-Steuer-Tools aus Systemdistributionen sowie von Drittanbietern kennen. Zu guter Letzt wird eine Minimalkonfiguration vorgestellt, mit der Sie die ordnungsgemäße Funktion des Servers überprüfen können.

## 5.1 Apache 2 starten und beenden

Nachdem Sie den Apache-Webserver nun (hoffentlich) auf die eine oder andere in Kapitel 4, »Apache kompilieren und installieren«, beschriebene Weise installiert haben, können Sie ihn starten. Je nach Betriebssystem und Plattform stehen dafür unterschiedliche Optionen zur Verfügung. Sie erfahren in diesem Abschnitt, wie sich der Server unter UNIX und Windows starten, beenden und neu starten lässt. Außerdem werden verschiedene Optionen des automatischen Starts beim Hochfahren besprochen.

### 5.1.1 Apache unter UNIX steuern

Auf UNIX-Systemen können Sie das ausführbare Programm `httpd` mit verschiedenen Parametern aufrufen, um den Webserver zu starten, neu zu starten, zu beenden oder Informationen über seinen Status zu erhalten. Die bevorzugte Variante ist allerdings die Verwendung des mitinstallierten Shell-Skripts `apachectl`, das eine genauere Kontrolle ermöglicht. Beide Varianten werden hier vorgestellt.

Ob der Webserver bereits ausgeführt wird, können Sie (als `root`) mit folgendem Kommando herausfinden:

```
# ps aux |grep httpd[1]
```

Bei einer Standardinstallation mit dem MPM `prefork` sollte das Ergebnis etwa so aussehen wie in Abbildung 5.1, falls Apache läuft. Andernfalls erhalten Sie höchstens eine Zeile, die besagt, dass `grep httpd` ausgeführt wird – eben der Suchfilter des Befehls, den Sie gerade eingegeben haben.

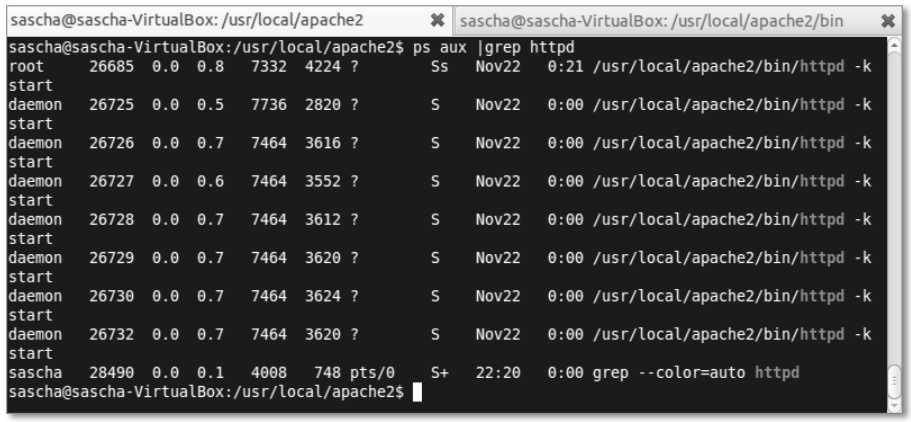



**Abbildung 5.1** Anzeige der laufenden Apache-Prozesse auf einem Linux-Host

### Parameter des Programms httpd

Der Funktionskern des Webservers Apache 2 ist das ausführbare Programm (Binary Executable) `httpd`. Falls Sie den Server nach der Anleitung aus Kapitel 4, »Apache kompilieren und installieren«, über die Option `--with-program-name` mit einem anderen Programmnamen kompiliert haben sollten, gilt dieser im Folgenden sinngemäß. Bei der mitgelieferten Version in manchen Linux-Distributionen heißt das Binary beispielsweise `apache2`. Je nach Installationslayout beziehungsweise eingestelltem Verzeichnis befindet sich das Programm an unterschiedlichen Stellen in Ihrem Dateisystem – genauer gesagt, im `SBINDIR` Ihrer Installation. Haben Sie beispielsweise das Standardlayout `Apache` verwendet, dann finden Sie die Datei im Verzeichnis */usr/local/apache2/bin*, beim `GNU`-Layout dagegen unter */usr/local/sbin*.

Um einen sauber installierten Apache-Webserver zu starten, sollte in der Regel der folgende Befehl ausreichen:

```
$ httpd
```

(Ob Sie dem Befehl zusätzlich eine Pfadangabe voranstellen müssen, hat natürlich damit zu tun, ob das `SBINDIR` sich in Ihrem `PATH` befindet oder nicht.)

1 Je nach konkreter UNIX-Variante können die Parameter für `ps` variieren; konsultieren Sie gegebenenfalls die Manpage des Kommandos, indem Sie `man ps` eingeben.

Wenn der Server auf diese einfache Weise nicht startet oder wenn Sie besondere Einstellungen für seinen Start vornehmen möchten, können Sie aus den folgenden Kommandozeilenparametern auswählen:

- **`-k Option`**
  Der Parameter –`k` mit einer der Optionen `shutdown`, `restart`, `graceful` oder `graceful-stop` dient dem Beenden beziehungsweise dem Neustart des Servers. Diese Befehle werden im Folgenden in der Beschreibung des Skripts `apachectl` näher erläutert.
- **`-D Name`**
  Dieser Parameter eröffnet Ihnen ein sehr praktisches Verfahren, in der Konfigurationsdatei `httpd.conf` optionale Direktiven unterzubringen und nach Belieben auszuführen oder auch nicht: Sie können Direktiven in einen Block nach dem Schema `<IfDefine Name>...</IfDefine>` einschließen. Sie werden nur dann ausgeführt, wenn der entsprechende `Name` beim Start von Apache mit diesem Parameter definiert wird. Die Verwendung von `<IfDefine>` in der Konfigurationsdatei wird in Kapitel 6, »Grundkonfiguration«, erläutert.
- **`-d Verzeichnis`**
  Mithilfe dieses Parameters lässt sich eine alternative `ServerRoot` angeben – es handelt sich um das Verzeichnis, in dem die Inhalts- und Konfigurationsdateien des Servers liegen.
- **`-f Datei`**
  Um Ihren Server mit einer anderen Konfigurationsdatei (`ServerConfigFile`) auszuführen als mit `httpd.conf` in Ihrem `SYSCONFDIR`, können Sie diese alternative Datei mithilfe der vorliegenden Option angeben.
- **`-C "Direktive"`**
  Diese Option ermöglicht die Angabe einer zusätzlichen Konfigurationsdirektive, die vor dem Einlesen der Konfigurationsdatei verarbeitet werden soll.
- **`-c "Direktive"`**
  Dieser Parameter ähnelt dem vorigen, mit einem Unterschied: Eine Direktive, die Sie hier angeben, wird erst nach der Verarbeitung der Konfigurationsdatei ausgeführt – dies ermöglicht das gezielte Überschreiben einer Direktive der Server-Konfiguration.
- **`-e Level`**
  Der Parameter gibt die Dringlichkeit beim Start auftretender Fehler an, ab der diese Fehler angezeigt werden sollen. Die möglichen Werte dieser Option sind die bekannten `syslog`-Fehlerprioritäten wie `notice`, `err` oder `alert`. Sie werden im Zusammenhang mit der Konfigurationsdirektive `LogLevel` noch näher erläutert werden.

- **-E Datei**
  Bestimmt, dass Startfehler nicht auf der Konsole (beziehungsweise `stderr`) angezeigt, sondern in die angegebene Datei geschrieben werden sollen.
- **-R Verzeichnis**
  Wenn Apache mit dem Compiler-Flag `SHARED_CORE` kompiliert wurde (siehe Kapitel 4, »Apache kompilieren und installieren«), gibt diese Option das Verzeichnis an, in dem sich die Shared Objects des Apache-Core befinden.
- **-X**
  Apache wird im Debug-Modus ausgeführt: Es ist nur ein einziger Worker-Prozess beziehungsweise -Thread verfügbar und die Verbindung zum startenden Terminal wird nicht gelöst, sodass eventuelle Fehlermeldungen und Warnungen auf der Konsole erscheinen.

Die Verwendung der folgenden Optionen startet den Server nicht, sondern dient der Ausgabe verschiedener Informationen:

- **-v**
  Dieser Parameter zeigt nur Versionsinformationen an.
- **-V**
  Zeigt ausführliche Versionsinformationen an sowie die Einstellungen, mit denen Apache 2 kompiliert wurde. Die Ausgabe sieht für 2.3-beta unter Ubuntu Linux beispielsweise so aus:

```
Server version: Apache/2.3.14 (Unix)
Server built:   Oct 24 2011 22:14:41
Server's Module Magic Number: 20110724:0
Server loaded:  APR 1.4.5, APR-UTIL 1.3.12
Compiled using: APR 1.4.5, APR-UTIL 1.3.12
Architecture:   32-bit
Server MPM:     prefork
  threaded:     no
    forked:     yes (variable process count)
Server compiled with....
 -D APR_HAS_SENDFILE
 -D APR_HAS_MMAP
 -D APR_HAVE_IPV6 (IPv4-mapped addresses enabled)
 -D APR_USE_SYSVSEM_SERIALIZE
 -D APR_USE_PTHREAD_SERIALIZE
 -D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
 -D APR_HAS_OTHER_CHILD
 -D AP_HAVE_RELIABLE_PIPED_LOGS
 -D DYNAMIC_MODULE_LIMIT=256
 -D HTTPD_ROOT="/usr/local/apache2"
 -D SUEXEC_BIN="/usr/local/apache2/bin/suexec"
```

```
-D DEFAULT_PIDLOG="logs/httpd.pid"
-D DEFAULT_SCOREBOARD="logs/apache_runtime_status"
-D DEFAULT_ERRORLOG="logs/error_log"
-D AP_TYPES_CONFIG_FILE="conf/mime.types"
-D SERVER_CONFIG_FILE="conf/httpd.conf"
```

Die ersten beiden Zeilen bilden übrigens die Ausgabe der Option `-v`.

Noch ein Wort zur »Module Magic Number«: Dieser Wert entspricht der Build-Nummer des vorliegenden Apache-Release. Er ist in der Quellcode-Datei *include/ap_mmn.h* durch die symbolischen `MODULE_MAGIC_NUMBER_MAJOR` und `MODULE_MAGIC_NUMBER_MINOR` definiert. Die Werte für Apache 2.3.14-beta sind `20110724` und `0`; bei Version 2.2 sind es `20051115` und `24`. DSO-Module mit derselben Magic Number (relevant ist die Zahl vor dem Doppelpunkt) können auf derselben Plattform im Binärformat installiert werden und brauchen nicht neu kompiliert zu werden.

- **`-h`**
  Zeigt eine Liste sämtlicher Kommandozeilenoptionen an (wie diese hier).
- **`-l`**
  Der Parameter zeigt eine Liste aller einkompilierten Module an.
- **`-L`**
  Mit dieser Option erhalten Sie eine Liste sämtlicher verfügbarer Konfigurationsdirektiven – welche das sind, hängt von den vorhandenen Modulen ab.
- **`-t -D DUMP_VHOSTS`**
  Die Option `-t -D Name` soll allgemein bestimmte Einstellungen anzeigen, die sich aus der bereits verarbeiteten Konfiguration ergeben. Zurzeit ist nur die spezielle Variante `-t -D DUMP_VHOSTS` verfügbar, die die entsprechenden Ergebnisse für virtuelle Hosts ausgibt.
- **`-S`**
  Dies ist eine Kurzfassung für die Option `-t -D DUMP_VHOSTS`.
- **`-t -D DUMP_MODULES`**
  Zeigt eine Liste der tatsächlich aktiven Module an, sowohl der fest einkompilierten als auch der DSO-Module, die durch eine `LoadModule`-Direktive (siehe Kapitel 6, »Grundkonfiguration«) geladen wurden. Diese Option steht erst seit Apache 2.1-beta zur Verfügung.
- **`-M`**
  Dies ist die Kurzschreibweise für `-t -D DUMP_MODULES`.
- **`-t`**
  Diese Option überprüft die gewählte Konfigurationsdatei (und eventuelle zusätzliche Direktiven) auf syntaktische Richtigkeit.

### Apache manuell beenden und neu starten

Da in Version 2 des Apache-Webservers das weiter unten behandelte Skript *apachectl* zur Verfügung steht, ist es eigentlich nicht mehr nötig, die hier beschriebenen manuellen Verfahren anzuwenden. Allerdings verdeutlichen sie die Technik, die diesem Shell-Skript zugrunde liegt. Deshalb ist ihre Kenntnis beispielsweise für eigene, angepasste Steuerskripte erforderlich.

Wie bei UNIX-Daemons üblich, reagiert der Steuerprozess – der ursprüngliche Parent-Prozess, der die Child-Prozesse zur Verarbeitung von Client-Anfragen startet – auf einige Standardsignale, die das Beenden beziehungsweise den Neustart bewirken. Solche Signale werden durch das Hilfsprogramm `kill` versendet, das den gleichnamigen Systemaufruf ausführt. Die allgemeine Syntax dieses Befehls lautet bekanntermaßen so:

```
# kill [-SIGNAL] PID
```

Das einzige Problem besteht darin, den Steuerprozess ausfindig zu machen. Wie Sie in Abbildung 5.1 gesehen haben, gibt es bei einem `prefork`-Apache mehrere Prozesse mit dem Namen *httpd*. Welcher von ihnen der Steuerprozess ist, lässt sich auf zweierlei Arten ermitteln:

- Es ist derjenige Prozess, der unter der User-ID `root` ausgeführt wird. Die Child-Prozesse für die Client-Bedienung laufen unter einer anderen UID, meistens `nobody` oder `daemon` (wird durch die Direktive `User` festgelegt; siehe Kapitel 6, »Grundkonfiguration«).
- Die zweite Methode ist die sicherste: Apache legt beim Start eine PID-Datei an. Diese enthält die Prozess-ID des Steuerprozesses. Diese Datei wird unter anderem dazu verwendet, um bei einem Startversuch festzustellen, ob der Server bereits läuft oder ob er womöglich unsauber beendet wurde. Letzteres ist der Fall, wenn die PID-Datei zwar noch existiert, aber kein Prozess unter der Nummer läuft, die darin steht.

  Wo sich die PID-Datei (*httpd.pid*) befindet, hängt wieder einmal vom verwendeten Verzeichnislayout ab. Sie liegt im `RUNTIMEDIR`: Beim `Apache`-Layout ist dies */usr/local/apache2/logs*, beim `GNU`-Layout */usr/local/var/apache2/run*.

  Bequemerweise können Sie zur Angabe der PID im `kill`-Befehl unmittelbar den Wert aus *httpd.pid* verwenden. Setzen Sie dazu einfach den Ausgabebefehl `cat /Pfad/von/httpd.pid` in `` `Backticks` ``; unter Verwendung des `GNU`-Layouts zum Beispiel so:

  ```
  `cat /usr/local/var/apache2/run/httpd.pid`
  ```

Nun brauchen Sie nur noch zu wissen, welche Signale verwendet werden, um die unterschiedlichen Verhaltensweisen des Servers zu bewirken:

- **TERM** beendet Apache 2 vollständig. Da TERM das Standardsignal für kill ist, brauchen Sie gar kein Signal anzugeben, um Apache zu beenden.

  Hier ein Beispiel mit der nach dem ersten Verfahren ermittelten PID aus Abbildung 5.1:

  ```
  # kill 23100
  ```

  Natürlich können Sie das Signal TERM auch explizit angeben, wenn Sie möchten:

  ```
  # kill -TERM 23100
  ```

- **HUP** bewirkt einen normalen Neustart des Webservers: Alle Child-Prozesse werden sofort beendet (wobei laufende Übertragungen natürlich abgebrochen werden); anschließend wird der Parent-Prozess neu gestartet und erzeugt wieder die vorgesehene Anfangszahl von Child-Prozessen.

  Das folgende Beispiel funktioniert, wenn Ihr Apache 2 das GNU-Layout verwendet:

  ```
  # kill -HUP `cat /usr/local/var/apache2/run/httpd.pid`
  ```

- **USR1** sorgt für einen unterbrechungsfreien Neustart (graceful restart) des Servers: Child-Prozesse, die sich gerade um Verbindungen kümmern, werden erst beendet, wenn der aktuelle Vorgang abgeschlossen ist; untätige Child-Prozesse werden sofort beendet. Der Parent-Prozess wird schließlich neu gestartet, nachdem alle Child-Prozesse sauber abgeschlossen wurden.

  Hier sehen Sie ein Beispiel für das Verzeichnislayout Apache:

  ```
  # kill -USR1 `/usr/local/apache2/logs/httpd.pid`
  ```

- WINCH schließlich führt zum sauberen Beenden (graceful shutdown) von Apache: Mit dem Beenden des Hauptprozesses wird gewartet, bis alle Worker-Prozesse oder -Threads mit der Verarbeitung ihrer Verbindungen fertig sind. Optional lässt sich mithilfe der Direktive GracefulShutdownTimeout (siehe Kapitel 6, »Grundkonfiguration«) eine maximale Wartezeit festlegen, nach der Apache auf jeden Fall beendet wird. Hier ein Beispiel; es gilt wieder für das Standardlayout Apache:

  ```
  # kill -WINCH `/usr/local/apache2/logs/httpd.pid`
  ```

#### Parameter des Skripts apachectl

Eine bequemere Steuerung des Apache-Webservers bietet das Shell-Skript apachectl, das bei der Kompilierung oder Binärinstallation von Version 2 mitgeliefert wird. Es enthält zahlreiche Optionen zum Starten, Beenden, Neustarten und Überwachen des Servers.

Das Skript befindet sich im BINDIR Ihres Verzeichnislayouts. Dies ist beim Apache-Layout */usr/local/apache2/bin*, beim GNU-Layout */usr/local/bin*. Die allgemeine Syntax lautet folgendermaßen:

```
# apachectl Befehl
```

apachectl bietet folgende Befehle an:

- **start**
  **-k start**
  Der Befehl start beziehungsweise -k start startet den Webserver mit Standardoptionen. Dies ist der Standardbefehl, wenn Sie apachectl ohne weitere Optionen aufrufen.
- **stop**
  **-k stop**
  Wenn Sie einen dieser beiden Befehle eingeben, wird der Server beendet.
- **restart**
  **-k restart**
  Diese Befehle senden nach dem oben beschriebenen Schema ein HUP-Signal an den HTTP-Server, um ihn ohne Rücksicht auf bestehende Verbindungen sofort neu zu starten.
- **graceful**
  **-k graceful**
  Diese beiden Befehle sorgen für einen unterbrechungsfreien Neustart (graceful restart) des Servers, der keine offenen Verbindungen fallen lässt.
- **graceful-stop**
  **-k graceful-stop**
  Mithilfe eines dieser Befehle können Sie den Server sauber beenden (graceful shutdown); er wird erst beendet, nachdem alle Worker-Prozesse oder -Threads ihre aktuellen Aufgaben abgeschlossen haben.
- **fullstatus**
  Wenn in Ihrem Webserver das Modul mod_status aktiv ist und der Textbrowser Lynx auf Ihrem System zur Verfügung steht, gibt dieser Befehl eine Statusmeldung des laufenden Servers inklusive aller zurzeit bedienten Client-Verbindungen aus.
- **status**
  Auch dieser Befehl funktioniert nur, wenn mod_status eingeschaltet und Lynx verfügbar ist. Er gibt eine einfache Statusmeldung ohne Verbindungsinformationen aus.

- `configtest`
  Dieser Befehl entspricht der Option `-t` des Programms `httpd`: Er testet die syntaktische Richtigkeit der aktuellen Konfigurationsdatei.

Neben den hier beschriebenen Befehlen können Sie auch für `apachectl` alle oben beschriebenen `httpd`-Optionen verwenden; diese werden entsprechend weitergereicht.

#### Den Start automatisieren

Jedes moderne UNIX-System verfügt über eine Möglichkeit, beim Booten beliebige Programme – insbesondere Daemons wie den Apache-Webserver – zu starten. Ein kleines Problem besteht nur darin, dass dieses Verfahren in den verschiedenen Systemvarianten unterschiedlich realisiert wurde. Historisch betrachtet lässt sich der Unterschied auf die beiden UNIX-Grundströmungen System V und BSD zurückführen, inzwischen zieht er sich aber – und dann auch noch mit einigen Variationen bezüglich der Verzeichniswahl – quer durch die Systeme und Distributionen (zumal es immer schwieriger wird, zu unterscheiden, welche aktuellen Systeme von System V abstammen und welche von der BSD).

Hier wird zunächst jedes der beiden grundsätzlichen Verfahren kurz vorgestellt; anschließend erhalten Sie Informationen darüber, wie sich der Server unter einem entsprechend beschaffenen System automatisch starten lässt.

- **System V Init**
  Diese Boot-Methode wird von immer mehr Betriebssystemen der UNIX-Familie verwendet, unter anderem auch von Linux. Systeme, die System V Init einsetzen, arbeiten mit unterschiedlichen Runlevels. Ein Runlevel ist ein Systemzustand, in dem jeweils nur bestimmte Prozesse laufen. Beim Wechsel des Runlevels über den Befehl `init LEVELNR`. werden bestimmte Skripte aufgerufen, die manche Programme starten und andere beenden. Einige Runlevel haben eine spezielle Bedeutung:
  - 0: heruntergefahrener Zustand
  - S (manchmal auch 1): Single-User-Modus (für Wartungsarbeiten)
  - 1 (bei vielen Systemen): Multi-User-Modus ohne Netzwerk
  - 2: Multi-User-Modus mit Netzwerk; nur Konsole
  - 3: Multi-User-Modus mit Netzwerk und GUI (klassisch)
  - 5: Multi-User-Modus mit Netzwerk und GUI (Linux)
  - 6: Systemneustart (Reboot)

  Betriebssysteme dieser Bauart besitzen für jedes Runlevel ein spezielles Init-Verzeichnis. Diese Verzeichnisse heißen */etc/rcLEVELNR.d*, also etwa */etc/*

*rc1.d* für Runlevel 1 oder */etc/rc5.d* für Runlevel 5. Die Shell-Skripte in diesen Verzeichnissen werden bei Erreichen des entsprechenden Levels automatisch ausgeführt, und zwar in alphabetischer Reihenfolge. Deshalb verwendet die übliche Konvention Namen, die mit *K* beginnen, für Kill-Skripte (die einen Prozess beenden) und solche mit *S* für Startskripte. Darüber hinaus bauen viele Daemons aufeinander auf. Deshalb wird hinter dem Anfangsbuchstaben eine zweistellige Zahl verwendet, die für eine bestimmte Reihenfolge sorgt.

In aller Regel sind die Einträge in diesen Verzeichnissen lediglich Symlinks auf Skripte, die sich eigentlich in einem anderen Verzeichnis befinden; meist in */etc/init.d* oder */sbin/init.d*. Für den Start und das Beenden des jeweiligen Prozesses wird normalerweise dasselbe Skript verwendet: Ein `S`-Symlink ruft es automatisch mit der Kommandozeilenoption `start` auf, ein `K` mit `stop`. Wie Sie bereits erfahren haben, erfüllt das Shell-Skript `apachectl` demzufolge die Voraussetzungen für diese Aufgabe.

Aus diesem Grund müssen Sie lediglich für Ihr Standard-Runlevel (3 oder 5) einen `S`-Symlink auf dieses Skript erzeugen. Für die Runlevel 0 und 6 (Herunterfahren beziehungsweise Neustart) können Sie entsprechend einen `K`-Link anlegen. Da von Apache in der Regel keine anderen Dienste abhängen, können Sie ihn recht spät starten (wählen Sie einen Symlink-Namen wie `S95apache`) und ziemlich früh beenden (`K15apache` dürfte in Ordnung gehen).

Begeben Sie sich also in das jeweilige Runlevel-Init-Verzeichnis, und erstellen Sie die nötigen symbolischen Links. Angenommen, Sie haben Apache mit dem `GNU`-Layout installiert und verwenden ein Linux-System mit dem Standard-Runlevel 5. Dann müssen Sie den folgenden Befehl für den Startskript-Link eingeben:

```
# ln -s /usr/local/bin/apachectl /etc/rc5.d/S95apache
```

Als Nächstes werden die Stopp-Links für die Runlevel 0 und 6 erzeugt. Wenn Sie das Layout `Apache` verwenden, sehen die beiden Befehle so aus:

```
# ln -s /usr/local/apache2/bin/apachectl /etc/rc0.d/K15apache
# ln -s /usr/local/apache2/bin/apachectl /etc/rc6.d/K15apache
```

In vielen Linux-Distributionen, beispielsweise openSUSE und Fedora Core beziehungsweise Red Hat, können Sie das Kommando `chkconfig` verwenden, um Apache 2 für den automatischen Start in den Runlevels 3 und 5 einzurichten. Erstellen Sie dazu als Erstes im Verzeichnis */etc/init.d* einen symbolischen Link auf `apachectl`. Beispiel:

```
# ln -s /usr/local/apache2/bin/apachectl /etc/init.d/apache2
```

Nun lässt sich der Symlink mithilfe von `chkconfig -a` aktivieren:

```
# chkconfig -a apache2
apache2 0:off 1:off 2:off 3:on 4:off 5:on 6:off
```

- **BSD-Startskript**
  BSD-basierte UNIX-Systeme verwenden im Gegensatz zu der bereits beschriebenen System-V-Methode einige zentrale Startskripte. Sie befinden sich in Verzeichnissen wie */etc* oder */etc/rc.d* und heißen *rc.boot*, *rc.local* usw. Interessant ist in diesem Zusammenhang das Skript *rc.local*, das Sie nach Belieben um weitere Startbefehle erweitern können.

  Unter einem solchen Betriebssystem brauchen Sie *rc.local* also nur mit einem Texteditor zu öffnen und können dann den Aufruf von `apachectl` mit dem Parameter `start` hinzufügen. In diesem Skript wird normalerweise mit einer Fallentscheidung nach dem Schema `if [-x PFAD]` überprüft, ob das aufzurufende Programm oder Skript überhaupt existiert. An diese Konvention sollten Sie sich halten. Falls Sie also das `GNU`-Layout verwenden, können Sie folgende Zeilen an *rc.local* anfügen:

```
# httpd starten
if [ -x /usr/local/bin/apachectl ]; then
   echo "Starting Apache httpd..."
   /usr/local/bin/apachectl start
fi
```

Es gibt hier keine Lösung, die »besser« ist – beide funktionieren in der Praxis, und bei jedem System ist es eine von beiden.

Noch leichter haben Sie es natürlich, wenn Sie den Webserver über ein Paket Ihres Systemdistributors installiert haben: Diese Installer-Befehle kümmern sich normalerweise selbst darum, Apache für den automatischen Start zu konfigurieren. Diese Einstellung können Sie bei manchen Systemen auch auf der grafischen Benutzeroberfläche vornehmen; zu diesem Zweck bieten einige Distributionen spezielle Verwaltungsprogramme an. Hier nur ein paar Beispiele:

- **openSUSE**
  Ein gutes Argument für den Einsatz von openSUSE (früher SUSE Linux) gegenüber anderen Linux-Distributionen ist schon seit Langem das komfortable Installations- und Konfigurationsprogramm `yast` (Yet Another Setup Tool). Die aktuelle Version openSUSE 11.0 bietet eine Rubrik für den automatischen Start und die komfortable grafische Konfiguration von Apache. Starten Sie dazu über ein Terminalfenster oder aus dem KDE-Menü das Programm `yast`. Hier finden Sie in der Rubrik Netzwerkdienste das entsprechende Icon HTTP-Server.

  Dieses Tool dient darüber hinaus dem Aktivieren oder Deaktivieren von Apache-Modulen. Dies muss bei der eingebauten Apache-Version von SUSE an

dieser Stelle geschehen; entsprechende Änderungen in den Konfigurationsdateien werden durch SUSEConfig nachträglich wieder überschrieben. Außerdem können Sie an dieser Stelle auch Konfigurationsdirektiven einstellen, wie sie ab Kapitel 6, »Grundkonfiguration«, ausführlich erläutert werden.

Bereits in früheren Versionen von SUSE Linux stand eine andere Möglichkeit zur Verfügung, die noch immer eingesetzt werden kann – vornehmlich, wenn Sie den Server selbst kompiliert haben, denn dann steht die `yast`-Methode nicht zur Verfügung: Über den *Runlevel-Editor* können Sie einstellen, welche Programme beziehungsweise Daemons in welchem Runlevel ausgeführt werden sollen. Starten Sie dazu wiederum `yast`, und wählen Sie Runlevel-Editor in der Kategorie System. Sie sollten Apache 2 in den Runlevels 3 und 5 aktivieren.

- **Mac OS X**
  Unter Mac OS X wird Apache normalerweise automatisch mit dem System installiert. Wählen Sie Systemeinstellungen • Netzwerk. Hier können Sie Apache unter Sharing • Web Sharing für den automatischen Start aktivieren beziehungsweise deaktivieren.

### 5.1.2 Apache unter Windows steuern

Unter Windows funktioniert die Steuerung des Webservers aufgrund der Plattformunterschiede ein wenig anders als auf UNIX-Systemen. Im Prinzip lassen sich drei verschiedene Konfigurationsarten unterscheiden, deren Funktionalität sich allerdings zum Teil überschneidet:

- Sie können das ausführbare Programm *httpd.exe* mit diversen Optionen aufrufen, die zum Teil dem Programm `httpd` und dem Skript `apachectl` unter UNIX entsprechen.
- Sie können Apache 2 als Dienst installieren. Ein Dienst ist die Windows-Entsprechung eines Daemons. Wenn Apache als Dienst ausgeführt wird, können Sie ihn über das Applet Dienste der Systemsteuerung beziehungsweise der Microsoft Management Console steuern.
- Im Verzeichnis `bin` des Servers finden Sie ein kleines Steuerprogramm namens `ApacheMonitor.exe`. Dieses Programm installiert ein kleines Steuer-Icon in den SysTray und stellt diverse Optionen zur Verfügung.

#### Optionen des Programms httpd.exe

Unter Windows heißt das ausführbare Webserver-Programm *httpd.exe* (bis Version 2.0 hieß es *Apache.exe*). Es befindet sich im Verzeichnis *bin* in Ihrer `ServerRoot` (zum Beispiel in *C:\Program Files\Apache Software Foundation*

*Apache2.2\bin*). Unter Windows 7 und Vista müssen Sie die Eingabeaufforderung als Administrator starten, um das Konsolenprogramm auszuführen. Klicken Sie dazu START • ALLE PROGRAMME • ZUBEHÖR • EINGABEAUFFORDERUNG mit der rechten Maustaste an, wählen Sie ALS ADMINISTRATOR AUSFÜHREN, und bestätigen Sie den Sicherheitsdialog. Im Folgenden sind alle Optionen dieses Programms aufgeführt. Sofern sie den bereits besprochenen Parametern von `httpd` beziehungsweise `apachectl` unter UNIX entsprechen, fällt die Beschreibung recht kurz aus:

- **`-D Name`**
  Definiert einen Namen für die Konfigurationsdirektive `<IfDefine Name>`.
- **`-d Verzeichnis`**
  Gibt eine alternative `ServerRoot` an.
- **`-f Datei`**
  Ermöglicht die Angabe einer alternativen Konfigurationsdatei.
- **`-C "Direktive"`**
  Führt vor dem Einlesen der Konfigurationsdatei die angegebene Konfigurationsdirektive aus.
- **`-c "Direktive"`**
  Führt die angegebene Direktive nach dem Verarbeiten der Konfigurationsdatei aus, ermöglicht also das nachträgliche Überschreiben vorhandener Direktiven.
- **`-k start`**
  Diese Option startet Apache. Wenn er als Dienst installiert ist, wird dieser gestartet; andernfalls startet der Server als Konsolenprogramm.
- **`-k runservice`**
  Mit diesem Befehl wird explizit ein bereits installierter Apache-Dienst gestartet.
- **`-k restart`**
  Diese Option führt einen unterbrechungsfreien Neustart des Servers durch. Wenn er als Konsolenprogramm läuft, müssen Sie ein weiteres Eingabeaufforderungsfenster öffnen, um den Befehl einzugeben.
- **`-k stop`**
  **`-k shutdown`**
  Jeder dieser beiden Befehle beendet Apache. Läuft er als Konsolenprogramm, dann können Sie den Befehl von einem anderen Fenster aus eingeben.
- **`-k install`**
  Installiert Apache als Dienst.
- **`-k config`**
  Diese Option kann verwendet werden, um zusammen mit anderen Befehlen die Konfiguration des Apache-Dienstes zu ändern.

- `-k uninstall`
  Deinstalliert den Apache-Dienst.
- `-n Name`
  Mithilfe dieser Option können Sie einen alternativen Namen angeben, unter dem der Apache-Dienst installiert werden soll (der Standardname ist `Apache2.2`). Hat er bereits einen anderen Namen, dann dient dieselbe Option dazu, den Dienst später für die Deinstallation oder für Konfigurationsänderungen anzusprechen.
- `-w`
  Wenn diese Option angegeben wird, bleibt das Konsolenfenster bei einem Fehler geöffnet. Dies ist vor allem dann nützlich, wenn Sie Apache aus einer Batchdatei heraus starten, die per Doppelklick oder automatisch ausgeführt wird.
- `-e level`
  Dringlichkeitsstufe, ab der Fehler beim Start des Servers angezeigt werden sollen
- `-E Datei`
  Schreibt Startfehlermeldungen in die angegebene Datei.
- `-v`
  Ausgabe von Versionsinformationen
- `-V`
  Ausgabe der Versionsinformationen und der Einstellungen, mit denen der Server kompiliert wurde
- `-h`
  Ausgabe einer Liste aller Kommandozeilenoptionen
- `-l`
  Ausgabe einer Liste der einkompilierten Module
- `-L`
  Auflisten der verfügbaren Konfigurationsdirektiven
- `-X`
  Apache im Debug-Modus ausführen
- `-t -D DUMP_VHOSTS`
  Ausgabe der verarbeiteten Einstellungen für virtuelle Hosts
- `-S`
  Kurzfassung von `-t -D DUMP_VHOSTS`
- `-t`
  Syntax der Konfigurationsdatei überprüfen

### Apache als Dienst betreiben

Bereits seit der alten Version 1.3 lässt sich Apache unter NT-basierten Windows-Betriebssystemen als Dienst installieren. Der Vorteil ist, dass der Server in dieser Konstellation unabhängig von einem angemeldeten Benutzer ausgeführt wird. Auch die Performance ist bei einem Dienst besser, als wenn Apache 2 als Konsolenanwendung ausgeführt wird.

Wenn Sie das in Kapitel 4, »Apache kompilieren und installieren«, beschriebene Windows-MSI-Paket installieren, können Sie die automatische Einrichtung von Apache als Dienst wählen; sie ist sogar standardmäßig vorgegeben. Haben Sie den Server dagegen selbst kompiliert oder die Installation als Dienst abgelehnt, müssen Sie den folgenden Befehl ausführen, um den Dienst nachträglich zu installieren:

```
> apache -k install
```

Der Dienst wird dadurch ein für allemal in die Liste der Systemdienste aufgenommen. Der Standardname des Dienstes ist `Apache2`. Um einen anderen Namen festzulegen, können Sie den Befehl mit der zusätzlichen Option `-n Name` verwenden, beispielsweise so:

```
> apache -k install -n Winnetou
```

Wenn Sie ihn später wieder entfernen möchten, können Sie dies mit diesem Befehl erledigen:

```
> apache -k uninstall
```

Falls Sie einen alternativen Namen festgelegt haben, müssen Sie diesen auch bei der Deinstallation angeben:

```
> apache -k uninstall -n Winnetou
```

Einen installierten Apache-Dienst können Sie auf der Kommandozeile auch mithilfe der Befehle `net start` und `net stop` steuern. Dazu müssen Sie in jedem Fall den Dienstnamen angeben, nicht den Namen des ausführbaren Programms. Beispiele:

```
> net start Apache2.2
> net stop Winnetou
```

Zur Verwaltung von Diensten wie dem Apache-Dienst bietet Windows das Verwaltungs-Applet Dienste. In Windows 7 und Vista befindet es sich in Start • Systemsteuerung • Verwaltung • Dienste; in Windows XP finden Sie es unter Start • Verwaltung • Dienste, unter Windows 2000 in der Systemsteuerung unter Verwaltung und bei Windows NT 4.0 unter Start • Einstellungen • Systemsteuerung • Dienste. Abbildung 5.2 zeigt das Applet unter Windows Vista.

Änderungen an der Konfiguration des Dienstes können Sie über die Schaltfläche Eigenschaften vornehmen; es handelt sich um die vierte Schaltfläche von links mit der kleinen Registerkarte, die Sie in der Symbolleiste in Abbildung 5.2 sehen können; alternativ ist auch ein Doppelklick auf den Dienst möglich. Dieselbe Symbolleiste enthält im Übrigen Schaltflächen zum schnellen Starten, Beenden und Neustarten des gerade ausgewählten Dienstes.

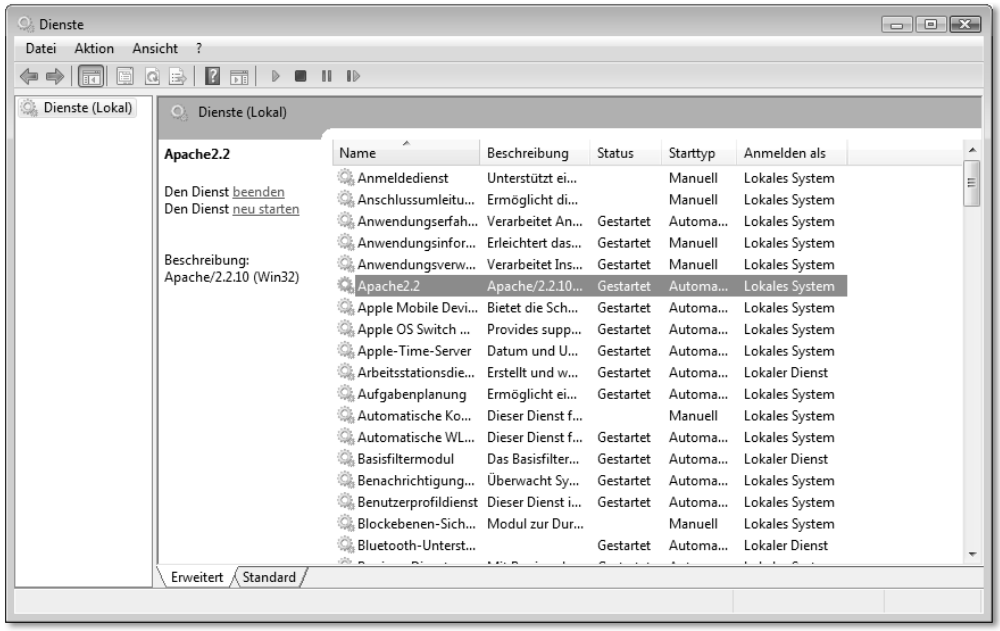



**Abbildung 5.2** Das Verwaltungs-Applet »Dienste« mit ausgewähltem Apache-Dienst unter Windows Vista

In Abbildung 5.3 werden nebeneinander die Inhalte der beiden ersten Registerkarten des Eigenschaften-Dialogs für den Apache-Dienst gezeigt: Allgemein und Anmelden. Die vier Registerkarten des Dialogs bieten die folgenden Einstellungsmöglichkeiten:

- Allgemein
  Hier werden die Grundeinstellungen für den Dienst vorgenommen, insbesondere ermöglicht der Starttyp die Einstellung, ob er automatisch gestartet werden soll. Im Einzelnen finden Sie hier folgende Felder und Schaltflächen:
  - Dienstname – Offizielle Bezeichnung des Dienstes; nicht änderbar
  - Anzeigename – Name des Dienstes, wie er in der Liste angezeigt wird; in Windows 7 und Vista ebenfalls nicht änderbar
  - Beschreibung – Nähere Information über den Dienst

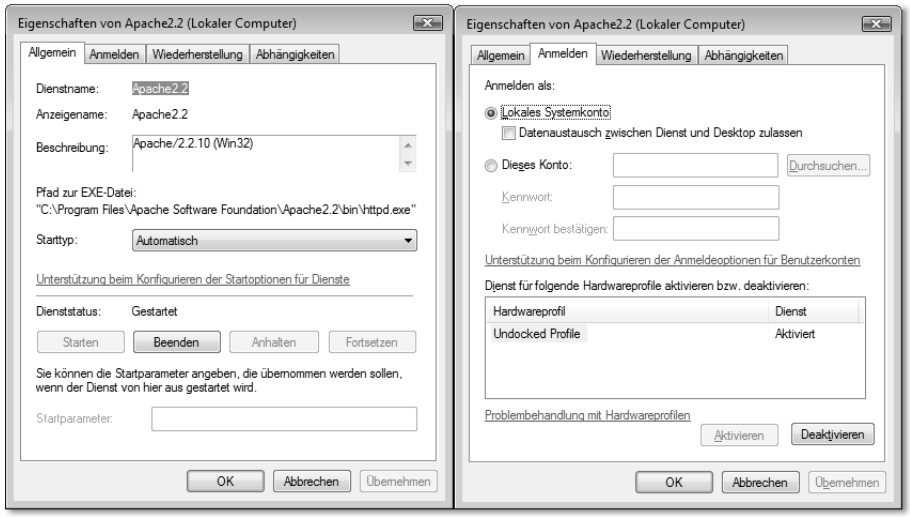



**Abbildung 5.3** Die beiden ersten (und wichtigsten) Registerkarten des Eigenschaften-Dialogs für den Apache-Dienst unter Windows XP

- Pfad zur EXE-Datei – Das ausführbare Programm, das dieser Dienst ausführt, mitsamt Kommandozeilenparametern
- Starttyp – Legt fest, wie dieser Dienst gestartet werden soll: Automatisch startet ihn automatisch, Manuell nur auf ausdrückliche Anforderung und Deaktiviert gar nicht.
- Dienststatus – Zeigt an, ob der Dienst zurzeit läuft, beendet wurde oder deaktiviert ist.
- Starten – Hier können Sie den Apache-Dienst starten, wenn Sie die Startart auf Manuell gesetzt oder ihn zuvor beendet haben.
- Beenden – Diese Schaltfläche beendet den Apache-Dienst.
- Anhalten – Einige Windows-Server-Dienste – meist von Microsoft selbst – können über diese Schaltfläche in eine Art »Administrationsmodus« versetzt werden: Der Dienst ist dann nur noch für Benutzer mit Administratorrechten erreichbar. Bei Apache wurde dieses Feature leider noch nicht eingebaut; hier müssen Sie sich mit dem oben erwähnten Debug-Modus oder mit einer Umkonfiguration der Authentifizierung (siehe Kapitel 9, »Authentifizierung«) behelfen. Daher ist die Schaltfläche beim Apache-Dienst deaktiviert.
- Fortsetzen – Das Gegenstück zur Schaltfläche Anhalten: Ein Dienst soll aus dem eingeschränkten Wartungsmodus wieder in den Normalbetrieb zurückversetzt werden. Für Apache ebenfalls nicht verfügbar.

- Anmelden
  Auf dieser Registerkarte wird festgelegt, unter welcher Benutzerkennung der Dienst `Apache2.2` ausgeführt wird. Die Standardeinstellung ist Lokales Systemkonto. Dies ist in den meisten Fällen in Ordnung, mit einer wichtigen Ausnahme:

  Aus Sicherheitsgründen sollte das Benutzerkonto des Apache-Dienstes niemals Berechtigungen für Netzwerkanwendungen erhalten. Falls der »Benutzer« Lokales Systemkonto diese für eine andere Anwendung benötigt, sollten Sie Apache unbedingt unter einer anderen Benutzerkennung betreiben. Dazu müssen Sie einen neuen Benutzer einrichten; dieser sollte keine Administratorrechte haben, sondern ein normaler Benutzer sein. Er benötigt aber die zusätzlichen Rechte »Anmelden als Dienst« und »Als Teil des Betriebssystems handeln«.

  Unter Windows 7, Vista und XP legen Sie einen neuen Benutzer über das Applet Benutzerkonten in der Systemsteuerung an, in Windows 2000 unter Verwaltung. Die erforderlichen Rechte können Sie über Gruppenrichtlinien oder über die lokalen Sicherheitseinstellungen in der Microsoft Management Console vergeben. Bei Windows NT 4.0 erledigen Sie beides mithilfe des Programms unter Start • Programme • Verwaltung (Allgemein) • Benutzer-Manager; die Rechte finden Sie dort unter Richtlinien • Benutzerrechte.

  Wählen Sie nach dem Erstellen des neuen Benutzerkontos die Option Dieser Benutzer. Geben Sie den Namen dieses Benutzers und zweimal sein Passwort ein.

  Unter der Einstellung des Benutzerkontos können Sie noch festlegen, in welchen Hardwareprofilen Apache aktiviert werden soll. Da Sie beim Booten ein bestimmtes Hardwareprofil auswählen können, bietet dieses Verfahren eine einfache Möglichkeit, Betriebssystemkonfigurationen mit und ohne aktivierten Apache-Webserver einzurichten. Eingerichtet werden Hardwareprofile übrigens in den Systemeigenschaften (Systemsteuerung • System oder rechte Maustaste auf das Symbol Computer (bis Windows XP Arbeitsplatz) und Eigenschaften auswählen).

- Wiederherstellen
  Auf dieser Registerkarte können Sie detailliert festlegen, was geschehen soll, wenn Apache ausfällt, das heißt beim Systemstart nicht ausgeführt werden kann oder unerwartet beendet wird. Dies ist natürlich besonders dann wichtig, wenn Sie den betreffenden Rechner hauptsächlich als Webserver für ein Intranet oder das Internet verwenden.

  Unter den Kategorien Erster Fehler, Zweiter Fehler und Weitere Fehler können Sie je eine der folgenden Optionen auswählen:

- Keine Aktion durchführen – Es soll nichts Besonderes veranlasst werden. Wenn Sie Apache nur zu Test- oder Entwicklungszwecken installiert haben, ist dies die passende Auswahl.
- Dienst neu starten – Es soll versucht werden, Apache neu zu starten. Für den ersten Fehlschlag bietet sich diese Möglichkeit an.
- Ein Programm ausführen – Wenn Sie diese Option auswählen, können Sie weiter unten ein Programm bestimmen, das ausgeführt werden soll. Zusätzlich besteht die Möglichkeit, dem Programm die Fehlernummer als Kommandozeilenparameter in der Form `/fail=%1 %` zu übergeben. Ein entsprechend präpariertes eigenes Programm kann diesen Parameter auswerten.
- Computer neu starten
  Diese extreme Option ergibt eigentlich nur dann einen Sinn, wenn Sie einen Windows-Rechner in Ihrem Netzwerk ausschließlich als Webserver einsetzen. Unter Neustartoptionen können Sie in diesem Fall eine Meldung eintragen; diese wird allen (Windows-)Benutzern im Netzwerk angezeigt, die zur Zeit des Neustarts mit diesem Host verbunden sind.

Zusätzlich können Sie noch die Option Fehlerzähler nach … Tagen zurücksetzen wählen. Die zusätzlichen Einstellungen Dienst nach … Minuten neu starten und Neustartoptionen sind nur verfügbar, wenn Sie für eine der Fehlernummern den Neustart des Dienstes beziehungsweise des Systems ausgewählt haben.

- Abhängigkeiten
  Auf dieser Registerkarte werden die Dienste angezeigt, von deren Funktionieren der aktuelle Dienst abhängt und umgekehrt. Von Apache hängen in der Regel keine anderen Dienste ab. Er selbst benötigt natürlich funktionierendes TCP/IP-Networking, was bei neueren Windows-Versionen auch als Voraussetzung angezeigt wird.

Beachten Sie, dass dieser Dialog unter Windows NT 4.0 nur aus einer einzigen Seite bestand. Auf dieser konnten Sie lediglich den Starttyp und das Benutzerkonto einstellen und natürlich den Dienst beenden und neu starten.

Windows 95, 98 und ME boten von Hause aus gar keine Dienste an. Aber obwohl der Einsatz von Apache unter diesen Systemen ohnehin nicht zu empfehlen ist, wurde der als Nächstes beschriebene Apache-Monitor so geschrieben, dass er auch mit einer Art »Dienstemulation« zusammenarbeitet, die auf diesen Systemen läuft.

### Der Apache-Monitor

Wenn Sie Apache über den MSI-Installer als Dienst installiert haben, wurde der Apache-Monitor automatisch eingerichtet und wird bei jedem Windows-Boot-

Vorgang mitgestartet. Bei anderen Installationsarten können Sie ihn selbst starten oder ebenfalls für den automatischen Start konfigurieren.

Das Programm trägt den Namen *ApacheMonitor.exe* und befindet sich im Verzeichnis *bin* des Apache-Installationsordners. Wenn Sie es per Doppelklick oder über die Konsole starten, bleibt es nur für die aktuelle Systemsitzung aktiv. Um es für den automatischen Start einzurichten, gibt es zwei Möglichkeiten:

- Sie können eine Verknüpfung zu dem Programm im Ordner *Autostart* des Startmenüs anlegen. Dazu genügt es, das Icon des Programms mit der rechten Maustaste in Start • Alle Programme • Autostart zu ziehen und beim Loslassen die Option Verknüpfung hier erstellen aus dem Kontextmenü zu wählen. Diese Variante verwendet übrigens auch der MSI-Installer automatisch.
- Die Alternative besteht darin, einen Eintrag für den Start des Monitors in der Registry zu erstellen. Wählen Sie dazu Start • Ausführen, und geben Sie `regedit` ein. Im linken Fensterbereich müssen Sie sich durch die Hierarchie zum Schlüssel *HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run* vorarbeiten. Wenn Sie den Ordner *Run* angeklickt haben, können Sie mit der rechten Maustaste in den rechten Fensterbereich klicken und Neu • Zeichenfolge aus dem Kontextmenü wählen. Geben Sie einen beliebigen Namen ein (in diesem Fall natürlich am besten ApacheMonitor). Doppelklicken Sie zu guter Letzt auf das Icon der neuen Zeichenfolge, und geben Sie als Wert den Pfad des Programms *ApacheMonitor.exe* ein.

Wenn der Apache-Monitor ausgeführt wird, macht er sich als kleines Apache-Icon im SysTray bemerkbar – dies ist der Bereich links in der Taskleiste, neben der Uhrzeit. Wenn Sie das Icon mit der linken Maustaste anklicken, stehen Ihnen die Optionen Start (den Apache-Dienst starten), Stop (zum Beenden des Dienstes) und Restart (Neustart, zum Beispiel nach einer Konfigurationsänderung) zur Verfügung. Ein Klick mit der rechten Maustaste ermöglicht dagegen das Öffnen des eigentlichen Monitorfensters, das in Abbildung 5.4 zu sehen ist.

Auch hier sind zunächst einmal wieder Schaltflächen zum Starten, Beenden und Neustarten zu erkennen. Darüber hinaus können Sie über Connect eine Verbindung zu einem anderen Windows-Rechner in Ihrem LAN herstellen, der Apache 2 ausführt, und dessen Apache-Dienst fernsteuern. Dazu benötigen Sie allerdings Administratorrechte, die auch für den entfernten Host gelten – entweder über ein entsprechendes Domänen-Benutzerkonto oder dadurch, dass Ihr aktueller Benutzername mit demselben Passwort und identischen Rechten auf dem anderen Computer existiert.

Die Schaltfläche Services schließlich öffnet unter Windows 2000 bis Windows 7 das bereits besprochene Betriebssystem-Applet Dienste; unter Windows NT 4.0 funktioniert dies leider nicht.

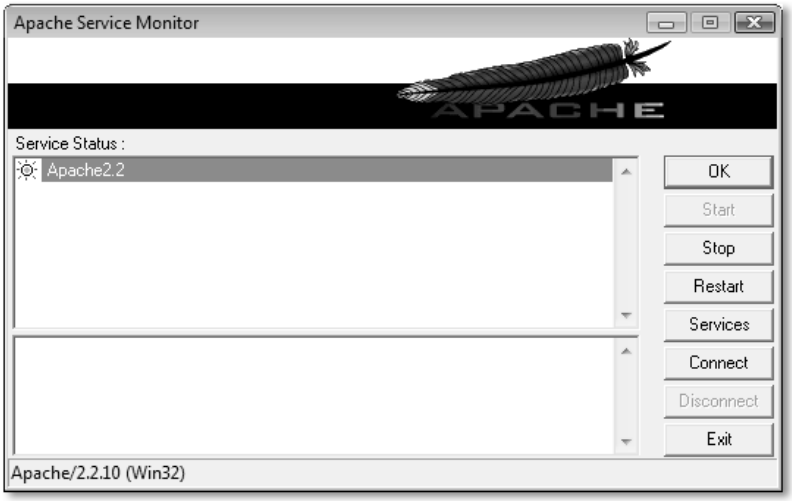



**Abbildung 5.4** Das Hauptfenster des Apache-Monitors

### 5.1.3 Apache-Hilfsprogramme

Das Programm `httpd` (UNIX) beziehungsweise `apache.exe` (Windows) ist nicht das einzige ausführbare Programm, das mit dem Webserver installiert wird. Es gibt zusätzlich einige nützliche Kommandozeilen-Tools. Die meisten von ihnen sind vor allem für Aufgaben nützlich, auf die erst in späteren Kapiteln näher eingegangen wird. Aus diesem Grund finden Sie hier nur eine kurze Übersicht über die Programme mit dem Hinweis, in welchem Kapitel sie jeweils behandelt werden.

Im Einzelnen stehen neben dem ausführbaren Server-Programm und dem Skript `apachectl` folgende Programme zur Verfügung:

- **`ab`** – das Apache-Benchmark-Programm (siehe Kapitel 12, »Skalierung und Performance-Tuning«)
- **`apxs`** – Hilfsprogramm zum nachträglichen Kompilieren von Modulen. Dieses Tool wurde bereits in Kapitel 4, »Apache kompilieren und installieren«, angesprochen.
- **`dbmmanage`** – Perl-Skript zur Verwaltung von Authentifizierungsdaten im DBM-Format (siehe Kapitel 9, »Authentifizierung«)

- `htcacheclean` – ein neues Tool zur Bereinigung des festplattenbasierten Web-Caches. Dieses Programm wird in Kapitel 13, »Proxy- und Cache-Funktionen«, beschrieben.
- `htdbm` – ein binäres Verwaltungsprogramm für DBM-Authentifizierungsdaten (siehe Kapitel 9, »Authentifizierung«)
- `htdigest` – Verwaltungsprogramm für Digest-Authentifizierungsdaten (Näheres in Kapitel 9, »Authentifizierung«)
- `htpasswd` – Verwaltungsprogramm für Basic-Authentifizierungsdaten. Auch dieses Hilfsprogramm wird in Kapitel 9, »Authentifizierung«, behandelt.
- `httxt2dbm` – einfaches Programm zur Umwandlung von Text- in DBM-Dateien als Nachschlagetabellen für `Rewrite`-Maps (URL-Umwandlung); siehe Kapitel 8, »Weiterleitungen und Indizes«.
- `logresolve` – Tool zur Ermittlung von Hostnamen zu den IP-Adressen in Log-Dateien (siehe Kapitel 11, »Logging«)
- `rotatelogs` – Programm zum automatischen, regelmäßigen Wechsel von Log-Dateien. Auch dieses Programm wird in Kapitel 11, »Logging«, näher betrachtet.
- `suexec` – CGI-Skripte unter anderer User-ID ausführen (siehe Kapitel 18, »Sicherheit«).
- `log_server_status` – Statusinformationen in eine Zeile packen und in eine Log-Datei schreiben (Näheres in Kapitel 11, »Logging«).
- `split-logfile` – Log-Dateien anhand bestimmter Regeln in mehrere Einzeldateien zerlegen (siehe ebenfalls Kapitel 11, »Logging«).
- `fcgistarter` – Daemon zum Starten von FastCGI-Skripten, die über `mod_proxy_fcgi` als Reverse Proxy ausgeliefert werden können (nur 2.4).

## 5.2 Apache testen

Nach Installation und Start sollten Sie überprüfen, ob Apache ordnungsgemäß funktioniert. In diesem kurzen Abschnitt werden zwei Methoden dafür beschrieben: das Überprüfen der mitgelieferten Startseite und das Einrichten einer eigenen Minimalkonfiguration.

### 5.2.1 Die automatische Startseite

Nachdem Sie den Webserver mit einer der hier beschriebenen Methoden gestartet haben, sollte er eigentlich funktionieren. Ob dies tatsächlich der Fall ist, kön-

nen Sie mit einem Webbrowser testen: Das vorkonfigurierte Website-Verzeichnis `htdocs` enthält zu diesem Zweck eine Testseite, die im Browser angezeigt werden sollte, wenn Sie die Wurzeladresse Ihres Servers eingeben.

Öffnen Sie also einen Browser, und geben Sie als URL *http://localhost* ein. Falls der Standardname `localhost` auf Ihrem System nicht unterstützt wird (zum Beispiel in älteren Windows-Versionen), müssen Sie stattdessen *http://127.0.0.1* eingeben. Abbildung 5.5 zeigt, wie die Seite bei Apache 2.2 aussehen sollte, wenn alles in Ordnung ist.

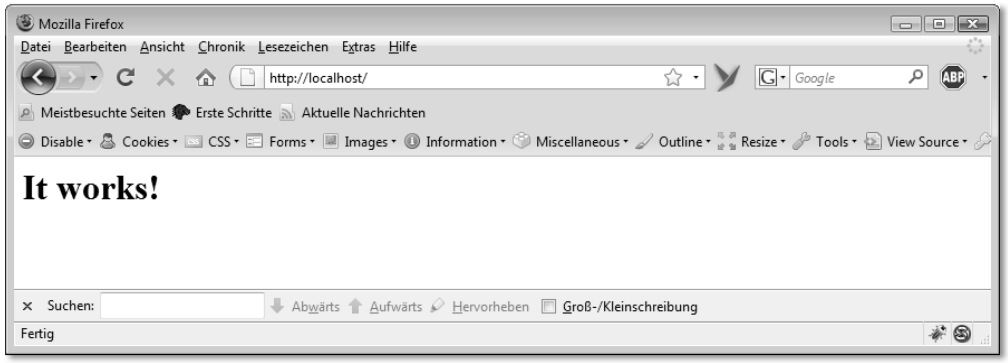



**Abbildung 5.5** Die Testseite des Apache-Webservers 2.2 nach erfolgreichem Start

### 5.2.2 Die erste Website

Es genügt natürlich nicht, den Apache-Webserver einfach so zu starten. Es sind zahlreiche Anpassungen der Konfigurationsdatei *httpd.conf* erforderlich, damit er genau nach Wunsch arbeitet. Da große Teile des Inhalts von *httpd.conf* davon abhängen, welche Module im Apache-Server installiert sind, werden in diesem Unterabschnitt nur einige wenige Konfigurationsdirektiven angesprochen, die das Ausliefern statischer Dokumente ermöglichen.

Wenn Sie eine neue Apache-Installation zum ersten Mal verwenden (und vor allem, wenn Sie Apache überhaupt das erste Mal einsetzen), sollten Sie nicht einfach ohne Weiteres der Öffentlichkeit eine Website präsentieren. Es gibt Unmengen von Konfigurationsanweisungen und Anpassungsmöglichkeiten. Aus diesem Grund empfiehlt es sich, einen neu installierten Server vor der Inbetriebnahme mit einer Test-Website zu überprüfen.

In diesem kurzen Abschnitt wird eine kleine Website mit einer minimalen Konfigurationsdatei erstellt. Einzelheiten zu den zahlreichen Optionen für die Datei *httpd.conf* finden Sie an entsprechenden Stellen im Buch; hier geht es erst einmal darum, überhaupt eine Website zu veröffentlichen.

Erstellen Sie als Erstes ein Verzeichnis, das den Stamm Ihrer Website bilden soll. Erstellen Sie darin eine HTML-Datei namens *index.html* mit beliebigem Inhalt (oder kopieren Sie den Inhalt des Verzeichnisses *testsite* von der DVD zum Buch hinein). Als Verzeichnis für den Test können Sie beispielsweise das offizielle *htdocs*-Verzeichnis Ihres Apache-Servers oder ein neu erstelltes Unterverzeichnis davon benutzen.

Benennen Sie die vorgefertigte Konfigurationsdatei *httpd.conf* um. Erstellen Sie für Apache 2.4 die folgende neue Minimaldatei (Erläuterung der austauschbaren Elemente siehe unten):

```
ServerName www.mynet.de
Listen 80
ServerRoot /usr/local/apache2

# Falls das MPM dynamisch geladen wird (Beispiel prefork):
LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
# Dieser Block wird nur bei
# DSO-basierter Modulinstallation benötigt
LoadModule dir_module modules/mod_dir.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule authz_host_module modules/mod_authz_host.so

# An Ihr Installationslayout anpassen
DocumentRoot /usr/local/apache2/htdocs

# Absicherung durch Absperren des Wurzelverzeichnisses
<Directory />
  Options None
  Require all denied
</Directory>

# Freigeben der DocumentRoot
<Directory /usr/local/apache2/htdocs>
  DirectoryIndex index.html
  Options All
  Require all granted
</Directory>
```

Für Version 2.2 müssen Sie eine etwas andere Version verwenden:

```
ServerName www.mynet.de
Listen 80
ServerRoot /usr/local/apache2
```

```
# Dieser Block wird nur bei
# DSO-basierter Modulinstallation benötigt
LoadModule dir_module modules/mod_dir.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule authz_host_module modules/mod_authz_host.so

# An Ihr Installationslayout anpassen
DocumentRoot /usr/local/apache2/htdocs

# Absicherung durch Absperren des Wurzelverzeichnisses
<Directory />
  Options None
  AllowOverride None
  Order deny,allow
  Deny from all
</Directory>

# Freigeben der DocumentRoot
<Directory /usr/local/apache2/htdocs>
  DirectoryIndex index.html
  Options All
  AllowOverride All
  Order allow,deny
  Allow from all
</Directory>
```

Die Verzeichnisangaben für die Direktiven `ServerRoot`, `DocumentRoot` und den `<Directory>`-Container müssen Sie natürlich den Gegebenheiten Ihrer Plattform und Ihrer Apache-Installation anpassen. Das obige Beispiel entspricht dem `Apache`-Layout unter UNIX.

Zu beachten ist in diesem Zusammenhang, dass auch unter Windows der Slash (/) und nicht der plattformtypische Backslash (\) als Pfadtrennzeichen benutzt werden muss. Angenommen, Apache ist auf Ihrem Windows-Rechner unter *C:\Program Files\Apache2.2* installiert, dann sieht die Datei folgendermaßen aus:

```
ServerName www.mynet.de
Listen 80
ServerRoot "C:/Program Files/Apache2.2"

LoadModule dir_module modules/mod_dir.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule authz_host_module modules/mod_authz_host.so

# An Ihr Installationsverzeichnis anpassen
DocumentRoot "C:/Program Files/Apache2.2/htdocs"
```

```
# Absicherung durch Absperren des Wurzelverzeichnisses
<Directory />
  Options None
  AllowOverride None
  Order deny,allow
  Deny from all
</Directory>

# Freigeben der DocumentRoot
<Directory "C:/Program Files/Apache2.2/htdocs">
  DirectoryIndex index.html
  Options All
  AllowOverride All
  Order allow,deny
  Allow from all
</Directory>
```

Sobald die finale Version von Apache 2.4 für Windows verfügbar ist und Sie diese einsetzen, müssen Sie wie oben im UNIX-Beispiel die Direktiven `Order`, `Allow` und `Deny` durch `Require all denied` beziehungsweise `Require all granted` ersetzen.

Wie Sie sehen, sind die drei `LoadModule`-Direktiven unter Windows nicht optional: Bei der Standardversion, die der in Kapitel 4, »Apache kompilieren und installieren«, vorgestellte MSI-Installer einrichtet, werden Module immer als DSOs geladen und sind niemals statisch einkompiliert.

Im Übrigen sollten Sie *www.mynet.de* für einen lokalen Test zu Ihrer *hosts*-Datei (UNIX: */etc/hosts*, Windows: *%Systemroot%\System32\drivers\etc\hosts*) hinzufügen. Andernfalls erhalten Sie beim Apache-Start die folgende Meldung:

```
Could not determine the Server's full qualified domain name.
Using 127.0.0.1 for Server Name.
```

Erstellen Sie in der Datei *hosts* einen Eintrag wie diesen, um den Server auf demselben Host zu testen:

```
127.0.0.1 www.mynet.de
```

Für einen Test über das Netzwerk müssen Sie statt *127.0.0.1* die IP-Adresse der entsprechenden Netzwerkschnittstelle verwenden. Bei einem Produktionsserver muss der Name gemäß den Erläuterungen in Kapitel 1, »IP-Netzwerke, Internet und WWW«, über einen DNS-Server aufgelöst werden, damit der Server im Internet unter diesem Namen erreichbar ist.

Starten Sie Apache 2 nun neu, indem Sie der weiter oben beschriebenen Anleitung für Ihre Plattform folgen. Falls Sie die Minimalkonfigurationsdatei unter einem anderen Namen als *httpd.conf* gespeichert haben, können Sie sie dabei mit der Option `-f Dateiname` angeben. Angenommen, Sie verwenden ein UNIX-System und die Datei */etc/httpd/httpd_minimal.conf*, dann sieht der entsprechende Befehl so aus:

```
# apachectl graceful -f /etc/httpd/httpd_minimal.conf
```

Auf einem Windows-Rechner müssen Sie dagegen diesen Befehl eingeben, falls Sie keine der bereits behandelten grafischen Methoden einsetzen und die Konfigurationsdatei *C:\Program Files\Apache2.2\conf\httpd_minimal.conf* verwenden:

```
> apache -k restart
  -f "C:/Program Files/Apache2.2/conf/httpd_minimal.conf"
```

Nach dem Neustart können Sie einen Browser öffnen und versuchen, die neue Startseite der Website aufzurufen. Falls alles in Ordnung ist, müssten Sie nun die eben erstellte HTML-Seite sehen.

Hier noch eine kurze Übersicht über die Direktiven, die in der Minimalkonfiguration verwendet werden – sie alle werden in Kapitel 6, »Grundkonfiguration«, genauer erläutert:

- **`ServerName www.mynet.de`**
  Hier wird der Name festgelegt, unter dem der Server im Internet oder Intranet erreichbar sein soll.
- **`Listen 80`**
  Diese Direktive legt den TCP-Port fest, an dem der Server auf HTTP-Verbindungsanfragen lauscht. Wie Sie bereits wissen, ist 80 der Standardport für das HTTP.
- **`ServerRoot /usr/local/apache2`**
  Die Konfigurationsanweisung `ServerRoot` legt das Verzeichnis fest, in dem sich die Konfigurations- und Ressourcendateien des Servers befinden, das heißt Verzeichnisse wie `conf`. Verschiedene andere Direktiven benötigen als Parameter eine Pfadangabe, die relativ zu diesem Verzeichnis angegeben werden kann. Wenn ein solcher Wert – wie im obigen Windows-Beispiel – Leerzeichen enthält, müssen Sie ihn in Anführungszeichen setzen.
- **`LoadModule dir_module modules/mod_dir.so`**
  Diese Zeile und die beiden nächsten werden auf einem UNIX-System nur benötigt, wenn alle Module (oder die drei Module `mod_dir`, `mod_autoindex` und `mod_authz_host` ausdrücklich) als DSOs installiert wurden. Mithilfe von `LoadModule` wird ein DSO-Modul beim Server-Start geladen und aktiviert. Es

werden jeweils ein schematischer Modulname und der Pfadname des Moduls (relativ zur `ServerRoot`) benötigt. Für den schematischen Namen müssen Sie das Präfix `mod` vom Dateinamen entfernen und stattdessen `_module` anfügen. Aus `mod_foo` würde also beispielsweise `foo_module` mit folgender Direktive:

```
LoadModule foo_module modules/mod_foo.so
```

- Unter Windows werden die `LoadModule`-Direktiven fast immer verwendet, weil Module auf dieser Plattform in aller Regel als DSOs installiert werden.

  `mod_dir` ist übrigens das Modul, das die Definition des Startseitennamens ermöglicht und darüber hinaus Verzeichnis-URLs ohne abschließenden `/` verarbeitet. `mod_autoindex` generiert dagegen automatisch einen Index aller Dateien im Verzeichnis, wenn keine Startseite vorhanden ist. `mod_authz_host` schließlich erlaubt Zugriffsbeschränkungen auf Hostnamens- und IP-Adressebene.

- **`DocumentRoot /usr/local/apache2/htdocs`**
  Dies ist das Verzeichnis, in dem sich die Website befindet, die der Server veröffentlicht.

- **`<Directory /> ... </Directory>`**
  Ein `<Directory>`-Container enthält die Konfigurationsoptionen für ein einzelnes Verzeichnis. Der Container für das Wurzelverzeichnis bestimmt die Voreinstellung für *alle* Verzeichnisse und URLs. Es gibt einige Direktiven, die nur in Verzeichnis-Containern stehen dürfen.

- **`Options None`**
  Die Direktive `Options` legt fest, welche »Dienstleistungen« in diesem Verzeichnis verfügbar sein sollen. Die Voreinstellung `None` für das Wurzelverzeichnis besagt, dass zunächst einmal sämtliche Optionen deaktiviert sind. Einzelheiten erfahren Sie in Kapitel 6, »Grundkonfiguration«.

- **`AllowOverride None`**
  Es besteht die Möglichkeit, Konfigurationsdirektiven in eine Datei innerhalb eines einzelnen Verzeichnisses oder sogar Unterverzeichnisses einer Website auszulagern. Diese Datei heißt standardmäßig `.htaccess` (kann durch die Direktive `AccessFileName` geändert werden). Am bekanntesten ist dieses Verfahren zur Erstellung passwortgeschützter Verzeichnisse; neben Authentifizierungsdirektiven lassen sich aber auch andere Einstellungen durch diese Datei lokal überschreiben.

  `AllowOverride` legt fest, welche Direktiven überschrieben werden dürfen. Dazu werden diese nicht einzeln angegeben, sondern es gibt einige spezielle Bezeichnungen für Funktionsgruppen, die im nächsten Kapitel ausgeführt werden. Die Voreinstellung `None` bedeutet natürlich, dass zunächst einmal kei-

nerlei Direktiven überschrieben werden dürfen. `.htaccess`-Dateien werden in diesem Fall sogar vollständig ignoriert.

- **`Require all denied`**
  Dies ist die aktuelle Syntax, um in Apache 2.4 allen Benutzern ausnahmslos den Zugriff zu verweigern. Dies ist die empfohlene Voreinstellung für alle Verzeichnisse; Sie sollten die Bereiche, auf die Benutzer zugreifen dürfen, jeweils explizit freigeben.

  `Require`

  Dies ist eine Core-Direktive, die alle Aspekte der Zugriffskontrolle regelt. Bis Apache 2.2 bildete die Host- und IP-Adressen-Zugriffskontrolle (einschließlich des Erlaubens oder Verbietens aller Zugriffe) die Ausnahme; ab 2.4 ist `Require` auch für diese zuständig.

- **`Order deny,allow`**
  Die Direktive `Order` aus dem Modul `mod_authz_host` bestimmt, in welcher Reihenfolge die Zugriffseinstellungen `Allow` und `Deny` verarbeitet werden. `allow,deny` bedeutet, dass die Verbote die Erlaubnisse einschränken. Bei `deny,allow` ist es umgekehrt. In Apache 2.4 ist diese Direktive zwar noch vorhanden (im Modul `mod_access_compat`), aber genau wie `Allow` und `Deny` veraltet; sie alle wurden durch `Require` ersetzt.

- **`Deny from all`**
  Die Direktive `Deny` stammt ebenfalls aus `mod_authz_host`. Diese rigorose Einstellung besagt, dass auf das Wurzelverzeichnis zunächst einmal *niemand* zugreifen darf. In Apache 2.4 ist auch diese Direktive veraltet und wurde nach `mod_access_compat` verschoben.

- **`<Directory /usr/local/apache2/htdocs> ... </Directory>`**
  Dieser Container enthält die Einstellungen für die freigegebene Website. Da das Wurzelverzeichnis sehr restriktiv abgesperrt wurde, müssen hier einige Einstellungen durch ausdrückliche Erlaubnisse überschrieben werden: `Options` und `AllowOverride` werden auf `All` gesetzt – unterhalb der `DocumentRoot` sind alle Verzeichnisoptionen verfügbar, und das Maximum an Direktiven darf in `.htaccess`-Dateien überschrieben werden. Order erhält den Wert `allow,deny` – die `Allow`-Einstellung soll Vorrang besitzen.

- **`DirectoryIndex index.html`**
  Diese durch das Modul `mod_dir` bereitgestellte Direktive definiert die Namen eines oder mehrerer Dokumente, die der Server ausliefert, wenn die URL in einer Anfrage nur einen Verzeichnis-, aber keinen Dateinamen enthält. Falls mehrere Dateien angegeben werden, sucht Apache in der angegebenen Reihenfolge nach ihnen und liefert das erste Dokument aus, das er findet. *index.html* ist die Standardeinstellung.

- `Require all granted`
  Diese Syntax kommt in Apache 2.4 zum Einsatz, um beliebigen Besuchern den Zugriff auf eine Ressource zu erlauben.
- `Allow from all`
  Genau wie `Deny` ist auch `Allow` in `mod_authz_host` definiert. Diese Zeile erlaubt allen Hosts aus dem gesamten Internet den Zugriff auf die Website. In Apache 2.4 ist diese Direktive veraltet und befindet sich im Modul `mod_access_compat`.

## 5.3 Zusammenfassung

Das ausführbare Programm, das den Kern des Apache-Webservers bildet, lässt sich – typisch für Software aus der UNIX-Welt – mit zahlreichen Parametern und Optionen aufrufen. Sie können den Server damit nicht nur einfach starten, sondern über die bestehende Konfigurationsdatei hinaus anpassen oder auch wichtige Informationen über den Status des HTTP-Servers erhalten. Dies gilt sowohl für Apache auf der UNIX-Plattform als auch auf Windows-Systemen.

Daneben gibt es zahlreiche Hilfsmittel zur Steuerung des Webservers. Auf UNIX-Systemen ist er mit dem Shell-Skript *apachectl* ausgestattet, das nicht nur den angepassten Start, sondern auch das Beenden und den Neustart von Apache 2 ermöglicht. Unter Windows wurden einige dieser Optionen zum ausführbaren Programm (*httpd.exe*) hinzugefügt.

Eine gewisse Herausforderung besteht in dem Problem, Apache beim Hochfahren des Systems automatisch zu starten. Auf einem UNIX-Rechner müssen Sie herausfinden, ob Ihr System System V Init oder die BSD-Boot-Methode verwendet, um die entsprechenden Startbefehle hinzuzufügen. Unter Windows empfiehlt sich zu diesem Zweck der Betrieb des Webservers als Dienst.

Nachdem Sie Ihren Server gestartet haben, sollten Sie mit einer Minimalkonfiguration wie der hier beschriebenen überprüfen, ob er auch ordnungsgemäß seine Arbeit erledigt, bevor Sie ihn praktisch im Internet oder Intranet einsetzen.

*»Die Grundlage ist das Fundament der Basis.«*
*– Le Corbusier*

# 6 Grundkonfiguration

Das wichtigste Element bei der Arbeit mit dem Apache-Webserver sind seine Konfigurationsdateien; die wichtigste von ihnen ist die Hauptkonfigurationsdatei *httpd.conf*. In diesem Kapitel erfahren Sie deshalb alles über ihre Syntax und über die wichtigsten Direktiven für die grundlegende Anpassung des Servers. In den restlichen Kapiteln dieses Buches werden Sie natürlich noch zahlreiche weitere Konfigurationsbefehle kennenlernen.

## 6.1 Aufbau der Apache-Konfigurationsdateien

Wie bei den meisten Programmen aus dem UNIX-Umfeld basiert auch die Konfiguration des Apache-Webservers auf einfachen Textdateien. Das einzige kleine Problem an dieser Art von Dateien ist, dass es aus historischen Gründen kein einheitliches Format für sie gibt: Wer heute ein neues Softwareprojekt startet, wird dessen Konfigurationsdatei höchstwahrscheinlich als XML-Format konzipieren – dies ist beispielsweise beim Servlet- und JSP-Server Tomcat der Fall (siehe Kapitel 15, »Technologien zur Webprogrammierung«). Als 1995 mit der Arbeit an Apache begonnen wurde, gab es einen solchen Standard dagegen noch nicht.

Das Format der Apache-Konfigurationsdateien wurde vom NCSA HTTPd übernommen, dessen Weiterentwicklung Apache ursprünglich war. Der NCSA-Server besaß drei Konfigurationsdateien: *httpd.conf* für die eigentliche Server-Konfiguration, *srm.conf* (Server Resource Map) für die Eigenschaften der Website und *access.conf* zur Verwaltung von Zugriffsrechten. Frühe Versionen des Apache-Webservers übernahmen diese Aufteilung. In Version 1.3 wurde sie aus Kompatibilitätsgründen noch optional unterstützt, aber nicht mehr empfohlen. In Apache 2.0 können Sie das Verfahren nachahmen, indem Sie die beiden anderen Dateien über `Include`-Direktiven in *httpd.conf* einbinden. Allerdings ist es üblicher, entweder alle Konfigurationsdaten in der Hauptdatei unterzubringen oder aber – die bevorzugte Methode ab Version 2.2 – zusätzliche Einstellungen nach Modulen beziehungsweise Aufgaben geordnet in eine Reihe von `Include`-Dateien auszulagern.

Sie können Ihre eigene Konfigurationsdatei übrigens auf zwei verschiedene Arten erstellen: Entweder modifizieren Sie die mitgelieferte Standarddatei, oder Sie schreiben eine ganz neue »from scratch«. Das Ändern der vorhandenen Datei geht schneller, während Sie bei einer völlig neuen Datei keinen unnötigen Ballast mitnehmen und nur die Konfigurationsdirektiven verwenden, die Sie wirklich benötigen. Die vorgefertigte Datei ist nämlich so eingerichtet, dass sie für eine Vielzahl von Anwendungsfällen des Webservers funktioniert. Das heißt aber eben nicht, dass der Server mit dieser Konfiguration gut oder gar optimal arbeitet. Mithilfe der Anleitung in diesem Kapitel und in späteren Kapiteln können Sie jedenfalls beide Möglichkeiten wählen.

### 6.1.1 Namen, Pfad und Aufgaben der Konfigurationsdateien

Grundsätzlich wird zunächst nur eine einzige Konfigurationsdatei verarbeitet, nämlich *httpd.conf* im `SYSCONFDIR` Ihres Verzeichnislayouts. Beim `Apache`-Layout ist dies */usr/local/apache2/conf*, beim `GNU`-Layout */usr/local/etc/apache2* usw. (siehe Kapitel 4, »Apache kompilieren und installieren«). Sollen weitere Konfigurationsdateien verwendet werden, müssen diese mithilfe der im Verlauf dieses Kapitels vorgestellten Direktive `Include` in diese Datei (oder verschachtelt ineinander) eingebunden werden.

Verschiedene Versionen beziehungsweise Distributionen von Apache verwenden unterschiedliche Zusatzkonfigurationsdateien. Zur Verdeutlichung hier einige Beispiele:

- In Apache 2.0 war alles sehr einfach: Die gesamte Konfiguration steht in der Hauptdatei *httpd.conf* – mit einer einzigen Ausnahme: Die `mod_ssl`-Konfiguration für gesicherte Verbindungen steht in der externen Datei *ssl.conf*. Diese enthält eine `<IfDefine>`-Direktive, durch die sie nur aktiviert wird, wenn der Server mit dem Parameter `-DSSL` gestartet wird.
- In Apache 2.2 wurden diverse Teile der Hauptkonfiguration ausgelagert; sie befinden sich aus der Sicht des Konfigurationsverzeichnisses im Unterverzeichnis *extra*. Damit sie aktiviert werden, müssen Sie das Kommentarzeichen # vor den jeweiligen `Include`-Direktiven in *httpd.conf* entfernen. Es handelt sich um folgende Dateien:
  - *httpd-mpm.conf* – MPM-spezifische Einstellungen, die vor allem die Performance betreffen; die zuständigen Direktiven werden noch in diesem Kapitel behandelt.
  - *httpd-multilang-errordoc.conf* – mehrsprachige Fehlermeldungsseiten. Die dafür notwendige Content-Negotiation (automatische Dokumentauswahl nach Browser-Präferenz) wird in Kapitel 7, »Header und MIME-Einstellun-

gen«, behandelt; das Einbinden benutzerdefinierter Fehlermeldungsseiten dagegen in Kapitel 8, »Weiterleitungen und Indizes«.

- *httpd-autoindex.conf* – Optionen für automatisch generierte Verzeichnis-Indizes (siehe Kapitel 8, »Weiterleitungen und Indizes«)
- *httpd-languages.conf* – Einstellungen für die Zuordnung zwischen zusätzlichen Dateiendungen und Dokumentsprachen (siehe Kapitel 7, »Header und MIME-Einstellungen«)
- *httpd-userdir.conf* – Veröffentlichung benutzerspezifischer Websites unter */~user*-URL-Pfaden (zuständig ist das in Kapitel 8, »Weiterleitungen und Indizes«, behandelte Modul `mod_userdir`)
- *httpd-info.conf* – Informationen und Statusmeldungen des Servers unter speziellen URLs veröffentlichen (siehe Kapitel 8, »Weiterleitungen und Indizes«)
- *httpd-vhosts.conf* – Einstellungen für virtuelle Hosts (Näheres in Kapitel 12, »Skalierung und Performance-Tuning«)
- *httpd-manual.conf* – Einbindung der Apache-Dokumentation unter dem URL-Pfad */manual*
- *httpd-dav.conf* – WebDAV-Einstellungen (siehe Kapitel 17, »Apache erweitern«)
- *httpd-default.conf* – einige zusätzliche Einstellungen für den Hauptserver (die meisten von ihnen werden im vorliegenden Kapitel behandelt)
- *httpd-ssl.conf* – SSL-Konfiguration (das Thema von Kapitel 10, »Gesicherte Verbindungen«)

- Die vorinstallierte Apache-2.2-Distribution unter openSUSE 12 verwendet eine andere Konfigurationsdateien-Aufteilung; diese ermöglicht das modulare Hinzufügen und Entfernen von Apache-Bestandteilen mithilfe von YaST. Beachten Sie, dass Änderungen in einigen dieser Dateien aus den Inhalten von */etc/sysconfig/apache2* überschrieben werden. Diese Dateien werden von YaST verwendet und haben eine andere Syntax als Apache-Konfigurationsdateien.

  Hier die Liste der einzelnen echten Apache-Konfigurationsdateien, die sich standardmäßig alle im Verzeichnis */etc/apache2/conf* beziehungsweise in den im Einzelnen angegebenen Unterverzeichnissen befinden:

  - *uid.conf* – Benutzer- und Gruppen-ID für Worker-Prozesse (siehe Direktiven `User` und `Group` weiter unten in diesem Kapitel)
  - *server-tuning.conf* – MPM-spezifische Prozess- und Performance-Einstellungen

- *sysconfig.d/loadmodule.conf* – zu ladende Module. Änderungen an dieser Datei werden allerdings nicht akzeptiert; verwenden Sie YaST, oder tragen Sie die gewünschten Modulkürzel (ohne `mod_`) in die Datei */etc/sysconfig/apache2* ein.
- *listen.conf* – IP-Adressen und Ports, an denen Apache lauschen soll (siehe Direktive `Listen` weiter unten)
- *mod_log_config.conf* – Definition der Log-Formate und -Dateien (siehe Kapitel 11, »Logging«)
- *sysconfig.d/global.conf* – Server-Grundeinstellungen (auch diese Konfiguration wird durch */etc/sysconfig/apache2* bestimmt)
- *mod_status.conf* – Statusinformationen (siehe Kapitel 8, »Weiterleitungen und Indizes«)
- *mod_info.conf* – Server-Informationen (siehe Kapitel 8, »Weiterleitungen und Indizes«)
- *mod_usertrack.conf* – Cookie-basiertes Usertracking (siehe Kapitel 11, »Logging«)
- *mod_autoindex-defaults.conf* – automatisch generierte Verzeichnisindizes (siehe Kapitel 8, »Weiterleitungen und Indizes«)
- *mod_mime-defaults.conf* – MIME-Einstellungen (siehe Kapitel 7, »Header und MIME-Einstellungen«)
- *errors.conf* – angepasste Fehlermeldungen (siehe Kapitel 8, »Weiterleitungen und Indizes«)
- *ssl-global.conf* – SSL-Konfiguration (siehe Kapitel 10, »Gesicherte Verbindungen«)
- *default-server.conf* – Optionen des Standardservers (beantwortet Anfragen, für die kein virtueller Host zuständig ist)
- *mod_userdir.conf* – benutzerspezifische Webverzeichnisse; wird von *default-server.conf* eingebunden (Kapitel 8, »Weiterleitungen und Indizes«).
- *conf.d/apache2-manual?conf* – Online-Dokumentation; dieses (ebenfalls in *default-server.conf* eingebundene) `Include` wird durch das `?` optional.
- *sysconfig.d/include.conf* – Includes für Ihre eigenen Erweiterungsdateien (wird durch */etc/sysconfig/apache2* festgelegt)
- *vhosts.d/*.conf* – Einstellungen für virtuelle Hosts; alle **.conf*-Dateien in diesem Verzeichnis werden automatisch geladen.

### 6.1.2 Grundlegendes zur Syntax

Apache-Konfigurationsdateien wie *httpd.conf* bestehen aus Konfigurationsdirektiven, die jeweils in einer eigenen Zeile stehen müssen. Leere Zeilen werden ignoriert, ebenso Zeilen, die mit einem # beginnen – genau wie in UNIX-Shells oder in Perl leitet dieses Zeichen einen Kommentar ein. Eine Direktive hat stets das folgende grundlegende Format:

```
Direktivenname Wert [Wert ...]
```

Der Name des jeweiligen Befehls wird also durch (beliebig viele) Leerzeichen von dem eingestellten Wert und eventuellen weiteren Werten getrennt. Hier zwei Beispiele (von deren Bedeutung Sie bereits in Kapitel 5, »Apache in Betrieb nehmen«, erfahren haben):

```
ServerName www.mynet.de
DirectoryIndex index.html index.htm
```

Zu lange Direktiven können Sie auf mehrere Zeilen aufteilen, wenn Sie die unterbrochenen Zeilen durch einen Backslash (\) abschließen. Beispiel:

```
LoadModule autoindex_module \
   modules/mod_autoindex.so
```

Zusätzlich gibt es einige spezielle Container für Konfigurationsoptionen, die nur einen bestimmten Kontext betreffen. Beispielsweise legen Direktiven zwischen `<Directory ...>` und `</Directory>` die Optionen für ein einzelnes freigegebenes Verzeichnis fest.

Insgesamt lassen sich die Konfigurationsanweisungen der Datei *httpd.conf* in die drei folgenden großen Abschnitte unterteilen:

1. **Programmeinstellungen**
   In den Kommentaren der vorgefertigten Konfigurationsdatei, die mit Apache installiert wird, heißt dieser Abschnitt »global environment«. Hier stehen alle Einstellungen, die das Programm selbst steuern. Die meisten Direktiven für den ersten Abschnitt werden in diesem Kapitel beschrieben.
2. **Konfiguration des Hauptservers**
   Hier werden die Optionen für die Hauptwebsite eingestellt, die dieser Server ausliefert. Die Direktiven für diesen Abschnitt finden Sie thematisch sortiert in verschiedenen Kapiteln dieses Buches.
3. **Virtuelle Hosts**
   Wie bereits erwähnt, ist das Einrichten virtueller Hosts eine der wichtigsten Fähigkeiten von Apache. Im letzten Abschnitt von *httpd.conf* steht die Konfiguration für die virtuellen Hosts. Das grundlegende Verfahren wird in diesem

Kapitel erläutert. Genaueres über virtuelle Hosts erfahren Sie in Kapitel 12, »Skalierung und Performance-Tuning«.

Beachten Sie, dass es in der Konfigurationsdatei keine formale Markierung für diese drei Teile gibt. Es handelt sich vielmehr um eine Konvention für die Reihenfolge der Konfigurationsanweisungen. In der vorgefertigten Datei sind sie lediglich durch entsprechende Kommentare gekennzeichnet. Dennoch sollten Sie sich an diese Abfolge halten, um Probleme zu vermeiden. Beispielsweise muss ein DSO-Modul bereits geladen sein, bevor die für dieses Modul definierten Direktiven benutzt werden können.

Konfigurationsdirektiven können viele unterschiedliche Arten von Werten annehmen. Einige der wichtigsten sind folgende:

- `On | Off`
  Manche Direktiven schalten eine bestimmte Funktion oder Eigenschaft grundsätzlich ein und aus. Dazu werden in der Regel die Werte `On` und `Off` verwendet. Beispielsweise schaltet die folgende Direktive grundsätzlich die `Rewrite`-Funktion (URL-Umformung mit regulären Ausdrücken) ein:

  ```
  RewriteEngine On
  ```

- **Numerisch**
  Der Wert mancher Direktiven ist eine ganze Zahl. Dies gilt insbesondere für globale Server-Einstellungen, die die Performance von Apache 2 beeinflussen. Beispielsweise gibt die folgende Direktive an, dass über eine persistente HTTP-Verbindung nacheinander maximal 100 Anfragen verarbeitet werden dürfen:

  ```
  MaxKeepAliveRequests 100
  ```

- **Pfad**
  Einige Direktiven benötigen einen Pfad im Dateisystem als Parameter. Ob es sich um einen absoluten Pfad handeln muss oder ob er relativ zur `ServerRoot` oder einem anderen Basisverzeichnis angegeben werden kann, unterscheidet sich je nach Direktive (und wird bei der Beschreibung in diesem Buch jeweils angegeben). Die folgende Option legt z. B. das Stammverzeichnis der Website fest:

  ```
  DocumentRoot /usr/local/httpd/htdocs
  ```

  Beachten Sie, dass auch auf Windows-Systemen der Slash (`/`) als Pfadtrennzeichen verwendet wird, nicht der plattformspezifische Backslash (`\`). Die `DocumentRoot`-Angabe könnte unter Windows also beispielsweise folgendermaßen aussehen:

  ```
  DocumentRoot "E:/Apache2/htdocs"
  ```

Bei allen Pfadangaben unter sämtlichen Betriebssystemen sind die Anführungszeichen zunächst einmal optional. Sie *müssen* sie nur dann verwenden, wenn ein Verzeichnis- oder Dateiname in einem Pfad ein Leerzeichen enthält. Dies ist bei `DocumentRoot` unter Windows beispielsweise der Fall, wenn Sie das Standardverzeichnis verwenden, das der MSI-Installer vorgibt:

```
DocumentRoot "C:/Programme/Apache Group/Apache2/htdocs"
```

- **Dateiendung**
  Viele Konfigurationsanweisungen weisen einer Dateiendung eine bestimmte Einstellung zu. In der Regel wird bei den Dateiendungen nicht zwischen Groß- und Kleinschreibung unterschieden, und Sie können den Punkt am Anfang nach Belieben setzen oder weglassen. Das folgende Beispiel stellt für Dokumente mit den Endungen `.htm` und `.html` den Zeichensatz `iso-8859-1` ein:

  ```
  AddCharset iso-8859-1 .htm .html
  ```

- **String**
  Bei bestimmten Direktiven ist der Wert eine Zeichenkette, die beispielsweise einen Text angibt, der ausgegeben werden soll. Dieser muss in Anführungszeichen stehen. Zum Beispiel können Sie für die HTTP-Fehlermeldungen anstelle der URL eines Fehlerdokuments auch einfach einen Text festlegen, hier etwa für den Fehler 404 (Dokument nicht gefunden):

  ```
  ErrorDocument 404 "So etwas haben wir nicht"
  ```

- **Feste Werte**
  Verschiedene Direktiven haben ihre eigenen, besonderen Werte. Ein Beispiel ist `AllowOverride`; diese Option bestimmt, welche Direktiven innerhalb eines Verzeichnisses durch eine *.htaccess*-Datei überschrieben werden dürfen. Dazu werden einige Namen definiert, die für bestimmte Gruppen von Direktiven stehen. Beispielsweise erlaubt die folgende Variante das Überschreiben von Authentifizierungs- und Dateioptionen:

  ```
  AllowOverride AuthConfig FileInfo
  ```

- **Regulärer Ausdruck**
  Manche Direktiven erlauben neben einem festgelegten Wert die Angabe eines Musters. Solche Muster werden in Form regulärer Ausdrücke (RegExp) angegeben. Details über die Syntax erhalten Sie in Anhang G. `mod_rewrite` macht z.B. intensiven Gebrauch von regulären Ausdrücken. Auch der Container `<DirectoryMatch>` verwendet einen regulären Ausdruck, um die Konfiguration für eine durch das Muster angegebene Gruppe von Verzeichnissen anzugeben statt nur für ein einzelnes Verzeichnis (durch einen `<Directory>`-Con-

tainer). Beispielsweise betrifft die folgende Formulierung alle Verzeichnisse unter `/usr/local/htdocs`, die `data`, gefolgt von einer Zahl, lauten:

```
<DirectoryMatch "^/usr/local/htdocs/data[0-9]+$">
   ...
</DirectoryMatch>
```

- **Komplexe Ausdrücke**
  Einige Direktiven verstanden schon vor der Einführung von Apache 2.4 komplexe Ausdrücke, insbesondere die URL-Umleitungsdirektiven des Moduls `mod_rewrite` (siehe Kapitel 8, »Weiterleitungen und Indizes«) sowie einige Einstellungen von `mod_ssl` (Kapitel 10, »Gesicherte Verbindungen«). Für die neue Version 2.4 wurde der Parser für solche Ausdrücke, `ap_expr`, aus den einzelnen Modulen ausgelagert und verallgemeinert. Genaueres über solche komplexen Ausdrücke erfahren Sie in der Beschreibung der Container-Direktive `<If>` im vorliegenden Kapitel. Das folgende Beispiel setzt für eine HTTP/1.0-Anfrage, die keinen `Host`-Header enthält, nachträglich einen Standardwert (bei HTTP/1.1 ist der `Host`-Header vorgeschrieben):

  ```
  <If "-z req(Host)">
    RequestHeader set Host www.mynet.de
  </If>
  ```

- **Kein Wert**
  Einige wenige Direktiven besitzen gar keinen Wert. Sie schalten durch ihre bloße Präsenz eine bestimmte Option ein beziehungsweise deaktivieren diese, wenn sie nicht vorhanden sind. Sie können beispielsweise folgendermaßen das Beispielmodul `mod_example` (das die Funktionsweise der Apache-API erläutert) aktivieren:

  ```
  Example
  ```

  Die Apache Group ist bemüht, diese Schreibweise auszumerzen und durch `On|Off` zu ersetzen. Bei den wenigen Direktiven, die noch ohne Wert geschrieben werden, wird dies aus Kompatibilitätsgründen aber erst nach und nach geschehen.[1]

### 6.1.3 Syntaxschema

Für jede in diesem Buch behandelte Apache-Direktive finden Sie vor der ausführlichen Beschreibung eine tabellarische Übersicht nach folgendem Schema:

1 Nichtsdestotrotz wurde im Apache-Release 2.0.49 die nur für Windows gültige Direktive `Win32DisableAcceptEx` neu eingeführt, die – eher versehentlich – ebenfalls ohne Wert geschrieben wird.

### SampleDirective

Hier steht die Zusammenfassung der Aufgabe dieser Direktive:

| Seit Version | 1.2; bis 1.3.18 in `mod_bar` |
|---|---|
| Modul | `mod_foo` |
| Kontext | Server, `<VirtualHost>` |
| Syntax | `SampleDirective On\|Off` |
| Standardwert | `Off` |

Hier eine Übersicht über die Bedeutung der einzelnen Informationen:

- **Seit Version**
  Gibt die Apache-Version an, seit beziehungsweise in der diese Direktive (und die damit verbundene Funktionalität) verfügbar ist oder in der wichtige Änderungen vorgenommen wurden. Bei Direktiven, die »schon immer« verfügbar waren (üblicherweise bei solchen, die noch aus den Zeiten des NCSA HTTPd stammen), fehlt die Zeile komplett.
- **Modul**
  Gibt Auskunft darüber, zu welchem Modul die Option gehört. `core` steht für Direktiven, die in den Funktionskern eingebaut sind. Nehmen Sie diese Angabe bitte äußerst ernst: Wenn in diesem Feld steht, dass die Direktive zu einem bestimmten Modul gehört, dann bedeutet dies, dass sie nicht zur Verfügung steht, falls dieses Modul nicht vorhanden oder nicht aktiviert ist! In diesem Fall wird die Konfigurationsanweisung nicht etwa stillschweigend übergangen, sondern Apache verweigert den Start mit einer entsprechenden Fehlermeldung. Verwenden Sie im Zweifelsfall `<IfModule>`-Container.
- **Kontext**
  Informiert darüber, in welchen der in Abschnitt 6.2, »Kontexte und Container«, beschriebenen Kontexten der Konfigurationsdatei die Direktive eingesetzt werden kann. Falls *.htaccess* dabei ist, wird in Klammern der `AllowOverride`-Wert angegeben, der für den Einsatz dieser Direktive in einer *.htaccess*-Datei erforderlich ist. Beachten Sie, dass bei Angabe der Container `<Directory>`, `<Location>` und `<Files>` auch die jeweiligen RegExp-Gruppenoptionen `<DirectoryMatch>`, `<LocationMatch>` und `<FilesMatch>` gemeint sind.
- **Syntax**
  Gibt einen Überblick über die Art der Werte, die für die Direktive zulässig sind. Ein Pipe-Zeichen (|) trennt in diesem Zusammenhang mehrere Alternativen voneinander; eckige Klammern bezeichnen optionale Komponenten.

- **Standardwert**
  Dies ist der Wert, der für diese Direktive implizit gesetzt wird, wenn sie gar nicht in der Konfigurationsdatei vorkommt. Falls eine Direktive völlig optional ist und damit unwirksam, wenn sie nicht verwendet wird, steht hier die Beschreibung »nicht gesetzt«.

## 6.2 Kontexte und Container

Nicht jede Konfigurationsanweisung ergibt in jedem Zusammenhang einen Sinn. Viele von ihnen können nur in bestimmten Kontexten oder speziellen Containern der Konfigurationsdatei stehen. Diese Container und Kontexte werden hier kurz vorgestellt.

### 6.2.1 Der Server-Kontext

Im Server-Kontext, das heißt außerhalb aller Container, stehen vor allem Direktiven, die das grundlegende Verhalten von Apache 2 selbst betreffen. Beispielsweise werden hier Module geladen oder MPM- und Performance-Einstellungen vorgenommen.

Abgesehen davon, können auch viele datei- und verzeichnisbezogene Optionen in der Server-Konfiguration statt in einem Verzeichniskontext stehen, was den Vorteil hat, dass sie sich dann zunächst einmal auf alle Dateien und Verzeichnisse beziehen, solange sie nicht in einem untergeordneten Kontext überschrieben werden.

### 6.2.2 Virtuelle Hosts

Virtuelle Hosts sind eine der wichtigsten Fähigkeiten des Apache-Webservers: Erst die Möglichkeit, mehrere unabhängige Websites mit eigenständigen Domain-Namen innerhalb einer Server-Installation zu betreiben, erlaubt kommerzielles Webhosting. Näheres zum Thema erfahren Sie in Kapitel 12, »Skalierung und Performance-Tuning«.

#### VirtualHost

Container mit Konfigurationsdirektiven für einen virtuellen Host

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server |
| **Syntax** | `<VirtualHost Name\|IP-Adresse\|* [:Port]> ... </VirtualHost>` |

Es gibt zwar nicht viele Direktiven, die sich ausschließlich auf virtuelle Hosts beziehen, aber einige der Direktiven für die Server-Konfiguration lassen sich für einzelne virtuelle Hosts überschreiben – Angaben wie das Website-Stammverzeichnis (`DocumentRoot`) oder der `ServerName` *müssen* bei einem virtuellen Host sogar eigene Werte erhalten.

Der virtuelle Host, für den die Konfiguration gelten soll, kann als Hostname oder als IP-Adresse angegeben werden. Zusätzlich können Sie hinter einem Doppelpunkt einen TCP-Port angeben. Beispiele:

```
<VirtualHost www.sales.mynet.de>
  ...
</VirtualHost>

<VirtualHost 196.17.8.52>
  ...
</VirtualHost>

<VirtualHost www.order.mynet.de:8080>
  ...
</VirtualHost>
```

Wenn der virtuelle Host die Standardadresse des Servers mit einem anderen Port verwenden soll, können Sie anstelle der Adresse einfach ein Sternchen (*) oder alternativ `_default_` verwenden:

```
<VirtualHost *:8000>
  ...
</VirtualHost>

<VirtualHost _default_:3128>
...
</VirtualHost>
```

### 6.2.3 Verzeichnis- und Datei-Container

In der Apache-Konfigurationsdatei enthalten die Verzeichnis- und Datei-Container die meisten Einstellungen für die veröffentlichte Website. Für jede der drei Container-Sorten `<Directory>` (lokales Verzeichnis), `<Location>` (URL) und `<Files>` (einzelne Dateien) gibt es ein Gegenstück mit dem Suffix `Match`: `<DirectoryMatch>`, `<LocationMatch>` und `<FilesMatch>` nehmen anstelle eines einfachen Pfadnamens einen regulären Ausdruck als Argument an, sodass Sie damit auf einfache Weise Einstellungen für eine ganze Gruppe von Verzeichnissen, URLs beziehungsweise Dateien vornehmen können.

Bitte beachten Sie, dass die Kontextangabe »Verzeichnis« in der Online-Dokumentation von Apache 2 nicht nur für <Directory>-Container steht, sondern – sofern nicht im Einzelfall anders angegeben – auch für <Location>- und <Files>-Bereiche. Darüber hinaus sind in der Regel auch noch die entsprechenden <*Match>-Varianten gemeint. In allen Direktivenreferenzen in diesem Buch (mit Ausnahme der Kurzreferenz im Anhang) werden die drei Kontexte dagegen einzeln angegeben.

### Directory

Konfigurationsanweisungen für ein Verzeichnis

| | |
|---|---|
| **Modul** | core |
| **Kontext** | Server |
| **Syntax** | <Directory Pfadname> ... </Directory> |

Die meisten Einstellungen, die das Verhalten der veröffentlichten Website betreffen, sind Verzeichnisoptionen. <Directory> ... </Directory> ist der wichtigste der entsprechenden Container. Das folgende Beispiel ist der Container für die Optionen der Standard-Website auf einem UNIX-Server mit GNU-Installationslayout (die DocumentRoot):

```
<Directory /usr/local/share/apache2/htdocs>
  ...
</Directory>
```

Unter Windows variiert die DocumentRoot je nach Apache-Installationsverzeichnis. Beispiel:

```
<Directory "E:/Apache2/htdocs">
  ...
</Directory>
```

Auch für andere Verzeichnisse als die gesamte Website werden übrigens <Directory>-Container verwendet, beispielsweise für die Optionen der Freigabe von Benutzer-Websites mittels mod_userdir.

Es ist wichtig, die Reihenfolge zu kennen, in der Konfigurationsdirektiven aus mehreren passenden <Directory>-Containern auf ein Verzeichnis angewendet werden: Es wird immer zuerst der allgemeinste (am weitesten übergeordnete) und zuletzt der am tiefsten verschachtelte Container verwendet. Angaben, die einem betroffenen Verzeichnis »näher« sind, überschreiben also »entferntere« (allgemeinere) Einstellungen. Diese Reihenfolge sollten Sie sich beispielsweise zunutze machen, um aus Sicherheitsgründen zunächst einmal für das allge-

meinste Verzeichnis (/) sämtliche Zugriffe auf den Server zu verbieten und nur für einzelne Verzeichnisse wieder zu erlauben:

```
<Directory />
  Order Deny,Allow
  Deny from All
</Directory>
...
<Directory /usr/local/share/apache2/htdocs>
  Order Allow,Deny
  Allow from All
  # Weitere Site-Direktiven ...
</Directory>
```

Wenn zusätzlich reguläre Ausdrücke (mittels `<DirectoryMatch>` oder `<Directory ~>`) ins Spiel kommen, werden diese erst nach allen Einzelverzeichnis-Direktiven verarbeitet.

Anstelle eines einzelnen Verzeichnisses können Sie übrigens auch ein einfaches Muster angeben. Die Syntax entspricht den Datei- und Verzeichnismustern in UNIX-Shells:

- Ein Fragezeichen (`?`) steht für genau ein beliebiges Zeichen. Beispielsweise kann `docs?` für `docs1`, `docs5` oder auch `docsx` stehen, nicht aber für `docs` oder `docs02`.
- Ein Sternchen (`*`) repräsentiert beliebig viele beliebige Zeichen. `docs*` kann also für `docs`, `docs1`, `docs100` oder `docs_test` stehen. Der Platzhalter steht aber niemals für mehrere Verzeichnisse. `docs/*scripts` steht also beispielsweise für `docs/perlscripts` und `docs/php_scripts`, aber nicht für `docs/test/scripts`.
- Eine Liste von Zeichen in eckigen Klammern bedeutet, dass eines dieser Zeichen gemeint ist. `[abc]` ist also eines der Zeichen `a`, `b` oder `c`. Alternativ können Sie auch einen Bereich angeben: `[0-9]` steht z. B. für eine beliebige Ziffer.

#### DirectoryMatch

Konfigurationsanweisungen für eine Gruppe von Verzeichnissen, auf die ein regulärer Ausdruck passt

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `<DirectoryMatch RegExp> ... </DirectoryMatch>` |

Im Grunde gelten für <DirectoryMatch>-Container dieselben Aussagen wie für <Directory>-Abschnitte: Auch sie dienen der Konfiguration von Verzeichnissen. Allerdings wird hier kein einzelnes Verzeichnis angegeben, sondern ein regulärer Ausdruck, der auf viele Verzeichnisse zutreffen kann. Ein Synonym, das aber aus Gründen der Übersichtlichkeit nicht zu empfehlen ist, ist übrigens folgende Variante von <Directory>:

```
<Directory ~ RegExp>
  ...
</Directory>
```

Die folgende Formulierung betrifft z. B. sämtliche Verzeichnisse unterhalb von */usr/local/apache2/*, deren Name mit `doc` beginnt:

```
<DirectoryMatch "^/usr/local/apache2/doc">
  ...
</DirectoryMatch>
```

Mithilfe dieses Containers können Sie alle Verzeichnisse unterhalb von */usr/local/apache2/htdocs* ansprechen, deren Name `shop`, `catalog` oder `order` enthält:

```
<DirectoryMatch \
  "^/usr/local/apache2/htdocs/.*(shop|catalog|order)">
  ...
</DirectoryMatch>
```

#### Location

Konfigurationsanweisungen für eine bestimmte URL

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `<Location URL-Pfad\|URL> ... </Location>` |

Der wesentliche Unterschied zwischen einem <Directory>- und einem <Location>-Container besteht darin, dass <Directory> sich auf ein tatsächliches Verzeichnis im lokalen Verzeichnis beziehen muss, während <Location> die angeforderte URL betrifft. Ein <Location>-Container kann also beispielsweise auch URL-Pfade beschreiben, die mithilfe einer `Alias`-Direktive in den Verzeichnisbereich der Website abgebildet wurden. Diese Konfigurationsabschnitte werden erst nach <Directory>- und <Files>-Containern verarbeitet, und zwar in der Reihenfolge, in der sie in der Konfigurationsdatei stehen.

Das folgende Beispiel betrifft lokale URLs unter dem Pfad */info*:

```
<Location /info>
  ...
</Location>
```

### LocationMatch

Konfigurationsanweisungen für ein URL-Muster, auf das ein regulärer Ausdruck zutrifft

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | <LocationMatch RegExp> ... </LocationMatch> |

<LocationMatch> bezieht sich auf eine Gruppe von URLs beziehungsweise URL-Pfaden, die durch einen regulären Ausdruck beschrieben werden. Das folgende Beispiel betrifft alle URLs, die den Teilpfad */info* enthalten:

```
<LocationMatch "/info">
  ...
</LocationMatch>
```

### Files

Konfigurationsanweisungen für Dateien mit dem angegebenen Namen

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, .htaccess (All) |
| Syntax | <Files Dateiname> ... </Files> |

Ein <Files>-Container enthält Konfigurationsanweisungen für alle Dateien im aktuellen Kontext, die den angegebenen Dateinamen tragen. Sie können einen einzelnen Dateinamen angeben oder Shell-Platzhalter verwenden. Das folgende Beispiel gilt für alle Dateien, die *info.html* heißen:

```
<Files info.html>
...
</Files>
```

Der folgende Container betrifft dagegen alle Dateien mit der Endung .gif:

```
<Files *.gif>
...
</Files>
```

#### FilesMatch

Konfigurationsanweisungen für eine Gruppe von Dateien, auf die ein regulärer Ausdruck passt

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, .htaccess (All) |
| Syntax | <FilesMatch RegExp> ... </Files> |

<FilesMatch> hat eine ähnliche Aufgabe wie <Files>, ist aber flexibler, weil Sie einen regulären Ausdruck angeben, der eine Reihe von Dateinamen beschreibt. Eine alternative Formulierung ist <Files ~ RegExp>. Das folgende Beispiel beschreibt Text-, HTML- und XML-Dateien:

```
<FilesMatch "\.(txt|html?|xml)$">
  ...
</FilesMatch>
```

### 6.2.4 Spezial-Container

Die Container in diesem Abschnitt ermöglichen es, Konfigurationsdirektiven von bestimmten Bedingungen abhängig zu machen.

#### If, ElseIf, Else

Container für Direktiven, die beliebigen zu überprüfenden Bedingungen genügen

| Seit Version | 2.3-beta |
|---|---|
| Modul | core |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, .htaccess (All) |
| Syntax | <If Ausdruck> ... </If><br>[<ElseIf Ausdruck> ... </ElseIf> ...]<br>[<Else> ... </Else>] |

Die in Version 2.3-beta neu eingeführten Container <If>, <ElseIf> und <Else> ermöglichen es, einen Konfigurationsabschnitt von beliebigen Bedingungen abhängig zu machen. Zur Formulierung der Bedingungen werden Ausdrücke verwendet, die durch den neuen Ausdrucks-Parser *ap_expr* interpretiert werden. Es ist davon auszugehen, dass auch andere Teile der Apache-Konfiguration, die Ausdrücke verarbeiten, nach und nach auf die allgemeingültige ap_expr-Syntax umgestellt werden. ap_expr-Ausdrücke können aus folgenden Elementen bestehen:

- *Literale*
  »Wörtlich gemeinte« Werte. String-Literale stehen in einfachen Anführungszeichen, zum Beispiel `'hello world'`. Numerische Literale werden ohne Kennzeichnung geschrieben, etwa `404`.
- *Variablen*
  Variablen werden im Format `%{Variablenname}` angegeben. In Tabelle 6.1 finden Sie eine Liste der in Ausdrücken unterstützten Variablen. Weitere Informationen über CGI-Umgebungsvariablen erhalten Sie in Kapitel 14, »CGI«. Außerdem gibt es zahlreiche Sondervariablen für SSL, die in Kapitel 10, »Gesicherte Verbindungen«, aufgelistet werden.

  Beispiel: `%{REQUEST_METHOD}` gibt die verwendete HTTP-Anfragemethode an, etwa `GET` oder `POST`.

| Variable | Bedeutung | Beispielwert |
|---|---|---|
| `HTTP_ACCEPT` | HTTP-Header `Accept` (vom Client akzeptierte MIME-Types) | `text/html, image/*, */*` |
| `HTTP_FORWARDED` | HTTP-Header `X-Forwarded-For` (vom Proxy gesetzte Hostnamen/IPs der Original-Clients) | `192.168.0.7, testclient` |
| `HTTP_HOST` | HTTP-Header `Host` (angesprochener Server-Hostname) | `static.mynet.de` |
| `HTTP_PROXY_CONNECTION` | HTTP-Erweiterungs-Header `Proxy-Connection` (Hostnamen/IPs von Proxies) | `192.168.0.29, 10.21.16.53` |
| `HTTP_REFERER` | HTTP-Header `Referer` (zuvor besuchte URL) | `http://www.mynet.de/info.html` |
| `HTTP_USER_AGENT` | HTTP-Header `User-Agent` (Identifikation des Clients) | `Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:9.0) Gecko/20100101 Firefox/9.0` |
| `REQUEST_METHOD` | HTTP-Anfragemethode | `GET` |
| `REQUEST_SCHEME` | Schema-Teil der HTTP-Anfrage | `http` |
| `REQUEST_URI, DOCUMENT_URI` | Pfad-Teil der HTTP-Anfrage | `/info.html` |

**Tabelle 6.1** Variablen zur Verwendung in ap_expr-Ausdrücken

| Variable | Bedeutung | Beispielwert |
|---|---|---|
| REQUEST_FILENAME, SCRIPT_FILENAME | Absoluter Pfad der angeforderten Ressource (falls verfügbar, ansonsten wie REQUEST_URI) | /var/www/htdocs/info.html |
| LAST_MODIFIED | Änderungsdatum und -uhrzeit der Datei im Format YYYYMMDD hhmmss | 20111012172152 |
| SCRIPT_USER | Eigentümer des Scripts | www-data |
| SCRIPT_GROUP | Gruppe des Scripts | www-data |
| PATH_INFO | PathInfo hinter dem Dateinamen der URL gem. AcceptPathInfo | /extra/path/info |
| QUERY_STRING | Query-String der Anfrage | ?user=klaus&passwd=geheim |
| IS_SUBREQ | Interne Unteranfrage? | true/false |
| THE_REQUEST | Komplette HTTP-Anfragezeile | GET/HTTP/1.1 |
| REMOTE_ADDR | IP-Adresse des Clients | 192.168.0.2 |
| REMOTE_HOST | Hostname des Clients | client0815.provider.com |
| REMOTE_USER | Username des authentifizierten Users, falls vorhanden | schmitz |
| REMOTE_IDENT | identd-Username, falls vorhanden | mueller |
| SERVER_NAME | Server-Name des aktuellen Servers/VHosts | www2.mynet.de |
| SERVER_PORT | TCP-Port des aktuellen Servers/VHosts | 80 |
| SERVER_ADMIN | ServerAdmin des aktuellen Servers/VHosts | admin@mynet.de |
| SERVER_PROTOCOL | Protokoll der Anfrage | HTTP/1.1 |
| DOCUMENT_ROOT | DocumentRoot des aktuellen Servers/VHosts | /var/www/static |
| AUTH_TYPE | Aktuelle AuthType-Einstellung (siehe Kapitel 9, »Auhentifizierung, Autorisierung und Zugriffskontrolle«) | basic |

**Tabelle 6.1** Variablen zur Verwendung in ap_expr-Ausdrücken (Forts.)

| Variable | Bedeutung | Beispielwert |
|---|---|---|
| CONTENT_TYPE | MIME-Type der Antwort | text/html |
| HANDLER | Handler, der die Antwort erzeugt (siehe Kapitel 7, »Header und MIME-Einstellungen«) | default-handler |
| HTTPS | Wird HTTPS verwendet? | on |
| IPV6 | Wird IPv6 verwendet? | off |
| REQUEST_STATUS | HTTP-Status der Anfrage | 200 |
| REQUEST_LOG_ID | ErrorLog-ID der Anfrage (siehe Kapitel 11, »Logging«) | 27 |
| CONN_LOG_ID | ErrorLog-ID der Verbindung (siehe Kapitel 11, »Logging«) | 45 |
| TIME_YEAR | Aktuelles Jahr | 2011 |
| TIME_MON | Aktueller Monat (1–12) | 10 |
| TIME_DAY | Aktueller Tag im Monat (1–31) | 13 |
| TIME_HOUR | Aktuelle Stunde (0–23) | 19 |
| TIME_MIN | Aktuelle Minute (0–59) | 23 |
| TIME_SEC | Aktuelle Sekunde (0–59) | 31 |
| TIME_WDAY | Aktueller Wochentag (0 = Sonntag bis 6 = Samstag) | 4 |
| TIME | Datum/Uhrzeit im Format YYYYMMDD hhmmss | 20111013192331 |
| SERVER_SOFTWARE | Versionsangabe des Servers | Apache/2.3.15 (Unix) |
| API_VERSION | Version der Server-API (Module Magic Number) | 20110724:5 |

**Tabelle 6.1** Variablen zur Verwendung in ap_expr-Ausdrücken (Forts.)

- *Operatoren*
  Für den Vergleich von Variablen und anderen Werten steht eine Reihe verschiedener Operatoren zur Verfügung. Tabelle 6.2 zeigt eine vollständige Liste. In der Spalte »Typ« bedeutet »binär«, dass der jeweilige Operator zwischen zwei Werten steht (beispielsweise einer Variablen und einem literalen Wert), während »unär« vor einem einzelnen Wert steht und eine bestimmte Eigenschaft dieses Werts überprüft. Die zusätzliche Eigenschaft »eingeschränkt« bedeutet, dass der Operator aus Sicherheitsgründen in einigen Modulen wie `mod_include` (siehe Kapitel 16, »SSI und Filter«) nicht zulässig ist.

| Operator | Typ | Bedeutung |
|---|---|---|
| ==, = | binär | String-Gleichheit |
| != | binär | String-Ungleichheit |
| < | binär | String kleiner als |
| <= | binär | String kleiner/gleich |
| > | binär | String größer als |
| >= | binär | String größer/gleich |
| -eq, eq | binär | Integer-Gleichheit |
| -ne, ne | binär | Integer-Ungleichheit |
| -lt, lt | binär | Integer kleiner als |
| -le, le | binär | Integer kleiner/gleich |
| -gt, gt | binär | Integer größer als |
| -ge, ge | binär | Integer größer/gleich |
| -ipmatch | binär | IP-Adresse entspricht Adresse/Netzmaske. |
| -strmatch | binär | String entspricht Muster mit Platzhaltern (*, ?, []; siehe z.B. <Directory>). |
| -strcmatch | binär | wie -strmatch ohne Unterscheidung von Groß-/Kleinschreibung |
| -fnmatch | binär | wie -strmatch, aber Platzhalter passen nicht auf / |
| -in | binär | String ist in der angegebenen Liste enthalten. Die Liste hat das Format {Wort1, ..., WortN}. |
| =~ | binär | String entspricht regulärem Ausdruck (Näheres siehe unten). |
| && | binär | Logisches Und zur Verknüpfung von Ausdrücken – der Gesamtausdruck trifft zu, wenn alle Teilausdrücke zutreffen. |
| \|\| | binär | Logisches Oder zur Verknüpfung von Ausdrücken – der Gesamtausdruck trifft zu, wenn mindestens ein Teilausdruck zutrifft. |
| -d | unär, eingeschränkt | Trifft zu, wenn das Argument ein Verzeichnis ist. |
| -e | unär, eingeschränkt | Trifft zu, wenn das Argument im Dateisystem existiert (Datei oder Verzeichnis). |

**Tabelle 6.2** Mögliche Variablen in ap_expr-Ausdrücken

| Operator | Typ | Bedeutung |
|---|---|---|
| `-f` | unär, eingeschränkt | Trifft zu, wenn das Argument eine reguläre Datei ist. |
| `-L`, `-h` | unär, eingeschränkt | Trifft zu, wenn das Argument ein Symlink ist. |
| `-F` | unär, eingeschränkt | Trifft zu, wenn das Argument eine gültige Datei ist, die mit den aktuellen Zugriffseinstellungen erreichbar ist (Achtung: Interne Unterabfrage, kann die Performance beeinträchtigen). |
| `-U`, `-A` | unär | Trifft zu, wenn das Argument eine gültige URL ist, die mit den aktuellen Zugriffseinstellungen erreichbar ist (Achtung: Interne Unterabfrage, kann die Performance beeinträchtigen). |
| `-n` | unär | String ist nicht leer. |
| `-z` | unär | String ist leer. |
| `-T` | unär | False, wenn der String leer, `"0"`, `"off"`, `"false"` oder `"no"` ist (Groß-/Kleinschreibung wird nicht beachtet); ansonsten true. |
| `-R` | unär | Entspricht `"%{REMOTE_ADDR} -ipmatch ..."`. |

**Tabelle 6.2** Mögliche Variablen in ap_expr-Ausdrücken (Forts.)

- *Funktionen*
  Funktionen werden in der Form `Funktionsname(Argument)` aufgerufen. Tabelle 6.3 enthält eine Liste aller standardmäßig vorhandenen Funktionen; Module können jedoch zusätzliche Funktionen definieren (`mod_ssl` tut dies beispielsweise). Die Funktionen `file` und `filesize` sind genauso eingeschränkt wie einige oben erwähnte Operatoren; sie stehen also aus Sicherheitsgründen nicht für `mod_include` und ähnliche Module zur Verfügung.

  Soweit ihre Rückgabewerte passen, können Funktionen beliebig ineinander verschachtelt werden. Beispielsweise liest `toupper(req('Host'))` den Anfrage-Header `Host` aus und wandelt ihn in Großbuchstaben um – aus `static.mynet.de` wird so beispielsweise `STATIC.MYNET.DE`.

| Funktion | Bedeutung |
|---|---|
| `req`, `http` | HTTP-Anfrage-Header |
| `resp` | HTTP-Antwort-Header |

**Tabelle 6.3** Standardfunktionen in ap_expr-Ausdrücken

| Funktion | Bedeutung |
|---|---|
| reqenv | Anfrage-Umgebungsvariable |
| osenv | Betriebssystem-Umgebungsvariable |
| note | RequestNote (Anmerkung zur Anfrage) |
| env | Durchsucht note, reqenv und osenv in dieser Reihenfolge und liefert den ersten Treffer. |
| tolower | String komplett in Kleinschreibung umwandeln |
| toupper | String komplett in Großschreibung umwandeln |
| escape | Sonderzeichen in %hex-Darstellung umwandeln |
| unescape | Sonderzeichen aus der %hex-Darstellung in Standarddarstellung umwandeln |
| file | Inhalt der angegebenen Datei auslesen |
| filesize | Dateigröße der angegebenen Datei (0, wenn die Datei nicht existiert oder keine reguläre Datei ist) |

**Tabelle 6.3** Standardfunktionen in ap_expr-Ausdrücken (Forts.)

- *Reguläre Ausdrücke*
  Komplexe Suchmuster, die im Format /muster/ oder m#muster# geschrieben werden. Mit vorangestelltem m sind auch andere Trennzeichen als / oder # möglich, beispielsweise Klammernpaare – Sie können sich so ein Trennzeichen aussuchen, das im regulären Ausdruck selbst nicht vorkommt, und brauchen dieses dann nicht zu escapen. Wenn Sie hinter dem schließenden Trennzeichen ein i setzen, wird beim Mustervergleich nicht zwischen Groß- und Kleinschreibung unterschieden. /[A-Za-z]+/ und m#[a-z]+#i stehen beispielsweise beide für einen oder mehrere Buchstaben in Groß- und/oder Kleinschreibung.

  Näheres über reguläre Ausdrücke erfahren Sie beispielsweise in der Beschreibung der Direktive <DirectoryMatch> oder in Kapitel 8, »Weiterleitungen und Indizes«, in der Dokumentation des Moduls mod_rewrite.

  Wenn innerhalb des regulären Ausdrucks Bereiche in runden Klammern stehen, werden die entsprechenden Teiltreffer in den Spezialvariablen $1 bis $9 zwischengespeichert, und $0 steht für den gesamten Treffer. In der Regel können Sie diese Variablen nur im selben Gesamtausdruck verwenden, in dem auch der entsprechende reguläre Ausdruck steht; eine Ausnahme bilden die Direktiven von mod_rewrite.

Innerhalb von Ausdrücken können Teilausdrücke in (runde) Klammern gesetzt werden, beispielsweise um eine bestimmte Rangfolge von Operatoren zu erzwingen.

Der zu prüfende Ausdruck steht in aller Regel in doppelten Anführungszeichen, weil er typischerweise Leerzeichen enthält.

Die Direktive `<If>` überprüft bei jeder Anfrage im entsprechenden Kontext den angegebenen Ausdruck, und wenn er zutrifft, werden die umschlossenen Direktiven für diese Anfrage beachtet. Auf `<If>` können beliebig viele `<ElseIf>`-Container folgen. Diese werden nur dann geprüft, wenn das vorherige `<If>` beziehungsweise `<ElseIf>` nicht zutrifft. Am Ende der Kette kann ein `<Else>`-Container folgen, der keinen Ausdruck enthält und beachtet wird, wenn keiner der `<If>`- oder `<ElseIf>`-Ausdrücke zutrifft.

Das folgende Beispiel untersucht den Beginn des Anfrage-Headers `Accept-Language` (die vom User-Agent bevorzugten Sprachen) mithilfe eines regulären Ausdrucks und fügt den Antwort-Header `X-Greeting` mit einem jeweils passenden Wert hinzu:

```
<If "req('Accept-Language') =~ /^de/">
  Header set X-Greeting "Guten Tag!"
</If>
<ElseIf "req('Accept-Language') =~ /^fr/">
  Header set X-Greeting "Bonjour!"
</ElseIf>
<Else>
  Header set X-Greeting "Hello!"
</Else>
```

Ein passender Ort für dieses Beispiel ist der `<Directory>`-Container für die `DocumentRoot`. Dass es funktioniert, können Sie nach einem Neustart von Apache mit dem Hilfsprogramm `curl` testen:

```
# curl -I -H "Accept-Language: de" http://localhost/
```

gibt die Header der Server-Antwort aus; in diesem Fall unter anderem:

```
X-Greeting: Guten Tag!
```

Das Beispiel mag recht sinnlos erscheinen, zeigt aber, wie mächtig die neuen Bedingungs-Direktiven sind, und es bietet einen Einstieg in das komplexe Thema Content-Negotiation (Beachten von Client-Vorlieben bei der Auslieferung von Content), das im nächsten Kapitel genau beschrieben wird.

#### IfDefine

Container für Direktiven, die über einen Kommandozeilenparameter beim Apache-Start aktiviert werden

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, .htaccess (All) |
| Syntax | <IfDefine [!]Wert> ... </IfDefine> |

<IfDefine> kann innerhalb eines beliebigen Kontextes der Konfigurationsdatei stehen. Die enthaltenen Konfigurationsanweisungen werden nur dann beachtet, wenn beim Start von Apache die Kommandozeilenoption -D mit dem entsprechenden Wert gesetzt beziehungsweise nicht gesetzt wird. Das folgende Beispiel setzt die globale Direktive LogLevel (Dringlichkeitsgrad, ab dem Vorgänge in das ErrorLog geschrieben werden; siehe Kapitel 11, »Logging«) auf debug, falls beim Start der Wert Debug eingestellt wurde, und ansonsten auf error:

```
<IfDefine Debug>
  LogLevel debug
</IfDefine>

<IfDefine !Debug>
  LogLevel error
</IfDefine>
```

Um Apache gewissermaßen im »Debug-Modus« zu starten, müssen Sie in der Kommandozeile den folgenden Befehl eingeben:

```
# httpd -DDebug
```

Für die Schreibweise unter Windows sowie für einen Neustart des laufenden Servers gelten die Informationen aus Kapitel 5, »Apache in Betrieb nehmen«.

**Hinweis**

Die Parameter oder Variablen, die für <IfDefine> und die nachfolgend beschriebenen neuen Direktiven Define und UnDefine verwendet werden, sind nicht dasselbe wie Umgebungsvariablen. Letztere werden in Kapitel 14, »CGI«, beschrieben.

### Define

Variablen definieren

| Seit Version | 2.3 |
|---|---|
| Modul | core |
| Kontext | Server, <VirtualHost>, <Directory>, <Location> |
| Syntax | Define Variable [Wert] |

Mit Define können Sie Variablen definieren, die an anderer Stelle durch <IfDefine>-Container abgefragt werden können. Die Definitionen sind äquiva-

lent zum Starten von Apache mit dem Kommandozeilenargument `-D`. Der Variablenname darf keinen Doppelpunkt enthalten, weil solche Namen von `RewriteMap` (siehe Kapitel 8, »Weiterleitungen un Indizes«) verwendet werden. Der Wert ist optional; wenn Sie ihn weglassen, ist die Variable einfach »gesetzt«.

Eine einmal definierte Variable ist global gültig; die Definition ist nicht auf den Konfigurationskontext beschränkt, in dem `Define` steht. Neben der Verwendung in `<IfDefine>` können Variablen auch in der Form `${Variable}` in Ausdrücken verwendet werden; siehe die Beschreibung der Direktive `<If>`. Beispiel:

```
Define Debug 1
```

#### UnDefine

Variablendefinitionen aufheben

| Seit Version | 2.3 |
|---|---|
| Modul | `core` |
| Kontext | Server, `<VirtualHost>`, `<Directory>`, `<Location>` |
| Syntax | `UnDefine Variable` |

Diese Direktive hebt eine zuvor mittels `Define` oder dem Kommandozeilenparameter `-D` erzeugte Variablendefinition wieder auf. Beispiel:

```
UnDefine Debug
```

#### IfModule

Container für Direktiven, die von einem Modul abhängen

| Modul | `core` |
|---|---|
| Kontext | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `.htaccess` (`All`) |
| Syntax | `<IfModule [!]Modul>` ... `</IfModule>` |

Direktiven innerhalb eines `<IfModule>`-Containers werden nur dann beachtet, wenn das angegebene Modul aktiviert beziehungsweise nicht aktiviert ist. Für das Modul muss entweder der bei `LoadModule` (siehe dort) verwendete kanonische Name (z.B. `dir_module` für das Modul `mod_dir`) oder aber der Name seiner C-Quellcode-Datei angegeben werden – der Grund dafür ist, dass dies die einzigen Möglichkeiten sind, für statisch einkompilierte und für DSO-Module dieselbe Bezeichnung zu verwenden. Für komplexere Module, die aus mehreren Quelldateien bestehen, müssen Sie die »Hauptdatei« verwenden. Es handelt sich um diejenige, in der die symbolische Konstante `STANDARD20_MODULE_STUFF` verwendet wird.

Dieser Container kann unter anderem für den Import externer Konfigurationsdateien für spezielle Module verwendet werden. Das folgende Beispiel stammt aus der vorgefertigten Konfigurationsdatei einer Apache-Installation:

```
<IfModule mod_ssl.c>
  Include conf/ssl.conf
</IfModule>
```

Falls das Modul `mod_ssl` aktiv ist, wird die ausgelagerte Datei *ssl.conf* mit Einstellungen für den SSL-Betrieb eingefügt. Hier die entsprechende Schreibweise mit dem kanonischen Namen, die in der mitgelieferten Konfigurationsdatei von Apache 2.2 bevorzugt wird:

```
<IfModule ssl_module>
  Include conf/ssl.conf
</IfModule>
```

#### Error

Verarbeitung der Konfiguration mit einer Fehlermeldung abbrechen

| | |
|---|---|
| **Seit Version** | 2.3.9 |
| **Modul** | `mod_cgi`, `mod_cgid` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`All`) |
| **Syntax** | Error Fehlermeldung |
| **Standardwert** | nicht gesetzt |

Diese Direktive bricht die Verarbeitung der Konfiguration mit einer benutzerdefinierten Fehlermeldung ab. Der wichtigste Einsatzzweck ist der Hinweis auf benötigte, aber nicht geladene Module, so dass `Error` vor allem zusammen mit `<IfModule>` verwendet werden kann. Das folgende Beispiel bricht die Verarbeitung ab, wenn `mod_ssl` nicht geladen wurde:

```
<IfModule !ssl_module>
  Error "mod_ssl wird benötigt. LoadModule hinzufügen!"
</IfModule>
```

#### Limit

Beschränkt Konfigurationseinstellungen auf einzelne HTTP-Anfragemethoden.

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `.htaccess` (`All`) |
| **Syntax** | `<Limit Methode [Methode ...]>` ... `</Limit>` |

Der Container <Limit> ermöglicht es, in einem beliebigen Kontext Konfigurationsdirektiven zusammenzufassen, die nur für Client-Anfragen über bestimmte HTTP-Zugriffsmethoden gelten. Ein gutes Beispiel ist ein besonders starker Schutz für PUT-Anfragen, falls Sie diese zulassen möchten. Das folgende Beispiel erlaubt die Methode PUT nur von IP-Adressen im Netzwerk 192.168.0.0/24 aus – dieser private Adressraum wird häufig für kleinere LANs genutzt:

```
<Limit PUT>
  Order deny,allow
  Deny from All
  Allow from 192.168.0
</Limit>
```

### LimitExcept

Beschränkt Konfigurationseinstellungen auf alle HTTP-Anfragemethoden außer den genannten.

| | |
|---|---|
| **Modul** | core |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, .htaccess (All) |
| **Syntax** | <LimitExcept Methode [Methode ...]> ... </LimitExcept> |

Dieser Container kehrt die Funktionsweise von <Limit> um: Die enthaltenen Direktiven gelten für alle Client-Anfragemethoden außer den aufgeführten. Auf diese Weise lässt sich das vorangegangene Beispiel noch restriktiver formulieren – alle Methoden außer POST und GET müssen aus dem lokalen Netz stammen:

```
<LimitExcept POST GET>
  Order deny,allow
  Deny from All
  Allow from 192.168.0
</LimitExcept>
```

### IfVersion

Die Gültigkeit von Direktiven auf bestimmte Versionen beschränken

| | |
|---|---|
| **Seit Version** | 2.0.56 und 2.1.0 |
| **Modul** | mod_version |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, .htaccess (All) |
| **Syntax** | <IfVersion [[!]Operator] Version> ... </IfVersion> |

Das Modul `mod_version` wurde für den neuen Versionszweig 2.2 (bereits ab 2.1-beta) entwickelt, steht aber seit Release 2.0.56 auch in der vorangegangenen Version zur Verfügung (ein sogenannter *Back-Port*). Das Modul definiert ausschließlich den Container `<IfVersion>`. Die darin enthaltenen Direktiven werden nur ausgeführt, wenn die laufende Apache-Version dem angegebenen Wert entspricht.

Der Operator kann einen der folgenden Werte besitzen:

- =, == oder ohne Operator – die Konfigurationseinstellungen gelten, wenn Apache genau die angegebene Version hat. Beispiel:

```
<IfVersion = 2.2.6>
  ...
</IfVersion>
```

- < – die Direktiven werden nur beachtet, falls der laufende Server älter ist als die Versionsnummer:

```
<IfVersion < 2.2.9>
  ...
</IfVersion>
```

- <= – die verschachtelten Einstellungen sind gültig, wenn eine Apache-Version bis einschließlich der angegebenen läuft. Hier ein Beispiel:

```
<IfVersion <= 2.2.6>
  ...
</IfVersion>
```

- > – diese Direktiven sollen nur ausgewertet werden, wenn die Versionsnummer höher ist als angegeben. Dies sieht beispielsweise so aus:

```
<IfVersion > 2.2.3>
  ...
</IfVersion>
```

- >= – hier muss Apache mindestens die angegebene Versionsnummer haben. Beispiel:

```
<IfVersion >= 2.2.0>
  ...
</IfVersion>
```

- ~ – die Direktiven sind gültig, wenn die Versionsangabe dem angegebenen regulären Ausdruck entspricht. Das folgende Beispiel gilt für Versionen ab 2.0.56 oder 2.2.7 bis 2.2.9:

```
<IfVersion ~ (2\.0\.56|2\.2\.([789]|\d\d+)>
  ...
</IfVersion>
```

Eine alternative Schreibweise für `~ RegExp` ist übrigens `= /RegExp/`.

Optional können Sie jedem Operator ein Ausrufezeichen voranstellen, um seine Bedeutung umzukehren. Dieses Beispiel gilt für alle Versionen außer 2.2.8:

```
<IfVersion != 2.2.8>
  ...
</IfVersion>
```

**Verschachtelung von Containern**

Beachten Sie, dass nicht jeder Container in jeden anderen verschachtelt werden darf. Für die Verschachtelungsreihenfolge gelten folgende Regeln:

- Ein `<VirtualHost>`-Container darf nur im Server-Kontext stehen und nicht in andere Container hineinverschachtelt werden.
- <Directory>-Container dürfen keine `<Location>`-Container enthalten und umgekehrt.
- In `<Files>`-Container dürfen nur noch `<Limit>`-Container verschachtelt werden.
- `<Limit>`-Container bilden stets die innerste Verschachtelung und dürfen keine anderen Container mehr enthalten.

Diese Regeln gelten auch für die jeweiligen Varianten wie `<DirectoryMatch>` oder `<LimitExcept>`.

### 6.2.5 .htaccess-Dateien

Alle bis jetzt vorgestellten Kontexte und Container befinden sich typischerweise in der zentralen Konfigurationsdatei *httpd.conf* oder in zusätzlichen Dateien, die über `Include`-Direktiven eingebunden wurden. *.htaccess*-Dateien bieten zusätzliche Flexibilität: Wenn sich in einem Verzeichnis unterhalb der `DocumentRoot` eine solche Datei befindet, werden die darin befindlichen Konfigurationsanweisungen für dieses Verzeichnis und alle Unterverzeichnisse beachtet. *.htaccess* ist der vorgegebene Name der Datei; er kann über die im Verlauf dieses Kapitels vorgestellte Direktive `AccessFileName` geändert werden. *.htaccess*-Dateien sind nur in Verzeichnissen (und deren Unterverzeichnissen) zulässig, für die die Konfigurationsanweisung `AllowOverride` gesetzt wurde. Diese bestimmt nicht nur grundsätzlich, ob innerhalb der Verzeichnisse des betreffenden Kontextes *.htaccess*-Dateien erlaubt sind, sondern auch, welche Arten von Einstellungen darin überschrieben werden dürfen. In der Referenz jeder Direktive in diesem Buch, die überhaupt in einer *.htaccess*-Datei stehen darf, steht der zugehörige Wert für `AllowOverride` in Klammern hinter der Kontextangabe *.htaccess*.

Aus Sicherheitsgründen sollten *.htaccess*-Dateien zunächst einmal generell verboten werden; später können Sie sie dann für einzelne Verzeichnisse gestatten. Deshalb sollte die *httpd.conf* auf jeden Fall folgenden Abschnitt enthalten (bei der vorgefertigten Variante einer normalen Apache-Installation ist er bereits vorhanden):

```
<Directory />
  AllowOverride None
  # ... hier weitere Voreinstellungen für alle Verzeichnisse
</Directory>
```

In den meisten Fällen ist es übrigens gar keine so gute Idee, mit *.htaccess*-Dateien zu arbeiten. Der wichtigste Grund dafür ist, dass sich diese nicht – wie die *httpd.conf* und eventuell zusätzlich importierte Konfigurationsdateien – im zentralen *conf*-Verzeichnis der Installation befinden, sondern in den entsprechenden Verzeichnissen selbst. Der ausufernde Einsatz führt auf Dauer zu einem heillosen Chaos, weil Sie nicht mehr durchblicken werden, wo überall Konfigurationseinstellungen stehen und welche letztendlich gültig sind.

Abgesehen davon, werden *.htaccess*-Dateien bei jeder einzelnen Client-Anfrage auf das entsprechende Verzeichnis interpretiert, was Performance-Nachteile mit sich bringt (nur ein generelles `AllowOveride No` verhindert dabei, dass überhaupt nach den Dateien gesucht wird). Andererseits sorgt dieses Verhalten dafür, dass Änderungen in *.htaccess*-Dateien ohne Neustart des Servers sofort wirksam werden.

Zahlreiche Anleitungen für nebenberufliche Webmaster, die im Internet zu finden sind, erwecken den Eindruck, *.htaccess*-Dateien seien nichts weiter als ein Ablageort für Zugriffsbeschränkungen und Authentifizierungseinstellungen. Aber erstens können in *.htaccess* je nach `AllowOverride`-Einstellung auch andere Direktiven stehen, und zweitens ist es überhaupt kein Problem (und in der Regel empfehlenswerter), Zugriffs- und Authentifizierungsoptionen direkt in die Datei *httpd.conf* zu schreiben.

Ein weiteres Problem besteht darin, dass sich die Dateien innerhalb der `DocumentRoot` befinden und deshalb standardmäßig mit denselben Berechtigungen öffentlich verfügbar sind wie die Verzeichnisse, in denen sie liegen. Das ist für die im Zusammenhang mit der Authentifizierung verwendeten *.htpasswd*-Dateien, die Benutzernamen und Passwörter enthalten, sogar noch gefährlicher. Deshalb sollte die Hauptkonfigurationsdatei immer folgenden Abschnitt enthalten, um den Zugriff auf diese Dateien zu verhindern:

```
<FilesMatch "^\.ht">
  Order deny,allow
  Deny from All
</FilesMatch>
```

Es gibt ein Szenario, in dem *.htaccess*-Dateien sogar sehr wichtig sind: Kommerzielle Webhoster, bei denen die Kunden-Websites als virtuelle Hosts auf den Server-Rechnern liegen, können ihren Kunden selbstverständlich keinen Zugriff auf die zentrale Konfigurationsdatei erlauben. Die Konfiguration einer solchen Site oder einzelner Verzeichnisse darin ist somit nur über *.htaccess* möglich.

Daneben sind *.htaccess*-Dateien für das Ausprobieren bestimmter Konfigurationsdirektiven nützlich: Da sie bei der Verarbeitung jeder einzelnen Anfrage neu eingelesen werden, braucht der Server nach einer Änderung in einer solchen Datei nicht neu gestartet zu werden. Wenn die getestete Konfiguration dann funktioniert, sollten Sie sie aber in den meisten Fällen in die Hauptkonfigurationsdatei verschieben und Apache neu starten.

Zur Auswertung der Einstellungen in *.htaccess*-Dateien ist noch anzumerken, dass die Einstellungen einer solchen Datei in einem übergeordneten Verzeichnis stets von einer *.htaccess*-Datei in einem untergeordneten Verzeichnis überschrieben werden.

### 6.2.6 Einfügen externer Konfigurationsdateien

Wie bereits erwähnt, ist es möglich, externe Konfigurationsdateien zu importieren. Dafür ist die Direktive `Include` zuständig.

#### Include

Einfügen externer Konfigurationsdateien

| | |
|---|---|
| **Seit Version** | 1.3; Platzhalter seit 2.0.41 |
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `Include Dateiname\|Muster` |
| **Standardwert** | nicht gesetzt |

Mithilfe dieser Direktive wird eine zusätzliche Konfigurationsdatei importiert. Sie können entweder einen absoluten oder einen relativen Pfad angeben; Letzterer bezieht sich wie üblich auf die `ServerRoot`. In neueren Apache-Versionen ist es auch möglich, ein einfaches Dateimuster mit Platzhaltern anzugeben (siehe Beschreibung von `<Directory>`). In diesem Fall werden die Dateien in alphabetischer Reihenfolge ausgewertet. Wenn die Pfadangabe ein Verzeichnis ist, werden sogar alle Dateien in diesem Verzeichnis eingelesen. Zu empfehlen ist dies aber nicht, da die Gefahr groß ist, dass in dem Verzeichnis eine andere Datei landet, die nicht der Syntax einer Apache-Konfigurationsdatei entspricht. Der Server kann in diesem Fall nicht starten.

Die Konfigurationsanweisungen der importierten Datei gelten in dem Kontext, in dem die `Include`-Anweisung steht, und müssen entsprechend in diesem Kontext erlaubt sein. Beispiele:

```
Include conf/ssl.conf
Include conf/dir*.conf
Include /usr/local/apache2/conf/dirs.conf
```

Auf UNIX-Systemen können Sie das Skript `apachectl` (siehe Kapitel 5, »Apache in Betrieb nehmen«) mit der Option `configtest` aufrufen, um eine Liste aller externen Konfigurationsdateien zu erhalten, die importiert werden:

```
# apachectl configtest
Processing config file: /usr/local/apache2/conf/ssl.conf
Processing config file: /usr/local/apache2/conf/special.conf
```

## 6.3 Allgemeine Konfigurationsdirektiven

In diesem Abschnitt werden die Direktiven behandelt, die für den grundlegenden Betrieb von Apache und einer einfachen Website aus statischen Dokumenten wichtig sind. Es handelt sich ausschließlich um solche Konfigurationsanweisungen, die zum Kern des Servers (`core`) gehören oder in MPM-Modulen beziehungsweise in solchen Modulen verfügbar sind, die standardmäßig zu Apache gehören und aktiviert sind.

### 6.3.1 Einrichten der Server-Umgebung

Wie bereits erwähnt, bilden die grundlegenden Programmeinstellungen den ersten Abschnitt der Datei *httpd.conf*. In der automatisch generierten Konfigurationsdatei einer normalen Apache-Installation bis 2.0 wurde der Abschnitt durch folgenden Kommentar charakterisiert:

```
### Section 1: Global Environment
#
# The directives in this section affect the overall operation
# of Apache, such as the number of concurrent requests it can
# handle or where it can find its configuration files.
```

Hier wird beispielsweise das grundlegende Verzeichnis wichtiger Server-Ressourcen (`ServerRoot`) festgelegt. Außerdem werden hier die DSO-Module geladen, und das grundlegende Verhalten des Servers bei der Verarbeitung von Anfragen wird konfiguriert. Die Direktiven werden hier in der Reihenfolge behandelt, in der sie in der Standarddatei stehen.

### ServerRoot

Stammverzeichnis der Apache-Installation; enthält unter anderem die Verzeichnisse *conf/* und *logs/*

| Modul | core |
|---|---|
| Kontext | Server |
| Syntax | ServerRoot Pfad |
| Standardwert | PREFIX der Installation |

Zahlreiche Pfade in der Apache-Konfiguration können relativ zu dem hier eingestellten Verzeichnis angegeben werden. Insbesondere die Verzeichnisse für die Konfigurationsdateien (*conf/*) und die Log-Dateien (*logs/*) müssen sich normalerweise unterhalb des hier eingestellten Verzeichnisses befinden. Welcher Wert hier eingestellt werden muss, variiert je nach Plattform und Installationslayout. In Kapitel 4, »Apache kompilieren und installieren«, finden Sie die passenden Angaben für verschiedene UNIX-Systeme und für Windows: Auf UNIX-Rechnern handelt es sich um das beim Kompilieren angegebene PREFIX, bei Windows um das übergeordnete Verzeichnis der Apache-Installation. Beispiele:

```
ServerRoot /usr/local/httpd                        # UNIX
ServerRoot "C:/Programme/Apache Group/Apache2"     # Windows
```

Beachten Sie, dass der Wert dieser Direktive (im Gegensatz zu zahlreichen anderen Konfigurationsanweisungen, deren Wert ein Verzeichnis ist) keinen abschließenden Slash enthalten darf.

### TimeOut

Wartezeit, bis die Verarbeitung einer Anfrage abgebrochen wird

| Modul | core |
|---|---|
| Kontext | Server |
| Syntax | TimeOut Sekunden |
| Standardwert | 300 |

Die Direktive TimeOut legt die Wartezeit für drei verschiedene Aspekte einer HTTP-Transaktion fest:

- die maximale Gesamtdauer einer GET-Anfrage
- die maximale Wartezeit zwischen zwei TCP-Paketen einer POST- oder PUT-Anfrage
- die maximale Wartezeit zwischen zwei Empfangsbestätigungen (ACK) für TCP-Pakete der Server-Antwort

In der Regel gibt es keinen Grund, an den (recht großzügig) voreingestellten 300 Sekunden etwas zu ändern. Für zukünftige Apache-Versionen ist allerdings geplant, die drei Werte getrennt konfigurierbar zu machen. Siehe auch »RequestReadTimeout« auf Seite 259.

#### KeepAlive

Persistente Verbindungen gemäß HTTP/1.1-Spezifikation

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `KeepAlive On\|Off` |
| **Standardwert** | `On` |

Diese Direktive regelt, ob Apache 2 grundsätzlich persistente HTTP-Verbindungen zulässt, also mehrere Anfragen über ein und dieselbe Verbindung abhandelt. Clients, die HTTP/1.1 verwenden, erwarten standardmäßig eine persistente Verbindung, HTTP/1.0-Clients dagegen nur, wenn sie den Header `Connection: Keep-Alive` senden. Es gibt im Allgemeinen keinen Grund, `KeepAlive` grundsätzlich abzuschalten. Für einzelne Clients, die damit Schwierigkeiten haben, können Sie über eine `BrowserMatch`-Direktive die Option `nokeepalive` setzen (siehe Kapitel 14, »CGI«).

#### MaxKeepAliveRequests

Zulässige Höchstzahl von Anfragen über eine persistente Verbindung

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `MaxKeepAliveRequests Anzahl` |
| **Standardwert** | `100` |

Diese Direktive legt fest, wie viele aufeinanderfolgende HTTP-Anfragen über ein und dieselbe persistente HTTP-Verbindung abgehandelt werden. Der Wert 0 steht für beliebig viele Anfragen. Er sollte nicht verwendet werden, weil er möglicherweise die Gefahr von Denial-of-Service-Attacken erhöht. Ein relativ hoher Wert ist dagegen gut für die Server-Performance, da persistente Verbindungen besonders die Auslieferung von HTML-Dokumenten mit zahlreichen eingebetteten Bildern beschleunigen. Beispiel:

```
MaxKeepAliveRequests 200
```

### KeepAliveTimeout

Maximale Zeitspanne für nachfolgende Anfragen innerhalb einer persistenten Verbindung

| | |
|---|---|
| **Seit Version** | 1.x; Zeitangaben in Millisekunden seit 2.3.2 |
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `KeepAliveTimeout Zeitwert[ms]` |
| **Standardwert** | `15` |

Mithilfe dieser Konfigurationsanweisung können Sie festlegen, wie viele Sekunden Apache eine persistente Verbindung nach dem Abschluss einer Anfrage offen halten und auf eine weitere Anfrage warten soll. Es ist ein wenig schwierig, den richtigen Wert zu finden: Die Zeitspanne darf nicht zu kurz sein, weil sie persistente Verbindungen ansonsten in der Praxis verhindert, sie sollte aber auch nicht zu lang sein, damit der Server nicht zu stark durch unnütz offen gehaltene Verbindungen belastet wird. Die Voreinstellung von 15 Sekunden ist in der Regel in Ordnung – bei einem sehr beschäftigten Server sollten Sie allerdings einen kürzeren Wert wählen.

Seit Version 2.3.2 können Sie den Zeitwert auch in Millisekunden angeben. Dazu müssen Sie hinter der Zahl das Kürzel `ms` notieren. Beispiel:

```
KeepAliveTimeout 1500ms
```

### RequestReadTimeout

Timeouts für den Empfang von Anfrage-Header und -Body von Clients

| | |
|---|---|
| **Seit Version** | 2.2.15 |
| **Modul** | `mod_reqtimeout` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `RequestReadTimeout [header=Timeout[-max.Timeout][,MinRate=Rate] [body=Timeout[-max.Timeout][,MinRate=Rate]` |
| **Standardwert** | `header=20-40,MinRate=500 body=20,MinRate=500` |

Seit Apache 2.2.15 ist das Modul `mod_reqtimeout` verfügbar, mit dem Timeouts und Mindestdatenraten für die Header und den Body einer Client-Anfrage gesetzt werden können. Das Modul definiert lediglich die Direktive `RequestReadTimeout`. Für die Header und den Body werden jeweils separate Werte angegeben. Es gibt drei mögliche Wertangaben:

- Ein numerischer Wert: Angabe des Gesamt-Timeout-Werts in Sekunden

- Zwei numerische Werte (`m-n` mit m < n): Der erste Wert gibt an, wie lange auf den Beginn der Datenübertragung gewartet wird. Findet dieser innerhalb der angegebenen Zeit statt, tritt der zweite Wert als Timeout für die gesamte Datenübertragung in Kraft.
- Zusatzangabe `MinRate=n`: Nach dem Empfang von n Byte wird der Timeout-Wert beziehungsweise der maximale Timeout-Wert um eine Sekunde erhöht.

Beispiele:

```
RequestReadTimeout header=5 body=20
```

Es wird maximal 5 Sekunden auf den Empfang des gesamten Headers und 20 Sekunden auf den Empfang des Bodys gewartet.

```
RequestReadTimeout header=5,MinRate=1000 \
body=20-40,MinRate=1000
```

Für den Header gelten 5 Sekunden, die pro 1.000 empfangenen Byte um eine Sekunde erhöht werden. Beim Body werden 20 Sekunden bis zum Beginn der Datenübertragung gewährt und insgesamt maximal 40 Sekunden; letztere werden wiederum pro 1.000 Byte um eine Sekunde erhöht.

```
RequestReadTimeout header=0 body=0
```

Deaktiviert die Funktionalität der Direktive – beispielsweise für einen VHost, bei dem Sie wissen, dass er mit besonders langsamen Clients zusammenarbeiten muss.

#### Listen

TCP-Port (und IP-Adresse), an der der Webserver lauscht

| | |
|---|---|
| **Seit Version** | 1.x; seit 2.0 vorgeschrieben; Protokoll seit 2.1.5 |
| **Modul** | beos, mpm_netware, mpm_winnt, mpmt_os2, prefork, worker, `event` |
| **Kontext** | Server |
| **Syntax** | `Listen [IP-Addr:]Port [Protokoll]` |
| **Standardwert** | `80` |

Dies ist eine der wichtigsten Konfigurationsdirektiven. Sie bestimmt, an welchen TCP-Ports der Apache-Webserver auf eingehende Verbindungen lauscht. Da die TCP/IP-Implementierung trotz der APR-Abstraktionsschicht plattformspezifisch ist, wurde diese Direktive in den MPM-Modulen und nicht im `core` implementiert.

Wird nur ein TCP-Port angegeben, dann lauscht der Server über alle Netzwerkschnittstellen an diesem Port. Wenn Sie Verbindungen nur über bestimmte Schnittstellen zulassen möchten, können Sie vor dem Port, durch einen Doppelpunkt getrennt, eine IP-Adresse angeben. Anstelle IPv4-Adressen können übri-

gens auch IPv6-Adressen verwendet werden; da diese Doppelpunkte enthalten, müssen sie zusätzlich in eckigen Klammern stehen, um sie korrekt von der Portangabe zu trennen. Wenn Ihr Server an mehreren Ports lauschen soll, müssen Sie mehrere `Listen`-Anweisungen untereinanderschreiben; die Direktive kann nur jeweils einen Wert haben. Wenn Sie ein anderes Protokoll verwenden möchten als das Standardprotokoll (zum Beispiel HTTP auf Port 80 oder HTTPS auf Port 443), dann können Sie es optional als zusätzlichen Parameter angeben.

Beachten Sie, dass `Listen` nur im Server-Kontext stehen darf. Wenn Sie virtuelle Hosts verwenden, müssen zunächst die IP-Adressen und Ports für den »Hauptserver« und alle virtuellen Hosts angegeben werden. Anschließend können Sie sie mithilfe von `<VirtualHost>`-Containern nach Wunsch aufteilen. Beispiele:

```
Listen 8080 http
Listen 196.23.17.3:80
Listen [fe80::3a0:93a2:5a5b:3ab1]:80
```

### Protocol

Protokoll für ein lauschendes Socket festlegen

| | |
|---|---|
| **Seit Version** | 2.1.5 (für Windows seit 2.3.3) |
| **Modul** | core |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `Protocol Protokoll` |
| **Standardwert** | nicht gesetzt |

Wenn Sie HTTP oder HTTPS nicht auf den Standard-Ports (80 beziehungsweise 443) betreiben, können Sie mit Hilfe dieser Direktive das Protokoll für den aktuellen Kontext angeben. Dies dient der Auswahl des passenden Moduls und der Verarbeitung protokollspezifischer Optionen durch AcceptFilter. Das folgende Beispiel stellt `https` für einen Nicht-Standard-Port ein:

```
Protocol https
```

Alternativ kann das Protokoll auch über `Listen` eingestellt werden.

### Mutex

Allgemeine Methode für alle Serialisierungsaufgaben

| | |
|---|---|
| **Seit Version** | 2.3.4 |
| **Modul** | core |
| **Kontext** | Server |

| Syntax | Mutex-Mechanismus Default\|Methode [OmitPID] |
|---|---|
| Standardwert | Default |

Manche Vorgänge in einem Webserver erfordern, dass sie nur jeweils einmal zur selben Zeit ausgeführt werden dürfen. Um dies sicherzustellen, wird ein Mutex-Verfahren verwendet (mutual exclusion, also gegenseitiger Ausschluss). Bis einschließlich Version 2.2 gibt es diverse Einzeldirektiven, die Mutex-Einstellungen für verschiedene Vorgänge vornehmen. In Version 2.3.4 wurde dann die zentrale Direktive `Mutex` eingeführt, die alle anderen Mutex-Direktiven ersetzt.

Der erste Parameter, der Mechanismus, bestimmt, für welchen der verschiedenen Anwendungsfälle Sie einen Mutex definieren möchten. Verfügbar sind zurzeit folgende (Drittanbieter-Module können jedoch zusätzliche Mechanismus-Namen definieren):

- `mpm_accept` (Module `prefork` und `worker`): Vermeidung des »Thundering-Herd«-Problems (donnernde Herde) beim Verbindungseingang; ersetzt die veraltete, im vorliegenden Kapitel beschriebene Direktive `AcceptMutex`.
- `authdigest_client` (`mod_auth_digest`): Zugriff auf die Client-Liste im Shared Memory (siehe Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«)
- `authdigest_opaque` (`mod_auth_digest`): Zugriff auf den Counter im Shared Memory (Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«)
- `ldap_cache` (`mod_ldap`): Zugriff auf den Cache für LDAP-Abfrage-Ergebnisse (Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«)
- `rewrite_map` (`mod_rewrite`): Vermeidung vermischter Ein-/Ausgabedaten beim Zugriff mehrerer Anfragen auf externe RewriteMap-Programme (Kapitel 8, »Weiterleitungen und Indizes«)
- `ssl_cache` (`mod_ssl`): Zugriff auf den SSL-Session-Cache (Kapitel 10, »Gesicherte Verbindungen«)
- `ssl_stapling` (`mod_ssl`): Zugriff auf den OCSP-Stapling-Cache (Kapitel 10, »Gesicherte Verbindungen«)
- `watchdog_callback` (`mod_watchdog`): Callback-Funktion des jeweiligen Client-Moduls (`mod_watchdog` ist ein neues Modul, das noch nicht dokumentiert wurde, und wird daher in diesem Buch noch nicht beschrieben)

Der zweite Parameter ist die Mutex-Methode. In Versionen vor 2.0 konnte diese nur einmalig bei der Kompilierung angepasst werden. Das funktioniert in Versionen ab 2.0 noch immer. Die Flags, die Sie für den jeweiligen Wert beim Aufruf des `configure`-Skripts setzen müssen, werden bei den entsprechenden Angaben für die Direktive genannt:

- `flock`: Apache verwendet den Systemaufruf `flock()`, um eine Dateisperre auf die mithilfe von `LockFile` definierte Datei zu errichten und später wieder aufzuheben. Um dies bei der Kompilierung als Standard zu setzen, müssen Sie `configure` mit folgender Option aufrufen:

  ```
  CFLAGS="-D APR_USE_FLOCK_SERIALIZE"
  ```

- `fcntl`: Der Systemaufruf `fctnl()` wird benutzt, um die durch `LockFile` festgelegte Datei zu sperren und wieder zu entsperren.

  ```
  configure: -D APR_USE_FCNTL_SERIALIZE
  ```

- `posixsem`: Apache verwendet mit POSIX kompatible Semaphore als Warteschlangen.

  ```
  configure: -D APR_USE_POSIXSEM_SERIALIZE
  ```

- `sysvsem`: Es werden mit System V kompatible Semaphore als Warteschlangen benutzt. Diese Einstellung wird zurzeit nur unter dem Betriebssystem IRIX verwendet.

  ```
  configure: -D APR_USE_SYSVSEM_SERIALIZE
  ```

- `sem`: Es wird jeweils die »beste« verfügbare Semaphoren-Implementierung ausgewählt, also je nach System entweder `posixsem` oder `sysvsem`. Einen eigenen `configure`-Wert hierfür gibt es logischerweise nicht.
- `pthread`: Apache benutzt POSIX-Mutexe, die der POSIX-Thread-Spezifikation genügen.

  ```
  configure: -D APR_USE_PTHREAD_SERIALIZE
  ```

- `Default`: Der bei der Kompilierung festgelegte Wert wird verwendet.

Zusätzlich können Sie noch das Schlüsselwort `OmitPID` angeben, um zu verhindern, dass die Prozess-ID des Parent-Prozesses an den Namen der Mutex-Datei angehängt wird.

#### AcceptFilter

Optimierung für die lauschenden Sockets bestimmter Protokolle

| Seit Version | 2.1.5 |
|---|---|
| Modul | `core` |
| Kontext | Server |
| Syntax | `AcceptFilter Protokoll Filter` |
| Standardwert | nicht gesetzt |

Unter einigen Betriebssystemen existieren spezielle Kernel-Filter, die die Performance von TCP/IP-Anwendungsprotokollen durch Pufferung verbessern können. In der Praxis sind dies die Accept-Filter von FreeBSD sowie die `TCP_DEFER_ACCEPT`-Filter von Linux.

Die passenden Werte unter FreeBSD sind:

```
AcceptFilter http httpready
AcceptFilter https dataready
```

`httpready` puffert die gesamte HTTP-Anfrage auf Kernel-Ebene und gibt sie erst dann an den Server weiter, wenn sie vollständig ist. Die Vollständigkeit von HTTPS-Anfragen kann der Kernel dagegen aufgrund der Verschlüsselung nicht erkennen; deshalb wird hier `dataready` verwendet. Näheres steht in den FreeBSD-Manpages zu `accf_http(9)` beziehungsweise `accf_data(9)`.

Auf Linux-Systemen steht dagegen nur der Filter `data` zur Verfügung:

```
AcceptFilter http data
AcceptFilter https data
```

Weitere Informationen über `TCP_DEFER_ACCEPT` liefert die Linux-Manpage zu `tcp(7)`.

Für Protokolle, die vor dem Senden auf eine Server-Meldung warten, empfiehlt sich dagegen der Wert `none` – die Apache-Online-Dokumentation nennt als Beispiel NNTP:

```
AcceptFilter nntp none
```

#### LoadModule

Das angegebene DSO-Modul laden und aktivieren

| | |
|---|---|
| **Modul** | `mod_so` |
| **Kontext** | Server |
| **Syntax** | `LoadModule Modul Dateiname` |
| **Standardwert** | nicht gesetzt |

Wenn Sie einen Apache-Webserver mit DSO-Unterstützung betreiben (siehe Kapitel 4, »Apache kompilieren und installieren«) und ein DSO-Modul laden möchten, wird dazu die Direktive `LoadModule` verwendet. Sie benötigt zwei Parameter: einen kanonischen Modulnamen sowie den Pfad und Dateinamen des gewünschten Moduls. Der Dateiname hat jeweils das Format `mod_xxx.so`.[2] Beim kanonischen Namen werden das Präfix `mod_` und die Dateiendung `.so` weggelas-

sen; dafür wird _module angehängt (z.B. xxx_module). Die Pfadangabe wird relativ zur ServerRoot interpretiert, falls sie keinen absoluten Pfad enthält. Das folgende Beispiel lädt das Modul mod_dir:

```
LoadModule dir_module modules/mod_dir.so
```

Bei manchen Modulen heißt die eigentliche DSO-Datei nicht mod_xxx.so, sondern anders. Dies ist beispielsweise bei PHP (siehe Kapitel 15, »Technologien zur Webprogrammierung«) der Fall:

```
LoadModule php5_module modules/libphp5.so
```

#### LoadFile

Die angegebene Objektdatei oder Bibliothek laden

| | |
|---|---|
| **Modul** | mod_so |
| **Kontext** | Server |
| **Syntax** | LoadFile Dateiname [Dateiname ...] |
| **Standardwert** | nicht gesetzt |

Einige Module benötigen Unterstützung durch zusätzliche Dateien. Eine solche externe Bibliothek oder Objektdatei wird mithilfe von LoadFile geladen. Sie können eine oder mehrere zu ladende Dateien angeben. Das folgende Beispiel lädt auf einem Windows-Rechner den Perl-Interpreter, der für das Modul mod_perl (siehe Kapitel 15, »Technologien zur Webprogrammierung«) benötigt wird, und anschließend mod_perl selbst:

```
LoadFile "C:/Perl/bin/perl58.dll"
LoadModule perl_module modules/mod_perl.so
```

#### TraceEnable

Bestimmt das Verhalten bei TRACE-Anfragen

| | |
|---|---|
| **Seit Version** | 1.3.34 und 2.0.55 |
| **Modul** | core |
| **Kontext** | Server |
| **Syntax** | TraceEnable on\|off\|extended |
| **Standardwert** | On |

2 Unter Windows handelt es sich bei DSO-Modulen formal um DLL-Dateien. Trotzdem wurde seit Apache 1.3.15 auch auf dieser Plattform die Dateiendung .so eingeführt.

Diese Direktive ermöglicht es, das Verhalten von Apache bei HTTP-Anfragen mit der Methode `TRACE`, die eine Liste der zum Erreichen einer URL verwendeten Proxy-Server zurückgibt (siehe Kapitel 2, »Funktionsweise von Webservern«), zu modifizieren. Die Standardeinstellung `on` besagt, dass `TRACE`-Anfragen erlaubt sind, aber im Einklang mit RFC 2616 keinen Anfrage-Body enthalten dürfen. `TraceEnable off` verbietet `TRACE`-Anfragen ganz und sendet dem Client eine Antwort mit dem Status `405 Method not allowed`.

Ein vom HTTP-Standard abweichendes Verhalten können Sie mit dem Wert `extended` erreichen: Die Anfragen dürfen einen Body enthalten. Dieser ist auf 64 Kilobyte beschränkt; hinzu kommen maximal 8 Kilobyte für Chunk-Header, wenn `Transfer-Encoding: chunked` verwendet wird. Diese Einschränkungen gelten nicht, wenn Apache als Proxy eingesetzt wird (siehe Kapitel 13, »Proxy- und Cache-Funktionen«). In diesem Modus gibt die Antwort im Body alle Header zurück. Diese Variante ist vor allem für Debugging- und Testzwecke geeignet.

### 6.3.2 Plattformspezifische Server-Einstellungen

Die Konfigurationsanweisungen in diesem Abschnitt betreffen ebenfalls das allgemeine Verhalten des Server-Programms. Sie sind allerdings nicht im Core, sondern in diversen MPM-Modulen definiert, weil sie nicht für alle Plattformen und Laufzeitmodelle von Apache festgelegt sind. Aufgrund ihrer besonderen Wichtigkeit wurde die Direktive `Listen` bereits im vorangegangenen Abschnitt behandelt, obwohl sie ebenfalls zu den MPM-Direktiven gehört.

Lesen Sie auch die Beschreibungen der diversen Multiprocessing-Module in Kapitel 3, »Apache 2 im Überblick«. Dort wird erläutert, auf welche Weise die MPMs Gebrauch von Prozessen und Threads machen. Dies verdeutlicht die Zusammenhänge zwischen den zahlreichen Direktiven, die deren Anzahl festlegen.

Beachten Sie, dass sich einige Direktiven in diesem Abschnitt nicht durch einen normalen Neustart von Apache 2 aktivieren lassen. Stattdessen müssen Sie dafür den Server komplett beenden und wieder starten. Dies betrifft vor allem die maximal konfigurierbaren Werte `ServerLimit` und `ThreadLimit`.

#### GracefulShutdownTimeout

Wartezeit, nach der Apache bei `graceful-stop` auf jeden Fall beendet wird

| | |
|---|---|
| **Seit Version** | 2.2 |
| **Modul** | prefork, worker, `event` |
| **Kontext** | Server |
| **Syntax** | `GracefulShutdownTimeout Sekunden` |
| **Standardwert** | `0` |

Ein neues Feature von Apache 2.2 ist das saubere Beenden, das heißt, der Hauptprozess wird erst beendet, wenn die Worker-Prozesse oder -Threads mit dem Verarbeiten der aktuellen Anfragen fertig sind (siehe Kapitel 5, »Apache in Betrieb nehmen«). Diese Direktive bestimmt optional eine maximale Wartezeit in Sekunden; sobald diese erreicht ist, wird Apache sofort beendet. Der Vorgabewert 0 bedeutet, dass beliebig lange auf die Worker gewartet wird. Das folgende Beispiel legt eine Höchstdauer von zwei Minuten fest:

```
GracefulShutdownTimeout 120
```

#### PidFile

Datei für die PID des Apache-Hauptprozesses

| | |
|---|---|
| **Modul** | beos, mpm_winnt, mpmt_os2, prefork, worker, `event` |
| **Kontext** | Server |
| **Syntax** | `PidFile Dateiname` |
| **Standardwert** | `logs/httpd.pid` |

Diese Direktive bestimmt den Pfad der PID-Datei: Wie die meisten UNIX-Daemons legt auch der Apache-HTTP-Server die Prozess-ID (PID) seines Hauptprozesses in einer Datei ab, um beispielsweise bei einem Startbefehl zu kontrollieren, ob der Server noch läuft (oder unsauber beendet wurde). In Kapitel 5, »Apache in Betrieb nehmen«, wird erläutert, wie diese Datei auf einem UNIX-Rechner dazu verwendet werden kann, um Apache zu beenden oder neu zu starten.

Da nicht jedes Betriebssystem UNIX-artige PIDs verwendet und das Verfahren deshalb nicht auf jede beliebige Plattform portierbar ist, ist die Direktive nicht im `core`, sondern in den passenden MPM-Modulen definiert. Hier ein Beispiel:

```
PidFile /var/run/httpd.pid
```

Wenn kein absoluter Pfad angegeben wird, interpretiert Apache die Angabe relativ zur `ServerRoot`.

#### User

UNIX-User-ID, unter der Apache Anfragen bearbeitet

| | |
|---|---|
| **Seit Version** | 1.x; seit 2.0 nur noch Server |
| **Modul** | prefork, worker, `event`; seit 2.3-beta `mod_unixd` |
| **Kontext** | Server |
| **Syntax** | `User Username` \| `#Usernr.` |
| **Standardwert** | `#-1` |

Die Direktive `User` funktioniert nur auf UNIX-Plattformen. Sie legt die User-ID fest, unter der Apache Client-Anfragen beantwortet. Das betrifft nicht den Hauptprozess, der für die Steuerung des Servers sowie für den Start der eigentlichen Arbeitsprozesse beziehungsweise -Threads zuständig ist – dieser läuft in der Regel unter der User-ID `root`.

Die User-ID kann entweder als Username oder numerisch angegeben werden. Wenn Sie eine Usernummer verwenden, muss diese hinter einer Raute stehen. Beispiele:

```
User htuser
User #103
```

Per Voreinstellung läuft Apache 2 unter der User-ID `-1`. Dies entspricht auf den meisten Systemen dem User `nobody`. Normalerweise ist dies in Ordnung; `nobody` ist meist eine Benutzerkennung ohne Shell. Selbst wenn sich also jemand über eine Sicherheitslücke die Rechte des Webservers aneignen sollte, wird er es nicht allzu leicht haben, unter dieser User-ID Schaden anzurichten.

Noch besser ist es, einen separaten User nur für den Webserver-Betrieb einzurichten. Auch in diesem Fall sollten Sie darauf achten, dass sich unter diesem Benutzerkonto niemand persönlich anmelden kann – dies erreichen Sie auf vielen Betriebssystemen, indem Sie die Login-Shell des Kontos auf */bin/false* setzen. Darüber hinaus sollte der User einer Group angehören, die ebenfalls exklusiv für den Einsatz von Apache eingerichtet wurde. Die beiden folgenden Befehle zur Erstellung einer solchen Group und eines Users sind für Linux geeignet und zeigen sinngemäß, wie es funktioniert:

```
# groupadd htgroup
# useradd -s /bin/false -g htgroup htuser
```

Anschließend müssen Sie die neu erstellte User- und Group-ID in der Apache-Konfigurationsdatei eintragen:

```
User htuser
Group htgroup
```

Bitte beachten Sie, dass Sie `User` *niemals* auf `root` setzen sollten!

#### Group

UNIX-Group-ID, unter der Apache Anfragen bearbeitet

| | |
|---|---|
| **Seit Version** | 1.x; seit 2.0 nur noch Server |
| **Modul** | beos, mpmt_os2, prefork, `worker`; seit 2.3-beta `mod_unixd` |

| Kontext | Server |
|---|---|
| Syntax | `Group Gruppenname \| #Gruppennr.` |
| Standardwert | `#-1` |

Unter UNIX wird ein Prozess nicht nur unter einer User-ID, sondern auch unter einer Group-ID ausgeführt. Diese Direktive setzt die Group-ID, unter der die Prozesse zur Verarbeitung von Anfragen laufen – der Hauptsteuerprozess ist hier, genau wie bei `User`, nicht betroffen. Gültige Werte für `Group` sind entweder ein Groupname oder eine Raute mit nachfolgender numerischer ID. Die Verwendung der Group-ID `root` ist natürlich wiederum tabu. Hier zwei Beispiele:

```
Group htgroup
Group #101
```

Auch für Group ist der Standardwert `-1` (`nobody`). Bereits bei der Beschreibung von `User` wurde gezeigt, wie Sie eine separate Group für den Webserver anlegen können, um die Sicherheit noch weiter zu erhöhen.

#### ListenBackLog

Gibt die maximale Länge der Warteschlange für eingehende Verbindungsanfragen an.

| Modul | beos, mpm_netware, mpm_winnt, mpmt_os2, prefork, worker, `event` |
|---|---|
| Kontext | Server |
| Syntax | `ListenBackLog Anzahl` |
| Standardwert | `511` |

Es wurde in diesem Buch bereits beschrieben, dass zur Verarbeitung eingehender Anfragen normalerweise immer ein gewisser Vorrat an Prozessen oder Threads (je nach MPM) vorhanden ist. Bei Spitzenbelastung des Servers kann es allerdings vorkommen, dass er die Reserve-Prozesse beziehungsweise -Threads nicht schnell genug erzeugt. Ist der Vorrat völlig aufgebraucht, dann werden weitere ankommende Anfragen in eine Warteschlange des lauschenden Sockets gestellt. Diese Direktive definiert die maximale Anzahl wartender Verbindungen in dieser Warteschlange. Sobald wieder Worker verfügbar sind, werden die Anfragen in Eingangsreihenfolge bearbeitet (FIFO-Prinzip).

Normalerweise brauchen Sie den voreingestellten Wert niemals anzupassen. Es kann höchstens angebracht sein, ihn kurzfristig massiv zu erhöhen, wenn Sie von einer SYN-Flood-Attacke betroffen sind (Überschwemmung durch eintreffende TCP-Pakete mit gesetztem `SYN`-Flag), damit die SYN-Flood-Pakete nicht den gesamten Vorrat für sich beanspruchen, was zum Denial of Service führt.

Der tatsächliche Maximalwert ist übrigens an die Kernel-Konstante SOMAXCONN gebunden, die angibt, wie viele Verbindungen ein lauschendes Socket überhaupt annehmen kann. Wenn für ListenBackLog ein Wert angegeben wird, der SOMAXCONN übersteigt, dann wird dieser ignoriert. Wie sich SOMAXCONN gegebenenfalls anpassen lässt, entnehmen Sie bitte der Dokumentation Ihres Betriebssystems.

#### AcceptMutex

Methode zur Serialisierung von Child-Prozessen, die Anfragen an Netzwerk-Sockets annehmen

| | |
|---|---|
| **Version** | 2.0 bis 2.2 |
| **Modul** | prefork, worker, event |
| **Kontext** | Server |
| **Syntax** | AcceptMutex Default\|Methode |
| **Standardwert** | Default |

Die verschiedenen Worker-Child-Prozesse oder -Threads rufen für jedes lauschende Socket select() auf, um in Erfahrung zu bringen, ob an diesem Socket eine Verbindungsanfrage wartet. Gleichzeitige select()-Aufrufe könnten dazu führen, dass einer oder mehrere von ihnen blockieren. Deshalb bestimmt diese Direktive, auf welche Weise Apache dafür sorgt, dass sie select() nur nacheinander aufrufen.

In Version 2.3.4 wurde die allgemeine Direktive Mutex eingeführt (siehe Seite 261), sodass AcceptMutex und andere spezifische Mutex-Direktiven abgeschafft wurden. Die möglichen Werte entsprechen den in der Beschreibung von Mutex angegebenen Methoden.

#### CoreDumpDirectory

Verzeichnis, in das Apache vor einem CoreDump zu wechseln versucht

| | |
|---|---|
| **Seit Version** | 2.0 |
| **Modul** | beos, mpm_winnt, prefork, worker, event |
| **Kontext** | Server |
| **Syntax** | CoreDumpDirectory Verzeichnis |
| **Standardwert** | je nach Installationslayout |

Wenn ein schwerwiegender Fehler auftritt, können Programme unter einigen Betriebssystemen einen CoreDump (Hauptspeicherauszug) in eine Datei schrei-

ben. Die Direktive gibt das Verzeichnis an, in dem der CoreDump abgelegt werden soll. Da die User-ID, unter der Apache ausgeführt wird, in der Regel keine Schreibrechte für die `ServerRoot` haben sollte, müssen Sie ein anderes Verzeichnis angeben, wenn Sie CoreDumps zu Debugging-Zwecken benötigen.

#### LockFile

Pfad der Lock-Datei zur Serialisierung angenommener Anfragen

| | |
|---|---|
| **Modul** | prefork, `worker` |
| **Kontext** | Server |
| **Syntax** | `LockFile Dateiname` |
| **Standardwert** | `logs/accept.lock` |

Wenn für `AcceptMutex` über eine der beiden Einstellungen `flock` oder `fcntl` eine Dateisperre verwendet wird, um die Socket-Zugriffe von Anfragen zu serialisieren, wird eine Datei benötigt, auf der die Sperre errichtet werden kann. Die Direktive `LockFile` legt diese Datei fest. Aus Sicherheitsgründen sollte sie sich nicht in einem öffentlich zugänglichen Verzeichnis unterhalb der `ServerRoot` befinden.

#### MaxClients (seit 2.3.13 MaxRequestWorkers)

Höchstzahl von Child-Prozessen, die zur Verarbeitung von Anfragen gestartet werden

| | |
|---|---|
| **Modul** | beos, prefork, worker, `event` |
| **Kontext** | Server |
| **Syntax** | `MaxClients Anzahl` |
| **Standardwert** | je nach MPM verschieden |

Diese Direktive legt die maximale Anzahl von Child-Prozessen beziehungsweise Threads fest, die Apache insgesamt zur Annahme von Client-Anfragen startet. Sie brauchen den hier festgelegten Wert nur dann zu erhöhen, wenn Sie durch Benchmarks oder Logfile-Auswertung feststellen sollten, dass Ihr Server mit der Anzahl der ankommenden Anfragen nicht zurechtkommt. Allerdings müssen Sie in diesem Fall zunächst die Direktive `ServerLimit` anpassen, um die maximal *einstellbare* Anzahl von Prozessen oder Threads zu erhöhen. Der Vorgabewert ist je nach MPM unterschiedlich:

- `prefork`: 256 Child-Prozesse
- `worker`, `event`: 16 (`ServerLimit`) x 25 (`ThreadsPerChild`)
- `beos`: 50 Threads

### MaxMemFree

Maximale Speichermenge, die die Speicherverwaltung ohne `free()`-Aufruf verwalten darf

| | |
|---|---|
| **Modul** | beos, mpm_netware, mpm_winnt, prefork, worker, `event` |
| **Kontext** | Server |
| **Syntax** | `MaxMemFree KBytes` |
| **Standardwert** | `0` |

Mithilfe dieser Direktive können Sie die Arbeitsspeichermenge begrenzen, über die die Hauptspeicher-Zuteilungsroutine des Servers verfügen darf, ohne mittels `free()` wieder Speicher freizugeben. Der Vorgabewert `0` bedeutet, dass keine Beschränkung existiert.

### MaxRequestsPerChild (seit 2.3.9 MaxConnectionsPerChild)

Maximale Anzahl von Anfragen, die ein Child-Prozess nacheinander abarbeitet

| | |
|---|---|
| **Modul** | mpm_netware, mpm_winnt, mpmt_os2, prefork, worker, `event` |
| **Kontext** | Server |
| **Syntax** | `MaxRequestsPerChild Anzahl` |
| **Standardwert** | meist `10000` (siehe Beschreibung) |

Die Direktive bestimmt indirekt die Lebensdauer eines Worker-Prozesses, der für die Bearbeitung von Client-Anfragen zuständig ist: Nach der Verarbeitung der angegebenen Anzahl von Anfragen (beziehungsweise persistenten Verbindungen) wird der Prozess beendet. Dies ermöglicht ein allmähliches Verringern der Prozessanzahl, wenn die Anfragelast nachlässt.

Der spezielle Wert 0 bedeutet, dass die Lebensdauer eines Prozesses nicht eingeschränkt wird. Bei `mpm_winnt` und `mpm_netware` ist dieser Wert aufgrund der Plattformbesonderheiten voreingestellt.

### MaxSpareThreads

Maximale Anzahl von Reserve-Threads

| | |
|---|---|
| **Modul** | beos, mpm_netware, mpmt_os2, worker, `event` |
| **Kontext** | Server |
| **Syntax** | `MaxSpareThreads Anzahl` |
| **Standardwert** | je nach MPM verschieden |

Diese Direktive legt bei Thread-basierten MPMs die Höchstzahl von Reserve-Threads fest, die für die Verarbeitung einer Verbindung bereitstehen. Für die verschiedenen MPMs sind folgende Werte vorgegeben:

- `worker`, `event`: 250. Hier geht es um die Anzahl der Threads insgesamt. Der Wert kann nicht kleiner als `MinSpareThreads` + `ThreadsPerChild` sein.
- `mpm_netware`: 100 (Gesamtzahl). Der Wert muss größer sein als `MinSpareThreads`.
- `beos`: 50 (Gesamtzahl)
- `mpmt_os2`: 10 (Gesamtzahl)

#### MinSpareThreads

Mindestzahl von Reserve-Threads

| | |
|---|---|
| **Modul** | beos, mpm_netware, mpmt_os2, worker, `event` |
| **Kontext** | Server |
| **Syntax** | `MinSpareThreads Anzahl` |
| **Standardwert** | je nach MPM verschieden |

Die Direktive `MinSpareThreads` bestimmt bei Thread-basierten MPMs die Mindestzahl der Reserve-Threads. Wie bei `MaxSpareThreads` gibt es Unterschiede zwischen den Standardwerten der einzelnen MPM-Module:

- `worker`, `event`: Gesamtwert 75
- `mpm_netware`: insgesamt 10
- `beos`: 1 (Gesamtwert)
- `mpmt_os2`: Gesamtwert 5

#### ScoreBoardFile

Pfad der Datei zur Koordination von Child-Prozessen

| | |
|---|---|
| **Modul** | `mpm_common` |
| **Kontext** | beos, mpm_winnt, prefork, worker, `event` |
| **Syntax** | `ScoreBoardFile Dateiname` |
| **Standardwert** | `logs/apache_status` |

Ein `ScoreBoardFile` ist eine Datei, die zur Kommunikation zwischen Parent- und Child-Prozessen eingesetzt wird. Wenn die Direktive nicht in der Konfigurationsdatei steht, wird anstelle einer Datei ein Shared-Memory-Segment verwendet.

Solange Sie kein externes Modul verwenden, das explizit auf das ScoreBoard zugreifen muss, ist dies aus Performance- und Sicherheitsgründen zu bevorzugen.

#### SendBufferSize

Größe des TCP-Sendepuffers

| | |
|---|---|
| **Modul** | beos, mpm_netware, mpm_winnt, mpmt_os2, prefork, worker, `event` |
| **Kontext** | Server |
| **Syntax** | `SendBufferSize Bytes` |
| **Standardwert** | `0` |

Die Größe des TCP-Sendepuffers wird auf den angegebenen Wert gesetzt. Die Voreinstellung 0 bedeutet, dass der Standardwert des zugrunde liegenden Betriebssystems verwendet werden soll. Das ist normalerweise in Ordnung. Es sei denn, Sie verwenden ein veraltetes Betriebssystem mit einer sehr schnellen Netzwerkverbindung – dann sollten Sie aus Performance-Gründen manuell einen höheren Wert angeben als die Systemvorgabe.

#### ReceiveBufferSize

Größe des TCP-Empfangspuffers

| | |
|---|---|
| **Modul** | beos, mpm_netware, mpm_winnt, mpmt_os2, prefork, worker, `event` |
| **Kontext** | Server |
| **Syntax** | `ReceiveBufferSize Bytes` |
| **Standardwert** | `0` |

Diese Direktive stellt analog zu SendBufferSize die Größe des TCP-Empfangspuffers ein. Auch hier steht der Standardwert 0 für den Standard des Betriebssystems.

#### ServerLimit

Absolute Höchstgrenze für die Anzahl von Prozessen

| | |
|---|---|
| **Modul** | prefork, `worker` |
| **Kontext** | Server |
| **Syntax** | `ServerLimit Anzahl` |
| **Standardwert** | je nach MPM verschieden |

Diese Direktive legt den Höchstwert fest, der überhaupt für `MaxClients` (beziehungsweise `NumServers` beim MPM `perchild`) eingestellt werden darf. Für die einzelnen MPM-Module hat dies unterschiedliche Auswirkungen:

- `prefork`: Die Voreinstellung ist 256. Wenn Sie für `MaxClients` – bei diesem MPM die Höchstzahl der Child-Prozesse insgesamt – einen höheren Wert einstellen möchten, müssen Sie zuerst `ServerLimit` auf diesen Wert setzen.
- `worker`, `event`: Der Standardwert ist 16 für die höchste einstellbare Anzahl von Thread-verwaltenden Child-Prozessen. Die Gesamtzahl der Worker-Threads ist das Produkt von `ServerLimit` und `ThreadsPerChild`. Erhöhen Sie im Bedarfsfall einen dieser beiden Werte, wenn Sie insgesamt mehr Threads benötigen.

Wie bereits erwähnt, muss Apache komplett beendet und wieder gestartet werden, um diesen Wert anzupassen; ein automatischer Neustart genügt nicht.

Wenn Sie einen ungewöhnlich hohen Wert einstellen möchten, müssen Sie `configure` bei der Kompilierung mit dem Compiler-Flag `HARD_SERVER_LIMIT` aufrufen. Beispiel:

```
./configure ... CFLAGS="-D HARD_SERVER_LIMIT=2048"
```

#### StartServers

Anzahl von Child-Prozessen, die beim Start erzeugt werden

| | |
|---|---|
| **Modul** | mpmt_os2, prefork, worker, `event` |
| **Kontext** | Server |
| **Syntax** | `StartServers Anzahl` |
| **Standardwert** | je nach MPM verschieden |

Über diese Direktive wird festgelegt, wie viele Child-Prozesse beim Start des Servers erzeugt werden. Diese Anzahl wird zur Laufzeit gemäß den Angaben in den anderen Direktiven für die Prozessanzahl der Server-Last angepasst. Deshalb gibt es in der Regel wenig Veranlassung, die Voreinstellung zu ändern. Für die einzelnen MPMs gelten folgende Standardwerte:

- `prefork`: 5
- `worker`: 3
- `mpmt_os2`: 2

### StartThreads

Anzahl von Threads, die beim Start erzeugt werden

| | |
|---|---|
| **Modul** | beos, `mpm_netware` |
| **Kontext** | Server |
| **Syntax** | `StartThreads Anzahl` |
| **Standardwert** | je nach MPM verschieden |

Bei den Thread-basierten MPMs `beos` und `mpm_netware` übernimmt diese Direktive die Aufgabe von `StartServers`: Sie bestimmt, wie viele Threads beim Start des Servers erzeugt werden. Da Apache bei Bedarf zusätzliche Threads startet, muss auch dieser Wert normalerweise nicht geändert werden. Hier die Standardwerte:

- `mpm_netware`: 50 (Gesamtzahl der gestarteten Threads)
- `beos`: 10 (ebenfalls insgesamt)

### ThreadLimit

Absolute Höchstzahl von Threads

| | |
|---|---|
| **Seit Version** | 2.0; für `mpm_winnt` seit 2.0.41 |
| **Modul** | mpm_winnt, worker, `event` |
| **Kontext** | Server |
| **Syntax** | `ThreadLimit Anzahl` |
| **Standardwert** | je nach MPM verschieden |

Dieser Wert bestimmt die maximale Anzahl von Threads, die für `ThreadsPerChild` angegeben werden kann. Sie sollten die Direktive nur anpassen, wenn Sie den Wert von `ThreadsPerChild` erhöhen müssen. Um kein unnötiges Shared Memory zu vergeuden, sollte `ThreadLimit` keinen höheren Wert erhalten als den für `ThreadsPerChild` benötigten.

Der Standardwert ist für `mpm_winnt` 1.920 und für alle anderen MPMs 64. Der Quellcode enthält einen fest einprogrammierten Maximalwert von 20.000 (beziehungsweise 15.000 bei `mpm_winnt`).

Es wurde bereits darauf hingewiesen, dass der Server komplett beendet und wieder gestartet werden muss, um diesen Wert anzupassen.

### ThreadsPerChild

Anzahl der Threads pro Child-Prozess

| | |
|---|---|
| **Modul** | mpm_winnt, worker, `event` |
| **Kontext** | Server |

| Syntax | `ThreadsPerChild Anzahl` |
|---|---|
| Standardwert | je nach MPM verschieden |

Diese Direktive legt fest, wie viele Threads ein einzelner Verwaltungs-Child-Prozess startet. Dies hat je nach MPM unterschiedliche Folgen:

- `worker`, `event`: Bei diesen MPM-Modulen existieren mehrere Child-Prozesse. Der Standardwert beträgt 25 und kann erhöht werden, wenn die Gesamtzahl der Worker (Child-Prozesse * Threads) für die Server-Last nicht ausreicht.
- `mpm_winnt`: Da unter Windows nur ein einziger Child-Prozess zur Verwaltung der Worker-Threads existiert, muss der hier angegebene Wert der maximalen Anzahl gleichzeitiger Anfragen entsprechen, die der Server bewältigen muss. Die Voreinstellung liegt bei 64, und es ist wahrscheinlich, dass Sie sie für einen Produktionsserver erhöhen müssen.

#### MaxSpareServers

Maximale Anzahl von Reserve-Child-Prozessen

| Modul | `mpm_prefork` |
|---|---|
| Kontext | Server |
| Syntax | `MaxSpareServers Anzahl` |
| Standardwert | `10` |

Diese Direktive bestimmt, wie viele unbeschäftigte Prozesse maximal als Reserve auf Anfragen warten sollen. Sollte die Anzahl über den hier eingestellten Wert hinausgehen, werden die überschüssigen Prozesse beendet.

Sie brauchen nur bei sehr stark beanspruchten Servern einen höheren Wert als voreingestellt. Wenn Sie einen Wert verwenden, der kleiner als `MinSpareServers` ist, wird er auf `MinSpareServers` +1 gesetzt.

#### MinSpareServers

Mindestzahl von Reserve-Child-Prozessen

| Modul | `mpm_prefork` |
|---|---|
| Kontext | Server |
| Syntax | `MinSpareServers Anzahl` |
| Standardwert | `5` |

Mithilfe von `MinSpareServers` können Sie festlegen, wie viele unbeschäftigte Child-Prozesse mindestens als Reserve auf Client-Anfragen warten sollen. Wenn

weniger Prozesse frei sind als hier eingestellt, erzeugt Apache neue, und zwar höchstens einen pro Sekunde. In der Regel muss der Wert nicht geändert werden. Nur bei besonders belasteten Sites müssen Sie ihn möglicherweise erhöhen.

#### MaxRequestsPerThread

Maximale Anzahl von Anfragen, die ein Thread nacheinander abarbeitet

| | |
|---|---|
| **Version** | 2.0 und 2.2 (nur BeOS) |
| **Modul** | `mpm_beos` |
| **Kontext** | Server |
| **Syntax** | `MaxRequestsPerThread Anzahl` |
| **Standardwert** | `0` |

Diese Direktive bestimmt unter BeOS indirekt die Lebensdauer eines Threads und ähnelt damit der für Prozesse zuständigen Einstellung `MaxRequestsPerChild`: Sie legt fest, wie viele Anfragen beziehungsweise persistente Verbindungen ein Thread verarbeitet, bevor er beendet wird. Wenn Sie die Voreinstellung 0 (unbegrenzte Lebensdauer) ändern, kann der Server schneller auf eine Verringerung der Anfragelast reagieren und so Ressourcen freigeben.

#### MaxThreads

Maximale Anzahl von Worker-Threads

| | |
|---|---|
| **Seit Version** | 2.0 (nur NetWare) |
| **Modul** | `mpm_netware` |
| **Kontext** | Server |
| **Syntax** | `MaxThreads Anzahl` |
| **Standardwert** | `2.048` |

Diese Direktive ist nur unter Novell NetWare verfügbar. Sie bestimmt die maximale Anzahl von Threads, die Anfragen beantworten, und ähnelt so der für andere MPMs definierten Direktive `ThreadsPerChild`. Da der Standardwert dem einkompilierten Maximum entspricht, können Sie den Wert lediglich verringern.

#### ThreadStackSize

Größe des Stacks pro Thread

| | |
|---|---|
| **Seit Version** | 2.0 (nur NetWare) |
| **Modul** | `mpm_netware` |
| **Kontext** | Server |

| Syntax | ThreadStackSize Anzahl |
|---|---|
| Standardwert | 65.536 |

Mit dieser Direktive wird die maximale Größe des Stacks für einen einzelnen Thread festgelegt. Sie muss nur erhöht werden, falls ein Stack-Overflow auftreten sollte.

#### Win32DisableAcceptEx

AcceptEx() deaktivieren, falls inkompatibel

| Seit Version | 2.0.49 bis 2.2.x |
|---|---|
| Modul | mpm_winnt |
| Kontext | Server |
| Syntax | Win32DisableAcceptEx |
| Standardwert | nicht gesetzt |

Unter Windows wird standardmäßig die Winsock2-Funktion AcceptEx() verwendet, die gegenüber dem klassischen BSD-Systemaufruf accept() einen Performance-Gewinn liefert. Dies kann allerdings zur Inkompatibilität mit einigen anderen Programmen führen, z.B. mit VPN-Software oder Antivirus-Programmen. Ein solches Problem kann sich in einer Fehlermeldung wie dieser äußern:

```
[error] (730038)An operation was attempted on something that
is not a socket.: winnt_accept: AcceptEx failed. Attempting
to recover.
```

Wenn Sie diese Direktive (ohne Wert) in Ihre Konfigurationsdatei setzen, wird AcceptEx() deaktiviert. Sie sollten sie also verwenden, falls das geschilderte Problem bei Ihnen auftritt. Ab Apache 2.3 wird AcceptFilter für die entsprechenden Einstellungen verwendet.

### 6.3.3 Konfiguration des »Hauptservers«

Der zweite Abschnitt der Konfigurationsdatei *httpd.conf* ist wie erwähnt für die Einstellungen der Standard-Website zuständig, die der Server veröffentlicht. Hier befinden sich also beispielsweise die DocumentRoot (das Basisverzeichnis) der Site sowie sämtliche Zugriffs- und Authentifizierungseinstellungen. In der offiziellen Vorlage der *httpd.conf* hieß es früher über diesen Abschnitt:

```
### Section 2: 'Main' server configuration
#
# The directives in this section set up the values used by the
# 'main' server, which responds to any requests that aren't
```

```
# handled by a <VirtualHost> definition.  These values also
# provide defaults for any <VirtualHost> containers you may
# define later in the file.
```

Der »Hauptserver« behandelt nach dieser Beschreibung sämtliche Anfragen, für die kein spezieller virtueller Host konfiguriert ist. Außerdem werden alle hier vorgenommenen Grundeinstellungen zunächst von jedem virtuellen Host übernommen, sofern sie in dessen Konfiguration nicht überschrieben werden. Umgekehrt gilt natürlich, dass viele der hier vorgestellten Direktiven für einzelne virtuelle Hosts überschrieben werden können und oft sogar sollten.

#### ServerAdmin

E-Mail-Adresse des Server-Administrators für automatisch generierte Fehlermeldungsseiten

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ServerAdmin E-Mail-Adresse` |
| **Standardwert** | nicht gesetzt |

Wenn Apache bei der Beantwortung von Anfragen auf Probleme stößt, erzeugt er angepasste Fehlermeldungsseiten (festgelegt durch die Direktive `ErrorDocument`). Auf Wunsch kann eine solche Seite einen Link auf Ihre E-Mail-Adresse enthalten, damit Benutzer der Website Ihnen Probleme mitteilen können. Die Direktive `ServerAdmin` legt diese Adresse fest. Beispiel:

```
ServerAdmin webmaster@mynet.de
```

#### ServerName

Hostname und TCP-Port des Servers

| | |
|---|---|
| **Seit Version** | 2.0 (früher `Port`) |
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ServerName voll.qualifizierter.domain.name [:Port]` |
| **Standardwert** | nicht gesetzt |

Mithilfe dieser Direktive wird der Domain-Name des Servers angegeben, optional gefolgt von einer TCP-Portnummer. Die Konfigurationsanweisung bestimmt nicht etwa, wie `Listen`, auf welche Anfragen Apache überhaupt reagiert. Sie legt den Server-Namen auch nicht nach außen fest – das funktioniert im Internet nur

über DNS. Der hier angegebene Name dient vielmehr der Selbstidentifikation des Servers. Ob er auch für selbst referenzierte URLs benutzt wird, regelt die Direktive `UseCanonicalName`.

Es ist auf jeden Fall wichtig, dass der festgelegte Name auf irgendeine Weise in eine IP-Adresse aufgelöst werden kann: bei einem öffentlich zugänglichen Produktionsserver natürlich über DNS; bei einem neu eingerichteten Server, den Sie testen möchten, können Sie den Namen dagegen in Ihre `hosts`-Datei eintragen (siehe Kapitel 5, »Apache in Betrieb nehmen«). Hier ein Beispiel:

```
ServerName www.mynet.de:80
```

#### UseCanonicalName

Legt fest, woher der Server seinen eigenen Hostnamen für selbst referenzierte URLs erhält.

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `UseCanonicalName On\|Off\|DNS` |
| **Standardwert** | `Off` (bis 2.0 `On`) |

Bei Weiterleitungen, automatisch generierten Fehlermeldungsseiten und ähnlichen Gelegenheiten muss Apache selbst referenzierte URLs erzeugen, also absolute URLs, die wieder auf den Server-Host selbst zeigen. Diese Direktive bestimmt, wie der Host- beziehungsweise Domain-Name für diese URLs bestimmt wird. Es gibt drei mögliche Werte:

- `On`: Apache verwendet den Hostnamen und den Port, der mithilfe der Direktive `ServerName` festgelegt wurde.
- `Off`: Es werden der Hostname und der Port benutzt, die der Client in der Anfrage übermittelt hat.
- `DNS`: Der Hostname wird durch einen Reverse-DNS-Lookup ermittelt, das heißt, er erfragt über das Domain Name System den Hostnamen zur IP-Adresse.

In der Regel ist die Einstellung `On` in Ordnung. Der Default-Wert wurde in Version 2.2 jedoch umgestellt, da `On` manchmal Probleme mit schlecht geschriebenen CGI-Skripten oder Webanwendungen bereitet, die für selbst referenzierte URLs nicht die Umgebungsvariable `SERVER_NAME` verwenden. Darüber hinaus kann `Off` nützlich sein, wenn die Clients im Intranet über einen anderen Hostnamen auf den Server zugreifen als diejenigen im Internet (z.B. *www* statt *www.mynet.de*), da die Intranet-Clients bei Weiterleitungen und ähnlichen Vor-

gängen sonst auf den externen Domain-Namen *www.mynet.de* umgeleitet würden. `DNS` sorgt für Kompatibilität mit HTTP/1.0-Clients, die keinen `Host`-Header senden, und sollte eigentlich nur beim Einsatz IP-basierter virtueller Hosts verwendet werden, zumal der Reverse Lookup für Performance-Nachteile sorgt.

#### UseCanonicalPhysicalPort

Legt fest, woher der Server die Portnummer für selbst referenzierte URLs erhält.

| | |
|---|---|
| **Seit Version** | 2.2 |
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `UseCanonicalPhysicalPort On\|Off` |
| **Standardwert** | `Off` |

Wenn Sie diese Direktive auf `On` setzen, wird bei der Erzeugung selbst referenzierter URLs der tatsächliche, physische TCP-Port der Anfrage gewürdigt. Die genaue Reihenfolge zur Ermittlung des Ports hängt auch von der Einstellung der Direktive `UseCanonicalName` ab. Hat diese den Wert `On`, dann versucht Apache, den Port in folgender Reihenfolge zu bestimmen:

1. Port aus dem Wert von `ServerName`
2. Physischer Port
3. Standardport

Hat `UseCanonicalName` dagegen einen der Werte `Off` oder `DNS`, wird folgende Reihenfolge verwendet:

1. Port aus dem `Host`-Header
2. Physischer Port
3. Port aus dem Wert von `ServerName`
4. Standardport

Bei `UseCanonicalPhysicalPort Off` wird der physische Port nicht berücksichtigt; ansonsten bleibt die Reihenfolge je nach Einstellung für `UseCanonicalName` gleich.

#### ServerTokens

Ausführlichkeit der Versionsangabe für den HTTP-Antwort-Header `Server`

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server |

| Syntax | ServerTokens Major \| Minor \| Min[imal] \| Prod[uctOnly] \| OS \| Full |
|---|---|
| Standardwert | Full |

Diese Direktive legt fest, wie ausführlich der Apache-Webserver seine Versionsinformationen im HTTP-Antwort-Header `Server` und in der durch `ServerSignature` definierten Fußzeile automatisch generierter Dokumente angibt. Folgende Werte sind dafür definiert:

- `Prod` oder `ProductOnly`: Es wird nur der Programmname ausgegeben: `Apache`.
- `Major`: Die Hauptversionsnummer wird angegeben, z.B. `Apache/2`.
- `Minor`: Haupt- und Unterversionsnummer werden kombiniert, z.B. `Apache/2.2`.
- `Min` oder `Minimal`. Es werden Hauptversions-, Unterversions- und Releasenummer angegeben. Beispiel: `Apache/2.2.10`.
- `OS`: Die vollständige Versionsnummer wie bei `Minimal` wird angegeben, zusätzlich steht in Klammern die Systemplattform. Beispiel:

  ```
  Apache/2.2.21 (UNIX)
  ```

- `Full`: Zusätzlich zur vollen Versionsnummer und zum Betriebssystem werden die Versionsdaten wichtiger Zusatzmodule angezeigt. Beispiel:

  ```
  Apache/2.2.21 (UNIX) mod_perl/2.0.4 Perl/v5.10.0 PHP/5.3.8
  ```

In den meisten Fällen sollten Sie sich für `Minimal` oder `OS` entscheiden. `Full` verrät womöglich zu viele Details über die Server-Konfiguration.

Falls Sie sich allerdings Hoffnungen machen, durch die Veröffentlichung von weniger Informationen (im Extremfall `ProductOnly`) Angriffe auf Ihren Webserver abzuwehren, handelt es sich um einen Fall von »Security by Obscurity«: Durch Verschleierung der Softwareinfrastruktur soll Crackern die Arbeit erschwert werden. Dies beeindruckt allerdings höchstens ein paar automatisierte Angriffs-Tools, die von Anfängern verwendet werden.

Aus diesem Grund wird auch keine Direktive angeboten, die eine grundlegende Änderung der Server-Informationen ermöglicht. Wenn Sie möchten, dass Ihr Server sich unter falschem Namen meldet, müssen Sie die entsprechenden Daten vor der Kompilierung im Quellcode ändern. Die Definitionen befinden sich am Ende der Datei `ap_release.h` im Verzeichnis `include` der Source-Distribution und lauten in der aktuellen Version folgendermaßen:

```
#define AP_SERVER_BASEVENDOR "Apache Software Foundation"
#define AP_SERVER_BASEPRODUCT "Apache"
#define AP_SERVER_MAJORVERSION "2"
#define AP_SERVER_MINORVERSION "2"
#define AP_SERVER_PATCHLEVEL "9"
```

Die Werte lassen sich natürlich beliebig ändern. Beispiel:

```
#define AP_SERVER_BASEVENDOR "Sirius Cybernetics Corp."
#define AP_SERVER_BASEPRODUCT "The Hitchhiker's Web Server"
#define AP_SERVER_MAJORVERSION "3"
#define AP_SERVER_MINORVERSION "1"
#define AP_SERVER_PATCHLEVEL "42"
```

Wenn Sie Apache anschließend gemäß den Anweisungen aus Kapitel 4, »Apache kompilieren und installieren«, kompilieren, meldet er sich als `The Hitchhiker's Web Server/3.1.42`.

#### ServerSignature

Fußzeile automatisch generierter Server-Dokumente

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`All`) |
| **Syntax** | `ServerSignature On\|Off\|Email` |
| **Standardwert** | `Off` |

Diese Direktive legt fest, ob Apache in automatisch generierte Dokumente wie Index- oder Fehlermeldungsseiten eine Fußzeile einfügen soll, die Informationen über die Server-Software, den Hostnamen und eventuell die E-Mail-Adresse des Administrators enthält. Es gibt drei mögliche Werte:

- `Off`: Apache erzeugt keine Informationsfußzeile.
- `On`: Es wird eine Fußzeile ohne E-Mail-Adresse generiert. Beispiel:

  *Apache/2.2.10 (UNIX) Server at www.mynet.de Port 80*
- `EMail`: Die Fußzeile sieht genauso aus wie bei `On`, allerdings ist der Hostname gleichzeitig ein Hyperlink auf die unter `ServerAdmin` angegebene E-Mail-Adresse.

Der Hostname wird gemäß der Direktive `UseCanonicalName` angegeben. Seit Apache 2.0.44 bestimmt `ServerTokens`, wie ausführlich die Information über den Server ausfällt.

#### DocumentRoot

Stammverzeichnis der Website

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>` |

| Syntax | DocumentRoot Verzeichnis |
|---|---|
| Standardwert | variiert je nach Layout |

Dies ist die wichtigste Konfigurationsanweisung für die Website des Hauptservers oder eines virtuellen Hosts: Sie legt das Basisverzeichnis fest, aus dem Apache die angeforderten Dokumente ausliefert. Das bedeutet, dass Pfadangaben aus der URL der Client-Anfrage an dieses Verzeichnis angehängt werden, um die gewünschte Ressource zu ermitteln. Angenommen, Sie haben `DocumentRoot` auf */usr/local/share/apache2/htdocs* gesetzt (Standard beim `GNU`-Installationslayout; siehe Kapitel 4, »Apache kompilieren und installieren«). Wenn ein Client nun beispielsweise die URL *http://www.mynet.de/info/news.html* anfordert, liefert der Server die Datei */usr/local/share/apache2/htdocs/info/news.html* aus – oder eine Fehlermeldung, falls das Dokument an der angegebenen Stelle nicht existiert. Beispiel:

```
DocumentRoot /usr/share/web
```

Mithilfe von Direktiven wie `Alias` oder `UserDir` (siehe Kapitel 8, »Weiterleitungen und Indizes«) können Sie übrigens auch solche Dateien und Verzeichnisse im Bereich der Website abbilden, die sich eigentlich außerhalb der `DocumentRoot` befinden.

#### DirectoryIndex

Name der Indexseite

| Modul | mod_dir |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| Syntax | DirectoryIndex lokale_URL [lokale_URL ...] |
| Standardwert | index.html |

Dies ist eine der beiden Direktiven, die das Standardmodul `mod_dir` zur Verfügung stellt. Sie definiert die Namen von Dokumenten, die Apache automatisch ausliefert, wenn anstelle einer Datei ein Verzeichnis angefordert wurde. Dies ist erforderlich, damit Benutzer einfach eine Website wie *http://www.mynet.de/* anfordern können, anstatt den Namen der gewünschten Datei selbst anzugeben. Wenn Sie hier mehrere Dateien angeben, sucht Apache in der angegebenen Reihenfolge nach ihnen und liefert die erste aus, die er findet. Beispiel:

```
DirectoryIndex index.html index.html.var index.php
```

Ist eine Datei mit dem Namen `index.html` vorhanden, dann wird diese an den Client geliefert. Bei der Angabe `index.html.var` handelt es sich um eine `TypeMap`

(siehe Kapitel 7, »Header und MIME-Einstellungen«) – je nach Sprach- oder Zeichensatzpräferenz des Clients können auf diese Weise unterschiedliche Dateien serviert werden. Der Eintrag `index.php` ist interessant, wenn Sie mit PHP (siehe Kapitel 15, »Technologien zur Webprogrammierung«) arbeiten – auf diese Weise können auch PHP-Skripte als Indexdateien dienen.

Selbstverständlich braucht der Name vor der Dateiendung nicht unbedingt `index` zu lauten – wenn Sie z. B. von Microsoft Internet Information Services auf Apache umgestiegen sind, möchten Sie vielleicht lieber den bei Microsoft üblichen Dateinamen `default.html` weiterverwenden, um Ihre alte Website mit möglichst wenigen Änderungen zu übernehmen.

Übrigens können Sie in der Liste auch einen absoluten URL-Pfad angeben. Dieser sollte natürlich den letzten Eintrag bilden, damit die entsprechende Datei immer dann ausgeliefert wird, wenn das aktuelle Verzeichnis keine eigene Indexdatei enthält. Das folgende Beispiel weist Apache an, die Datei `index.html` im angeforderten Verzeichnis an die Clients zu liefern oder aber die Datei `defaultindex.html` in der `DocumentRoot`, falls `index.html` nicht existiert:

```
DirectoryIndex index.html /defaultindex.html
```

Auf Wunsch kann Apache auch automatisch einen Verzeichnisindex generieren – diese Funktion wird vom Modul `mod_autoindex` bereitgestellt, das in Kapitel 8, »Weiterleitungen und Indizes«, ausführlich beschrieben wird.

**mod_dir und Verzeichnisanfragen**

- Neben der Definition der Indexseite hat das Modul `mod_dir` übrigens auch noch die wichtige Aufgabe, Anfragen umzuleiten, die ein Verzeichnis betreffen.
- Wenn eine Anfrage mit einer URL ohne abschließenden Slash eintrifft, behandelt Apache diese normalerweise nicht als Anforderung eines Verzeichnisses. Angenommen, ein Client fordert die Ressource /test an. Der Server sucht daraufhin in der `DocumentRoot` nach einer Datei mit diesem Namen.
- Falls aber keine Datei, sondern nur ein Verzeichnis mit diesem Namen vorhanden ist, geht es unterschiedlich weiter: Wenn `mod_dir` nicht aktiv ist, erhält der Client einfach die Fehlermeldung »404 Not Found«. Ist `mod_dir` dagegen geladen, wird eine Weiterleitung (301 Moved Permanently) gesendet, die einen Location-Header mit der kompletten URL einschließlich des Slashs enthält, in diesem Fall also z. B. *http:// www.mynet.de/test/* (der Hostname wird gemäß der bereits beschriebenen Direktive `UseCanonicalName` gebildet). So gut wie alle Browser folgen dieser Weiterleitung automatisch.

Wenn ein Client ein Verzeichnis mit abschließendem Slash anfordert, beispielsweise */test/*, wird natürlich sofort dessen Index geliefert.

### DirectorySlash

Weiterleitung bei fehlendem Abschluss-Slash ein-/ausschalten

| Seit Version | 2.0.51 |
|---|---|
| Modul | mod_dir |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| Syntax | DirectorySlash On\|Off |
| Standardwert | On |

Diese Direktive erlaubt es Ihnen in Ausnahmefällen, die unter »mod_dir und Verzeichnisanfragen« soeben beschriebene Weiterleitung bei fehlendem Slash zu deaktivieren. Dazu müssen Sie im entsprechenden Kontext folgende Direktive setzen:

```
DirectorySlash Off
```

Da die URL ohne abschließenden Slash nun nicht mehr auf Inhalte verweist, könnte es sinnvoll sein, für diesen Kontext einen speziellen Handler zu setzen, der sich um die Beantwortung solcher Anfragen kümmert (siehe Kapitel 7, »Header und MIME-Einstellungen«).

### AccessFileName

Name der ausgelagerten Verzeichniskonfigurationsdatei

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | AccessFileName Dateiname |
| Standardwert | .htaccess |

Mithilfe dieser Direktive können Sie den Namen einer Datei angeben, die innerhalb eines Verzeichnisses der DocumentRoot zum Überschreiben von Konfigurationsdirektiven dient. Standardmäßig ist der Name dieser Datei .htaccess, und es gibt im Allgemeinen keinen Grund, dies zu ändern.[3] Erlaubt ist jeder beliebige Dateiname – er sollte aber zumindest mit einem Punkt beginnen, um wenigstens einen geringfügigen Schutz zu gewährleisten. Beispiel:

```
AccessFileName .override
```

3 Wieder einmal ein Fall von »Security by Obscurity«, der allenfalls vor Skript-Kiddies schützt.

Vergessen Sie nicht, die Auslieferung von Dateien mit dem gewählten Namen an Clients generell zu deaktivieren:

```
<Files .override>
  Order Deny,Allow
  Deny from all
</Files>
```

### 6.3.4 Wichtige Verzeichniseinstellungen

Auch die Konfigurationsanweisungen, die in diesem Abschnitt besprochen werden, gehören zur zweiten Abteilung der Datei *httpd.conf*. Der Unterschied besteht darin, dass sie üblicherweise innerhalb von `<Directory>`-Containern stehen, um die Einstellungen für die `DocumentRoot` und ihre Unterverzeichnisse sowie für einige andere Verzeichnisse festzulegen.

Hier werden zunächst alle entsprechenden Direktiven behandelt; im Anschluss daran finden Sie einige Beispiele für übliche Verzeichniskonfigurationen.

#### Options

Verzeichnisoptionen

| Modul | `core` |
|---|---|
| Kontext | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`Options`) |
| Syntax | `Options All\|[+\|-]Option [[+\|-]Option ...]` |
| Standardwert | `All` |

Die Direktive `Options` legt fest, welche besonderen Eigenschaften ein bestimmtes Verzeichnis (mitsamt seinen Unterverzeichnissen) besitzen soll. Das nähere Verhalten einiger dieser Optionen wird durch andere Direktiven geregelt; hier geht es zunächst einmal darum, ob eine bestimmte Eigenschaft grundsätzlich unterstützt werden soll oder nicht. Im Einzelnen sind folgende Optionen definiert, die Sie als Werte der Direktive angeben können:

- `None`: Schaltet sämtliche Optionen ab.
- `Indexes`: Wenn die angeforderte URL ein Verzeichnis ist, wird die mittels `DirectoryIndex` festgelegte Indexdatei ausgeliefert oder – falls diese nicht vorhanden ist – ein durch `mod_autoindex` generierter Index.
- `FollowSymLinks`: Symbolische Links innerhalb des Verzeichnisses werden aufgelöst – Apache liefert das entsprechende Ziel des Links aus.

- `SymLinksIfOwnerMatch`: Symbolische Links werden nur dann verfolgt, wenn der Eigentümer des SymLinks demjenigen der Zieldatei entspricht.
- `ExecCGI`: Innerhalb des Verzeichnisses soll die Ausführung von CGI-Skripten gestattet sein: Dateien mit bestimmten Endungen oder MIME-Types werden von Apache als ausführbare Skripte betrachtet; ihre Ausgabe wird als dynamisch erzeugtes Dokument an den Client ausgeliefert. Es ist erheblich sicherer, über die Direktive `ScriptAlias` separate CGI-Verzeichnisse einzurichten und diese Option für Verzeichnisse innerhalb der `DocumentRoot` zu deaktivieren. Näheres zur CGI-Konfiguration erfahren Sie in Kapitel 14, »CGI«.
- `Includes`: Server Side Includes (siehe Kapitel 16, »SSI und Filter«) sind in diesem Verzeichnis gestattet.
- `IncludesNOEXEC`: Auch in einem Verzeichnis mit dieser Option sind SSI grundsätzlich erlaubt, allerdings mit Ausnahme von `#exec` (Programmausführung) und `#exec cgi` (CGI-Ausführung).
- `MultiViews`: Aktiviert die Auslieferung alternativer Dokumente für unterschiedliche Sprach-, Zeichensatz- oder Dateityp-Präferenzen eines Clients durch `mod_negotiation`. Näheres dazu finden Sie in Kapitel 7, »Header und MIME-Einstellungen«.
- `All`: Alle genannten Optionen außer `MultiViews`. Wenn `Options` für ein Verzeichnis nicht angegeben wird, ist dies der Standardwert.

Wenn Sie Optionen für mehrere ineinander verschachtelte Verzeichnisse festlegen, gelten für ein gegebenes Verzeichnis jeweils die speziellsten (für das am weitesten untergeordnete Verzeichnis angegebenen) Werte. Näheres zur Reihenfolge der Abarbeitung unterschiedlicher Container erfahren Sie in Abschnitt 6.2, »Kontexte und Container«. Das folgende Beispiel illustriert ein Missverständnis, das in diesem Zusammenhang auftreten kann:

```
<Directory /usr/share/web>
  Options FollowSymLinks Indexes IncludesNOEXEC
  # ... weitere Einstellungen für /usr/share/web
</Directory>

<Directory /usr/share/web/test>
  Options Includes
  # ... weitere Einstellungen für /usr/share/web/test
</Directory>
```

Im Verzeichnis */usr/share/web* sind die Optionen `FollowSymLinks`, `Indexes` und `IncludesNOEXEC` aktiviert. Im Unterverzeichnis */usr/share/web/test* gilt dagegen *nur* die Option `Includes` – das ist wahrscheinlich nicht das gewünschte Verhal-

ten. Um `Includes` hinzuzufügen, anstatt die anderen Optionen durch `Includes` zu ersetzen, müssten Sie für */usr/share/web/test* diese Variante angeben:

```
<Directory /usr/share/web/test>
  Options FollowSymLinks Indexes Includes
  # ... weitere Einstellungen für /usr/share/web/test
</Directory>
```

Natürlich können Sie die im übergeordneten Verzeichnis gesetzte Option `IncludesNOEXEC` hier zufälligerweise weglassen, da `Includes` deren Fähigkeiten enthält (und ausweitet).

Speziell für untergeordnete Verzeichnisse wird die spezielle Schreibweise `+Option` beziehungsweise `-Option` definiert: Damit lässt sich eine einzelne Option zu den bereits im übergeordneten Kontext gesetzten Optionen hinzufügen beziehungsweise hiervon entfernen. Mithilfe dieser Möglichkeit lässt sich das gerade zitierte Beispiel einfacher schreiben:

```
<Directory /usr/share/web>
  Options FollowSymLinks Indexes IncludesNOEXEC
  # ... weitere Einstellungen für /usr/share/web
</Directory>

<Directory /usr/share/web/test>
  Options +Includes
  # ... weitere Einstellungen für /usr/share/web/test
</Directory>
```

#### AllowOverride

Festlegung der lokal überschreibbaren Konfigurationsdirektiven

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | `<Directory>` |
| **Syntax** | `AllowOverride All\|None\|Direktiventyp [Direktiventyp ...]` |
| **Standardwert** | `All` |

Diese Direktive legt fest, welche Arten von Konfigurationsdirektiven in *.htaccess*-Dateien überschrieben werden dürfen. Da diese Dateien nur in Verzeichnissen innerhalb der `DocumentRoot` erlaubt sind, ist auch `AllowOverride` nur in `<Directory>`-Abschnitten gültig.

Es ist nicht möglich, die in *.htaccess*-Dateien gestatteten Direktiven einzeln anzugeben. Stattdessen definiert `AllowOverride` zahlreiche Gruppenbezeichnungen, die jeweils Konfigurationsanweisungen mit bestimmten Aufgaben zusammenfas-

sen. Damit Sie genau wissen, welche das jeweils sind, finden Sie in diesem Buch bei der Übersicht jeder einzelnen Direktive, die den Kontext *.htaccess* besitzt, in Klammern den Wert, den `AllowOverride` beinhalten muss, um diese Direktive in einer *.htaccess*-Datei zuzulassen.

Hier die einzelnen Werte, die Sie mit Ausnahme von `All` und `None` beliebig mischen können:

- `None`: Im angegebenen Verzeichnis und in allen Unterverzeichnissen ohne anderweitige Einstellung akzeptiert Apache gar keine *.htaccess*-Dateien.
- `FileInfo`: Ermöglicht das Überschreiben der Direktiven, die sich um Dateitypen und -inhalte kümmern.
- `Indexes`: Gestattet das Überschreiben von Direktiven zur automatischen Erzeugung von Verzeichnisindizes (meist in `mod_autoindex` definiert).
- `Limit`: Diese Option erlaubt das Überschreiben der Direktiven `Order`, `Allow` und `Deny` zur hostbasierten Zugriffskontrolle.
- `AuthConfig`: Wenn dieser Wert angegeben wird, dürfen alle Direktiven zur Authentifizierung überschrieben werden (siehe Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«).
- `Options`: Ermöglicht das Überschreiben der bereits besprochenen Direktive `Options` und anderer Verzeichnisoptions-Direktiven.
- `All`: Diese Einstellung gestattet die Verwendung aller bereits genannten Einzelgruppen sowie einiger zusätzlicher Direktiven.

Hier ein Beispiel:

```
<Directory /usr/local/share/apache2/htdocs>
  AllowOverride FileInfo Limit AuthConfig
  # ... weitere Einstellungen für die DocumentRoot
</Directory>
```

Näheres zu *.htaccess*-Dateien finden Sie in Abschnitt 6.2, »Kontexte und Container«.

#### Order

Reihenfolge, in der `Allow`- und `Deny`-Direktiven beachtet werden

| | |
|---|---|
| **Modul** | `mod_access_compat`; bis 2.2.x: `mod_authz_host`; bis 2.0.x: `mod_access` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`Limit`) |
| **Syntax** | `Order Deny,Allow \| Allow,Deny \| Mutual-failure` |
| **Standardwert** | `Deny,Allow` |

*Hinweis*: Diese Direktive ist seit Version 2.3-beta veraltet; sie wurde durch zusätzliche Werte für die in Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«, besprochene Direktive `Require` ersetzt. Zurzeit wird sie aus Kompatibilitätsgründen noch im Modul `mod_access_compat` unterstützt, dürfte aber mittelfristig ganz abgeschafft werden.

Bis Version 2.2 ist diese Direktive allerdings äußerst wichtig. `Order` legt die Reihenfolge fest, in der die Regeln aus den Konfigurationsanweisungen `Allow` und `Deny` angewendet werden, die wiederum bestimmen, welche Hosts auf die entsprechende Ressource zugreifen dürfen.

- `Deny,Allow`: Diese Reihenfolge ist die Voreinstellung. Sie besagt, dass zuerst die `Deny`-Liste und dann die `Allow`-Liste ausgewertet werden. Das bedeutet in der Praxis, dass die Regeln nach folgendem Schema ausgewertet werden: »Der Zugriff ist den angegebenen Hosts verboten mit den folgenden Ausnahmen: …« Die sinnvolle Verwendung setzt natürlich voraus, dass für `Deny` eine allgemeinere Angabe gemacht wird (vorzugsweise `Deny from all`) als für `Allow`. Wenn Sie keine weiteren Direktiven verwenden, ist der Standard bei dieser Einstellung `Allow from all` – alle Hosts dürfen zugreifen.
- `Allow,Deny`: Dies ist die umgekehrte Reihenfolge, die dafür sorgt, dass die `Allow`-Liste vor der `Deny`-Liste ausgewertet wird. Mit anderen Worten: Der Zugriff wird zunächst den angegebenen (üblicherweise allen) Hosts gestattet, bevor er einer kleineren Anzahl verboten wird. Das Standardverhalten ist in diesem Fall `Deny from all`, das heißt, alle müssen zunächst draußen bleiben.
- `Mutual-failure`: veraltetes Synonym für `Allow,Deny`

Bitte beachten Sie, dass zwischen den Aufzählungen `Deny,Allow` und `Allow,Deny` kein Leerzeichen nach dem Komma stehen darf.

#### Allow

Bestimmt, welche Hosts auf einen Site-Bereich zugreifen dürfen.

| | |
|---|---|
| **Modul** | `mod_access_compat`; bis 2.2.x: `mod_authz_host`; bis 2.0.x: `mod_access` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`Limit`) |
| **Syntax** | `Allow from all \| Host \| env=Umgebungsvariable [Host \| env=Umgebungsvariable ...]` |
| **Standardwert** | nicht gesetzt |

*Hinweis*: Diese Direktive ist seit Version 2.3-beta veraltet; sie wurde durch zusätzliche Werte für die in Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«, ersetzt. Zurzeit wird sie aus Kompatibilitätsgründen noch im Modul `mod_access_compat` unterstützt, dürfte aber mittelfristig ganz abgeschafft werden.

Bis Apache 2.2 legt die Direktive `Allow` fest, wer auf eine bestimmte Ressource des Servers zugreifen darf. Die Werte können eines der folgenden Formate annehmen:

- **All**
  Grundsätzlich darf jeder Host auf die Inhalte des Kontextes zugreifen, in dem die Direktive definiert ist. Die entsprechende Apache-2.4-Schreibweise lautet:

  ```
  Require all granted
  ```

- **Domain-Name**
  Bezeichnet Hosts, deren Name der angegebenen Domain und ihren Subdomains entspricht. Apache führt zur Ermittlung des Hostnamens eines zugreifenden Rechners zunächst einen Reverse-DNS-Lookup durch, um den zur IP-Adresse gehörenden Hostnamen zu ermitteln. Anschließend wird zur Sicherheit noch einmal ein Forward-Lookup ausgeführt, um sicherzugehen, dass der Hostname wiederum der IP-Adresse entspricht. Die DNS-Lookups finden in diesem Fall auch dann statt, wenn `HostNameLookups` (siehe Kapitel 11, »Logging«) deaktiviert ist.

  Betrachten Sie das folgende Beispiel:

  ```
  Allow from office.mynet.de
  ```

  Client-Rechner aus den Subdomains *office.mynet.de* und *billing.office.mynet.de* dürfen zugreifen, während Clients aus der Domain *sales.mynet.de* der Zugriff verweigert wird.

  Die 2.4-Variante ist:

  ```
  Require host office.mynet.de
  ```

- **Vollständige IP-Adresse**
  Ein solcher Wert gibt die IP-Adresse eines einzelnen Hosts an, der auf den Server zugreifen darf. Beispiel:

  ```
  Allow from 196.17.8.3
  ```

  Schreiben Sie für Apache 2.4 stattdessen:

  ```
  Require ip 196.17.8.3
  ```

  Wenn Sie in einem Bereich Ihrer Website (oder auf dem ganzen Server) zu Testzwecken nur Zugriffe durch den lokalen Rechner selbst gestatten möchten, müssen Sie im entsprechenden Kontext folgende Direktivenfolge verwenden:

```
Order Deny,Allow
Deny from all
Allow from 127.0.0.1
```

In 2.4 funktioniert es folgendermaßen:

```
Require ip 127.0.0.1
```

- **Unvollständige IP-Adresse**
  Wenn Sie Zugriffe auf ein bestimmtes IP-Teilnetz beschränken möchten, genügt es oft, den gemeinsamen Netzwerkteil der entsprechenden IP-Adressen anzugeben. Das folgende Beispiel erlaubt Zugriffe aus dem Netz 196.17.8.0/24:

  ```
  Allow from 196.17.8
  ```

  Apache 2.4:

  ```
  Require ip 196.17.8
  ```

- **CIDR-Adresse**
  Sie können ein Teilnetz auch nach der CIDR-Logik angeben (siehe Kapitel 1, »TCP/IP-Netzwerke, Internet und WWW«). Wenn die Grenze zwischen Netzwerk- und Hostteil im gewünschten IP-Teilnetz nicht genau an einer Byte-Grenze liegt, ist dies sogar erforderlich. Dieses Beispiel gestattet Zugriffe aus dem Teilnetz 160.76.32.0/19 (Adressbereich 160.76.32.0 bis 160.76.63.255):

  ```
  Allow from 160.76.32.0/19
  ```

  Version 2.4:

  ```
  Require ip 160.76.32.0/19
  ```

- **IP-Adresse/Teilnetzmaske**
  Dies ist eine alternative Schreibweise für CIDR-Adressen. Beispielsweise können Sie das Netzwerk 160.76.32.0/19 auch folgendermaßen angeben:

  ```
  Allow from 160.76.32.0/255.255.224.0
  ```

  Ab 2.3-beta:

  ```
  Require ip 160.76.32.0/255.255.224.0
  ```

- **env=Umgebungsvariable**
  Diese spezielle Form erlaubt Zugriffe nur dann, wenn die angegebene Umgebungsvariable gesetzt ist. Sinnvoll ist dieses Verfahren im Zusammenhang mit der Direktive `SetEnvIf` (siehe Kapitel 14, »CGI«). Diese setzt eine Variable, wenn die Client-Anfrage eine bestimmte Bedingung erfüllt. Das folgende Beispiel erlaubt den Zugriff nur, wenn das `Referer`-Feld der Client-Anfrage mit

*http://www.mynet.de* beginnt, das heißt, wenn die aktuelle Anfrage durch einen Hyperlink von der eigenen Website aus zustande kam:

```
SetEnvIf Referer "^http://www.mynet.de" locallink
Order Deny,Allow
Deny from all
Allow from env=locallink
```

Dies ist eine mögliche Methode, um Gefahren durch Cross-Site-Scripting zu verringern.

Ab Version 2.3 kann folgende Variante verwendet werden:

```
SetEnvIf Referer "^http://www.mynet.de" locallink
Require env locallink
```

Oder sogar noch kürzer:

```
Require expr "%{HTTP_REFERER} =~ m#^http://www.mynet.de"
```

### Deny

Legt fest, welche Hosts nicht auf einen Site-Bereich zugreifen dürfen.

| | |
|---|---|
| **Modul** | `mod_access_compat`; bis 2.2.x: `mod_authz_host`; bis 2.0.x: `mod_access` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`Limit`) |
| **Syntax** | `Deny from all \| Host \| env=Umgebungsvariable [Host \| env=Umgebungsvariable ...]` |
| **Standardwert** | nicht gesetzt |

*Hinweis*: Diese Direktive ist seit Version 2.3-beta veraltet; sie wurde durch zusätzliche Werte für die in Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«, ersetzt. Zurzeit wird sie aus Kompatibilitätsgründen noch im Modul `mod_access_compat` unterstützt, dürfte aber mittelfristig ganz abgeschafft werden.

Die Syntax von `Deny` ist mit `Allow` identisch. Die Direktive bestimmt, welche Hosts vom Zugriff auf eine Server-Ressource ausgeschlossen sind. In welcher Reihenfolge `Allow` und `Deny` ausgewertet werden, regelt die bereits besprochene Direktive `Order`.

Das folgende Beispiel verbietet zunächst einmal allen Hosts den Zugriff und erlaubt ihn anschließend Hosts aus den Domains *office.mynet.de* und *external.mynet.de*:

```
Order Deny,Allow
Deny from all
Allow from office.mynet.de external.mynet.de
```

In Apache 2.4 genügt dazu folgende Zeile:

```
Require host office.mynet.de external.mynet.de
```

Dieses Beispiel erlaubt dagegen allen Hosts den Zugriff, die nicht zur Domain *unsere-konkurrenz.de*[4] gehören:

```
Order Allow,Deny
Allow from all
Deny from unsere-konkurrenz.de
```

In Versionen ab 2.3-beta können Sie dies folgendermaßen ausdrücken:

```
<RequireAll>
  Require all granted
  Require not host unsere-konkurrenz.de
</RequireAll>
```

#### AllowMethods

Legt fest, welche HTTP-Zugriffsmethoden in einem bestimmten Kontext erlaubt sind.

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_allowmethods` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `AllowMethod HTTP-Methode [HTTP-Methode ...]` |
| **Standardwert** | nicht gesetzt |

Das neue Modul `mod_allowmethods` stellt die Direktive `AllowMethods` zur Verfügung. Diese ermöglicht innerhalb eines Verzeichniskontextes die Angabe der HTTP-Anfragemethoden, mit denen Benutzer auf die Ressourcen innerhalb des Verzeichnisses zugreifen dürfen. Dies ist einfacher als mit `<Limit>`-Containern, aber zurzeit gilt das Modul noch als experimentell.

Das folgende Beispiel erlaubt im aktuellen Kontext die Methoden `GET`, `POST` und `HEAD`:

```
AllowMethods GET POST HEAD
```

4 Natürlich nützt das in dieser Form nichts, weil die normalen Büro-PCs der auszuschließenden Konkurrenzfirma wahrscheinlich mit temporären IP-Adressen und Domain-Namen wie xdsl-client-08-15.t-online.de im Netz unterwegs sind. Überdies könnten Sie rechtliche Probleme bekommen (unlauterer Wettbewerb), wenn Sie die Konkurrenz explizit vom Zugriff auf Seiten ausschließen, die ansonsten für alle Welt zugänglich sind.

### Voreinstellung für alle Verzeichnisse

Aus Sicherheitsgründen sollten Sie – wie in einigen speziellen Zusammenhängen bereits erwähnt – zunächst einmal Einstellungen für das Wurzelverzeichnis (/) vornehmen. Sämtliche Zugriffe und Optionen sollten hier verboten werden. Diese Einstellungen werden von allen unspezifizierten Verzeichnissen sowie allen `<Directory>`-, `<Files>`- und `<Location>`-Containern automatisch übernommen. Anschließend können Sie für einzelne untergeordnete Ressourcen jeweils die benötigten Optionen freischalten.

Ein üblicher Grundeinstellungs-Container sieht beispielsweise so aus:

```
<Directory />
  # Alle Optionen deaktivieren
  Options None
  # Sämtliches Überschreiben durch .htaccess dekativieren
  AllowOverride None
  # Zugriffe aller Clients verbieten
  Require all denied
  # Bis 2.2 werden statt "Require all denied"
  # folgende zwei Zeilen verwendet:
  #Order Deny,Allow
  #Deny from all
</Directory>
```

Die vorgefertigte Konfigurationsdatei enthält normalerweise nicht die Angabe `Options None`, sondern `Options FollowSymLinks`. Auf einem Server, den Sie allein verwalten, ist dies kein großes Problem, da Sie selbst entscheiden können, welche symbolischen Links Sie überhaupt anlegen möchten. Andererseits schadet es auch nichts, `FollowSymLinks` global zu deaktivieren und dann zusammen mit den anderen Optionen jeweils im Einzelfall einzuschalten.

### Sinnvolle Einstellungen für die DocumentRoot

Die `DocumentRoot` bildet, wie erwähnt, das Wurzelverzeichnis der Website, die Apache ausliefert, wenn kein virtueller Host angesprochen wurde. Für dieses Verzeichnis müssen Sie demzufolge einige Einstellungen der globalen Verzeichniskonfiguration überschreiben, um die Website im Internet zu veröffentlichen. Hier zunächst ein typisches Beispiel:

```
<Directory /usr/local/share/apache2/htdocs>
  Options Indexes FollowSymLinks MultiViews
  AllowOverride None
  Require all granted
```

```
  # Bis 2.2 folgende zwei Zeilen anstelle von "Require all granted":
  #Order allow,deny
  #Allow from all
</Directory>
```

Bei dieser Konfiguration sind die Optionen `Indexes`, `FollowSymLinks` und `MultiViews` eingeschaltet. Wenn Sie mit Server Side Includes arbeiten möchten (siehe Kapitel 16, »SSI und Filter«), müssen Sie zusätzlich `Includes` aktivieren. Für CGI sollte dagegen in der Regel nicht `ExecCGI` verwendet werden, sondern ein per `ScriptAlias` in die `DocumentRoot` abgebildetes *cgi-bin*-Verzeichnis (Genaueres dazu in Kapitel 14, »CGI«).

Die Verwendung von *.htaccess*-Dateien wurde hier ebenfalls abgeschaltet. In Abschnitt 6.2, »Kontexte und Container«, wurde bereits begründet, warum dies in den meisten Fällen ratsam ist.

Zu guter Letzt wurde der Zugriff auf die Site für beliebige Hosts gestattet.

## 6.4 Zusammenfassung

Die Konfigurationsdatei *httpd.conf* ist einer der wichtigsten Bestandteile des Apache-Webservers. Sie legt detailliert fest, in welchem Umfang und auf welche Weise sich der Server um Client-Anfragen kümmern soll. Sie ist ab Werk in drei Abschnitte unterteilt:

- Konfiguration der Umgebung
- Konfiguration des »Hauptservers«
- Einstellungen für virtuelle Hosts

Die Datei besteht aus zahlreichen Konfigurationsanweisungen oder Direktiven. Einige von ihnen gehören zum Funktionskern von Apache (`core`), andere zu den plattformspezifischen MPM-Modulen. Viele sind dagegen in den zahlreichen Zusatzmodulen definiert und stehen nur zur Verfügung, wenn diese Module aktiviert sind. Jede Direktive besteht aus ihrem festgelegten Namen und einem oder mehreren durch Leerzeichen getrennten Werten. Die Direktiven müssen jeweils in einer eigenen Zeile stehen; zu lange Direktiven können Sie aufteilen, wenn Sie an das Ende einer umbrochenen Zeile einen Backslash (`\`) setzen. Der Inhalt einer Zeile nach einer Raute (#) gilt als Kommentar und wird ignoriert.

Die wichtigsten Grunddirektiven für die ersten beiden Abschnitte wurden in diesem Kapitel behandelt; virtuelle Hosts kommen erst in Kapitel 12, »Skalierung und Performance-Tuning«, ausführlich zur Sprache.

Nicht jede Konfigurationseinstellung kann in jedem beliebigen Kontext stehen. Im Wesentlichen gibt es vier verschiedene Kontexte für Direktiven:

- den Server-Kontext
- `<VirtualHost>`-Container für virtuelle Hosts
- Verzeichnis- und Datei-Container: `<Directory>`, `<Location>`, `<Files>`
- ausgelagerte Konfigurationsanweisungen in *.htaccess*-Dateien

Mit den Direktiven, die in diesem Kapitel vorgestellt wurden, lässt sich ein einfacher Server betreiben, der statische Dokumente ausliefert. Die wichtigsten Raffinessen für den Betrieb einer realistischen Site lernen Sie in den beiden folgenden Kapiteln kennen.

*»Was ist der Körper,*
*wenn das Haupt ihm fehlt?«*
*– William Shakespeare*

# 7 Header und MIME-Einstellungen

Bereits in Kapitel 2, »Funktionsweise von Webservern«, wurde die Bedeutung der HTTP-Header erläutert. In diesem Kapitel werden sämtliche Direktiven und Module behandelt, die die Manipulation dieser Header ermöglichen. Zu den wichtigsten Headern gehören die diversen MIME-Einstellungen wie Medientyp oder Sprache. Unter anderem werden sie zur Content-Negotiation herangezogen, die ebenfalls in diesem Kapitel behandelt wird.

## 7.1 HTTP-Header manipulieren

Apache bietet Ihnen die Möglichkeit, HTTP-Header auf vielfältige Weise zu manipulieren. Es gibt Module und Direktiven, mit denen sich zusätzliche Header erstellen oder vorhandene ändern lassen.

Das wichtigste Modul zur Header-Manipulation ist `mod_headers`. Es enthält Direktiven, mit denen sich beinahe jeder Header der HTTP-Antwort vor der Auslieferung an den Client hinzufügen, entfernen oder modifizieren lässt. Auch das veraltete Modul `mod_cern_meta`, das eine ähnliche Aufgabe erfüllt, gehört noch heute zur Apache-Distribution. Das Spezialmodul `mod_expires` sowie die beiden Core-Direktiven `ContentDigest` und `FileETag`, die alle ganz spezielle Header setzen, werden ebenfalls in diesem Abschnitt behandelt.

### 7.1.1 MD5-Digest und ETag

Im `core` sind zwei Konfigurationseinstellungen definiert, die besondere Header erzeugen: `ContentDigest` generiert den Header `Content-MD5` (einen MD5-Hash des Dateiinhalts), `FileETag` bestimmt den Aufbau des Headers `ETag`.

**ContentDigest**

Erzeugen von `Content-MD5`-Headern

| Modul | Core |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Options) |
| Syntax | ContentDigest On\|Off |
| Standardwert | Off |

Wie in Kapitel 2, »Funktionsweise von Webservern«, bereits beschrieben wurde, enthält der HTTP-Entity-Header `Content-MD5` einen MD5-Message-Digest des gelieferten Dokuments. Dieser Code wird nach einem komplexen Verfahren berechnet und repräsentiert einen – viel längeren – ursprünglichen Inhalt eindeutig. Eine Änderung des Dokuments hat demzufolge auch eine Änderung des MD5-Digests zur Folge. Deshalb wird MD5 häufig dann verwendet, wenn es darum geht, die Integrität von Inhalten zu gewährleisten – beispielsweise bei Open-Source-Downloads von verschiedenen Mirror-Servern (wie bei der Apache Software Foundation selbst) oder bei verschiedenen Formen der verschlüsselten Kommunikation.

Browser oder Proxys können diesen Wert verwenden, um zu überprüfen, ob der gelieferte Inhalt durch einen Übertragungsfehler oder mutwillig verändert wurde. Ein `Content-MD5-Header` sieht beispielsweise so aus:

```
Content-MD5: UdtczArEb7L8iy4UUKLJeg==
```

Der Webserver erstellt nur für diejenigen Dokumente einen MD5-Digest, die der `core`-Handler liefert. Für dynamische Inhalte steht diese Möglichkeit also nicht zur Verfügung. Da für jede Anfrage ein neuer Digest errechnet wird, sollten Sie dieses Verfahren auf einem besonders stark beanspruchten Server nicht einsetzen.

Bitte beachten Sie, dass im August 2004 Sicherheitsmängel an MD5 festgestellt wurden. Für die reine Digest-Anwendung im HTTP-Header sind diese zwar nicht relevant, aber in sicherheitsrelevanteren Bereichen dürfte der Algorithmus mittelfristig ersetzt werden.

### FileETag

Informationen, die in die `ETag`-Erzeugung einbezogen werden

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | FileETag All\|None\|Komponente [Komponente ...] |
| Standardwert | INode MTime Size (ab 2.3 nur noch MTime Size) |

Ein `ETag` (Entity-Tag) ist ein hexadezimaler Wert, der Auskunft über die Identität einer Datei gibt. Diese Information wird vor allem zur Cache-Verwaltung verwendet: Ein gleich gebliebenes `ETag` besagt in der Regel, dass die im Cache befindliche Kopie noch aktuell ist.

Mit der Direktive `FileETag` können Sie bestimmen, ob ein ETag erzeugt wird und welche Dateiinformationen einbezogen werden sollen. Bis auf `All` und `None` können Sie die Werte beliebig mischen. Folgende Angaben sind möglich:

- `None`
  Es wird gar kein ETag erzeugt.
- `INode`
  Die `INode`-Nummer der Datei wird für die Berechnung verwendet.
- `MTime`
  Datum und Uhrzeit der letzten Änderung werden in die ETag-Berechnung einbezogen.
- `Size`
  Die Dateigröße wird zur Erzeugung des ETags benutzt.
- `All`
  Alle drei Werte werden verwendet. Dies ist eine Kurzfassung für die folgende Schreibweise:

```
FileETag INode MTime Size
```

### 7.1.2 mod_headers

Das Modul `mod_headers` ist eine Neuentwicklung für Apache 2.0, die im Verlauf der 2.0- und 2.2-Revisionen mehrmals geringfügig erweitert wurde. Das Modul ermöglicht die direkte Manipulation von HTTP-Headern. Zu diesem Zweck definiert es die beiden Direktiven `Header` und `RequestHeader`; sie ermöglichen die Änderung der HTTP-Header für die Antwort beziehungsweise die Client-Anfrage. Sie können nach Belieben Header hinzufügen, ergänzen, ändern oder entfernen.

#### Header

HTTP-Antwort-Header setzen/ändern

| Seit Version | 2.0, `onsuccess\|always` seit 2.0.51, `early` seit 2.1-beta, `edit` seit 2.2.4, `merge` seit 2.2.9 |
|---|---|
| Modul | `mod_headers` |
| Kontext | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |

| Syntax | `Header [onsuccess\|always]`<br>`set\|append\|add\|unset\|echo\|edit\|merge`<br>`Headername [Wert] [Ersatz]`<br>`[early\|env=[!]Variable]` |
|---|---|
| Standardwert | nicht gesetzt |

Mithilfe von `Header` können Sie die meisten HTTP-Header der Server-Antwort modifizieren.

Die optionale Bedingung (`onsuccess` oder `always`) ist seit Version 2.0.51 verfügbar. Sie wählt die interne Header-Tabelle aus, für die die Direktive gesetzt werden soll: `onsuccess` modifiziert den Header nur bei erfolgreichen Anfragen (HTTP-Statuscode 2xx), `always` (Standard) dagegen bei allen Anfrageergebnissen.

Mit dem anschließenden Befehlsargument der Direktive bestimmen Sie, welche Operation Sie mit dem gewünschten Header durchführen möchten:

- `Header set Headername Wert`
  Der angegebene Header wird gesetzt. Falls bereits ein Header mit diesem Namen vorhanden war, wird sein Wert durch den hier angegebenen ersetzt. Andernfalls wird der Header neu hinzugefügt.
- `Header append Headername Wert`
  Falls der gewünschte Header bereits existiert, wird der Wert, den Sie angeben, durch ein Komma getrennt an den bisherigen Inhalt des Headers angefügt. HTTP gestattet solche Wertlisten bei den meisten Headern (beachten Sie hierzu Kapitel 2, »Funktionsweise von Webservern«, vor allem wegen möglicher Ausnahmen). Sollte der Header noch gar nicht existieren, wird er auch bei `append` neu angelegt.
- `Header add Headername Wert`
  Apache erzeugt einen neuen Header mit der angegebenen Bezeichnung unabhängig davon, ob ein solcher bereits existiert. Das kann natürlich dazu führen, dass es zwei Header mit demselben Namen gibt. Da diese Situation bei vielen wichtigen Headern problematisch sein kann, sollten Sie in aller Regel `append` bevorzugen.
- `Header unset Headername`
  Der oder die Header mit dem angegebenen Namen werden komplett entfernt, falls sie existieren.
- `Header echo RegExp`
  Falls die Client-Anfrage den angegebenen Header enthält, wird dieser in der Antwort mit demselben Wert zurückgesendet. Dies ist der einzige `Header`-Befehl, bei dem das nächste Argument (die Header-Bezeichnung) ein regulärer Ausdruck sein darf.

- `Header edit Headername RegExp Ersatzstring`
- Falls der bestehende Wert des angegebenen Headers dem als Wert verwendeten regulären Ausdruck entspricht, wird dieser Wert durch den Ersatzstring ersetzt. Wenn der reguläre Ausdruck Teilausdrücke in runden Klammern enthält, können die aufgrund dieser Teilausdrücke gefundenen Treffer – von links nach rechts und von außen nach innen gezählt – durch `$1`, `$2` usw. als sogenannte *Back-Referenzen* im Ersatzstring verwendet werden.
- `Header merge Headername Wert`

  Dieses neueste Kommando ist eine intelligentere Alternative zu `append`: Der Wert wird nur dann zum angegebenen Header hinzugefügt, wenn dessen durch Komma getrennte Wertliste diesen Wert nicht bereits enthält.

Als Header-Namen können Sie fast alle offiziellen HTTP-Header angeben (Ausnahmen siehe Kasten »Einschränkungen«). Eine vollständige Liste der Header finden Sie in Kapitel 2, »Funktionsweise von Webservern«. Abgesehen davon können Sie auch eigene Header einfügen, die nicht zum HTTP-Standard gehören. Dies ist oft sehr nützlich für Log-Dateien oder zur Kommunikation mit speziellen Client-Programmen, die keine Browser sind. Um Konflikte mit zukünftigen HTTP-Erweiterungen auszuschließen, sollten alle Zusatz-Header mit `X-` beginnen (z. B. `X-My-Header`). Header mit diesem Präfix werden ausdrücklich als Erweiterungs-Header betrachtet.

Übrigens spielt Groß- und Kleinschreibung in Header-Bezeichnungen keine Rolle, dennoch sollten Sie sich an die vorgegebene Schreibweise der Standard-Header halten und diese Konvention für Ihre eigenen Header übernehmen. Wenn Sie möchten, können Sie den Header-Namen mit dem Doppelpunkt abschließen, der diesen vom Wert trennt. Das ist aber nicht erforderlich.

**Einschränkungen**

Bitte beachten Sie, dass sich die folgenden Header nicht überschreiben – also weder ändern noch entfernen – lassen, weil sie in den meisten Fällen erst nach der Verarbeitung der Antwort durch `mod_headers` hinzugefügt werden:

```
Content-Type
Content-Length
Server
Date
```

Sie sollten auch von dem Versuch absehen, mittels `Header add` einen zusätzlichen Header mit einem dieser Titel hinzuzufügen – dies dürfte in vielen Fällen zu Problemen führen.

(Selbstverständlich kann auch die HTTP-Statuszeile, die erste Zeile der Server-Antwort, nicht auf diese Weise verändert werden. Sie ist aber ohnehin kein richtiger Header.)

Der Wert, den Sie bei `set`, `add`, `append` und `merge` angeben können, darf im Prinzip jeder beliebige String sein. Falls er Leerzeichen enthält, müssen Sie ihn wie

üblich in Anführungszeichen setzen. Wenn Sie einen Standard-HTTP-Header setzen, sollten Sie sich natürlich an den RFC-2616-Standard halten, der die Syntax des entsprechenden Headers bestimmt. Das Format aller vordefinierten Header wird in Kapitel 2, »Funktionsweise von Webservern«, genau beschrieben.

`mod_headers` definiert drei spezielle Platzhalter, die Sie zusätzlich zum gewöhnlichen Text in den Wert eines Headers setzen können:

- `%t`: der Zeitpunkt, zu dem die Anfrage empfangen wurde. Die Angabe erfolgt in Mikrosekunden seit EPOCH (01.01.1970, 00:00 Uhr UTC). Apache stellt dem Wert das Präfix `t=` voran.

  Beispiel: `t=1079190280851376`

- `%D`: die Zeitspanne zwischen dem Empfang der Anfrage und dem Versand der Antwort (beziehungsweise der Erzeugung des betreffenden Headers) in Mikrosekunden mit vorangestelltem `D=`.

  Beispiel: `D=350504`

- `%{EnvVar}e`: Dies fügt den Wert der angegebenen Umgebungsvariablen ein. Eine Liste aller Standard-Umgebungsvariablen sowie eine Anleitung zur Definition eigener Variablen finden Sie in Kapitel 14, »CGI«. Beispielsweise könnte `%{SERVER_SOFTWARE}e` (abhängig von der Direktive `ServerTokens` – siehe Kapitel 6, »Grundkonfiguration«) die folgende Ausgabe liefern: `Apache/2.2.9 (Unix)`.

- `%{SSLEnvVar}s` (verfügbar seit Version 2.1-beta): Diese Option fügt eine der in Kapitel 10, »Gesicherte Verbindungen«, beschriebenen SSL-Variablen ein.

Nach der theoretischen Beschreibung der Direktive hier einige konkrete Beispiele:

- `Header append Content-Language jp`
  Diese Anweisung hängt an den Header `Content-Language` das Sprachkürzel `jp` (Japanisch) an. Falls dieser Header noch nicht existiert, wird er mit dem angegebenen Wert neu erzeugt.

  An dieser Stelle ist der Unterschied zwischen `append` und dem neuen Befehl `merge` interessant. Angenommen, der bisherige Inhalt des Headers `Content-Language` sieht so aus:

  ```
  Content-Language: de, jp, en
  ```

  Dann erzeugt die `append`-Anweisung folgenden Header:

  ```
  Content-Language: de, jp, en, jp
  ```

Das dürfte für Clients zwar kein großes Problem sein, aber eleganter ist dennoch folgende Variante:

```
Header merge Content-Language jp
```

Da der Wert `jp` bereits vorhanden ist, wird er auf diese Weise nicht neu hinzugefügt.

- `Header set Cache-Control no-cache`
  Hier wird der Header `Cache-Control` mit dem Wert `no-cache` überschrieben (oder neu erstellt). Clients und Proxys, die sich danach richten, werden dieses Dokument nicht zwischenspeichern.
- `Header add X-Time "Request Received at %t."`
  Diese Formulierung fügt einen Header namens `X-Time` hinzu, und zwar auch dann, wenn bereits ein solcher existiert (was in diesem Fall natürlich unwahrscheinlich ist). Der Wert des Headers besteht aus dem String `Request Received at` sowie dem Zeitpunkt, zu dem Apache die Anfrage empfangen hat. Beispiel: `Request Received at t=1079190287526532`.
- `Header echo ^X\-`
  Sämtliche Erweiterungs-Header aus der Client-Anfrage (die mit `X-` beginnen) werden wieder an den Client zurückgeschickt.
- `Header edit Content-Language de(\-[A-Z]+)? de`
  Hier werden beliebige `de`-Varianten des Headers `Content-Language` (z. B. `de-DE` für Deutschland oder `de-AT` für Österreich) durch ein einfaches `de` (Deutsch allgemein) ersetzt. Details über die Funktionsweise regulärer Ausdrücke finden Sie im Rahmen der Beschreibung von `mod_rewrite` im nächsten Kapitel sowie in einer Kurzübersicht in Anhang G.

`Header` kann noch ein optionales Zusatzargument erhalten: entweder `early` (seit Version 2.1-beta) oder `env=Variable`. Die Option `early` ändert den angegebenen Header bereits zu Beginn der Anfrageverarbeitung. Dies ist für Entwickler nützlich, um Modulen, die erst danach zum Einsatz kommen, geänderte Bedingungen zu liefern. Auf einem Produktionsserver nützt dieser Modus dagegen nichts, weil die Wahrscheinlichkeit groß ist, dass die entsprechenden Header während der Verarbeitung noch geändert werden. Beachten Sie zudem, dass `early` nur im Server- und `<VirtualHost>`-Kontext eingesetzt werden kann, weil die Umwandlung der URL in einen Dateipfad zu diesem Zeitpunkt noch nicht stattgefunden hat.

`env=Variable` führt die Header-Operation nur aus, wenn die angegebene Umgebungsvariable gesetzt ist; `env=!Variable` dagegen nur dann, wenn sie *nicht* gesetzt ist. Dies dient vornehmlich der Überprüfung selbst gesetzter Variablen.

Direktiven wie `SetEnv` oder `SetEnvIf`, mit denen Sie Variablen definieren können, werden in Kapitel 14, »CGI«, näher erläutert. Hier nur ein Beispiel:

- `SetEnvIf X-Get-Duration .+ WANT_DURATION`
  `Header set X-Duration "Request took %D." env=WANT_DURATION`
  `SetEnvIf` überprüft, ob die Anfrage den Erweiterungs-Header `X-Get-Duration` enthält (das ist der Fall, wenn dessen Wert aus mindestens einem Zeichen besteht; als regulärer Ausdruck: `.+`). Wenn dies also zutrifft, wird die Variable `WANT_DURATION` ohne besondere Wertangabe gesetzt. Die zweite Zeile wird nur ausgeführt, wenn `WANT_DURATION` gesetzt ist, und erzeugt in diesem Fall einen Erweiterungs-Header namens `X-Duration`. Sein Wert enthält die Verarbeitungsdauer der Anfrage in Mikrosekunden und lautet insgesamt z. B. so: `Request took D=419382`.

  Dieses Beispiel könnte nützlich sein, um ein eigenes Benchmark-Programm zu schreiben, das als Client fungiert und den Header `X-Get-Duration` sendet. Da Apache den Antwort-Header `X-Duration` nur in diesem speziellen Fall erzeugt, werden »normale« Clients nicht damit behelligt.

Mehr Anwendungsbeispiele finden Sie an verschiedenen passenden Stellen weiter hinten in diesem Buch.

#### RequestHeader

Ankommende Anfrage-Header vor der Verarbeitung setzen oder ändern

| | |
|---|---|
| **Seit Version** | 2.0, `early` seit 2.1-beta, `edit` seit 2.2.4, `merge` seit 2.2.9, `edit*` seit 2.3-beta |
| **Modul** | `mod_headers` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `RequestHeader set\|append\|add\|unset\|edit\|edit*\|merge`<br>`  Headername [Wert]`<br>`  [early\|env=[!]Variable]` |
| **Standardwert** | nicht gesetzt |

Neben Antwort-Headern kann Apache auch Anfrage-Header manipulieren, und zwar bevor die Anfrage normal verarbeitet wird. Das erscheint auf den ersten Blick wenig sinnvoll, weil gerade die vom Client gesendeten Header darüber entscheiden sollten, wie eine Anfrage behandelt wird. Dennoch kann es mitunter praktisch sein, bestimmte Teile der Konfiguration gezielt über die Eigenschaften der Client-Anfrage zu täuschen.

Die Syntax von `RequestHeader` ist weitgehend identisch mit derjenigen der Direktive `Header`. Lediglich die Anweisung `echo` ist nicht vorhanden, da sie in diesem Zusammenhang keinen Sinn ergeben würde. Außerdem können Sie die Änderung eines Anfrage-Headers nicht von einer Umgebungsvariablen abhängig machen.

Für Apache 2.4 wurde zudem die Variante `edit*` eingeführt. Der Unterschied zum »normalen« `edit` besteht darin, dass hier *jede* Fundstelle des regulären Ausdrucks ersetzt wird und nicht nur die erste. Beispiele:

- `RequestHeader unset Accept-Charset`
  Der Header `Accept-Charset` wird aus der Anfrage entfernt. Damit enthält die Anfrage keinerlei Zeichensatzvorgaben mehr. Dies könnte für ein bestimmtes Verzeichnis Ihrer Site sinnvoll sein, das ohnehin nur Dokumente mit einem bestimmten Zeichensatz enthält.
- `RequestHeader set X-Filter On`
  Dies ergänzt die Anfrage um den Erweiterungs-Header `X-Filter: On`. Dies könnte sinnvoll sein, um in einem bestimmten Kontext die Verarbeitung aller Anfragen durch einen `InputFilter` zu erzwingen (siehe Kapitel 16, »SSI und Filter«).

Bei Anfrage-Headern bedeutet die Option `early`, dass die Header unmittelbar nach Eintreffen der Anfrage geändert werden; andernfalls werden sie erst in der sogenannten *Fixup-Phase* (unmittelbar vor der Erzeugung der Server-Antwort) modifiziert. Näheres über die Phasen der HTTP-Anfrageverarbeitung lesen Sie in Kapitel 17, »Apache erweitern«.

### 7.1.3 mod_expires

Für Caching- und Proxy-Operationen ist es sehr wichtig, zu wissen, wie lange ein Dokument gültig sein soll. Das Standardmodul `mod_expires` kümmert sich darum, den Antwort-Header `Expires` zu generieren, der über dieses »Haltbarkeitsdatum« Auskunft gibt.

#### ExpiresActive

Schaltet die Erzeugung von `Expires`-Headern ein/aus

| | |
|---|---|
| **Modul** | `mod_expires` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`Indexes`) |
| **Syntax** | `ExpiresActive On\|Off` |
| **Standardwert** | nicht gesetzt |

Mithilfe von ExpiresActive wird für einen bestimmten Kontext (und seine Subkontexte) zunächst einmal grundsätzlich bestimmt, ob Expires-Header erzeugt werden sollen. Wenn Sie diese Information in einem Kontext benötigen, müssen Sie die folgende Zeile hinzufügen:

```
ExpiresActive On
```

Wenn Sie die Erstellung von Expires-Headern in einem Subkontext, beispielsweise in einem <Location>-Container oder einer .htaccess-Datei, deaktivieren möchten, müssen Sie dort Folgendes schreiben:

```
ExpiresActive Off
```

#### ExpiresDefault

Verfahren und Zeitangabe für das Verfallsdatum

| | |
|---|---|
| **Modul** | mod_expires |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| **Syntax** | ExpiresDefault M<Sekunden>\|A<Sekunden> |
| **Standardwert** | nicht gesetzt |

Diese Direktive legt fest, wie der Wert für den Expires-Header berechnet werden soll. Die beiden möglichen Angaben sind die Buchstaben M oder A, gefolgt von einer Zeitangabe in Sekunden.

- M (Modified) bedeutet, dass die Berechnung auf Datum und Uhrzeit der letzten Änderung des Dokuments basieren soll. Alle Clients und Proxys erhalten also dieselbe Gültigkeitsinformation. Das ist besonders nützlich für Dokumente, die regelmäßig automatisch generiert werden. Beispielsweise könnten Sie für eine Datei, die jede Nacht um 00:00 Uhr (das heißt alle 24 Stunden oder alle 86.400 Sekunden) aktualisiert wird, Folgendes schreiben:

  ```
  ExpiresDefault M86400
  ```

- A (Access) legt der Berechnung der Gültigkeit den Zeitpunkt der Client-Anfrage zugrunde. Das Dokument bleibt also für die angegebene Zeitspanne seit seiner Auslieferung an einen bestimmten Client oder Proxy gültig. Dieses Verfahren empfiehlt sich eher für Dokumente, die in unregelmäßigen Abständen manuell geändert werden. Das folgende Beispiel definiert eine Gültigkeitsdauer von drei Stunden seit der Anfrage:

  ```
  ExpiresDefault A10800
  ```

In beiden Fällen wird die angegebene Sekundenzahl zu dem entsprechenden Zeitpunkt hinzugerechnet und als Verfallsdatum zurückgegeben, weil der Wert des Headers `Expires` keine Zeitspanne ist, sondern eine RFC-1123-Zeitangabe. Diese sieht beispielsweise so aus:

```
Expires: Sun, 17 Aug 2008 23:00:00 GMT
```

Neben der bisher vorgestellten Art der Zeitangabe für `ExpiresDefault` gibt es noch ein alternatives Verfahren, dessen Syntax sich wie folgt zusammenfassen lässt:

```
ExpiresDefault "<Basis> [plus] <Anzahl> <Typ>
[<Anzahl> <Typ> ...]"
```

Die Anführungszeichen sind wie üblich erforderlich, weil das Argument Leerzeichen enthält. Die einzelnen Bestandteile bedeuten Folgendes:

- `<Basis>` ist eine ausführlichere Schreibweise für `A` oder `M`:
  - `modification` ist das Änderungsdatum
  - `access` oder `now` stehen für den Zugriffszeitpunkt
- Das Schlüsselwort `plus` ist optional. Es verdeutlicht die Tatsache, dass die nachfolgende Zeitangabe zum Basiszeitpunkt addiert wird.
- Die `<Anzahl>` ist eine beliebige positive ganze Zahl.
- Als `<Typ>` können Sie folgende Angaben verwenden:
  - `year`/`years` (Jahre)
  - `month`/`months` (Monate)
  - `week`/`weeks` (Wochen)
  - `day`/`days` (Tage)
  - `hour`/`hours` (Stunden)
  - `minute`/`minutes` (Minuten)
  - `second`/`seconds` (Sekunden)

Dieses Beispiel bestimmt eine Gültigkeitsdauer von vier Tagen seit dem Client-Zugriff:

```
ExpiresDefault "access plus 4 days"
```

Mit der folgenden Angabe wird dagegen festgelegt, dass die Dateien im aktuellen Kontext eineinhalb Tage seit ihrer letzten Änderung gültig bleiben:

```
ExpiresDefault "modification plus 1 day 12 hours"
```

#### ExpiresByType

Setzt `Expires`-Header je nach MIME-Type

| Modul | `mod_expires` |
|---|---|
| Kontext | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`Indexes`) |
| Syntax | `ExpiresByType MIME-Type M<Sekunden>\|A<Sekunden>` |
| Standardwert | nicht gesetzt |

Mithilfe von `ExpiresByType` lässt sich das Verfallsdatum in einem bestimmten Kontext je nach MIME-Type bestimmen. Das erste Argument ist der gewünschte MIME-Type, der nach sämtlichen weiter unten in diesem Kapitel erläuterten Kriterien ermittelt wird. Der zweite Parameter ist eine Methoden- und Sekundenangabe wie bei `ExpiresDefault`.

Diese beiden Beispiele legen als Gültigkeitsdauer für PDF-Dokumente im aktuellen Kontext 14 Tage seit dem Client-Zugriff fest:

```
ExpiresByType application/pdf A1209600
ExpiresByType application/pdf "now plus 2 weeks"
```

Die folgenden äquivalenten Beispiele bestimmen dagegen, dass GIF-Bilder zweieinhalb Stunden seit ihrer letzten Änderung gültig sein sollen:

```
ExpiresByType image/gif M9000
ExpiresByType image/gif "modification plus 2 hours 30 minutes"
```

### 7.1.4 mod_asis

Wenn Sie das Modul `mod_asis` aktivieren, steht Ihnen der im nächsten Abschnitt beschriebene Handler `send-as-is` zur Verfügung. Dieser sorgt dafür, dass Apache für die entsprechenden Dokumente keine Standard-Antwort-Header generiert. Diese werden dann als Bestandteil der Datei selbst erwartet und vom sonstigen Inhalt durch eine Leerzeile getrennt.

In der Regel wird der Handler `send-as-is` für die spezielle Dateiendung `.asis` aktiviert:

```
AddHandler send-as-is asis
```

Anschließend können Sie beispielsweise das folgende Dokument unter dem Namen `test.html.asis` abspeichern:

```
Status: 200 OK
Content-type: text/html
```

```
Content-language: de
Cookie: test=Testwert

<html>
<head>
<title>Testdokument</title>
</head>
<body>
Dies ist nur ein Test
</body>
</html>
```

Apache fügt lediglich die Header `Server`, `Date` und `Content-Length` hinzu. Außerdem wird der Pseudo-Header `Status` aus dem `asis`-Dokument in eine HTTP-Statuszeile (je nach Client-Anfrage mit HTTP/1.0 oder HTTP/1.1) umgewandelt. Die tatsächlich gesendeten Header sehen also folgendermaßen aus:

```
HTTP/1.1 200 OK
Date: Tue, 19 Aug 2008 22:12:16 GMT
Server: Apache/2.2.10 (Unix)
Content-language: de
Cookie: test=Testwert
Content-Length: 101
Content-Type: text/html; charset=ISO-8859-1
```

Praktisch kann dieser Handler für CGI-Skripte sein, die ihre Header dynamisch erzeugen. Das Verfahren ersetzt hier die in früheren Apache-Versionen unterstützten NPH-CGIs (Non-parsed Header).

### 7.1.5 mod_cern_meta

Wie der Name schon sagt, stellt `mod_cern_meta` eine Fähigkeit zur Verfügung, mit der bereits der ursprüngliche CERN-Webserver ausgestattet war: das Hinzufügen von HTTP-Antwort-Headern aus sogenannten *Meta-Dateien*. Das Verfahren wird nur noch aus Kompatibilitätsgründen unterstützt und sollte bei einer neuen Website durch die entsprechenden Funktionen des bereits vorgestellten, viel leistungsfähigeren Moduls `mod_headers` ersetzt werden.

Sie müssen für jede Datei, die zusätzliche Header erhalten soll, eine eigene Meta-Datei erstellen. Diese Datei muss sich in dem durch `MetaDir` definierten Unterverzeichnis des aktuellen Verzeichnisses befinden; der vorgeschriebene Name der Datei entspricht dem Namen der Originaldatei mit der zusätzlichen Endung, die durch `MetaSuffix` angegeben wird. Standardmäßig ist das Meta-Verzeichnis `.web` und das Suffix `.meta`. Die Meta-Datei für die Datei `hallo.html` in einem gegebenen Verzeichnis wäre also – relativ dazu – *.web/hallo.html.meta*.

Die Meta-Datei selbst ist eine einfache Textdatei, die pro Zeile einen HTTP-Header enthält. Beachten Sie, dass sich Header mit diesem Verfahren nur hinzufügen, aber nicht modifizieren lassen. Beispiel:

```
Expires: Mon, 18 Aug 2008 20:37:00 GMT
X-More-Info: "Header from a Meta File"
```

### MetaFiles

CERN-Meta-Informationen ein-/ausschalten

| | |
|---|---|
| **Modul** | mod_cern_meta |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| **Syntax** | MetaFiles on\|off |
| **Standardwert** | off |

Nachdem das Modul mod_cern_meta aktiviert wurde, können Sie mithilfe dieser Direktive die grundsätzliche Unterstützung für Meta-Dateien einschalten oder in einem Subkontext wieder ausschalten. Beispiel:

```
MetaFiles on
```

### MetaDir

Name des Verzeichnisses mit CERN-Meta-Daten

| | |
|---|---|
| **Modul** | mod_cern_meta |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| **Syntax** | MetaDir Verzeichnis |
| **Standardwert** | .web |

Diese Direktive gibt den Namen des Unterverzeichnisses an, in dem sich die Meta-Dateien für die Dokumente des aktuellen Verzeichnisses befinden. Die Angabe ist relativ zum jeweiligen Verzeichnis. Wenn Sie die Meta-Dateien in demselben Verzeichnis ablegen möchten wie die eigentlichen Dateien, müssen Sie Folgendes schreiben:

```
MetaDir .
```

### MetaSuffix

Dateiendung für Dateien mit CERN-Meta-Informationen

| Modul | mod_cern_meta |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| Syntax | MetaSuffix Dateiendung |
| Standardwert | .meta |

Mit dieser Direktive wird die zusätzliche Dateiendung festgelegt, die eine Datei als CERN-Meta-Datei ausweist. An der Voreinstellung `.meta` müssen Sie in der Regel nichts ändern.

## 7.2 MIME-Konfiguration

Webserver und Browser setzen MIME-Informationen ein, um sich gegenseitig über die gelieferten beziehungsweise bevorzugten Dateitypen, Komprimierungsarten, Zeichensätze und Sprachen zu informieren. Zu diesem Zweck verwendet der Server die verschiedenen `Content-*`-Header, während Clients diverse `Accept-*`-Header benutzen.

Diese Informationen sind für eine funktionsfähige Website äußerst wichtig. Deshalb können Sie bei Apache 2 zahlreiche Einstellungen vornehmen, die bestimmen, welche MIME-Informationen an Clients übermittelt werden sollen:

- Im Core sind Direktiven definiert, mit denen die Standardwerte für die verschiedenen MIME-Aspekte eingestellt werden. Beispielsweise legt `DefaultType` den MIME-Type für den `Content-Type`-Header fest, den Apache ausliefert, wenn sich der Typ einer Datei nicht anderweitig ermitteln lässt.
- Das Modul `mod_mime` liefert die größte Fülle von sich auf MIME beziehende Direktiven. Ihnen allen ist gemeinsam, dass sie MIME-Einstellungen und andere Zuordnungen auf der Basis von Dateiendungen vornehmen. Die wichtigste Konfigurationsanweisung ist `TypesConfig`: Sie bestimmt den Pfad der Hauptzuordnungsdatei für MIME-Types.

  Beachten Sie, dass eine bestimmte Gruppe von `mod_mime`-Konfigurationseinstellungen nicht in diesem Kapitel behandelt wird: Die Direktiven zur Filterzuweisung finden Sie vollständig in Kapitel 16, »SSI und Filter«.
- Eine interessante Aufgabe erfüllt das Modul `mod_mime_magic`: Falls sich der MIME-Type nicht aus der Dateiendung oder einer verzeichnisweiten Voreinstellung ermitteln lässt, versucht es, den Dateityp aus dem Inhalt der Datei zu ermitteln. Die einzige Direktive dieses Moduls ist `MimeMagicFile`. Wenn diese vorhanden ist, wird die MIME-Magic-Funktionalität auf der Basis der hier angegebenen Datei aktiviert.

Die Auswertung der Varianten einer Ressource, die ein Client bevorzugt, wird nicht im vorliegenden Abschnitt beschrieben, sondern in Abschnitt 7.3, »Content-Negotiation«. Die MIME-Konfiguration für die `Content-*`-Header von HTTP-Antworten ist nämlich auch dann wichtig, wenn Sie keine Content-Negotiation betreiben möchten.

**Anmerkungen zu Dateiendungen**

Bevor in den folgenden Abschnitten die einzelnen auf MIME bezogenen Konfigurationsoptionen behandelt werden, sollten noch ein paar allgemeine Worte über die Bedeutung von Dateiendungen gesagt werden. Selbstverständlich verwendet Apache, wie fast alle Internetprogramme, die standardmäßigen Zuordnungen der offiziell bei der IANA registrierten Dateierweiterungen zu den entsprechenden MIME-Types: Einer Datei mit der Endung `.html` wird der Typ `text/html` zugewiesen, bei der Erweiterung `.jpg` wird `image/jpeg` übermittelt. Diese Einstellungen stehen in der mitgelieferten *TypesConfig*-Datei, die sich normalerweise unter *conf/mime.types* befindet.

Die Besonderheit beim Apache-Webserver besteht allerdings darin, dass Sie Dateien mit mehreren Endungen verwenden können, um an ein und derselben Datei verschiedene MIME-Einstellungen vorzunehmen. Dies betrifft eben nicht die gerade erwähnten offiziellen Endungen, sondern z. B. spezielle Sprach- oder Zeichensatzendungen. Die Standard-Konfigurationsdatei enthält bereits zahlreiche `mod_mime`-Direktiven mit solchen Zuordnungen. Beispielsweise erhalten Dateien mit dem Endungspartikel `.de` die Spracheinstellung `de` (Deutsch); die Endungen `.latin1` oder `.iso8859-1` ergeben die Zeichensatzzuweisung ISO-8859-1.

Die Reihenfolge der einzelnen Endungen ist dabei normalerweise egal: Wenn Sie Ihren Server für die soeben beschriebenen Endungen `.html`, `.de` und `.latin1` konfiguriert haben, wird jede der drei folgenden Dateien mit dem Typ `text/html`, der Spracheinstellung Deutsch und dem Zeichensatz ISO-8859-1 ausgeliefert:

```
deutsch1.html.de.latin1
deutsch2.de.latin1.html
deutsch3.latin1.html.de
```

Eine Bedeutung hat die Reihenfolge allerdings dann, wenn mehrere MIME-Types oder zeichensatzrelevante Endungen gleichzeitig verwendet werden: Da das Feld `Content-Type` eines HTTP-Antwort-Headers nur je einen Typ und einen Zeichensatz transportieren kann, gilt hier jeweils die rechts ganz zuletzt stehende Endung: *index.html.txt* würde unter dem MIME-Type `text/plain` ausgeliefert; *index.txt.html* dagegen als `text/html`. Bei Sprach-Dateiendungen ist das anders: Der Header `Content-Language` kann eine Liste mehrerer Sprachkürzel transportieren.

### 7.2.1 MIME-Type-Einstellungen

Die wichtigste aller MIME-Informationen ist natürlich der MIME-Type. Er wird als Inhalt des HTTP-Headers `Content-Type` an den Client übermittelt. Wie bereits

erwähnt, versucht Apache zunächst, den MIME-Type mithilfe der Dateiendung zu ermitteln; dazu dienen die Zuordnungsdatei `mime.types` sowie zusätzliche Typzuweisungen, die Sie mittels `AddType` hinzufügen können. Falls das nicht funktioniert, kann das Modul `mod_mime_magic` einspringen und versuchen, den Typ aus dem Dateiinhalt zu erraten. Schlägt auch dies fehl, wird die Datei mit dem `DefaultType` versehen. Wenn Sie möchten, können Sie in einem bestimmten Kontext auch `ForceType` verwenden und damit alle anderen Typzuweisungen überschreiben.

**DefaultType**

Standard-MIME-Type

| | |
|---|---|
| **Seit Version** | 1.x; `none` seit 2.2.7; ab 2.3.x alle anderen Werte unzulässig |
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `DefaultType MIME-Type\|none` |
| **Standardwert** | `text/plain`; ab 2.3.x `none` |

Diese Direktive legt den auszuliefernden `Content-Type` fest, wenn Apache keine andere Möglichkeit findet, den MIME-Type eines Dokuments zu ermitteln. Wenn Sie hier keine eigene Angabe machen, wird `text/plain` verwendet. Wenn Sie wissen, dass ein bestimmtes Verzeichnis unabhängig von der Dateiendung nur Dateien eines bestimmten Typs enthält – beispielsweise MP3-Files –, können Sie den `DefaultType` für diesen Kontext explizit setzen:

```
DefaultType audio/mpeg
```

Wenn Ihre Site oder ein bestimmter Bereich innerhalb Ihrer Site vorwiegend Binärdateien enthält, die zum Download angeboten werden, empfiehlt es sich, diesen Wert einzustellen:

```
DefaultType application/octet-stream
```

Beachten Sie, dass `DefaultType` nur auf Server-Antworten angewendet wird, die ihren `Content-Type`-Header nicht auf anderem Weg erhalten haben. Um den bereits gesetzten MIME-Type zu überschreiben, müssen Sie `ForceType` verwenden.

Seit der Version 2.2.7 steht der Wert `none` zur Verfügung. Wenn dieser gesetzt wird, gibt es keinen Fallback-MIME-Type mehr. Ab Version 2.3-alpha ist nur noch der Wert `none` zulässig, das heißt, die Direktive wird außer Dienst gestellt und ist nur noch aus Gründen der Abwärtskompatibilität vorhanden.

### ForceType

Erzwingen eines bestimmten MIME-Types

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `ForceType MIME-Type\|None` |
| **Standardwert** | nicht gesetzt |

`ForceType` erfüllt eine ähnliche Aufgabe wie `DefaultType`: Die Direktive setzt in Server-Antworten den angegebenen MIME-Type als `Content-Type`-Header. Der Unterschied besteht darin, dass dies im aktuellen Kontext ausnahmslos für *alle* Dateien gilt – auch dann, wenn der tatsächliche MIME-Type bereits ermittelt wurde. Das folgende Beispiel legt für URLs unter dem Verzeichnis */flash* den MIME-Type SWF (Flash-Movies) fest:

```
<Location /flash>
  ForceType application/x-shockwave-flash
</Location>
```

Um die Einstellung aus einem übergeordneten Kontext wieder durch das Standardverhalten zu ersetzen, können Sie die spezielle Einstellung `None` verwenden. Beispiel:

```
<Location /flash/data>
  ForceType None
</Location>
```

### AddType

MIME-Type nach Dateiendung festlegen

| | |
|---|---|
| **Modul** | `mod_mime` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `AddType MIME-Type Dateiendung [Dateiendung ...]` |
| **Standardwert** | nicht gesetzt |

`AddType` weist einer oder mehreren Dateiendungen einen bestimmten MIME-Type zu. Das ist in der Regel nicht erforderlich, weil die mit Apache gelieferte und durch die Direktive `TypesConfig` eingebundene Datei *mime.types* Definitionen für so gut wie alle bekannten Dateitypen enthält. Falls Sie eine neue Typzuordnung benötigen, empfiehlt es sich, sie mittels `AddType` in Ihre *httpd.conf*-Datei zu schreiben, anstatt die vorgefertigte Zuordnungsdatei zu ergänzen.

Wenn Sie für eine Dateiendung, die bereits in `mime.types` steht, eine Zuordnung hinzufügen, wird diese überschrieben.

Das folgende Beispiel legt für PDF-Dateien den Typ `application/octet-stream` fest. Dies soll den Browser zwingen, diese Dateien herunterzuladen und lokal zu speichern, anstatt sie direkt anzuzeigen:

```
<Location pdf_load>
  AddType application/octet-stream .pdf
</Location>
```

Beachten Sie, dass der Microsoft Internet Explorer solche Einstellungen ignoriert, da er versucht, den Dateityp unabhängig vom `Content-Type`-Header selbst zu ermitteln.

#### RemoveType

MIME-Type-Zuordnung für Dateien mit bestimmten Endungen entfernen

| | |
|---|---|
| **Modul** | `mod_mime` |
| **Kontext** | `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `RemoveType Dateiendung [Dateiendung ...]` |
| **Standardwert** | nicht gesetzt |

Diese Direktive dient dazu, eine spezielle MIME-Type-Zuweisung aus einem übergeordneten Kontext wieder aufzuheben. Dies betrifft nur Zuordnungen, die mittels `AddType` in der Konfigurationsdatei vorgenommen wurden. Nachdem die Typzuweisung entfernt wurde, gilt daher wieder die automatische Zuordnung aus der Datei *mime.types*, falls für die Dateiendung eine solche existiert. Ist dies nicht der Fall, dann wird die `DefaultType`-Einstellung verwendet.

Hier ein Beispiel in vollem Umfang:

```
<Location /textload>
  # Textdateien ausdrücklich zum Download anbieten
  AddType application/octet-stream .txt
</Location>

<Location /textload/config>
  # Textdateien hier wieder normal behandeln
  RemoveType .txt
</Location>
```

### TypesConfig

Pfad der Datei *mime.types*

| | |
|---|---|
| **Modul** | `mod_mime` |
| **Kontext** | Server |
| **Syntax** | `TypesConfig Dateipfad` |
| **Standardwert** | `conf/mime.types` |

Die Direktive `TypesConfig` legt fest, wo sich die Standard-Zuordnungsdatei für MIME-Types und Dateiendungen befindet. Die Pfadangabe ist relativ zur `ServerRoot`.

Die standardmäßig voreingestellte Datei *mime.types* wird mit der Apache-Distribution mitgeliefert und erfüllt im Prinzip alle Bedürfnisse. Jede Zeile in der Datei enthält durch Leerzeichen getrennt einen MIME-Type und eine oder mehrere Dateiendungen. Beispiel:

```
image/jpeg jpeg jpg jpe
```

In der Datei sind sämtliche bei der IANA registrierten MIME-Types enthalten. Die jeweils aktuelle Liste – deren Format allerdings inkompatibel zu *mime.types* ist – finden Sie beispielsweise unter *http://www.isi.edu/innotes/iana/assignments/media-types/media-types*. Auch in Anhang E dieses Buches befindet sich eine entsprechende Liste.

### ModMimeUsePathInfo

Bestimmt, dass `mod_mime` `PathInfo`-Daten beachten soll

| | |
|---|---|
| **Seit Version** | 2.0.41 |
| **Modul** | `mod_mime` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `ModMimeUsePathInfo On\|Off` |
| **Standardwert** | `Off` |

Diese Konfigurationseinstellung bestimmt, ob auch an die eigentliche URL angehängte Pfadinformationen (`PathInfo`) auf relevante Dateiendungen für `mod_mime`-Einstellungen untersucht werden sollen.

Wenn ein Client beispielsweise eine URL mit dem Pfad */info/test.html/weitere/info.txt* anfordert und */info/test.html* eine vorhandene Datei ist, dann wird die Zusatzinformation */weitere/info.txt* nur verarbeitet, wenn `ModMimeUsePathInfo`

auf `On` gesetzt wurde. In diesem Fall wird also die zuletzt gefundene Endung `.txt` zur MIME-Type-Ermittlung genutzt.

Näheres über `PathInfo` finden Sie in der Beschreibung der Direktive `AcceptPathInfo` in Kapitel 14, »CGI«.

### MimeMagicFile

MIME-Magic-Funktionalität auf der Basis der angegebenen Datei einschalten

| | |
|---|---|
| **Modul** | `mod_mime_magic` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `MimeMagicFile Dateipfad` |
| **Standardwert** | `conf/magic` |

Diese Direktive aktiviert die Funktion des Moduls `mod_mime_magic` und legt gleichzeitig die Datei fest, aus der es seine Informationen beziehen soll. Die vorgefertigte MIME-Magic-Datei befindet sich im *conf*-Verzeichnis und heißt *magic*, sodass dies der Vorgabewert ist. Wie bei den meisten Direktiven mit Pfadwerten bezieht sich ein relativer Pfad auf die `ServerRoot`.

`mod_mime_magic` versucht, den MIME-Type von Dateien aus deren Inhalt zu ermitteln. Das Verfahren basiert auf dem UNIX-Dienstprogramm `file(1)`, insbesondere auf der völlig neu geschriebenen System-V-Fassung von Ian Darwin aus dem Jahr 1987. `mod_mime_magic` selbst wurde von Cisco Systems entwickelt. Inzwischen gehört das Modul zum Lieferumfang von Apache.

Die *magic*-Datei selbst ist ein einfaches Textdokument. Jede Zeile besteht aus vier bis fünf Spalten mit folgender Bedeutung:

- **Startbyte**
  Hier wird das Byte angegeben, ab dem der Inhalt der Datei untersucht werden soll. Der Nummer kann ein Größer-als-Zeichen (>) vorangestellt werden, wenn die Bedingung von einer zuvor formulierten Bedingung abhängt.
- **Datentyp**
  In dieser Spalte wird das Format der Daten bestimmt, die in der Datei gesucht werden. Folgende Formate sind definiert:
  - `byte`: einzelnes Zeichen
  - `short`: 16-Bit-Integer (Systemreihenfolge)[1]
  - `long`: 32-Bit-Integer (Systemreihenfolge)

1 »Systemreihenfolge« bedeutet unbestimmt, ob Little Endian oder Big Endian gemeint ist.

- `date`: Datum/Uhrzeit in Sekunden seit EPOCH (Systemreihenfolge)
- `string`: Zeichenkette beliebiger Länge
- `beshort`: 16-Bit-Integer (Big Endian)
- `belong`: 32-Bit-Integer (Big Endian)
- `bedate`: Datum/Uhrzeit in Sekunden seit EPOCH (Big Endian)
- `leshort`: 16-Bit-Integer (Little Endian)
- `lelong`: 32-Bit-Integer (Little Endian)
- `ledate`: Datum/Uhrzeit in Sekunden seit EPOCH (Little Endian)

- **Inhalt**
  Hier steht, nach welchem Inhalt in der Datei gesucht werden soll.
- **MIME-Type**
  Der hier aufgeführte MIME-Type wird gesetzt, wenn eine Übereinstimmung gefunden wird.
- **MIME-Codierung**
  In dieser optionalen Spalte kann eine MIME-Codierung angegeben werden; dient in der Regel zur Kennzeichnung komprimierter Dateien.

Die folgenden Zeilen aus der offiziellen *magic*-Datei kennzeichnen beispielsweise HTML-Dokumente (vorsichtshalber auch solche mit unvollständiger Einleitung):

```
# html:  file(1) magic for HTML (HyperText Markup Language) docs
#
# from Daniel Quinlan <quinlan@yggdrasil.com>
# and Anna Shergold <anna@inext.co.uk>
#
0   string      \<!DOCTYPE\ HTML    text/html
0   string      \<!doctype\ html    text/html
0   string      \<HEAD      text/html
0   string      \<head      text/html
0   string      \<TITLE     text/html
0   string      \<title     text/html
0   string      \<html      text/html
0   string      \<HTML      text/html
0   string      \<!--       text/html
0   string      \<h1        text/html
0   string      \<H1        text/html
```

Diese Zeile erkennt dagegen JPEG-Bilder:

```
# JPEG images
0   beshort     0xffd8      image/jpeg
```

### 7.2.2 Zeichensatzeinstellungen

Als das Web erfunden wurde, machten sich seine Entwickler noch keinerlei Gedanken über Internationalisierung. Die ersten Versionen von HTTP und HTML kamen deshalb nur mit reinem 7-Bit-ASCII zurecht. Dies genügt allenfalls für englischen Text. Für eine leichte Verbesserung behalf man sich mit der Definition von Entity-Referenzen für die meisten Sonderzeichen aus dem Zeichensatz ISO-8859-1. Zeichencodierungen wie `ä` für ä oder `ß` für ß dürften Ihnen bestens bekannt sein. Damit lassen sich Dokumente in einigen mitteleuropäischen Sprachen erstellen. Für lateinisch geschriebene Sprachen mit spezielleren Sonderzeichen oder gar völlig andere Schriften nützt natürlich auch das nichts.

Das Zusammenwirken von HTTP/1.1 und (mindestens) HTML 4.0 ermöglicht endlich die Definition des Zeichensatzes, in dem ein Dokument verfasst ist. Apache stellt die in diesem Unterabschnitt beschriebenen Konfigurationsdirektiven für die Einstellung der Zeichensatzinformationen zur Verfügung. Die zugehörige Liste der Zeichensätze finden Sie in Anhang E.

Die Zeichensatzeinstellung wird im Header `Content-Type` vom eigentlichen MIME-Type durch ein Semikolon getrennt übertragen. Beispiel:

```
Content-Type: text/html; charset=utf-8
```

Ein unter Webdesignern und -entwicklern bekannteres Verfahren für die Definition des Zeichensatzes, der für ein HTML-Dokument verwendet werden soll, ist ein HTML-Meta-Tag nach folgendem Schema:

```
<meta http-equiv="Content-type"
content="text/html; charset=iso-8859-1" />
```

Die meisten Browser interpretieren ein Meta-Tag mit dem Attribut `http-equiv` als gleichwertig zum entsprechenden HTTP-Antwort-Header. Sicher sein können Sie sich dabei allerdings nicht; es ist immer besser, den Zeichensatz über einen echten HTTP-Header (und deshalb mithilfe der Server-Konfiguration) festzulegen.

Eine weitere Stelle, an der der verwendete Zeichensatz unbedingt (zusätzlich) angegeben werden sollte, ist die Kopfzeile von XML-Dokumenten – und damit auch von modernen XHTML-Dateien. Beispiel:

```
<?xml version="1.0" encoding="utf-8"?>
```

Lassen Sie sich nicht durch die Bezeichnung `encoding` verwirren: Es handelt sich trotzdem um einen Zeichensatz und nicht etwa um eine MIME-Codierung (`Content-Encoding`).

### AddDefaultCharset

Standardwert für den Zeichensatz im HTTP-Header

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | AddDefaultCharset On\|Off\|Zeichensatzname |
| Standardwert | Off |

Mithilfe der Direktive `AddDefaultCharset` wird gesteuert, ob Apache zum Header aller Antworten Zeichensatzinformationen hinzufügen soll, falls noch keine enthalten sind. Wenn Sie dieses Feature aktivieren, wird der Header `Content-Type` nach folgendem Schema erweitert:

```
Content-Type: text/html
```

wird zu

```
Content-Type: text/html; charset=iso-8859-1
```

Sie können drei verschiedene Werte einstellen:

- `Off`
  Es wird kein Zeichensatzwert zum Header hinzugefügt, wenn noch keiner vorhanden ist.
- `On`
  Apache fügt seinen fest einprogrammierten Standardzeichensatz `iso-8859-1` hinzu. Dieser Zeichensatz ist für Englisch, Deutsch und die meisten anderen mitteleuropäischen Sprachen geeignet.
- **Zeichensatzname**
  Sie können den gewünschten Zeichensatz explizit angeben. Beispiele: `iso-8859-5` (ASCII + Kyrillisch), `utf-8`.

### AddCharset

Einer Dateiendung einen Zeichensatz zuordnen

| Modul | mod_mime |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | AddCharset Zeichensatz Dateiendung [Dateiendung ...] |
| Standardwert | nicht gesetzt |

Diese Direktive legt verbindlich die Zeichensatzinformation fest, die Apache für eine oder mehrere angegebene Dateiendungen im aktuellen Kontext zum Header `Content-Type` hinzufügen soll. Eine eventuell bereits vorhandene Zeichensatzangabe wird überschrieben. Die Standard-Konfigurationsdatei enthält einige vorgefertigte Zuweisungen für Dateien, die als zusätzliche Endung den jeweiligen Zeichensatz enthalten. Beispiel:

```
AddCharset ISO-8859-1  .iso8859-1 .latin1
```

Alternativ können Sie – wenn auch mit Vorsicht – Zeichensatzangaben für die mittels `AddLanguage` festgelegten `Content-Language`-Einstellungen festlegen. Das folgende Beispiel definiert für die Dateiendung `.zh-cn` die Sprache vereinfachtes Chinesisch (`zh-CN`) und den passenden Zeichensatz `GB2312`:

```
AddLanguage   zh-CN   .zh-cn
AddCharset    GB2312  .zh-cn
```

#### RemoveCharset

Zeichensatzzuordnung für Dateien mit bestimmten Endungen entfernen

| | |
|---|---|
| **Seit Version** | 2.0.24 |
| **Modul** | `mod_mime` |
| **Kontext** | `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `RemoveCharset Dateiendung [Dateiendung ...]` |
| **Standardwert** | nicht gesetzt |

Die Direktive `RemoveCharset` entfernt eine Zeichensatzzuordnung für eine bestimmte Dateiendung, die in einem übergeordneten Kontext definiert wurde.

Das folgende Beispiel definiert für PDF-Dokumente unter dem URL-Pfad */turkish* den für Türkisch passenden Zeichensatz `iso-8859-9`. In dessen Unterverzeichnis *extra* wird die Zuweisung wieder aufgehoben:

```
<Location /turkish>
  AddCharset iso-8859-9 .pdf
</Location>

<Location /turkish/extra>
  RemoveCharset .pdf
</Location>
```

### 7.2.3 MIME-Codierung

Die MIME-Codierung bestimmt den Wert des Entity-Headers `Content-Encoding`, der eine spezielle Codierung – in aller Regel die Komprimierung – des Bodys einer HTTP-Übertragung kennzeichnet. Es spart natürlich Netzwerkbandbreite, wenn Dokumente komprimiert versendet und dann vom Client automatisch extrahiert werden. Allerdings beherrschen nicht alle Browser diese Fähigkeit. Deshalb ist es besser, den `Accept-Encoding`-Header von Client-Anfragen auszuwerten und entsprechend darauf zu reagieren. Näheres erfahren Sie in Abschnitt 7.3, »Content-Negotiation«.

#### AddEncoding

MIME-Codierung nach Dateiendung festlegen

| | |
|---|---|
| **Modul** | `mod_mime` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `AddEncoding MIME-Codierung Dateiendung [Dateiendung ...]` |
| **Standardwert** | nicht gesetzt |

Mithilfe dieser Direktive wird die MIME-Codierung für bestimmte Dateiendungen festgelegt. `AddEncoding` ändert die Codierung auch dann, wenn sie bereits mithilfe einer anderen Methode ermittelt wurde. In der vorgefertigten Konfigurationsdatei befinden sich die beiden folgenden Zeilen:

```
AddEncoding x-compress .Z
AddEncoding x-gzip     .gz .tgz
```

`x-compress` und `x-gzip` sind die traditionellen Bezeichnungen, die von älteren Browsern benötigt werden. Moderne Browser verwenden dagegen die aktuellen Bezeichnungen `compress` und `gzip`, verstehen aber auch die Varianten mit vorangestelltem `x-`. Deshalb empfiehlt es sich, immer `x-compress` und `x-gzip` zu schreiben. Für die neuere `deflate`-Komprimierung gilt das nicht, weil alte Browser sie ohnehin nicht verstehen. Wenn Apache den Header `Accept-Encoding` einer Client-Anfrage auswertet, ignoriert er ein eventuelles `x-`.

Da nicht jeder Browser in der Lage ist, solche komprimierten Dokumente automatisch zu entpacken, werden diese Anweisungen aber zunächst einmal auskommentiert. Damit der Browser selbst entscheiden kann, wie er mit den Dateien umgehen soll, werden stattdessen mittels `AddType` die entsprechenden `Content-Type`-Werte angegeben:

```
AddType application/x-compress .Z
AddType application/x-gzip     .gz .tgz
```

Beachten Sie, dass diese Einstellungen nur für bereits komprimierte Dokumente gelten. Apache komprimiert sie aufgrund dieser Einstellungen nicht automatisch für den Versand. Eine solche Aufgabe erfüllt das in Kapitel 16, »SSI und Filter«, behandelte Modul `mod_deflate`.

#### RemoveEncoding

Codierungszuordnung für Dateien mit bestimmten Endungen entfernen

| | |
|---|---|
| **Modul** | `mod_mime` |
| **Kontext** | `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `RemoveEncoding Dateiendung [Dateiendung ...]` |
| **Standardwert** | nicht gesetzt |

Mithilfe von `RemoveEncoding` können Sie eine MIME-Codierung entfernen, die in einem übergeordneten Kontext für eine bestimmte Dateiendung gesetzt wurde. Da die Komprimierungs-Header in aller Regel nur speziellen Dateiendungen zugeordnet werden, spielt dies in der Praxis keine große Rolle.

### 7.2.4 Spracheinstellungen

Der HTTP-Header `Content-Language` gibt Auskunft über die Sprache beziehungsweise Sprachen, in denen das gelieferte Dokument verfasst ist. `mod_mime` liefert die Direktiven, mit denen Sie dies einstellen können. Eine Tabelle der ISO-Sprachkürzel, die als Werte für diese Konfigurationseinstellungen angegeben werden können, finden Sie in Anhang E.

#### DefaultLanguage

Standardspracheinstellung

| | |
|---|---|
| **Modul** | `mod_mime` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `DefaultLanguage Sprachkürzel` |
| **Standardwert** | nicht gesetzt |

Diese Direktive legt die Standardsprache für alle Dokumente im aktuellen Kontext fest, die keine mittels `AddLanguage` konfigurierte Sprach-Dateiendung besit-

zen. Diese Einstellung wird als `Content-Language`-Header mit dem Dokument versendet. Wenn Sie diese Direktive nicht setzen, wird bei Dateien ohne Sprachkennzeichnung gar kein Wert übermittelt. Das folgende Beispiel stellt Deutsch als Standardsprache ein:

```
DefaultLanguage de
```

### AddLanguage

Sprachzuordnung nach Dateiendung festlegen

| | |
|---|---|
| **Modul** | `mod_mime` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `AddLanguage Sprachkürzel Dateiendung [Dateiendung ...]` |
| **Standardwert** | nicht gesetzt |

Mithilfe dieses Headers wird einer Dateiendung eine bestimmte Sprache zugeordnet. Dieses ISO-Sprachkürzel wird einerseits als Wert des HTTP-Antwort-Headers `Content-Language` gesendet und dient andererseits der Content-Negotiation. Die vorgefertigte *httpd.conf*-Datei enthält einige vorgefertigte Sprachzuweisungen wie beispielsweise folgende:

```
AddLanguage de .de
AddLanguage en .en
```

Natürlich steht es Ihnen frei, in einem bestimmten Kontext auch eine Standard-Dateiendung mit einer Sprache zu verknüpfen. Das folgende Beispiel stellt für alle HTML-Dokumente unter dem URL-Pfad */italian* die Sprache Italienisch ein:

```
<Location /italian>
  AddLanguage it .htm .html
</Location>
```

### RemoveLanguage

Sprachzuordnung für Dateien mit bestimmten Endungen entfernen

| | |
|---|---|
| **Seit Version** | 2.0.24 |
| **Modul** | `mod_mime` |
| **Kontext** | `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `RemoveLanguage Dateiendung [Dateiendung ...]` |
| **Standardwert** | nicht gesetzt |

Die Direktive `RemoveLanguage` ermöglicht das Entfernen einer Sprachzuordnung, die in einem übergeordneten Kontext mit `AddLanguage` vorgenommen wurde.

### 7.2.5 Handler festlegen

Mithilfe der Direktiven `SetHandler`, `AddHandler` und `RemoveHandler` können Sie bestimmen, von welchem Handler bestimmte Dateien verarbeitet werden sollen. Ein Handler ist der Mechanismus, der in Gang gesetzt wird, wenn ein Client eine URL anfordert. Standardmäßig ist für fast alle Dateien der im Core definierte `default-handler` aktiviert. Seine Aufgabe besteht darin, der angeforderten Datei einige HTTP-Header voranzustellen und sie an den Client auszuliefern. Einige weitere Handler werden durch Standardmodule bereitgestellt:

- `send-as-is`
  Apache fügt keinerlei eigene HTTP-Header hinzu, sondern sendet die gewünschte Datei, »wie sie ist«. Dieser Handler wird durch das bereits besprochene Modul `mod_asis` zur Verfügung gestellt.
- `cgi-script`
  Der Handler wird von `mod_cgi` beziehungsweise `mod_cgid` definiert. Er sorgt dafür, dass die angeforderte Ressource als ausführbares CGI-Skript behandelt wird, dessen Standardausgabe Apache dann an den Client ausliefert (siehe Kapitel 14, »CGI«).
- `server-parsed`
  In der Datei werden Server Side Includes (SSI) ausgeführt. Der Handler wird von `mod_include` bereitgestellt. Dies wird in Kapitel 16, »SSI und Filter«, besprochen.
- `imap-file`
  Dieser Handler behandelt angeforderte Dateien als Definitionen für serverseitige Image Maps und wird durch `mod_imagemap` (bis Version 2.0 irreführenderweise `mod_imap`) angeboten. Näheres dazu steht in Kapitel 8, »Weiterleitungen und Indizes«.
- `server-info`
  Unter einer URL, für die dieser Handler gesetzt ist, wird kein Dokument ausgeliefert. Stattdessen liefert Apache die durch `mod_info` generierten Informationen über die Server-Konfiguration aus. Stellen Sie sicher, dass Sie den betreffenden Container (typischerweise eine spezielle `<Location>`) vor öffentlichem Zugriff schützen. Auch auf dieses Thema wird im nächsten Kapitel näher eingegangen.
- `server-status`
  Ähnlich wie `server-info` bestimmt auch `server-status`, dass die entsprechende URL eine besondere Bedeutung hat: Sie liefert Statusinformationen

über den Server (Performance, Belastung usw.). Auch diese Information geht die Öffentlichkeit in der Regel nichts an, sodass Sie den Container absichern müssen. Der Handler ist nur verfügbar, wenn `mod_status` aktiv ist. Genaueres dazu finden Sie ebenfalls in Kapitel 8, »Weiterleitungen und Indizes«.

- `type-map`
  Die URL wird als Type-Map-Datei für Content-Negotiation behandelt (dazu mehr weiter unten in diesem Kapitel). Der Handler wird durch das Modul `mod_negotiation` zur Verfügung gestellt.

Mithilfe der Direktive `Action` können Sie eigene Handler-Bezeichnungen definieren und einem CGI-Skript zuordnen. Dieses Verfahren wird in Kapitel 14, »CGI«, genauer erläutert. In Kapitel 17, »Apache erweitern«, wird auch die Nutzung vorhandener und die Erstellung eigener Handler in selbst programmierten Modulen erläutert.

#### SetHandler

Bestimmt, dass der angegebene Handler alle Dateien im Kontext verarbeitet

| | |
|---|---|
| **Seit Version** | 1.x; seit 2.0 im Core |
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `SetHandler Handlername\|None` |
| **Standardwert** | nicht gesetzt |

Mithilfe von `SetHandler` wird ein Handler festgelegt, der alle Dateien im aktuellen Kontext verarbeitet. Ein praktisches Anwendungsbeispiel ist die Definition einer spezifischen URL für Server-Informationen:

```
<Location /_info>
  SetHandler server-info
  Order deny,allow
  Deny from all
  Allow from 196.23.17
</Location>
```

Unter der URL *www.mynet.de/_info* werden auf diese Weise Informationen über den Server ausgegeben. Es wurde bereits darauf hingewiesen, dass solche sensiblen Daten nicht jedem beliebigen Client verfügbar gemacht werden sollten. In diesem Beispiel wurde diese Empfehlung durch eine Beschränkung auf das eigene IP-Teilnetz realisiert. Zusätzlich sollten Sie die URL auch noch durch Authentifizierung absichern – dies wird ausführlich in Kapitel 9, »Authentifizierung«, beschrieben.

Das nächste Beispiel erzeugt einen URL-Alias namens `scripts` und setzt den Handler `cgi-script`. Alle Dateien in diesem Verzeichnis werden als CGI-Skripte behandelt:

```
Alias /scripts/ "/usr/local/apache2/cgi/"
<Location /scripts>
  SetHandler cgi-script
</Location>
```

Eine Kurzfassung für diese beiden Schritte ist die in Kapitel 14, »CGI«, behandelte Direktive `ScriptAlias`. Damit reduziert sich der Aufwand auf diese eine Zeile:

```
ScriptAlias /scripts/ "/usr/local/apache2/cgi/"
```

Wenn Sie einen im übergeordneten Kontext gesetzten Handler wieder loswerden möchten, können Sie den speziellen Wert `None` verwenden. Das folgende Beispiel entfernt den Handler `cgi-script` aus dem Unterverzeichnis *userdata* der eben definierten URL */scripts*:

```
<Location /scripts/userdata>
  SetHandler None
</Location>
```

#### AddHandler

Handler nach Dateiendung festlegen

| | |
|---|---|
| **Modul** | `mod_mime` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `AddHandler Handler-Name Dateiendung [Dateiendung ...]` |
| **Standardwert** | nicht gesetzt |

Diese Direktive legt einen Handler für eine bestimmte Dateiendung fest. Anders als `SetHandler` können Sie diese Direktive also problemlos in den Server-Kontext setzen und bestimmte Dateien auf diese Weise kennzeichnen. Beispielsweise könnten alle Dateien mit den Endungen `.cgi` und `.pl` als CGI-Skripte behandelt werden:

```
AddHandler cgi-script .cgi .pl
```

Das ist zwar sehr praktisch, aber vom Sicherheitsstandpunkt aus nicht uneingeschränkt empfehlenswert. Eine detaillierte Diskussion des Für und Wider finden Sie in Kapitel 14, »CGI«.

**RemoveHandler**

Handler-Zuordnung für Dateien mit bestimmten Endungen entfernen

| | |
|---|---|
| **Modul** | mod_mime |
| **Kontext** | <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| **Syntax** | RemoveHandler Dateiendung [Dateiendung ...] |
| **Standardwert** | nicht gesetzt |

Mithilfe von RemoveHandler können Sie den Handler für eine bestimmte Dateiendung entfernen. Auf diese Weise können Sie eine Handler-Zuordnung aus einem übergeordneten Kontext lösen. Das folgende Beispiel deaktiviert in URLs unter dem Verzeichnis */no-cgi* die allgemein definierte cgi-script-Zuweisung:

```
AddHandler cgi-script .cgi

<Location /no-cgi>
  RemoveHandler .cgi
</Location>
```

## 7.3 Content-Negotiation

Eine der praktischsten Fähigkeiten moderner Webserver wie Apache ist die automatische Anpassung an Vorgaben aus der Client-Anfrage. Genauer gesagt kann aus mehreren verfügbaren Varianten einer Ressource die passende ausgesucht werden. Die Grundlage für die Auswahl bilden die diversen Accept-*-Header der Client-Anfrage. Das Verfahren wird als *Content-Negotiation* bezeichnet; Apache stellt es durch das Modul mod_negotiation zur Verfügung. Dieses Modul muss für sämtliche in diesem Abschnitt beschriebenen Themen aktiviert sein.

Grundsätzlich können die folgenden vier Eigenschaften einer Ressource in unterschiedlichen Varianten vorliegen:

- **Medientyp** (MIME-Type)
  Die Auswahl erfolgt auf der Basis des Anfrage-Headers Accept.
- **Sprache**
  Der Anfrage-Header Accept-Language ist die Grundlage für die Entscheidung.
- **Zeichensatz**
  Der Client teilt dem Server seine diesbezüglichen Wünsche über den Header Accept-Charset mit.

- **Codierung**
  Ob und wie die angeforderte Ressource komprimiert sein darf, entscheidet Apache anhand des Anfrage-Headers `Accept-Encoding`.

Grundsätzlich unterstützt Apache zwei verschiedene Verfahren der Content-Negotiation. Beide treten in Kraft, sobald `mod_negotation` eingebunden wird. Es handelt sich um folgende Methoden:

- **Servergesteuerte Content-Negotiation**
  Apache interpretiert die vom Client übermittelten `Accept-*`-Header und entscheidet daraufhin selbst, welche Variante der Ressource am besten für den Client geeignet ist.
- **Transparente Content-Negotiation**
  Wenn der Client einen `Negotiate`-Header sendet, schickt ihm Apache einfach eine Antwort mit dem HTTP-Statuscode `300 Multiple Choices`, die eine Liste der verfügbaren Alternativen enthält.

### 7.3.1 Servergesteuerte Content-Negotiation

Bei dieser klassischen Variante der Content-Negotiation wertet Apache die `Accept-*`-Header der Client-Anfrage aus. Auf dieser Grundlage entscheidet er nach dem weiter unten erläuterten Apache-Negotiation-Algorithmus, welche Variante er an den Client übermitteln soll. Sie können die Liste der verfügbaren Alternativen auf zweierlei Arten erstellen:

- **Type-Maps**
  Eine Type-Map ist eine separate Datei, je nach Konfiguration meist mit der Endung `.var`, die eine Liste aller Varianten enthält.
- **MultiViews**
  Apache generiert die Type-Map automatisch anhand der Dateiendungen und der entsprechenden Einstellungen (siehe Abschnitt 7.2, »MIME-Konfiguration«).

#### Type-Maps

Wenn Sie Type-Maps verwenden möchten, müssen Sie im entsprechenden Kontext zunächst einmal eine Dateiendung für diese definieren. Dazu wird die entsprechende Endung mit dem speziellen Handler `type-map` verknüpft. Üblicherweise ist die Dateiendung für Type-Maps `.var`; die passende `AddHandler`-Direktive sieht so aus:

```
AddHandler type-map .var
```

Eine Datei mit der zusätzlichen Endung `.var` wird nun als Type-Map betrachtet. Diese enthält Verweise auf URIs (Pfadbestandteile der URL), die verschiedene Varianten der angeforderten Ressource darstellen.

In Apache 2.0 wurde alternativ auch die Möglichkeit eingeführt, den Body der verschiedenen Varianten direkt in die Type-Map zu schreiben – Sie brauchen dann also keine unterschiedlichen Dateien mehr für verschiedene Varianten. Dieses Verfahren wird beispielsweise für die internationalisierten Fehlermeldungsseiten verwendet, die bei der Apache-Distribution mitgeliefert werden.

Bevor das Format einer Type-Map ausführlich erläutert wird, sehen Sie hier ein einfaches Beispiel. Die Type-Map-Datei `test.html.var` verweist auf verschiedene Sprachversionen und sieht folgendermaßen aus:

```
URI: test.html.en
Content-Type: text/html
Content-Language: en

URI: test.html.de
Content-Type: text/html
Content-Language: de

URI: text.html.tr
Content-Type: text/html; charset=iso-8859-9
Content-Language: tr
```

Diese Einträge bedeuten Folgendes: Wenn ein Client die URI `test.html.var` anfordert, hängt es von dem Header `Accept-Language` ab, welche der drei Varianten geliefert wird:

- `Accept-Language: fr`
  Der Client erhält die französischsprachige Datei `test.html.fr`.
- `Accept-Language: de`
  In diesem Fall wird die deutschsprachige Datei `test.html.de` ausgeliefert.
- `Accept-Language: tr`
  Hier wird die türkische Variante `test.html.tr` übertragen; sie enthält zusätzlich die passende Zeichensatzeinstellung `iso-8859-9`.

Die Variante mit in die Type-Map integriertem Body sieht beispielsweise so aus wie im folgenden Dokument `hello.txt.var`:

```
Content-Language: fr
Content-Type: text/plain; charset=iso-8859-1
Description: "Text Document (iso-8859-1, French)"
Body: -------fr----------
```

```
Bonjour monde!
-------fr----------

Content-Language: de
Content-Type: text/plain; charset=iso-8859-1
Description: "Text Document (iso-8859-1, German)"
Body: -------de----------
Hallo Welt!
-------de----------

Content-Language: tr
Content-Type: text/plain; charset=iso-8859-9
Description: "Text Document (iso-8859-9, Turkish)"
Body: -------tr----------
Merhaba dünya!
-------tr----------
```

Die Zeilen zwischen `Body: <Name der Endmarkierung>` und dem Abschluss durch `<Name der Endmarkierung>` in einer separaten Zeile werden hier als Inhalt des Dokuments in der jeweils durch die Header beschriebenen Variante betrachtet.

In einer Type-Map dürfen die folgenden Header beziehungsweise Felder verwendet werden:

- `Content-Type`: Der MIME-Type der Ressource, optional gefolgt von einem oder mehreren der folgenden Zusatzwerte:
- `charset=Zeichensatz`
  - `level=Nr.`: ein Vorzugsfaktor (automatisch gilt `2` für `text/html` und `0` für alle anderen Medientypen)
  - `qs=Qualität`: Die »Quality of Source« gibt den Qualitätsfaktor der jeweiligen Variante an. Der Wert ist eine Fließkommazahl zwischen `0.0` und `1.0`. Je höher die Qualität einer verfügbaren Ressource ist, desto höher sollte diese Zahl eingestellt werden. »Qualität« heißt in diesem Fall vor allem die Entscheidung, welche Version Sie am liebsten ausliefern möchten. Für die Content-Negotiation wird dieser Wert mit dem `q`-Wert aus dem `Accept`-Header der Anfrage multipliziert.
- `Content-Language`: die Sprache, in der das Dokument verfasst ist
- `Content-Length`: Länge des Bodys in Byte
- `Content-Encoding`: MIME-Codierung (Komprimierungsart) der Ressource
- `Description`: eine optionale Beschreibung der Ressource. Apache baut sie in die Antwort ein, wenn dem Client eine Auswahlliste gesendet wird – dies

geschieht entweder, wenn keine passende Variante verfügbar ist, oder bei der transparenten Content-Negotiation.

- `URI`: die URI der Variante, auf die die genannten Header passen
- `Body`: der Inhalt der Variante, die zuvor durch Header charakterisiert wurde. Hinter `Body:` muss der Name einer Markierung stehen, die den Block abschließen wird. Traditionell wird als Name der Markierung so etwas wie `----<variierte Eigenschaft>----` gewählt, also beispielsweise `----de----` oder `----iso-8859-9----`. Die Anzahl der Bindestriche oder die Bezeichnung eines Blocks spielen aber überhaupt keine Rolle. Von Bedeutung ist lediglich, dass die Endmarkierung des Blocks ganz links am Zeilenanfang beginnen und mit dem Text hinter `Body:` übereinstimmen muss. Beachten Sie außerdem, dass die letzte Zeile der Datei, die den Block abschließt, mit einem Zeilenumbruch enden muss. Die Vorgehensweise entspricht den aus Shell- und Perl-Skripten bekannten HIER-Dokumenten.

Beachten Sie, dass Sie sich zwischen `URI` und `Body` entscheiden müssen; beide Felder auf einmal ergeben keinen Sinn. `Body` ist nur für kurze Inhalte geeignet, weil die maximale Länge von Type-Map-Dateien beschränkt ist. Die `URI`-Methode sollte also bevorzugt verwendet werden.

Hier sehen Sie ein weiteres Beispiel für eine Type-Map, in der zwei verschiedene Varianten eines Bildes durch `qs`-Parameter unterschieden werden – die JPEG-Version von `photo` hat eine höhere Qualität als die GIF-Version.

```
URI: photo.jpg
Content-Type: image/jpeg; qs=1.0
Description: "High-quality photograph (JPEG)"

URI: photo.gif
Content-Type: image/gif; qs=0.5
Description: "Low-quality representation of photograph (GIF)"
```

#### MultiViews

Die MultiViews-Suche generiert anhand der Dateiendungen und der bereits beschriebenen MIME-Grundeinstellungen automatisch eine Type-Map. Damit dies funktioniert, muss die Option `MultiViews` für den jeweiligen Kontext aktiviert werden (Näheres zur Direktive `Options` finden Sie in Kapitel 6, »Grundkonfiguration«). Beachten Sie, dass diese Option *nicht* in der Einstellung `Options All` enthalten ist! Angenommen, Ihre `Options`-Einstellung für den gewünschten Kontext sieht bisher so aus:

```
Options Indexes FollowSymLinks
```

Dann müssen Sie die Zeile folgendermaßen erweitern:

```
Options Indexes FollowSymLinks MultiViews
```

In einem untergeordneten Kontext lässt sich `MultiViews` auch zu den »geerbten« Optionen hinzufügen:

```
Options +MultiViews
```

Die MultiViews-Suche funktioniert nach folgendem Schema: Angenommen, ein Client fordert eine URI namens `test.html` an. Wenn eine Datei mit diesem Namen nicht vorhanden ist, wird die Content-Negotiation aktiviert und Apache liefert je nach den `Accept-*`-Headern der Client-Anfrage eine passende Variante aus, z. B. die Sprachvarianten `test.html.de`, `test.html.en` oder die komprimierte Fassung `test.html.gz`.

Natürlich ist die manuelle Erstellung von Type-Maps flexibler, beispielsweise durch die `qs`-Faktoren beim `Content-Type`. Abgesehen davon ist sie auch schneller, weil Apache nicht zuerst den Verzeichnisinhalt auslesen und selbst eine Type-Map erstellen muss. `MultiViews` sollten Sie daher nur einsetzen, wenn relativ wenige Varianten vorhanden sind.

#### Der Apache-Negotiation-Algorithmus

Die Auswahl der »besten« Variante für die jeweilige Client-Anfrage erfolgt nach einem ziemlich komplexen Verfahren. Diese Methode wird als *Apache-Negotiation-Algorithmus* bezeichnet.

Als Erstes werden die `Accept-*`-Header der Anfrage ausgelesen. Jeder Variante werden die `q=Faktor`-Qualitätswerte aus diesen Headern zugewiesen. Falls eine Variante laut `Accept-*` *gar nicht* akzeptabel ist (Qualitätsfaktor 0 beziehungsweise weder ausdrücklich noch durch Platzhalter erwähnt), wird sie aus der Liste gestrichen.

Sollte nach diesem Schritt bereits gar keine Variante mehr übrig bleiben, sendet der Server eine Antwort mit dem Statuscode `406 No Acceptable Variant`. Der Body enthält dabei eine Liste aller verfügbaren Varianten. Eine Ausnahme bilden die Sprachvarianten – über Direktive `ForceLanguagePriority` können Sie festlegen, dass die von *Ihnen* bevorzugte Sprachversion geliefert werden soll, wenn keine der vom Client gewünschten Sprachen verfügbar ist.

Wenn nach der Auswertung der `Accept-*`-Header noch Varianten verfügbar sind, durchläuft der Algorithmus nacheinander die folgenden Schritte. Sollte nach einem dieser Schritte nur noch eine Variante übrig bleiben, wird diese ausgeliefert.

1. Die Qualitätsfaktoren (`q`) der vom Client akzeptierten MIME-Types werden mit den Qualitätseinstellungen der Type-Map (`qs`) multipliziert. Dabei wird jeder Wert, der nicht explizit angegeben wurde, automatisch auf `1.0` gesetzt. Der Algorithmus wählt dann sämtliche Varianten mit dem höchsten Ergebnis aus.

   Übrigens ergänzt Apache die Vorgabe der HTTP/1.1-Spezifikation folgendermaßen: Dem Generalplatzhalter `*/*` für jeden beliebigen `Content-Type` wird die äußerst niedrige Priorität `q=0.01` zugewiesen; für `type/*` (alle Untertypen eines Haupttyps, z. B. `text/*` für sämtliche Textvarianten) wird `q=0.02` gesetzt. Dies gilt natürlich nur dann, wenn die Anfrage keine ausdrücklichen Angaben für diese Fälle enthält, und bewirkt dann, dass der Webserver nur im äußersten Notfall ein typfremdes Dokument ausliefert. Diese Erweiterung wird in der Apache-Originaldokumentation ausführlicher erläutert, und zwar im Abschnitt *Fiddling with Quality Values* des Dokuments *Content-Negotiation*.
2. Aus den verbleibenden Kandidaten werden die Varianten gewählt, die in der Client-Anfrage den höchsten `q`-Wert für die Sprache erhielten.
3. Wenn keine oder mehrere identische Sprachqualitätsfaktoren gefunden wurden, wird das beste Dokument nach der Sprachreihenfolge in `Accept-Language` gewählt. Ist auch diese nicht verfügbar, wird die eingestellte `LanguagePriority` des Servers betrachtet. Beachten Sie die Beschreibungen von `LanguagePriority` und `ForceLanguagePriority` weiter unten in diesem Abschnitt. Seit Apache 2.0.47 wird auch die spezielle Umgebungsvariable `prefer-language` in Betracht gezogen, die Sie z. B. mittels `SetEnvIf` setzen können.
4. Als Nächstes werden die Dokumente mit der höchsten `level`-Angabe im Header `Content-Type` ausgewählt. Wie bereits erwähnt, haben HTML-Dokumente die Voreinstellung `level=2` und alle anderen Dateien `level=0`.
5. Nun werden die Dokumente mit dem besten Zeichensatz gemäß den Qualitätsfaktoren oder der Reihenfolge nach aus dem Header `Accept-Charset` ausgewählt. Dokumente in `iso-8859-1` werden ebenfalls immer mitgeliefert, falls die Anfrage den Zeichensatz nicht ausdrücklich auf `q=0` setzt. `text/*`-Dokumente ohne Zeichensatzzuordnung gelten dabei automatisch als `iso-8859-1`.
6. Nun werden alle verbleibenden Varianten mit expliziter Zeichensatzinformation ausgewählt, die *nicht* `iso-8859-1` ist. Andernfalls werden alle Varianten gewählt.
7. Wenn es komprimierte Varianten gibt, die dem Wert von `Accept-Encoding` entsprechen, werden nur diese ausgewählt. Gibt es unpassende komprimierte sowie unkomprimierte Dateien, verbleiben nur die unkomprimierten in der Auswahl. Wenn es dagegen nur eine Sorte gibt, werden alle Dokumente gewählt.

8. Nun werden die Varianten mit der kleinsten `Content-Length` ausgewählt.
9. Falls an dieser Stelle noch immer mehrere Varianten übrig bleiben, wird die erste genommen. Dies ist entweder das erste in der Type-Map erwähnte Dokument oder das erste, das in Zeichensatzreihenfolge im Verzeichnis steht.

Irgendwann im Laufe der neun Verarbeitungsschritte hat sich ein Dokument als das »beste« herauskristallisiert. Diese Datei wird nun ausgeliefert. Die Antwort enthält den HTTP-Entity-Header `Vary`, der angibt, welche der vier Kategorien zur Content-Negotiation herangezogen wurden. Diese Information kann von Clients und Proxys für das Caching der Variante ausgewertet werden. Das folgende Beispiel zeigt die Ausgabe, wenn Dateityp (`accept`) und Sprache (`accept-language`) aus mehreren Alternativen ausgewählt wurden. Beispiel:

```
Vary: negotiate, accept, accept-language
```

Außerdem wird – ebenfalls für Caching-Zwecke – der Header `Content-Location` gesendet, der die konkrete URI der gewählten Variante enthält. Hier ein Beispiel:

```
Content-Location: test.html.de
```

### 7.3.2 Transparente Content-Negotiation

Apache beherrscht eine Teilmenge der transparenten Content-Negotiation (TCN) gemäß RFC 2295. Genauer gesagt wird die Content-Negotiation der vier bereits erläuterten Kategorien auch nach diesem Verfahren unterstützt, während die zusätzlich in RFC 2295 beschriebene Feature-Negotiation nicht verfügbar ist. Als Bestandteil der TCN kann Apache auch den *Remote Variant Selection Algorithm* (RVSA/1.0) ausführen, der in RFC 2296 abgehandelt wird. Er ähnelt dem Apache-Negotiation-Algorithmus und wird hier nicht näher beschrieben.

Die Besonderheit bei TCN besteht darin, dass der Server sich nicht auf die `Accept-*`-Header der Anfrage verlässt, sondern dem Client von vornherein eine Auswahlliste mit dem Status `300 Multiple Choices` oder `406 No Acceptable Variant` sendet.

Auch für TCN muss `mod_negotiation` aktiv sein. Die in Abschnitt 7.2, »MIME-Konfiguration«, beschriebenen MIME-Einstellungen und die Type-Map- beziehungsweise MultiViews-Optionen gelten grundsätzlich auch hier. TCN wird verwendet, wenn der Client einen `Negotiate`-Header sendet. Dieser hat die folgende Syntax:

```
Negotiate: trans | vlist | guess-small | RVSA-Versionsnr. | *
```

Die einzelnen Werte bedeuten Folgendes:

- `trans`: Der Client unterstützt transparente Content-Negotiation.
- `vlist`: wie `trans`, aber zusätzlich soll jede Antwort eine Liste von Varianten enthalten.
- `guess-small`: Wenn die vom Server ermittelte beste Variante nicht viel größer ist als eine Liste aller Varianten, darf der Server dieses Dokument liefern.
- RVSA-Versionsnummer: Der Server darf zur Ermittlung des besten Dokuments den RVSA-Algorithmus mit derselben Hauptversionsnummer ausführen.
- `*`: Der Server darf jede beliebige RVSA-Version oder einen eigenen Algorithmus verwenden, um die beste Variante zu ermitteln.

Die Liste, mit der Apache antwortet, enthält ebenfalls zwei zusätzliche Header:

- `TCN`
  Der `TCN`-Header gibt grundsätzlich an, dass transparente Content-Negotiation durchgeführt wurde. Für eine einfache Liste von Alternativen besitzt er den Wert `list`. Andere wichtige Werte sind `choice` (ein vom Server ausgewähltes Dokument mit zusätzlichem Variantenlisten-Header) und `adhoc` (einfaches Dokument ohne TCN zur Beantwortung inkompatibler Anfragen).
- `Alternates`
  Dies ist die eigentliche Liste der verfügbaren Varianten. Die Einträge werden, wie bei HTTP-Headern üblich, durch Kommas getrennt und haben folgendes Format:

  ```
  {"URI" qs {type MIME-Type} {language xx} {encoding Codierung}
  {length Bytes}}
  ```

  Natürlich sind sämtliche Informationen außer der URI optional, weil sie nur durch entsprechende Type-Map-Einstellungen oder MultiViews-Dokumente zur Verfügung gestellt werden.

Selbstverständlich gehört auch der `Vary`-Header hier wieder dazu. Im Folgenden sehen Sie ein konkretes Beispiel für ein solches Dokument, das dem Browser drei Varianten eines Dokuments zur Auswahl stellt:

```
HTTP/1.1 300 Multiple Choices
Date: Sun, 25 Aug 2008 17:21:45 GMT
Server: Apache/2.2.9 (Unix)
Alternates: {"test.html.de.gz" 1 {type text/html} {language de}
{encoding gzip} {length 231}}, {"test.html.de" 1 {type text/html}
{language de} {length 647}}, {"test.html.fr" 1 {type text/html}
{language fr} {length 783}}
Vary: negotiate,accept-language,accept-encoding
```

```
TCN: list
Content-Length: 519
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>300 Multiple Choices</title>
</head><body>
<h1>Multiple Choices</h1>
Available variants:
<ul>
<li><a href="test.html.de.gz">test.html.de.gz</a> ,
type text/html, language de, encoding gzip</li>
<li><a href="test.html.de">test.html.de</a> , type text/html,
language de</li>
<li><a href="test.html.fr">test.html.fr</a> , type text/html,
language fr</li>
</ul>
</body></html>
```

Wie Sie sehen, wird im Body des Dokuments eine manuelle Auswahlmöglichkeit in Form von Hyperlinks zur Verfügung gestellt. Das Negativum an der transparenten Content-Negotiation ist nämlich, dass sie bisher von den wenigsten Browsern unterstützt wird.

### 7.3.3 Konfigurationseinstellungen für Content-Negotiation

Die wichtigsten Informationen über Content-Negotiation haben Sie bereits in den vorangegangenen Unterabschnitten erhalten. Die grundsätzliche Aktivierung des Verfahrens erfolgt dabei nicht durch spezielle Direktiven. Sobald `mod_negotiation` aktiv ist, steht Content-Negotiation zur Verfügung. Alles Weitere wird durch Type-Maps oder MultiViews erledigt. Dennoch gibt es einige wenige Direktiven für die Einstellung bestimmter Aspekte der Content-Negotiation. Diese werden hier kurz vorgestellt.

#### MultiviewsMatch

Dateitypen, nach denen für MultiViews gesucht wird

| Seit Version | 2.0.26 |
|---|---|
| Modul | `mod_mime` |
| Kontext | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |

| Syntax | MultiviewsMatch Any \| NegotiatedOnly \| Filters \| Handlers [Handlers\|Filters] |
|---|---|
| Standardwert | NegotiatedOnly |

MultiviewsMatch bestimmt, welche Dateiendungen als Treffer für die MultiViews-Suche gelten sollen. Sie können folgende Werte angeben:

- NegotiatedOnly
  Dies ist der Standardwert. Er legt fest, dass nur diejenigen Dateiendungen für MultiViews verwendet werden, deren Eigenschaften mithilfe der weiter oben behandelten mod_mime-Direktiven bestimmt wurden.
- Handlers
  Neben den vier Negotiation-Kategorien werden auch die Handler-Dateiendungen in die Auswahl mit einbezogen.
- Filters
  Dieser Wert bestimmt, dass auch Dateiendungen betrachtet werden, die für Filter (siehe Kapitel 16, »SSI und Filter«) verwendet werden.
- Any
  Beliebige Dateiendungen werden in Betracht gezogen. Aus Sicherheitsgründen sollten Sie von dieser Variante absehen – andernfalls laufen Sie Gefahr, dass sogar automatische Sicherheitskopien mit Endungen wie .bak oder .old ausgeliefert werden, die nicht für die Öffentlichkeit bestimmt sind.

Die beiden Werte Handlers und Filters können Sie auch kombinieren, wenn Sie beide Optionen benötigen. Der Umfang von NegotiatedOnly ist ohnehin bereits in beiden enthalten. Any umfasst logischerweise alle anderen Werte.

### CacheNegotiatedDocs

Bestimmt, ob Content-Negotiation-Dokumente durch Proxys zwischengespeichert werden dürfen

| Seit Version | 1.x; Syntaxänderung in 2.0 |
|---|---|
| Modul | mod_negotiation |
| Kontext | Server, <VirtualHost> |
| Syntax | CacheNegotiatedDocs On\|Off |
| Standardwert | Off |

Diese Direktive bezieht sich lediglich auf HTTP/1.0-Anfragen, weil HTTP/1.1 – wie bereits angesprochen – eine erheblich genauere Cache-Steuerung zulässt. Wenn

Sie die Einstellung `CacheNegotiatedDocs On` vornehmen, wird das Caching von Content-Negotiation-Dokumenten erlaubt. Bei `Off` sendet Apache einen `Expires`-Header mit der aktuellen Uhrzeit, um das Caching ausdrücklich zu verhindern.

### LanguagePriority

Rangfolge für Sprachvarianten

| | |
|---|---|
| **Modul** | `mod_negotiation` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `LanguagePriority Sprachkürzel [ Sprachkürzel ...]` |
| **Standardwert** | nicht gesetzt |

Mit `LanguagePriority` können Sie eine Reihenfolge von Sprachversionen festlegen. Diese wird durch den Apache-Negotiation-Algorithmus in Betracht gezogen, wenn der Client keine Sprachpräferenzen übertragen hat. Beispiel:

```
LanguagePriority de en fr
```

### ForceLanguagePriority

Festlegung der Sprachpriorität, wenn mehr als ein Dokument passt

| | |
|---|---|
| **Seit Version** | 2.0.30 |
| **Modul** | `mod_negotiation` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `ForceLanguagePriority None \| Prefer \| Fallback [Prefer\|Fallback]` |
| **Standardwert** | `Prefer` |

Die Direktive `ForceLanguagePriority` legt fest, in welcher Weise die `LanguagePriority`-Liste verwendet werden soll. Die beiden möglichen Werte bedeuten Folgendes:

- `Prefer`
  Wenn mehr als eine Sprachversion mit derselben Präferenz auf die Anfrage passt, wird keine Antwort mit dem Status `300 Multiple Choices` zurückgegeben, sondern das Dokument, dessen Sprache als erste in der `LanguagePriority` steht.

- `Fallback`
  Ist keine Sprachversion vorhanden, die der Client-Anfrage entspricht, wird statt `406 No Acceptable Variant` die Datei mit der höchsten `LanguagePriority` ausgeliefert.

Sie können beide Werte angeben, um beide Optionen zu aktivieren.

## 7.4 Zusammenfassung

Die Manipulation von HTTP-Headern ist ein sehr mächtiges Hilfsmittel. Gerade mithilfe des modernen Moduls `mod_headers` können Sie beinahe jede beliebige Änderung an HTTP-Headern durchführen. Wenn Sie noch mehr Flexibilität benötigen, können Sie CGI-Skripte durch das Modul `mod_asis` mit dem Handler `send-as-is` versehen und deren Header als Teil der Ausgabe dynamisch erzeugen. Weitere Direktiven und Module ermöglichen die Berechnung und Ausgabe spezieller Header wie `ETag` oder `Expires`.

Noch wichtiger für den ernsthaften Betrieb einer Website ist die korrekte Einstellung der MIME-Header (`Content-*`) für Medientyp, Zeichensatz usw. Apache macht es Ihnen hier leicht; er kann diese Header automatisch anhand von Dateiendungen setzen. Dazu stehen die vielseitigen Funktionen und Direktiven des Moduls `mod_mime` zur Verfügung. Wenn alles andere fehlschlagen sollte, gibt es darüber hinaus immer noch die Möglichkeit, `mod_mime_magic` zu verwenden, um zumindest den MIME-Type aus dem Inhalt einer Datei zu ermitteln.

Unmittelbar auf die `mod_mime`-Einstellungen baut schließlich die Content-Negotiation auf. Sie wird durch das Standardmodul `mod_negotiation` bereitgestellt und ermöglicht Ihnen den Betrieb einer vielseitigen, internationalen Website: Die Wünsche der Clients, ausgedrückt durch die verschiedenen `Accept-*`-Anfrage-Header, können durch die Lieferung der jeweils passenden Dokumentvariante berücksichtigt werden. Zu diesem Zweck können Sie Type-Map-Dateien erstellen; diese müssen mit dem Handler `type-map` als solche deklariert werden. Alternativ können Sie auch dafür sorgen, dass Apache die Type-Map automatisch anhand von Dateiendungen erstellt.

*»Wie oft schlägt man einen Weg ein und wird davon abgeleitet!«*
*– Johann Wolfgang von Goethe*

# 8 Weiterleitungen und Indizes

In diesem Kapitel erhalten Sie weitere wichtige Informationen, die Sie für den Betrieb einer komplett ausgestatteten Website benötigen. Im ersten Abschnitt erfahren Sie alles über Aliase und Weiterleitungen – zwei etwas verschiedene Verfahren, die es ermöglichen, die URL der Client-Anfrage in etwas anderes umzusetzen als den entsprechenden Datei- oder Verzeichnispfad unterhalb der `DocumentRoot`. Zunächst wird das klassische Verfahren mit den Direktiven des Moduls `mod_alias` erläutert, danach geht es um das erheblich mächtigere und flexiblere `mod_rewrite`. Anschließend wird noch das Veröffentlichen von Benutzerverzeichnissen mit `mod_userdir` sowie die Umleitung zu angepassten Fehlermeldungsseiten erläutert.

Im erweiterten Zusammenhang dieses Themas ist es wichtig, zu wissen, dass Apache über das Modul `mod_autoindex` automatisch Indexseiten mit Verzeichnisinhalten generieren kann. Aussehen und Vollständigkeit dieses Indexes können sehr genau gesteuert werden. Indizes und die vergleichbaren serverseitigen Image Maps werden im zweiten Abschnitt dieses Kapitels behandelt.

## 8.1 Aliase und Weiterleitungen

Für eine Website ist es oft wichtig, die mit einer Anfrage gesendete URL zu manipulieren. Es gibt grundsätzlich zwei verschiedene Verfahren dafür:

- **Alias**
  Die URL-Änderung findet automatisch hinter den Kulissen statt, ohne dass der Client davon unterrichtet wird. Dies ist beispielsweise nützlich, um Dateien und Verzeichnisse, die sich aus Sicherheits- oder Organisationsgründen außerhalb der `DocumentRoot` befinden, einer URL zuzuordnen.
- **Weiterleitung (Redirect)**
  Der Client erhält eine Nachricht mit einem `3xx`-Statuscode. In einem `Location`-Header wird die neue URL des Dokuments mitgeteilt. Dieses Verfah-

ren sollten Sie z. B. immer dann wählen, wenn eine Datei sich nicht mehr an ihrem früheren Ort befindet.

Apache implementiert beide Methoden durch zwei unterschiedliche Module: den Klassiker `mod_alias` und das modernere, auf einer leistungsfähigen RegExp-Engine basierende Modul `mod_rewrite`. Beide werden in diesem Abschnitt vorgestellt. Einen Sonderfall von Aliasen haben Sie bereits in Kapitel 6, »Grundkonfiguration«, kennengelernt: das Einbinden der Benutzerverzeichnisse durch `mod_userdir`.

### 8.1.1 mod_alias

Dieses Modul enthält eine Reihe von Direktiven für die namensgebenden Aliase, aber auch für die Weiterleitung: `Alias` und `AliasMatch` dienen der URL-Änderung hinter den Kulissen, während die diversen `Redirect*`-Direktiven für Weiterleitungen zuständig sind.

#### Alias

Verzeichnis im Dateisystem einer URL zuordnen

| | |
|---|---|
| **Modul** | `mod_alias` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `Alias URL-Pfad Dateipfad\|Verzeichnispfad` |
| **Standardwert** | nicht gesetzt |

Die Direktive `Alias` ordnet eine Datei oder ein Verzeichnis, das außerhalb der `DocumentRoot` liegt, einer URL zu. Das erste Argument ist die gewünschte URL (genauer gesagt der URL-Pfad), das zweite ist der Pfad der Datei oder des Verzeichnisses im lokalen Dateisystem. Beispiel:

```
Alias /extern /usr/local/mydocs
```

Fordert ein User nach der Definition dieses Aliases die URL *http://www.mynet.de/extern/photo.jpg* an, dann wird die Datei */usr/local/mydocs/photo.jpg* geliefert.

Wenn der URL-Pfad mit einem Slash endet, muss dies auch für den Verzeichnispfad gelten. Beispiel:

```
Alias /images/ /usr/local/mydocs/images/
```

In diesem Fall wird allerdings eine Anfrage nach *www.mynet.de/images* (ohne abschließenden Slash) nicht, wie sonst üblich, durch eine automatische Weiterleitung auf die echte Verzeichnis-URL *www.mynet.de/images/* beantwortet, sondern mit `404 Not Found`.

Wenn Sie die in Kapitel 6, »Grundkonfiguration«, beschriebene restriktive Voreinstellung für das Wurzelverzeichnis vorgenommen haben, müssen Sie das Ziel des Aliases explizit freischalten – die Einstellungen für die `DocumentRoot` gelten in diesem Fall nicht. Hier ein Beispiel:

```
Alias /test/ /usr/local/mydocs/test/
<Directory /usr/local/mydocs/test>
  Options Indexes FollowSymLinks
  AllowOverride All
  Order allow,deny
  Allow from all
</Directory>
```

### AliasMatch

RegExp-Verzeichnismuster einem URL-Muster zuordnen

| | |
|---|---|
| **Modul** | `mod_alias` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `AliasMatch RegExp Dateipfad\|Verzeichnispfad` |
| **Standardwert** | nicht gesetzt |

`AliasMatch` funktioniert im Prinzip genau wie `Alias`. Der Unterschied besteht darin, dass als URL-Pfad ein regulärer Ausdruck verwendet wird. Geklammerte Teilausdrücke können im Datei- oder Verzeichnispfad durch `$1` bis `$9` wiedergegeben werden. Das folgende Beispiel bildet das Verzeichnis `/usr/local/mydocs/images/` auf die URI `/images` ab:

```
AliasMatch ^/images/(.*) /usr/local/mydocs/images/$1
```

Dieses Beispiel nutzt die Fähigkeiten von `AliasMatch` aus und ordnet die Verzeichnisse `/usr/local/mydocs/text0` bis `/usr/local/mydocs/text9` den URIs `/text0` bis `/text9` zu:

```
AliasMatch ^/text([0-9])/(.*) /usr/local/mydocs/text$1/$2
```

Die Optionen der mittels `AliasMatch` eingebundenen Verzeichnisse lassen sich mithilfe von `<DirectoryMatch>`- oder `<LocationMatch>`-Containern festlegen.

**Hinweis**

Beachten Sie bitte, dass zwei weitere `Alias`-Direktiven nicht hier, sondern in Kapitel 14, »CGI«, behandelt werden: `ScriptAlias` und `ScriptAliasMatch`. Sie funktionieren im Prinzip genau wie die Konfigurationsanweisungen `Alias` beziehungsweise `AliasMatch`, mit dem Unterschied, dass zusätzlich alle Dateien in den eingebundenen Verzeichnissen als CGI-Skripte betrachtet werden.

### Redirect

Sendet dem Client eine Umleitungsmitteilung

| Modul | mod_alias |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | Redirect [Status] URL-Pfad URL |
| Standardwert | nicht gesetzt |

Redirect ändert die angeforderte URL nicht serverseitig, sondern sendet dem Client eine Mitteilung über den geänderten Aufenthaltsort der angeforderten Ressource. Pflichtangaben sind der URL-Pfad, der in der Anfrage gefunden werden soll, und die URL, zu der die Umleitung erfolgen soll. Diese URL muss absolut sein, das heißt, sie muss mit *http://* und einem Hostnamen beginnen. Sie können daher nicht nur Weiterleitungen innerhalb Ihrer eigenen Site, sondern auch auf andere Websites durchführen.

Das folgende Beispiel leitet Anfragen für Dateien und Unterverzeichnisse unter */test* an die URL *http://www.mynet.de/woanders* weiter:

```
Redirect /test http://www.mynet.de/woanders
```

Das bedeutet, dass alles, was in der angeforderten URL hinter */test* stand, an *http://www.mynet.de/woanders* angehängt wird. Eine Anforderung des Bildes */test/logo.gif* würde mit folgender Weiterleitung auf *http://www. mynet.de/woanders/logo.gif* beantwortet:

```
HTTP/1.1 302 Found
Date: Tue, 26 Aug 2008 11:31:03 GMT
Server: Apache/2.2.9 (Unix)
Location: http://www.mynet.de/woanders/logo.gif
Content-Length: 221
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.mynet.de/woanders/
logo.gif">here</a>.</p>
</body></html>
```

Der wichtigste Teil der Antwort ist der `Location`-Header, der dem Client den geänderten Aufenthaltsort der angeforderten Ressource mitteilt. Für Clients, die nicht in der Lage sind, Weiterleitungen automatisch zu folgen, enthält der Body zusätzlich einen entsprechenden Hyperlink.

Ein optionales Argument für `Redirect` ist der HTTP-Statuscode, mit dem die Weiterleitungsmitteilung an den Client versendet wird. Für die folgenden vier Statuscodes existiert eine spezielle Bezeichnung:

- `permanent`
  Die Mitteilung wird als dauerhafte Weiterleitung mit dem Statuscode `301 Moved Permanently` versendet. Das folgende Beispiel leitet Anfragen für */old* dauerhaft nach *http://www.newaddress.org/new* um:

  ```
  Redirect permanent /old http://www.newaddress.org/new
  ```

  Dies ist z. B. nützlich, wenn sich Ihr Haupt-Domain-Name aus irgendeinem Grund geändert haben sollte. Angenommen, Sie haben Ihr Studium beendet und ziehen mit Ihrer Website von *www.uni-abc.de/~meinname* auf die neu reservierte Domain *www.meinname.de* um. In einer *.htaccess*-Datei in Ihrem Webverzeichnis könnten Sie die folgende Umleitung unterbringen:

  ```
  Redirect permanent /~meinname http://www.meinname.de
  ```

  Wie Sie sehen, muss der umzuleitende URL-Pfad auch dann mit einem `/` beginnen und den Beginn der URL bis zum aktuellen Verzeichnis wiedergeben, wenn die Direktive in einer *.htaccess*-Datei oder in einem Container-Kontext steht.

- `temp`
  Wie Sie am ersten Beispiel weiter oben sehen konnten, ist dies der Standardwert; Sie können ihn auch weglassen. Er erzeugt eine vorübergehende Weiterleitung – aus Kompatibilitätsgründen mit dem älteren Statuscode `302 Found`, nicht mit `307 Temporary Redirect`.

- `seeother`
  Die Meldung erhält den Statuscode `303 See Other`.

- `gone`
  Dies erzeugt den Status `410 Gone`, der anzeigt, dass die angeforderte Ressource nicht mehr existiert. Für diesen Statuscode ergibt eine URL keinen Sinn; deshalb darf sie auch nicht angegeben werden.

Andere Statuscodes als diese vier können Sie numerisch angeben. Beachten Sie, dass alle `3xx`-Codes eine Umleitungs-URL für den `Location`-Header benötigen. Alle anderen Fehlercodes erhalten dagegen keine URL. Das folgende Beispiel sendet für alle URIs unter `/nichtda` mutwillig ein `404 Not Found`:

```
Redirect 404 /nichtda
```

**Vorsicht Falle!**

Bei `Redirect` können Sie in die hier beschriebene Falle tappen: Eine Weiterleitung in ein Unterverzeichnis des aktuellen Verzeichnisses funktioniert nicht ohne Weiteres. Schauen Sie sich die folgende `Redirect`-Anweisung an, die zunächst völlig harmlos aussieht:

```
Redirect /info http://www.mynet.de/info/neu
```

Nehmen Sie zur Verdeutlichung an, der Server erhält eine Anfrage mit der URL *www.mynet.de/info/muster.gif*. Laut `Redirect`-Direktive sollen alle URIs, die mit */info* beginnen, auf *http://www.mynet.de/info/neu* weitergeleitet werden. Die Weiterleitung erzeugt also eine neue Anfrage mit folgender Adresse: *http://www.mynet.de/info/neu/muster.gif*. Da der `Redirect`-Befehl jedoch auch auf diese Anfrage zutrifft, wird erneut eine Weiterleitungsmitteilung gesendet – diesmal mit der URL *http://www.mynet.de/info/neu/neu/muster.gif*! Das geht so lange weiter, bis der Client oder der Server aufgibt oder bis die maximale URL-Länge erreicht ist …

Wenn Sie diese Art der Weiterleitung tatsächlich benötigen, funktioniert sie entweder über einen `Alias` – der eben keine neue Anfrage erzeugt – oder aber mithilfe der komplexen Regeln von `mod_rewrite`.

### RedirectMatch

Sendet dem Client eine Umleitungsmitteilung, wenn die URL einem RegExp entspricht

| | |
|---|---|
| **Modul** | `mod_alias` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `RedirectMatch [Status] RegExp URL` |
| **Standardwert** | nicht gesetzt |

`RedirectMatch` funktioniert genau wie `Redirect`. Allerdings können Sie als URL-Pfad einen regulären Ausdruck verwenden. Das folgende Beispiel leitet Anfragen nach URIs unter den Verzeichnissen */images00* bis */images99* permanent an die Verzeichnisse */00* bis */99* auf dem Host *bilder.mynet.de* weiter:

```
RedirectMatch permanent ^/images([0-9]{2})/(.*) \
http://bilder.mynet.de/$1/$2
```

Die Anfrage nach *www.mynet.de/images5/photo7.jpg* wird also beispielsweise mit einer Weiterleitung auf *bilder.mynet.de/5/photo7.jpg* beantwortet.

Anders als bei einem einfachen `Redirect` kann das Weiterleitungsziel von `RedirectMatch` auch etwas anderes sein als ein geändertes Präfix mit demselben Pfad. Das liegt daran, dass dieser Pfad bei `RedirectMatch` nicht automatisch angehängt wird. Deshalb könnten Sie beispielsweise alle Anfragen, die mit einem bestimmten URL-Pfad beginnen, auf eine einzige Datei umleiten. Beispiel:

```
RedirectMatch temp ^/news http://www.mynet.de/newsinfo.html
```

Diese Zeile könnten Sie z. B. einsetzen, falls sich der gesamte Inhalt des Unterverzeichnisses */news* gerade in einer Überarbeitung befinden sollte. Für jede Anfrage, deren URI mit */news* beginnt, würde daraufhin die Datei */newsinfo.html* angezeigt.

Hier noch ein Beispiel, das auf den ersten Blick etwas seltsam erscheint:

```
RedirectMatch ^/search/(.*) http://www.google.de/search?q=$1
```

Wie Sie sehen, werden sämtliche Anfragen, deren URL-Pfad mit */search/* beginnt, an die Suchmaschine Google weitergeleitet, genauer gesagt als Wert des Query-Feldes `q` für die URI */search* – dies ist der Suchbegriff. Mit anderen Worten: Google sucht nach dem Begriff, den Sie an *www.mynet.de/search/* angehängt haben. Das Prinzip ist also nicht weiter kompliziert.

Die weitaus interessantere Frage ist, wozu man so etwas überhaupt benötigt. Es könnte Sie beispielsweise ärgern, dass es keine Spuren in Ihren Log-Dateien (Näheres dazu in Kapitel 11, »Logging«) hinterlässt, wenn ein User einen externen Hyperlink anklickt. Besonders bedauerlich ist das, wenn Sie so etwas wie einen Amazon-Partner-Shop betreiben – Sie wissen nie, ob sich jemand für Ihre Angebote interessiert. Dies können Sie auf eine ähnliche Weise ändern. Zunächst einmal benötigen Sie eine `RedirectMatch`-Direktive wie diese:

```
RedirectMatch ^/elsewhere/(.*) http://$1
```

Anschließend müssen Sie die URLs der externen Hyperlinks in Ihren HTML-Dokumenten, für die Sie Log-Einträge benötigen, nach folgendem Schema ändern: Aus `http://andere-site.com/pfad/usw` wird `/elsewhere/andere-site.com/pfad/usw`.

Ein *Amazon.de*-Partner-Link auf das vorliegende Buch hat beispielsweise folgende Original-URL:

*http://www.amazon.de/dp/3836213257/my-partner-id*

Wenn die genannte Direktive in Kraft ist, müssen Sie sie in HTML-Dateien durch folgende Link-URL ersetzen:

*/elsewhere/www.amazon.de/dp/3836213257/my-partner-id*

Sobald jemand den entsprechenden Link anklickt, enthält Ihre `AccessLog`-Datei einen Eintrag wie diesen:

```
217.63.82.109 - - [29/Aug/2008:13:11:07 +0100] "GET /elsewhere/
www.amazon.de/dp/3836213257/my-partner-id HTTP/1.1" 302 249 "-"
```

```
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 4.0; .NET CLR
1.0.3705)"
```

In Kapitel 11, »Logging«, werden einige nützliche Perl-Skripte und Hilfsprogramme vorgestellt, mit denen sich Log-Dateien nach solchen Informationen filtern lassen. Beachten Sie die `302` in der Log-Zeile – sie weist darauf hin, dass der Browser des Besuchers keine lokale Datei erhalten hat, sondern eine Weiterleitungsmitteilung (`302 Found`).

#### RedirectPermanent

Sendet dem Client eine Mitteilung über eine permanente Umleitung

| | |
|---|---|
| **Modul** | `mod_alias` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `RedirectPermanent URL-Pfad URL` |
| **Standardwert** | nicht gesetzt |

Dies ist lediglich eine spezielle Schreibweise für `Redirect permanent`, die aus Kompatibilitätsgründen beibehalten wurde.

#### RedirectTemp

Sendet dem Client eine Mitteilung über eine vorübergehende Umleitung

| | |
|---|---|
| **Modul** | `mod_alias` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `RedirectTemp URL-Pfad URL` |
| **Standardwert** | nicht gesetzt |

Diese Direktive ist nichts weiter als eine veraltete Alternative zu `Redirect temp`.

### 8.1.2 mod_rewrite

Wie Sie gesehen haben, ist `mod_alias` zwar nützlich, stößt aber in puncto Flexibilität relativ schnell an seine Grenzen. Hier hat das Modul `mod_rewrite` erheblich mehr zu bieten. Es wurde 1996 von Ralf S. Engelschall geschrieben und gehört seit 1997 zur Apache-Distribution. In der Apache-Dokumentation wird es zu Recht als »Swiss Army-Knife of URL manipulation« bezeichnet. Es kann die volle Funktionalität von `mod_alias` übernehmen, bietet aber noch unzählige weitere Optionen.

Der Kern des Moduls ist eine leistungsfähige Engine für mit Perl kompatible reguläre Ausdrücke.[1] Technisch gesehen werden die Funktionen des Moduls an zwei Stellen während der Verarbeitung einer Anfrage aufgerufen: `Rewrite`-Direktiven, die in der Hauptkonfigurationsdatei stehen, werden während der Umwandlung der angeforderten URL in einen Dateipfad ausgeführt. Stehen dagegen `Rewrite`-Anweisungen in *.htaccess*-Dateien, werden diese erst in der sogenannten *Fixup-Phase* befolgt: Da *.htaccess*-Dateien unterhalb der `DocumentRoot` liegen, können sie erst nach der eigentlichen URL-Umwandlung gelesen werden. Da es an dieser Stelle also eigentlich bereits zu spät für URL-Modifikationen ist, erzeugt eine hier befindliche `Rewrite`-Regel intern eine Art neue Anfrage.

In diesem Abschnitt werden nun zunächst die Direktiven von `mod_rewrite` beschrieben. Anschließend werden einige ausführliche Beispiele behandelt. Beides ist lediglich eine Einführung, denn: »Despite the tons of examples and docs, `mod_rewrite` is voodoo. Damned cool voodoo, but still voodoo.« (Brian Moore; zitiert in der `mod_rewrite`-Dokumentation).

Um erfolgreich mit `mod_rewrite` arbeiten zu können, benötigen Sie Kenntnisse über reguläre Ausdrücke. Eine Übersicht über die wichtigsten Konstrukte finden Sie in Anhang G. Eine sehr ausführliche Anleitung bietet [FRIED2007]. Relativ gründlich können Sie sich auch von der POD-Dokumentation der Sprache Perl informieren lassen, wenn diese auf Ihrem System installiert ist. Geben Sie dazu auf der Konsole Folgendes ein:

```
$ perldoc perlre
```

#### RewriteEngine

`Rewrite`-Funktionalität ein-/ausschalten

| | |
|---|---|
| **Seit Version** | 1.3 |
| **Modul** | `mod_rewrite` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `RewriteEngine on\|off` |
| **Standardwert** | `off` |

Bevor Sie die Funktionalität von `mod_rewrite` nutzen können, müssen Sie sie mithilfe dieser Direktive einschalten:

1 Beachten Sie, dass die Bibliothek PCRE (Perl-Compatible Regular Expressions), die nicht nur in Apache, sondern in vielen verschiedenen Programmen und Sprachen eingesetzt wird, nicht ganz denselben Leistungsumfang besitzt wie das Pattern Matching in Perl selbst.

```
RewriteEngine On
```

Anders als zahlreiche andere Konfigurationseinstellungen wird diese Aktivierung von `RewriteEngine` nicht an Subkontexte vererbt: Wenn Sie die `RewriteEngine` beispielsweise im Server-Kontext einschalten, gilt dies nicht für `<VirtualHost>`-, `<Directory>`- oder sonstige Container. Die Funktion muss deshalb in jedem gewünschten Kontext explizit aktiviert werden. Insofern hat das ausdrückliche Ausschalten mittels `RewriteEngine Off` in der Praxis keine besonders große Bedeutung.

#### RewriteRule

Definiert `Rewrite`-Regeln

| | |
|---|---|
| **Seit Version** | 1.3; `cookie`-Flag seit 2.0.40 |
| **Modul** | `mod_rewrite` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `RewriteRule [!]RegExp Ersatztext [Flag, ...]` |
| **Standardwert** | nicht gesetzt |

Dies ist die wichtigste Direktive von `mod_rewrite`. Sie kann unabhängig von anderen Direktiven dieses Moduls aufgerufen werden und funktioniert dann ähnlich wie `AliasMatch` oder `RedirectMatch`. Standardmäßig werden allerdings alle untereinanderstehenden `RewriteRule`-Anweisungen nacheinander ausgeführt. Dabei wird die URL unter Umständen mehrfach umgeschrieben.

Das erste Argument ist ein beliebiger Perl-kompatibler regulärer Ausdruck. Die `Rewrite`-Regel wird angewendet, wenn das angegebene Muster im URL-Pfad der Client-Anfrage gefunden wird. Optional können Sie dem regulären Ausdruck ein Ausrufezeichen voranstellen. In diesem Fall wird die `RewriteRule` nur beachtet, wenn er *nicht* zutrifft.

Der Ersatztext ist ein beliebiger String. Darin können Sie eine Reihe besonderer Elemente verwenden:

- `$1` bis `$9`: Diese Platzhalter entsprechen den im regulären Ausdruck gefundenen Teilausdrücken, die Sie in Klammern setzen. Die Nummerierung erfolgt von links nach rechts, bei verschachtelten Klammern von außen nach innen.
- `%1` bis `%9`: Mithilfe dieser Platzhalter können Sie Bezug auf geklammerte Ausdrücke aus `RewriteCond`-Direktiven nehmen.
- `%{VAR}`: Dies ermöglicht das Einbinden der angegebenen Server-Variablen. An dieser Stelle sind die folgenden Variablen zulässig:

- Einige HTTP-Anfrage-Header. Ihr Name wird komplett in Großbuchstaben gesetzt; Bindestriche werden durch Unterstriche ersetzt. Die zulässigen Header sind `HTTP_USER_AGENT`, `HTTP_REFERER`, `HTTP_COOKIE`, `HTTP_FORWARDED`, `HTTP_HOST`, `HTTP_PROXY_CONNECTION` und `HTTP_ACCEPT` (ihre Bedeutung können Sie in Kapitel 2, »Funktionsweise von Webservern«, nachlesen).
- Informationen über Anfrage und Verbindung: `REMOTE_ADDR`, `REMOTE_HOST`, `REMOTE_USER`, `REMOTE_IDENT`, `REQUEST_METHOD`, `SCRIPT_FILENAME`, `PATH_INFO`, `QUERY_STRING` und `AUTH_TYPE`. Näheres über die meisten dieser Variablen steht in Kapitel 14, »CGI«.
- Informationen über den Server selbst: `DOCUMENT_ROOT`, `SERVER_ADMIN`, `SERVER_NAME`, `SERVER_ADDR`, `SERVER_PORT`, `SERVER_PROTOCOL` und `SERVER_SOFTWARE`. Viele dieser Variablen dürften Ihnen bekannt vorkommen, weil es ähnlich lautende Konfigurationsdirektiven gibt (siehe Kapitel 6, »Grundkonfiguration«). Ansonsten werden auch diese Variablen in Kapitel 14, »CGI«, erläutert.
- Datum und Uhrzeit: `TIME_YEAR`, `TIME_MON`, `TIME_DAY`, `TIME_HOUR`, `TIME_MIN`, `TIME_SEC`, `TIME_WDAY` und `TIME`. Die Namen dieser Variablen dürften selbsterklärend sein.
- `API_VERSION`: Diese Nummer entspricht der in Kapitel 5, »Apache in Betrieb nehmen«, angesprochenen Module Magic Number; sie gibt unter anderem Auskunft darüber, welche Versionen kompilierter DSO-Module mit der aktuellen Server-Version kompatibel sind. Beispielwert: `20051115:15` (Apache 2.2.9).
- `THE_REQUEST`: Dies ist die vollständige Startzeile der HTTP-Client-Anfrage. Beispiel: `GET /dir/file HTTP/1.1`
- `REQUEST_URI`: die angeforderte Ressource
- `REQUEST_FILENAME`: der lokale Pfad- und Dateiname der angeforderten Ressource
- `IS_SUBREQ`: Diese Variable besitzt den Wert `true`, wenn es sich bei der aktuellen HTTP-Anfrage nicht um eine Client-Anfrage, sondern um eine interne Unteranfrage handelt.

Abgesehen davon gibt es noch einige Sonderformen:

- `%{ENV:Variable}` liefert die angegebene Umgebungsvariable. Variablen ohne das Präfix `ENV` sind keine Umgebungsvariablen, sondern spezielle `mod_rewrite`-Server-Variablen – selbst dann, wenn sie denselben Namen tragen (und in aller Regel natürlich denselben Wert liefern).

  - `%{HTTP:Header}` liefert den Wert des angegebenen HTTP-Anfrage-Headers – beispielsweise erhalten Sie über `%{HTTP:User-Agent}` den Namen des anfragenden Client-Programms. Hier stehen auch Header zur Verfügung, für die nicht eine Standardform wie `%{HTTP_REFERER}` definiert ist.
  - `%{LA-U:Variable}` führt eine URL-basierte interne Unteranfrage durch, um den Wert der angegebenen Variablen zu ermitteln. Dieser Mechanismus wird für einige Variablen benötigt, deren Wert erst nach der URL-Umwandlung (die Phase, während der `mod_rewrite` im Server-Kontext arbeitet) feststeht. Ein gutes Beispiel ist der Benutzername des entfernten Benutzers, `%{LA-U:Remote-User}` – Authentifizierungsinformationen stehen erst später in der Anfrageverarbeitung zur Verfügung.[2]
  - `%{LA-F:Variable}` ist eine dateinamenbasierte Unterabfrage. In der Regel besteht kein Unterschied zur URL-basierten Form `%{LA-U:Variable}`.
- `${Map-Name:Schlüssel[|Standardwert]}`: Mithilfe von `RewriteMap` können Sie eine externe Datei definieren, in der sich Schlüssel-Wert-Paare befinden. Diese Formulierung ermöglicht es Ihnen, auf die durch `Map-Name` bezeichnete `RewriteMap` zuzugreifen und daraus den Wert auszulesen, der für `Schlüssel` definiert ist. Optional können Sie einen `Standardwert` angeben, der verwendet wird, wenn der Schlüssel in der Map nicht gefunden wird.

Als drittes Argument können optional zahlreiche Flags angegeben werden. Diese werden durch Kommas voneinander getrennt und stehen in eckigen Klammern. Folgende Flags sind definiert:

- `redirect|R[=Statuscode]`: Normalerweise sorgt `RewriteRule` für eine einfache interne URI-Änderung. Wenn Sie eine Weiterleitung durchführen möchten, müssen Sie die Ersatz-URL mit `http://` beginnen. Mithilfe dieses Flags können Sie zusätzlich einen bestimmten Statuscode setzen: `temp` (`302 Found` – Standard), `permanent` (`301 Moved Permanently`), `seeother` (`303 See Other`) beziehungsweise einen numerischen Wert aus dem `3xx`- oder `4xx`-Bereich.
- `forbidden|F`: Dies setzt den Statuscode `403 Forbidden`: Wenn der Mustervergleich zutrifft, wird dem Client der Zugriff auf die angeforderte Ressource ausdrücklich verboten.
- `gone|G`: Durch den Statuscode `410 Gone` wird dem Client mitgeteilt, dass die angeforderte Ressource dauerhaft entfernt worden ist.
- `proxy|P`: Dieses Flag beendet das Rewriting – nachfolgende `RewriteRule`-Anweisungen werden nicht mehr beachtet. Die Ziel-URL wird an `mod_proxy`

2 Das ist nicht der Fall, wenn Sie `RewriteRule` in einem Verzeichniskontext einsetzen – hier können Sie einfach `%{REMOTE_USER}` verwenden.

weitergereicht und erzeugt eine Proxy-Anfrage. Beachten Sie, dass der Ersatztext eine vollständige URL sein muss, die mit *http://* beginnt.

- `last|L`: Mit diesem Flag können Sie dafür sorgen, dass das Rewriting sofort beendet wird; weitere `RewriteRules` werden nicht mehr beachtet.
- `END` (seit Version 2.3): Wirkt wie `L`, verhindert aber zusätzlich die Ausführung eventueller `RewriteRules` in .htaccess-Dateien.
- `next|N`: Dies sorgt dafür, dass das komplette Rewriting erneut durchgeführt wird, beginnend bei der ersten `RewriteRule` in der Reihe. Dies ist beispielsweise für das mehrfache Suchen und Ersetzen eines Musters nützlich. Die Bedingung beziehungsweise das Suchmuster muss allerdings so beschaffen sein, dass irgendwann ein Abbruch stattfindet – andernfalls kommt es zu einer Endlosschleife. Ein `[L]`-Flag in einer weiter oben stehenden Regel kann das Problem natürlich ebenfalls lösen.
- `chain|C`: Dieses Flag ermöglicht eine Art logisches Und mit Short-Circuit-Verfahren: Die nächste `RewriteRule` wird nur dann überhaupt geprüft, wenn die aktuelle zutrifft. Andernfalls werden alle mittels `[C]`-Flags verketteten Regeln übersprungen.
- `type|T=MIME-Type`: Dies sorgt dafür, dass der Zieldatei der angegebene MIME-Type für den `Content-Type`-Header zugewiesen wird.
- `nosubreq|NS`: Die Regel soll nur angewendet werden, wenn die aktuell verarbeitete Anfrage eine echte Client-Abfrage und keine interne Unterabfrage ist.
- `nocase|NC`: Beim Suchmuster wird nicht zwischen Groß- und Kleinschreibung unterschieden.
- `qsappend|QSA`: Wenn der Ersatztext ein Fragezeichen enthält und auf diese Weise einen Query-String bildet, sollen die entsprechenden Daten den bereits vorhandenen Query-String nicht ersetzen, sondern an diesen angehängt werden.
- `qsdiscard|QSD` (seit Version 2.4): Falls die ursprüngliche URL einen Query-String enthält, wird dieser nicht wie üblich beibehalten, sondern entfernt. `QSD` hat dabei Vorrang vor `QSA`.
- `noescape|NE`: Im Ersatztext soll kein URL-Escaping durchgeführt werden. Dies ist z. B. nützlich, wenn `%`-Zeichen nicht codiert werden, sondern manuelles Escaping bewirken sollen.
- `passthrough|PT`: Das Ergebnis der `RewriteRule` wird so aufbereitet, dass der nächste Handler, der sich um die Umsetzung der Anforderungs-URL in einen Dateinamen kümmert, damit zurechtkommt. Intern wird das Feld `request_rec->uri` wieder auf den Wert von `filename` gesetzt. Dieses Flag ist beispielsweise erforderlich, wenn `mod_rewrite` und `mod_alias` gemeinsam verwendet werden.

- `skip|S=n`: Die nächsten `n` `RewriteRules` sollen übersprungen werden. Dieses Flag ist für `if/else`-artige Fallentscheidungen geeignet.
- `env|E=VAR:VAL`: Setzt die angegebene Umgebungsvariable (`VAR`) auf den genannten Wert (`VAL`).
- `cookie|CO=Name:Wert[:Lifetime[:Pfad]]`: Dies sorgt dafür, dass ein Cookie gesetzt wird. `Name` und `Wert` sind obligatorisch, während `Lifetime` (Gültigkeitsdauer) und `Pfad` (Geltungsbereich) des Cookies optional sind. Cookies ohne Gültigkeitsdauer sind sogenannte *Session-Cookies*, die am Ende der Client-Sitzung automatisch gelöscht werden.

Beachten Sie zu guter Letzt Folgendes: Wenn Sie `RewriteRule` in Verzeichniskontexten einsetzen, wird der Pfad bis zum entsprechenden Verzeichnis vor dem Rewriting aus der URL entfernt und erst danach wieder hinzugefügt. Eine Ausnahme tritt ein, wenn der Ersatzstring mit *http://* beginnt.

#### RewriteCond

Bedingung, unter der `Rewrite` durchgeführt wird

| | |
|---|---|
| **Seit Version** | 1.3 |
| **Modul** | `mod_rewrite` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `RewriteCond Teststring RegExp` |
| **Standardwert** | nicht gesetzt |

Diese Direktive formuliert gewissermaßen eine Vorbedingung für nachfolgende `RewriteRule`-Anweisungen, die zusätzlich zu dem RegExp-Match zutreffen muss. Wenn Sie mehrere `RewriteCond`-Direktiven untereinanderschreiben, müssen sie standardmäßig alle zutreffen, damit die `RewriteRule`-Anweisungen beachtet werden.

Das erste Argument, der `Teststring`, ist einfacher Text, der zusätzlich folgende Konstrukte enthalten kann:

- `$1` bis `$9`: Referenz auf einen geklammerten Teilausdruck aus dem regulären Ausdruck der nachfolgenden `RewriteRule`-Direktive
- `%1` bis `%9`: Referenz auf einen geklammerten Teilausdruck im regulären Ausdruck der vorigen `RewriteCond`
- `${Map-Name:Schlüssel[|Standardwert]}`: Referenz auf den angegebenen `Schlüssel` in der durch `Map-Name` bezeichneten `RewriteMap`

- `%{VAR}`: Zugriff auf die benannte Server-Variable (Näheres weiter oben in der Beschreibung der Direktive `RewriteRule` und in Kapitel 14, »CGI«)

Das zweite Argument von `RewriteCond` ist ein mit Perl kompatibler regulärer Ausdruck, mit dem das erste Argument verglichen wird. Diesem können Sie optional ein Ausrufezeichen voranstellen – die `RewriteCond` gilt dann nur als zutreffend, wenn der Ausdruck *nicht* passt. Alternativ können Sie statt eines regulären Ausdrucks auch folgende Spezialformulierungen verwenden:

- `<String`: Trifft zu, wenn das erste Argument im Alphabet (genauer gesagt in der Zeichensatzreihenfolge) vor dem hier angegebenen String steht.
- `>String`: Wahr, wenn das erste Argument in der Zeichensatzreihenfolge nach diesem Text folgt.
- `=String`: Zutreffend, wenn das erste Argument genau mit dem Teststring identisch ist. Dies ist eine praktische Kurzfassung für den regulären Ausdruck `^String$`.
- `-d`: Wahr, wenn das erste Argument ein Verzeichnis (Directory) ist.
- `-f`: Ist zutreffend, wenn das erste Argument eine normale Datei (File) ist.
- `-s`: Zutreffend, wenn das erste Argument eine Datei mit Inhalt ist – erkennbar an einer Dateigröße (Size) von über 0 Byte.
- `-l`: Wahr, wenn das erste Argument ein symbolischer Link ist.
- `-F`: Dieses Argument führt eine Unterabfrage durch, um zu überprüfen, ob es sich beim ersten Argument um eine tatsächlich verfügbare Datei handelt.
- `-U`: Führt ebenfalls eine Unterabfrage durch und prüft, ob das erste Argument eine verfügbare URL ist.

Alle diese Sonderargumente können Sie ebenfalls mit einem vorangestellten Anführungszeichen verneinen.

Das optionale dritte Argument sind auch hier Flags, die durch Kommata getrennt werden und in eckigen Klammern stehen. Die folgenden beiden Flags sind hier definiert:

- `NC|nocase`
  Im regulären Ausdruck des zweiten Arguments soll nicht zwischen Groß- und Kleinschreibung unterschieden werden.
- `OR|ornext`
  Zwischen der aktuellen und der nächsten `RewriteCond`-Anweisung wird ein logisches Oder gesetzt – es genügt, wenn eine von ihnen zutrifft. Der Normalfall ist dagegen ein logisches Und: Alle aufeinanderfolgenden `RewriteCond`-Direktiven müssen zutreffen, damit die nachfolgenden `RewriteRule`-Anweisungen beachtet werden.

### RewriteBase

Basis-URL für `Rewrite`-Verzeichnisse

| | |
|---|---|
| **Seit Version** | 1.3 |
| **Modul** | `mod_rewrite` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `RewriteBase URL-Pfad` |
| **Standardwert** | aktuelles Verzeichnis (Näheres siehe Text) |

Diese Direktive gibt den Basis-Verzeichnispfad für `Rewrite`-Operationen an. Dies kann für `RewriteRule`-Operationen in `<Directory>`-Containern verwendet werden.

Der Standardwert ist der tatsächliche Verzeichnispfad im Dateisystem. Wie bereits erwähnt, wird bei `RewriteRule`-Direktiven in untergeordneten Kontexten zunächst der URL-Pfad dieser Kontexte entfernt; nach erfolgtem `Rewrite` wird er wieder hinzugefügt.

In der Regel ist die Voreinstellung in Ordnung. Geändert werden muss sie nur, wenn der URL-Pfad des Containers nicht dem Verzeichnispfad entspricht – dies ist z. B. bei Verzeichnissen der Fall, die durch `Alias`-Anweisungen eingebunden wurden.

Betrachten Sie dazu folgendes Beispiel:

```
# Einbinden des Verzeichnisses /woanders unter der URL /test
Alias /test /woanders

# Einstellungen für das Verzeichnis /woanders
<Directory /woanders>
  # URL-Pfad für Rewrites muss auf /test gesetzt werden
  RewriteBase /test
  RewriteRule ...
</Directory>
```

### RewriteMap

Schlägt einen `Rewrite`-Schlüssel in einer Map-Datei nach

| | |
|---|---|
| **Seit Version** | 1.3; DBM-Auswahl seit 2.0.41 |
| **Modul** | `mod_rewrite` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `RewriteMap Map-Name Map-Typ:Map-Datei` |
| **Standardwert** | nicht gesetzt |

Diese Direktive bindet eine Datei mit Schlüssel-Wert-Paaren für das Nachschlagen von `Rewrite`-Regeln ein. Solche `RewriteMap`-Dateien sind praktisch, wenn Sie in `RewriteRule`-Ersatztexten oder `RewriteCond`-Teststrings häufiger bestimmte Textbausteine benötigen: Sie können den Schlüssel angeben, und der entsprechende Wert wird automatisch eingesetzt.

In einer `RewriteRule`- oder `RewriteCond`-Direktive erfolgt der Zugriff auf die `RewriteMap` nach folgendem Schema:

```
${Map-Name:Schlüssel}
```

Aus Sicherheitsgründen ist es besser, zusätzlich einen Standardwert anzugeben. Dieser wird eingesetzt, wenn der Schlüssel nicht gefunden wird. Die Schreibweise sieht so aus:

```
${Map-Name:Schlüssel|Standardwert}
```

Der `Map-Name` entspricht dem ersten Argument von `RewriteMap`. Es handelt sich um einen beliebigen String, den Sie sich aussuchen können. Das zweite Argument ist die durch Doppelpunkt getrennte Kombination aus dem Typ der `RewriteMap` und dem entsprechenden Dateinamen. Folgende Typen sind definiert:

- `txt`: einfache Textdatei
  Dies ist die einfachste Art einer Map-Datei. Jede Zeile enthält ein durch Leerzeichen getrenntes Schlüssel-Wert-Paar. Leerzeilen und Kommentare (Text nach einer *#*) werden ignoriert. Sollten Sie einen Schlüssel mehrfach definieren, gilt das erste Vorkommen. Hier ein Beispiel für eine solche Definition:

  ```
  RewriteMap mailmap txt:/Pfad/von/mailmap.txt
  ```

  Die Datei könnte beispielsweise folgendermaßen aussehen:

  ```
  # mailmap.txt - E-Mail-Adressen von Mitarbeitern
  ps  peterschmitz@mynet.de   # Peter Schmitz, Vertrieb
  bm  bertmueller@mynet.de    # Bert Müller, Technik
  ab  antonbecker@mynet.de    # Anton Becker, IT
  ```

  Wenn Sie nun in einer URL – z. B. in einem Query-String – eine der E-Mail-Adressen benötigen, funktioniert der Zugriff in etwa so:

  ```
  ${mailmap:ps|info@mynet.de}
  ```

  Damit nicht die Gefahr entsteht, gar keine E-Mail-Adresse zu erhalten, wird die Hauptfirmenadresse `info@mynet.de` als Standardwert verwendet.

- `rnd`: Textdatei mit Werten zur zufälligen Auswahl
  Auch dieser Map-Typ ist eine gewöhnliche Textdatei. Die Besonderheit besteht darin, dass Sie für jeden Schlüssel mehrere, durch Pipe-Zeichen (`|`) getrennte Werte angeben können. Bei jedem Zugriff auf den Schlüssel wählt Apache einen der Werte zufällig aus.

  Interessant ist dieses Verfahren für vielfältige Zwecke: Naheliegend wäre die zufällige Auswahl eines Bildes, eines Zitats oder einer URL aus einer Linkliste. Selbst für einfaches Load-Balancing taugt das Verfahren: Aus einer Liste von Host-Namensbestandteilen wie `www1` bis `www9` könnte bei jeder Anfrage ein anderer ausgewählt werden. Näheres dazu finden Sie in Kapitel 12, »Skalierung und Performance-Tuning«.

  Das folgende Beispiel ermöglicht die Auswahl eines Bildes und einer Link-URL aus Alternativenlisten:

  ```
  RewriteMap linkmap rnd:/Pfad/von/linkmap.txt
  ```

  Die zugehörige Datei sieht möglicherweise so aus:

  ```
  # linkmap.txt - zufällige Bild- und Link-URLs
  bild  photo.jpg|grafik.gif|bild.png|beispiel.jpg
  link  /info/index.html|/news/index.html|/catalog/index.html
  ```

  Die folgende `RewriteRule` wandelt einen Aufruf von */images/zbild* in ein zufällig ausgewähltes Bild um:

  ```
  RewriteRule ^/images/zbild /images/${linkmap:bild|default.gif}
  ```

- `dbm`: DBM-Datei anstelle der einfachen Textdatei
  Wenn Sie eine `RewriteMap` mit sehr vielen Zuordnungen benötigen, kann es sich günstig auf die Performance auswirken, statt der einfachen Textdateien datenbankartige DBM-Dateien zu verwenden. Der Inhalt einer solchen Datei ist mit der Textdatei identisch, wird aber binär gespeichert und mit einem Index versehen.

  Je nach Plattform stehen nicht alle Formen von DBM-Dateien (SDBM, GDBM, NDBM usw.) zur Verfügung. Recht empfehlenswert sind SDBM-Dateien: Zwar sind einige andere DBM-Typen noch schneller, aber dafür wird SDBM auf vielen Plattformen unterstützt und ist sowohl in Apache als auch beispielsweise in Perl automatisch eingebaut. Andere DBM-Typen müssen Sie mithilfe der in Kapitel 4, »Apache kompilieren und installieren«, vorgestellten `configure`-Einstellungen wie `--with-gdbm` bei der Kompilierung hinzufügen.

  Es ist wichtig, dass Sie folgende Beschränkung von SDBM-Dateien beachten: Ein Schlüssel und sein Wert dürfen zusammen nicht länger als 1.000 Byte sein. Natürlich ist es unwahrscheinlich, dass Sie in URLs jemals eine solche

Textmenge benötigen – aber bei der per `Rewrite` gesteuerten Erzeugung von Query-Strings könnte es immerhin vorkommen.

Sie können beispielsweise das folgende kleine Skript verwenden, um eine solche Datei aus Benutzereingaben zu erzeugen (es befindet sich zusätzlich auf der beiliegenden DVD-ROM):

```
#!/usr/bin/perl -w
use strict;
use Fcntl;
use SDBM_File;

# Name der MAP-Datei aus Kommandozeilenargument / Standard
my $mapfile = $ARGV[0] || 'map';

# Hash %map mit tie() an $mapfile binden
tie(my %map, 'SDBM_File', $mapfile, O_RDWR|O_CREAT, 0666)
|| die "Kein Zugriff auf $mapfile: $!\n";

# Bisherige Einträge anzeigen
print "Vorhandene Werte in $mapfile:\n";
foreach my $key (keys %map) {
   my $entry = $map{$key};
   print "$key\t$entry\n";
}

# Eingabe neuer Einträge
print "Bitte 'Name Wert'-Paare eingeben.\n"
print "[Einfaches ENTER zum Beenden.]\n";
while (my $line = <>) {
   chomp $line;
   last if ($line eq '');
   if ($line !~ /^[^\s]+\s+[^\s]+/) {
      print "Falsches Format: $line\n";
      next;
   }
   my ($key, $val) = split (/\s+/, $line);
   $map{$key} = $val;
}
```

Wenn Sie das Programm ausführen, werden zunächst die vorhandenen Werte angezeigt. Anschließend können Sie so lange neue eingeben, bis Sie einfach ↵ drücken. In der Ausführung sieht das etwa so aus:

```
$ chmod a+x sdbmedit.pl
$ ./sdbmedit.pl mymap
```

```
Vorhandene Werte in mymap:
logo      /images/common/logo.gif
photo     /images/photos/photo1.jpg
Bitte 'Name Wert'-Paare eingeben.
[Einfaches ENTER zum Beenden.]
head /images/common/headline.gif
bullet /images/common/bullet.gif
logo /images/common/neulogo.gif
```

Wenn Sie den Namen eines bereits vorhandenen Schlüssels eingeben (hier z. B. `logo`), wird dessen bisheriger Wert überschrieben.

Die praktische Funktion `tie()`bindet eine Variable an ein Package beziehungsweise eine Klasse. Im vorliegenden Beispiel führen Änderungen der Variablen dazu, dass auch die SDBM-Datei automatisch geändert wird. Bei `SDBM_File` werden folgende Argumente für `tie()` benötigt:

- Name der Hash-Variablen
- der Klassenname `SDBM_File`
- der Dateiname für die SDBM-Datei. Sie sollten einen Namen ohne Endung verwenden, weil automatisch die beiden Dateien *NAME.pag* und *NAME.dir* erzeugt werden.
- `open`-Konstanten für Lesen, Schreiben usw. Die symbolischen Konstanten werden durch `Fcntl` bereitgestellt. `O_RDWR` öffnet die Datei zum Lesen und Schreiben; `O_CREAT` erzeugt sie neu, falls sie noch nicht existieren sollte.
- Berechtigungen für die Datei. `0666` bedeutet, dass zunächst einmal jeder die Datei lesen und hineinschreiben darf. Diese Berechtigungen werden durch die `umask` des aktuellen Benutzers modifiziert.

Wenn eine Eingabe nicht mindestens ein Zeichen für den Schlüssel, ein Leerzeichen und ein Zeichen für den Wert enthält, wird sie mit einer Fehlermeldung zurückgewiesen; `next()` überspringt den Rest des Schleifenrumpfes und beginnt sofort den nächsten Schleifendurchlauf.

Alternativ steht seit Apache 2.1-beta das neue Hilfsprogramm `httxt2dbm` zur Verfügung, das textbasierte Map-Dateien automatisch in DBM-Dateien umwandelt. Das Syntaxschema dieses Programms sieht wie folgt aus:

```
httxt2dbm [-v] [-f DBM-Typ] -i Eingabedatei -o Ausgabedatei
```

Die Kommandozeilenparameter haben die nachfolgende Bedeutung:

- `-v` (verbose) sorgt für eine ausführlichere Ausgabe.
- `-f` ermöglicht Ihnen die Angabe des DBM-Typs (`GDBM`, `SDBM`, `DB` oder `NDBM`). Wenn Sie keinen Typ angeben, wird der APR-Standardwert gewählt.

- `-i` gibt die Textdatei an, aus der die Eingabedaten stammen sollen. Das Format muss den Angaben entsprechen, die weiter oben für den Map-Typ `txt` gemacht wurden.
- `-o` dient der Angabe eines Namens für die Ausgabedatei.

Das folgende Beispiel wandelt die Datei *mymap.txt* in die SDBM-Datei *mymap* um:

```
# httxt2dbm -f SDBM -i mymap.txt -o mymap
```

Die fertige SDBM-Datei können Sie folgendermaßen als Map-Datei definieren:

```
RewriteMap mymap dbm:/Pfad/von/mymap
```

- `int`: Interne Funktion verwenden

  Statt einer richtigen `map` soll eine der folgenden internen Funktionen verwendet werden:

  - `toupper`: Der bisherige Wert des »Schlüssels« wird komplett in Großbuchstaben konvertiert. `NurEinTest` würde zu `NUREINTEST`.
  - `tolower`: Der Wert wird in Kleinbuchstaben umgewandelt. Aus `NurEinTest` wird in diesem Fall `nureintest`.
  - `escape`: Sonderzeichen im Text werden URL-codiert. `Günther Möller` wird z. B. durch `G%FCnther+M%F6ller` ersetzt.
  - `unescape`: Eventuelle URL-codierte Sonderzeichen werden in ihre normale Form umgewandelt. Das vorige Beispiel könnte in diesem Fall umgekehrt ausgeführt werden.

  Die Einbindung einer solchen Funktion als Map funktioniert z. B. so:

  ```
  RewriteMap klein int:tolower
  ```

- `prg`: eigenes Programm ausführen

  Der letzte Map-Typ ist der komplizierteste, aber auch der vielseitigste: Es wird ein externes Programm aufgerufen, das beliebige Manipulationen am URL-Inhalt durchführen kann. Für ein solches Programm gelten folgende Voraussetzungen:

  - Der zu ändernde Text (der jeweilige »Schlüssel«) wird automatisch über `STDIN` eingegeben. Das Ergebnis muss auf `STDOUT` geschrieben werden. Da das Programm nur einmal beim Start von Apache aktiviert wird, benötigt es eine Eingabeschleife, die die verschiedenen Anforderungen nacheinander entgegennimmt.

- Es darf keine gepufferte Ausgabe stattfinden. Da bei einer solchen Änderung so gut wie nie genügend Text zusammenkommt, um den Ausgabepuffer zu füllen, bleibt das Programm – und damit die Anfrage – hängen. In Perl können Sie für `STDOUT` den Autoflush-Modus aktivieren, um dies zu verhindern. Fügen Sie dazu am Anfang Ihres Skripts folgende Zeilen ein:

```
select (STDOUT);
$| = 1;
```

- Um verschiedene Zugriffe auf Ihr Map-Programm zu serialisieren, müssen Sie mit der gleichnamigen Direktive eine `RewriteLock`-Datei definieren.

Hier ein recht nützliches Beispiel: Das folgende Perl-Skript liefert für den Schlüssel `now` (und als Standardwert für nicht gefundene Schlüssel) das aktuelle Datum im Format `2008-08-26`. `yesterday` bringt das ebenso formatierte Datum des Vortags; Schlüsselnamen wie `minus3` oder `minus7` gehen die entsprechende Anzahl von Tagen zurück. Wenn Sie jeden Tag eine bestimmte Datei erzeugen, in deren Namen für spätere Archivzwecke das aktuelle Datum vorkommt, ist es praktischer, sie über eine solche `Rewrite`-Option zu verlinken, als später umzubenennen. Hier der Code für das Beispiel:

```
#!/usr/bin/perl -w
# datemap.pl - Datum für URL-Umwandlung ermitteln
use strict;
select (STDOUT);
$| = 1;                          # Autoflush-Modus
my $daysec = 24 * 60 * 60;       # 1 Tag in Sekunden
# Monatsnamen aus scalar(localtime) in Nummern umwandeln:
my %months = (Jan => '01', Feb => '02', Mar => '03',
              Apr => '04', May => '05', Jun => '06',
              Jul => '07', Aug => '08', Sep => '09',
              Oct => '10', Nov => '11', Dec => '12');
# "Schlüssel" von STDIN lesen
while (my $key = <STDIN>) {
   chomp $key;
   my $now = time();             # Aktuelles Datum/Uhrzeit
   if ($key eq 'yesterday') {
      $now -= $daysec;           # 1 Tag abziehen
   } elsif ($key =~ /^minus(\d+)/) {
      $now -= $daysec * $1;      # n Tage abziehen
   }
   # Datum umwandeln -
   # liefert z.B. "Thu Aug 28 12:06:22 2008"
   my $timestr = scalar (localtime ($now));
   # Ergebnis in Einzelteile zerlegen
```

```
    my ($wd, $month, $day, $t, $year) =
         split (/\s+/, $timestr);
    # Monatsnummer ermitteln
    my $monthnr = $months{$month};
    # '0' vor dem Tagesdatum, falls < 10
    $day = ($day < 10) ? "0$day" : $day;
    # Komplettes Datum zusammensetzen - z.B. 2008-08-19
    my $output = "${year}-${monthnr}-${day}";
    # Ausgabe auf STDOUT
    print $output;
}
```

Verwenden können Sie dieses Skript z. B. folgendermaßen:

```
RewriteMap dates prg:/Pfad/von/datemap.pl
# Zugriffe auf /news/now in /news/JJJJ-MM-DD umwandeln
RewriteRule ^/news/now/(.*) /news/${dates:now}/$1
# Zugriffe auf /news/yesterday umwandeln
RewriteRule ^/news/yesterday/(.*) /news/${dates:yesterday}/$1
```

Noch effizienter wird es, wenn der Teil des Verzeichnispfades hinter `/news/` direkt an die Map durchgereicht wird:

```
RewriteRule ^/news/([^/]+)/(.*) /news/${dates:$1}/$2
```

Unter Windows müssen Sie eine wichtige Änderung durchführen, damit ein solches Map-Skript ausgeführt werden kann: Da ein Perl-Skript formal kein ausführbares Programm ist, können Sie als Quelle Ihres Programms nicht einfach so etwas wie *C:/Verzeichnis/skript.pl* verwenden, sondern müssen den Pfad des Perl-Interpreters angeben.

Hier sieht die `RewriteMap`-Anweisung z. B. so aus:

```
RewriteMap dates \
"prg:C:/Perl/bin/perl.exe C:/Apache2/conf/datemap.pl"
```

Aber auch mit dieser Option ist auf Windows-Systemen von Map-Programmen abzuraten. Besonders die Inkompatibilität mit Zeilenumbrüchen sorgt dafür, dass sie auf dieser Plattform oft hängen bleiben.

#### RewriteLock

Lock-Datei zur `RewriteMap`-Synchronisation

| Version | 1.3 bis 2.2 |
|---|---|
| Modul | `mod_rewrite` |
| Kontext | Server |

| Syntax | RewriteLock Dateipfad |
|---|---|
| Standardwert | nicht gesetzt |

Wenn Sie selbst geschriebene Programme als RewriteMap verwenden, benötigen Sie eine mithilfe dieser Direktive definierte Lock-Datei. Diese sorgt dafür, dass nicht mehrere Anfragen gleichzeitig auf das Programm zugreifen. Beispiel:

```
RewriteLock /var/run/apache2/rewrite.lck
```

Ab Apache 2.3.4 wird der allgemeine Wert der Direktive Mutex (siehe Kapitel 6, »Grundkonfiguration«) für das Rewrite-Locking verwendet.

#### RewriteOptions

Zusätzliche Rewrite-Optionen

| Seit Version | 1.3; MaxRedirects seit 2.0.45 und nicht mehr ab 2.1-beta |
|---|---|
| Modul | mod_rewrite |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | RewriteOptions Option [Option ...] |
| Standardwert | MaxRedirects=10 |

Diese Direktive ermöglicht es, zusätzliche Rewrite-Optionen zu setzen. Folgende zwei Optionen sind möglich:

- inherit
  Sämtliche Rewrite-Einstellungen wie RewriteRule, RewriteMap usw. werden aus dem übergeordneten Kontext übernommen.
- MaxRedirects=Anzahl
  Beschränkt die Verarbeitung auf eine maximale Anzahl interner Weiterleitungen, um der Gefahr einer Endlosschleife zu entgehen. Wenn die angegebene Anzahl überschritten wird, wird eine Meldung mit dem Status 500 Internal Script Error geliefert. Die Voreinstellung ist 10; in den allermeisten Fällen sollte sie ausreichen. Seit Apache 2.2 ist diese Option nicht mehr verfügbar.

#### Rewrite-Beispiele

Wie Sie gesehen haben, wurden in den Beschreibungen der einzelnen Rewrite-Direktiven noch keine ausführlichen Beispiele gezeigt. Da die Direktiven äußerst komplex sind und vielfältige Optionen besitzen, war es ratsam, sie zunächst im Überblick zu beschreiben.

Dafür finden Sie hier eine kleine Sammlung von Beispielen. In den späteren Kapiteln dieses Buches wird noch des Öfteren auf `mod_rewrite` zurückgegriffen, da es praktische Lösungen für zahlreiche Situationen bietet. Ein etwas umfangreicheres »`Rewrite`-Kochbuch« finden Sie in der Apache-Online-Konfiguration: Der »URL Rewriting Guide« von Ralf S. Engelschall, Autor des Moduls, befindet sich unter der URL *http://httpd.apache.org/docs2.2/misc/rewriteguide.html*.

- **Einfacher Alias**
  Selbstverständlich beherrscht `mod_rewrite` alle Fähigkeiten von `Alias`. Das folgende Beispiel bindet das Verzeichnis */usr/local/apache2/mydocs* unter dem URL-Pfad */text* ein:

  ```
  RewriteEngine On
  RewriteRule ^/text(.*) /usr/local/apache2/mydocs$1
  ```

  Auch die erweiterten Funktionen von `AliasMatch` sind natürlich in `RewriteRule` enthalten – schließlich ist der Vergleichstext ohnehin ein regulärer Ausdruck. Dieses Beispiel leitet alle Anfragen nach Dokumenten mit der Endung *.htm* oder *.html* auf entsprechende *.php*-Dateien um (eine nette Spielerei, die die verwendete Technologie für Server-Anwendungen zumindest vordergründig versteckt):

  ```
  RewriteEngine On
  RewriteRule ^(.*)\.html?$ $1.php
  ```

- **Einfache Weiterleitung**
  Auch Weiterleitungen im Stil von `Redirect` und `RedirectMatch` sind mit `RewriteRule` natürlich überhaupt kein Problem. Dazu muss die Direktive mit dem Flag `[R]` aufgerufen werden. Das folgende Beispiel leitet *alle* Anfragen auf die Site *www.ersatz-server.de* weiter; Sie könnten so etwas brauchen, wenn Sie unaufschiebbare Wartungsarbeiten an Ihrem Server durchführen müssen:

  ```
  RewriteEngine On
  RewriteRule .* http://www.ersatz-server.de [R]
  ```

  Das nächste Beispiel leitet dagegen Anfragen, die mit dem Verzeichnis `/sales` beginnen, an den Server `sales.mynet.de` weiter – so etwas ist ideal, wenn ein ehemaliges Unterverzeichnis auf einen virtuellen Host oder gar einen eigenen Server-Rechner ausgelagert wird:

  ```
  RewriteEngine On
  RewriteRule ^/sales(.*) http://sales.mynet.de$1 [R]
  ```

- **Serverseitige Browser-Weiche**
  Wenn Sie schon einmal in JavaScript programmiert haben, kennen Sie sicherlich die berüchtigten Browser-Weichen, die je nach Browser-Modell und -Version unterschiedliche Teile eines Skripts ausführen. Wenn Sie für bestimmte Browser völlig separate Seiten anbieten möchten, können Sie das Ganze auch serverseitig erledigen. Beispielsweise werden mit den beiden folgenden Konfigurationsanweisungen sämtliche Anfragen von Usern, die den Internet Explorer verwenden, mit ansonsten gleichen Pfad- und Dateiangaben in das Unterverzeichnis */msie* umgeleitet:

  ```
  RewriteEngine On
  RewriteCond %{HTTP_USER_AGENT} msie [NC]
  RewriteRule ^/(.*) /msie/$1
  ```

  Die Server-Variable `HTTP_USER_AGENT`, die den Wert des Anfrage-Headers `User-Agent` enthält, wird mit dem regulären Ausdruck `msie` verglichen – vorsichtshalber mit dem Flag `[NC]`, damit die Groß- und Kleinschreibung nicht beachtet wird.

- **Happy Hour**
  Angenommen, ein E-Commerce-Shop möchte seinen Kunden je nach Uhrzeit spezielle Sonderangebote machen, sofern diese sofort bestellt werden. Die folgende Kombination aus `RewriteCond` und `RewriteRule` leitet Anfragen für das Dokument */angebot.html* auf */happyhour.html* um, wenn die Stunde den Wert 19 hat (wenn es also zwischen 19:00 und 19:59 Uhr ist):

  ```
  RewriteEngine On
  RewriteCond %{TIME_HOUR} =19
  RewriteRule ^/angebot.html$ /happyhour.html
  ```

  Zu allen anderen Uhrzeiten wird die `RewriteRule`-Direktive übergangen, weil die Bedingung in der `RewriteCond`-Anweisung nicht zutrifft.

- **»Microsoft-Free Friday«**
  Unter zahlreichen Adressen im Web können Sie ein Apache-Modul herunterladen, das freitags alle User aussperrt, die den Microsoft Internet Explorer verwenden (suchen Sie mit einer Suchmaschine Ihrer Wahl nach »Microsoft-Free Friday«). Das ist zwar ein nettes Beispiel für die Modulprogrammierung, aber eigentlich überhaupt nicht nötig – mit zwei `RewriteCond`- und einer `RewriteRule`-Direktive lässt sich nämlich dasselbe erreichen:

  ```
  RewriteEngine On
  RewriteCond %{TIME_WDAY} =5
  RewriteCond %{HTTP_USER_AGENT} msie [NC]
  RewriteRule .* /no-ms.html
  ```

- **Session-Tracking**
  Sicher haben Sie schon einmal eine URL wie diese gesehen: *http://www.mynet.de/6FC9387432BA02E4/info.html*. Die lange Hexadezimalzahl dürfte hier in aller Regel kein echtes Verzeichnis sein, sondern eine Session-ID. Sie dient dazu, aufeinanderfolgende Anfragen desselben Users verfolgen zu können – dies ist z. B. wichtig, um Bestellinformationen über mehrere Seiten hinweg gespeichert zu halten. Es ist praktischer, Session-IDs in den URL-Pfad zu integrieren, als sie im Query-String zu transportieren. Auch das Logging jeder einzelnen Session wird auf diese Weise erleichtert.

  Die folgende Lösung generiert beim ersten Zugriff auf eine beliebige URL der Website eine neue Session-ID und leitet anschließend jede Anfrage mit Session-ID auf das eigentliche Dokument um, das diese ID natürlich nicht hat.

  Das erste Hilfsmittel ist eine `RewriteMap` in Form eines kleinen Perl-Skripts, das auf Anfrage eine 16-stellige hexadezimale Zufallszahl produziert. Dieses Skript heißt *session.pl* und sieht so aus:

```
#!/usr/bin/perl -w
use strict;
# Autoflush
select (STDOUT);
$| = 1;
# Liste mit Hex-Ziffern für die Session-ID
my @hexnums = qw (0 1 2 3 4 5 6 7 8 9 A B C D E F);
# Hauptschleife
while (<STDIN>) {
   # Der Wert des Schlüssels ist egal -
   # es wird auf jeden Fall eine Session-ID erzeugt
   my $id = '';
   for (my $i = 0; $i < 16; $i++) {
      $id .= $hexnums [int (rand (16))];
   }
   # Neue Session-ID ausgeben
   print "$id\n";
}
```

  Dieses Map-Skript muss nun als `RewriteMap` registriert werden. Dies funktioniert folgendermaßen:

```
RewriteMap session prg:/usr/local/apache2/mytools/session.pl
```

  Den Pfad müssen Sie natürlich anpassen. Bei der Beschreibung von `RewriteMap` wurde bereits erwähnt, dass unter Windows auch der Pfad des Interpreters angegeben werden muss, beispielsweise so:

```
RewriteMap session "prg:C:/Perl/bin/perl.exe \
C:/Apache2/mytools/session.pl"
```

Allerdings ist die Wahrscheinlichkeit dennoch groß, dass es auf Windows-Systemen trotz alledem nicht funktioniert.

Als Nächstes müssen Sie dafür sorgen, dass alle Anfrage-URLs, die noch keine Session-ID enthalten, sich eine solche aus dieser `RewriteMap` besorgen. Dazu dient folgende `RewriteRule`:

```
RewriteRule !^/[A-Z0-9]{16}/(.*) /${session:id}/$1 [L]
```

Zu guter Letzt müssen alle Anfragen mit Session-ID auf die entsprechende Ressource *ohne* ID umgelenkt werden:

```
RewriteRule ^/[A-Z0-9]{16}/(.*) /$1
```

### 8.1.3 Benutzerverzeichnisse veröffentlichen

Auf Multiuser-Systemen können Sie es den einzelnen Benutzern ermöglichen, ihre privaten Webseiten unter einer URL nach dem Schema *http://hostname/~username* zu veröffentlichen. Dies ist vor allen Dingen bei Universitätsrechenzentren der Fall; Millionen von Studenten auf der ganzen Welt betreiben Websites mit einer solchen Adresse. Apache stellt diese Funktionalität über das Modul `mod_userdir` bereit. Es sorgt dafür, dass ein bestimmtes Verzeichnis im Home-Verzeichnis des jeweiligen Benutzers automatisch auf diese URL abgebildet wird. Der Name dieses Verzeichnisses wird über die Direktive `UserDir` festgelegt.

Beachten Sie, dass der Begriff »Home-Verzeichnis« nicht auf jeder Plattform so eindeutig definiert ist wie unter UNIX. Im Einzelnen ist damit Folgendes gemeint:

- Unter allen UNIX-Systemen außer Mac OS X befindet sich das Home-Verzeichnis eines normalen Benutzers unter */home/username*. Das Home-Verzeichnis von `root` – das in aller Regel nicht ins Web gehört – ist dagegen normalerweise */root*.
- Mac OS X verwendet statt */home* das Verzeichnis */Users*; das Home-Verzeichnis eines Benutzers ist also */Users/username*.
- Unter Windows 7 und Vista liegen die Benutzerverzeichnisse unter *%HOMEDRIVE%\Users\username* (die Umgebungsvariable `HOMEDRIVE` steht für den Laufwerksbuchstaben der Festplatte, auf der sich Windows befindet). In der deutschen Version wird das Verzeichnis *Users* durch den Aliasnamen *Benutzer* maskiert. Die persönlichen Dokumente eines Benutzers liegen dort im Unterverzeichnis *Documents*. Deshalb ist es ratsam, als `UserDir` ein Verzeichnis unterhalb davon anzugeben, also beispielsweise *Documents/public_html*.

- Bei einer deutschen Version von Windows 2000 und XP befinden sich die Verzeichnisse mit den Daten der einzelnen Benutzer unter *%HOMEDRIVE%\Dokumente und Einstellungen\username*; die Dokumente eines Benutzers finden Sie im Unterverzeichnis *Eigene Dateien*. Ein guter Platz für das `UserDir` wäre also etwa *Eigene Dateien/public_html*.

  Für eine englischsprachige Windows-Version gelten diese Angaben sinngemäß, allerdings lautet der vollständige Pfadname des persönlichen Verzeichnisses (»Eigene Dateien«) hier *%HOMEDRIVE%\Documents and Settings\username\My Documents*.
- Unter Windows NT 4.0 befanden sich die persönlichen Dokumente ebenfalls im Verzeichnis *Eigene Dateien* beziehungsweise *My Documents*, allerdings war das eigentliche Benutzerverzeichnis hier *%SystemRoot%\Profiles\username* (`SystemRoot` ist übrigens in allen Windows-Versionen das Systemverzeichnis, z. B. *C:\WinNT*).

### UserDir

Verzeichnisname der User-Webverzeichnisse

| Modul | `mod_userdir` |
|---|---|
| Kontext | Server, `<VirtualHost>` |
| Syntax | `UserDir Verzeichnis \|`<br>`enabled [user ...] \| disabled [user ...]` |
| Standardwert | `public_html`<br>(seit 2.1.4 standardmäßig deaktiviert) |

Je nach Plattform sind unterschiedliche Verzeichnisnamen ratsam. Gemäß den obigen Ausführungen über die Home-Verzeichnisse auf den unterschiedlichen Plattformen können Sie unter UNIX beispielsweise Folgendes einstellen:

```
UserDir mysite
```

Damit wird z. B. das Verzeichnis */home/winnetou/mysite* als *http://www.mynet.de/~winnetou* veröffentlicht.

Auf einem Windows-XP-Rechner sollten Sie dagegen eine Einstellung wie diese vornehmen:

```
UserDir "Eigene Dateien/mysite"
```

Auf diese Weise würde das Verzeichnis *C:\Dokumente und Einstellungen\winnetou\Eigene Dateien\mysite* als *http://www.mynet.de/~winnetou* bereitgestellt.

Um den Zugriff auf die User-Verzeichnisse zu steuern, benötigen Sie anschließend einen Konfigurationsabschnitt wie diesen:

```
<Directory /home/*/mysite>
  Options Indexes SymLinksIfOwnerMatch
  AllowOverride AuthConfig Limit FileInfo
  <LimitExcept GET POST>
    Order Deny,Allow
    Deny from all
  </LimitExcept>
  <Limit GET POST>
    Order Allow,Deny
    Allow from all
  </Limit>
</Directory>
```

Dies beschränkt die Optionen und `AllowOverride`-Einstellungen auf ein relativ sicheres Minimum. Der `<LimitExcept>`-Container sperrt sämtliche Zugriffe durch HTTP-Methoden außer `GET` und `POST`, während der `<Limit>`-Abschnitt umgekehrt alle `GET`- und `POST`-Anfragen gestattet.

Die Verzeichnisangabe im `<Directory>`-Container muss natürlich je nach Plattform angepasst werden:

- Mac OS X:

```
<Directory /Users/*/mysite>
  ...
</Directory>
```

- Windows 7/Vista:

```
<Directory "C:/Users/*/Documents/mysite">
  ...
</Directory>
```

- Windows 2000/XP:

```
<Directory "C:/Dokumente und Einstellungen/*/Eigene
Dateien/mysite">
  ...
</Directory>
```

- Windows NT 4.0:

```
<Directory "C:/WinNT/Profiles/*/Eigene Dateien/mysite">
...
</Directory>
```

Um gezielt zu steuern, welche Benutzer ein Webverzeichnis veröffentlichen dürfen, können Sie die speziellen Werte `enabled` und/oder `disabled` verwenden, optional gefolgt von einem oder mehreren Benutzernamen.

- `disabled` ohne Angabe von Benutzernamen deaktiviert das Feature ganz:

  ```
  UserDir disabled
  ```

- `disabled` ohne Benutzernamen und ein anschließendes `enabled` mit einigen Namen erlaubt nur den ausdrücklich angegebenen Benutzern die Veröffentlichung ihres Webverzeichnisses:

  ```
  UserDir disabled
  UserDir enabled hans klaus peter
  ```

- Die umgekehrte Abfolge erlaubt die `UserDir`-Funktionalität im Allgemeinen und verbietet sie speziellen Benutzern:

  ```
  UserDir enabled
  UserDir disabled dieter thomas udo
  ```

### 8.1.4 Fehlerbehandlung

Eine spezielle Form der Weiterleitung stellt der Umgang mit diversen Fehlern dar, die bei der Verarbeitung von Client-Anfragen auftreten können: Schließlich meldet Apache im Fehlerfall nicht nur den entsprechenden Statuscode (z. B. `404 Not Found`), sondern liefert auch ein Dokument aus, das den Fehler näher beschreibt. Welches Dokument dies für den jeweiligen Fehler sein soll, wird durch die Core-Direktive `ErrorDocument` festgelegt.

#### ErrorDocument

Dokument, das bei einem bestimmten Fehler gesendet werden soll

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `ErrorDocument Fehlercode URL\|Pfad\|Meldungstext` |
| **Standardwert** | Einfache Textmeldung für jeden Fehler |

Mit dieser Direktive können Sie einstellen, welche Fehlermeldung Apache für einen bestimmten Fehlercode an den Client senden soll. Dies betrifft sowohl Anfragefehler (`4xx`-Statuscodes) als auch Server-Fehler (`5xx`-Statuscodes). Wenn Sie `ErrorDocument` für einen bestimmten Fehler nicht setzen, wird eine vorgegebene Textmeldung gesendet.

Sie können für jeden Fehler eine von drei möglichen Meldungsquellen festlegen:

- **Eigene Textmeldung**
  Geben Sie dazu einen beliebigen String in Anführungszeichen an (die alte 1.3-Syntax, bei der die Meldung lediglich mit einem Anführungszeichen *beginnen* musste, gilt nicht mehr). Beispiel:

  ```
  ErrorDocument 404 "Nae, esu jet hammer aevver nit!"[3]
  ```

- **Interne Weiterleitung**
  Der Wert ist ein beliebiger URL-Pfad, der relativ zur `DocumentRoot` ausgewertet wird. Beispiel:

  ```
  ErrorDocument 500 /cgi-bin/debug.pl
  ```

  Dies ist die beliebteste Verwendung von `ErrorDocument` – mit ihrer Hilfe können Sie die Fehlermeldungen dem Layout Ihrer Website anpassen. Beachten Sie in diesem Zusammenhang, dass der Microsoft Internet Explorer solche Dokumente nur dann anzeigt (und nicht durch seine eigenen Fehlermeldungen ersetzt), wenn sie eine bestimmte Mindestlänge haben – in der Regel liegt diese bei 512 Byte.

- **Externe Weiterleitung**
  Als Wert wird eine externe URL angegeben, die mit *http://* beginnt. Beachten Sie, dass Sie diesen Typ nicht für den Status `401 Authorization Required` verwenden können, weil Browser über Domain-Grenzen hinweg keine Anmeldeinformationen anfordern. Beispiel:

  ```
  ErrorDocument 403 http://im.parkverbot.de
  ```

Das Dokument für die interne Weiterleitung kann eine Type-Map sein (siehe Kapitel 7, »Header und MIME-Einstellungen«). Dies ermöglicht internationalisierte Fehlermeldungsseiten. Entsprechende Dokumente, bei denen die Body-Abschnitte für die unterschiedlichen Sprachen bereits enthalten sind, werden mit Apache 2 geliefert. Sie befinden sich im Verzeichnis `error` unter der `ServerRoot` und heißen `HTTP_<Fehler>.html.var`, also beispielsweise `HTTP_NOT_FOUND.html.var` für Fehler 404. Die Standard-Konfigurationsdatei enthält bereits einige Zeilen, um sie zu aktivieren – bei Bedarf können Sie einfach das Kommentarzeichen # entfernen.

### 8.1.5 Rechtschreibkorrektur in URLs mit mod_speling

Eine etwas abenteuerliche Form der automatischen Weiterleitung steht Ihnen zur Verfügung, wenn Sie das Modul `mod_speling` (der »Fehler« im Namen ist Absicht) aktivieren: Wenn unter einer angeforderten URL kein Dokument zu finden ist,

3 Hochdeutsch: »Nein, so etwas haben wir aber nicht!«

versucht es, ein Dokument mit einem ähnlichen Namen zu liefern und auf diese Weise Rechtschreibfehler in URLs zu korrigieren.

Das Modul ignoriert in einem ersten Schritt die Groß- und Kleinschreibung der angeforderten URL und akzeptiert im zweiten Schritt alle Varianten, die höchstens einen weiteren Fehler aus folgenden Kategorien enthalten:

- **Zusätzliches Zeichen**
  Angefordert wurde *info.html*; vorhanden ist nur *infos.html*.
- **Fehlendes Zeichen**
  Die angeforderte URL war *books.html*, aber nur *book.html* ist vorhanden.
- **Falsches Zeichen**
  Die Anfrage enthielt den Dateinamen *test.html*; lieferbar ist *text.html*.
- **Dreher**
  Der Client fragte irrtümlich nach *speil.html*; der Server kann *spiel.html* liefern.
- **Andere Dateiendungen**
  Der Client fordert *index.html* an; gefunden wird *index.php*.

Wenn nach einer entsprechenden Überprüfung des Verzeichnisses keine Variante gefunden wird, erhält der Client genau wie im Fall ohne `mod_speling` die Antwort `404 Not Found`. Findet Apache genau ein passendes Dokument, liefert er es ohne weitere Nachfrage aus. Bei mehreren gleichwertigen Treffern wird dagegen eine Liste mit dem Status `300 Multiple Choices` geliefert.

### CheckSpelling

Korrektur von URL-Schreibfehlern aktivieren

| | |
|---|---|
| **Seit Version** | 1.3 (vor 1.3.2 nur im Server- und VHost-Kontext) |
| **Modul** | `mod_speling` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`Options`) |
| **Syntax** | `CheckSpelling On\|Off` |
| **Standardwert** | `Off` |

Die Funktionalität von `mod_speling` wird im entsprechenden Kontext und in seinen Unterkontexten nur dann aktiviert, wenn Sie sie mit dieser Direktive explizit einschalten:

```
CheckSpelling On
```

Bei Bedarf können Sie sie in einem untergeordneten Kontext wieder ausschalten, indem Sie dort folgende Anweisung einsetzen:

```
CheckSpelling Off
```

**CheckCaseOnly**

Nur Änderung der Groß- und Kleinschreibung

| Seit Version | 2.2.0 |
|---|---|
| Modul | mod_speling |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Options) |
| Syntax | CheckCaseOnly On\|Off |
| Standardwert | Off |

Wenn Sie diese Direktive zusätzlich zu `CheckSpelling On` ebenfalls auf `On` setzen, wird die Funktionalität von `mod_speling` eingeschränkt: Es werden nur noch Änderungen in der Groß- und Kleinschreibung vorgenommen; alle anderen Optionen werden ausgeschaltet.

### 8.1.6 Status- und Konfigurationsinformationen über den Server

Apache 2 wird mit zwei sehr praktischen Modulen geliefert, die im weitesten Sinne auch mit Weiterleitung zu tun haben, da ähnlich wie bei `Alias`-Direktiven externe Daten in den URL-Bereich des Servers eingebunden werden: `mod_info` liefert die Konfigurationsdaten des laufenden Apache-Servers als HTML-Dokument; `mod_status` stellt dagegen Statusinformationen zur Verfügung. Beide sind ideal zur (Fern-)Diagnose des Servers geeignet. Mit den Informationen von `mod_info` können Sie beispielsweise schnell überprüfen, ob irgendein Server-Problem mit der Konfiguration zu tun hat. `mod_status` erlaubt Ihnen z. B. die Analyse von Performance-Engpässen.

Wenn Sie die Informationen von `mod_info` einbinden möchten, müssen Sie dafür eine `<Location>` verwenden, die keinem existierenden Verzeichnis entspricht – weder unterhalb der `DocumentRoot` noch per `Alias` eingebunden. Damit unter dieser URL Server-Informationen geliefert werden, muss der Handler `server-info` gesetzt werden. Abgesehen davon sollte die URL vor Zugriffen durch Unbefugte geschützt werden. Zusätzlich zu der hier gezeigten Beschränkung auf das lokale Netzwerk sollten Sie auch noch eine Benutzeranmeldung verlangen (siehe Kapitel 9, »Authentifizierung«). Das folgende Beispiel stellt die Konfigurationsinformationen unter dem URL-Pfad */_info* zur Verfügung:

```
<Location /_info>
  SetHandler server-info
  Order deny,allow
  Deny from all
  Allow from 192.168.0
```

```
  # Hier Benutzeranmeldung hinzufügen!
</Location>
```

Als berechtigter User können Sie nun einen beliebigen Browser öffnen und erhalten die Konfigurationsdaten unter der URL *http://www.mynet.de/_info*.

Seit Version 2.1-beta können Sie einen Query-String an die URL anhängen, um einen der folgenden Befehle zur Steuerung der Ausgabe auszuwählen:

- `?Modulname` – gibt nur Informationen über das angegebene Modul aus.
- `?config` – gibt die Konfigurationsinformationen allein, nicht nach Modulen sortiert, aus.
- `?hooks` – zeigt nur die Hooks an, mit denen die einzelnen Module verknüpft sind (Näheres über Hooks erfahren Sie in Kapitel 17, »Apache erweitern«).
- `?list` – gibt nur eine einfache Liste der aktivierten Module aus.
- `?server` – zeigt nur die Grundinformationen über den Server an.

Die Funktionalität des Moduls `mod_status` wird auf ähnliche Weise eingeschaltet; hier ist lediglich der Handler `server-status` zuständig. Hier ein Beispiel, das die Statusinformationen unter dem URL-Pfad */_status* (im Browser also unter einer URL wie *http://www.mynet.de/_status*) zugänglich macht:

```
<Location /_status>
   SetHandler server-status
   Order deny,allow
   Deny from all
   Allow from 192.168.0
   # Hier Benutzeranmeldung hinzufügen!
</Location>
```

Der optionale Query-String-Parameter `?refresh=N` aktualisiert die Seite alle `N` Sekunden. Sie können z. B. die URL *http://www.mynet.de/_status?refresh=10* angeben, um die Informationen alle zehn Sekunden neu zu laden.

#### AddModuleInfo

Zusätzliche HTML-Information über ein Modul in der Server-Info

| | |
|---|---|
| **Seit Version** | 1.3 |
| **Modul** | `mod_info` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `AddModuleInfo Modulname String` |
| **Standardwert** | nicht gesetzt |

Dies ist die einzige Konfigurationsdirektive des Moduls `mod_info`; seine Grundfunktion stellt es wie erwähnt durch den Handler `server_info` bereit. Die Direktive `AddModuleInfo` gibt den als zweiten Parameter angegebenen HTML-String aus, wenn das entsprechende Modul (erstes Argument) aktiv ist. Den Modulnamen müssen Sie genau wie bei `<IfModule>`-Containern (siehe Kapitel 6, »Grundkonfiguration«) in Form der Quelltextdatei des Moduls angeben, da nur dieser bei fest einkompilierten Modulen und bei DSO-Modulen identisch ist. Das folgende Beispiel verweist auf die Online-Dokumentation von `mod_rewrite`, wenn dieses Modul vorhanden ist:

```
AddModuleInfo mod_rewrite.c \
'N&auml;heres zu mod_rewrite <a \
href="http://httpd.apache.org/docs-2.2/mod/mod_rewrite.html">\
hier</a>'
```

#### ExtendedStatus

Statusinformationen für jede einzelne Anfrage

| | |
|---|---|
| **Seit Version** | 1.3.2 |
| **Modul** | `mod_status`; ab 2.3 `core` |
| **Kontext** | Server |
| **Syntax** | `ExtendedStatus On\|Off` |
| **Standardwert** | `Off` |

Wenn Sie die Direktive `ExtendedStatus` explizit auf `On` setzen, erhalten Sie nicht nur eine Statistik über alle Anfragen, sondern zusätzlich auch Informationen über jede einzelne Anfrage. Beachten Sie, dass dies die Performance Ihres Webservers beeinträchtigen kann; Sie sollten diese Option nur vorübergehend bei aktuellem Debug-Bedarf einschalten.

#### SeeRequestTail

Das Ende statt des Beginns jeder Anfrage anzeigen

| | |
|---|---|
| **Seit Version** | 2.2.7 |
| **Modul** | `mod_status`; ab 2.3 `core` |
| **Kontext** | Server |
| **Syntax** | `SeeRequestTail On\|Off` |
| **Standardwert** | `Off` |

Wenn Sie `ExtendedStatus` einschalten, werden standardmäßig die ersten 63 Zeichen jeder Anfrage gespeichert und angezeigt. `SeeRequestTail On` sorgt dafür, dass stattdessen die letzten 63 Zeichen gewählt werden.

Angenommen, eine Anfrage lautet wie folgt:

```
GET /data/media/images/photographs/fruit-and-vegetables/papaya.png
HTTP/1.1
```

Dann würde standardmäßig der folgende Teilstring angezeigt:

```
GET /data/media/images/photographs/fruit-and-vegetables/p
```

`SeeRequestTail On` würde dagegen folgenden Teil der Anfrage verwenden:

```
images/photographs/fruit-and-vegetables/papaya.png HTTP/1.1
```

## 8.2 Indizes

Bereits in Kapitel 6, »Grundkonfiguration«, wurde die Direktive `DirectoryIndex` vorgestellt. Diese legt die Namen von Dateien fest, die Apache automatisch ausliefern soll, wenn statt einer Datei ein Verzeichnis angefordert wird. Wenn kein solcher Index vorhanden ist, kann Apache ihn selbst on-the-fly erzeugen. Dieser Index wird vom Modul `mod_autoindex` bereitgestellt. Er ist ein HTML-Dokument mit Hyperlinks auf alle Dateien im aktuellen Verzeichnis (Einschränkungen sind möglich). Sie können das Aussehen dieser Indexseiten durch die zahlreichen Direktiven anpassen, die in diesem Abschnitt vorgestellt werden. Abbildung 8.1 zeigt zunächst einen klassischen Index, in dem nur die verlinkten Dateinamen angezeigt werden.

**Abbildung 8.1** Ein einfacher, durch »mod_autoindex« generierter Verzeichnisindex

Mit den Indizes verwandt sind die serverseitigen Image Maps. Sie werden durch das Modul `mod_imagemap` (bis Version 2.0 `mod_imap`) zur Verfügung gestellt und in Abschnitt 8.2.2, »Serverseitige Image Maps mit mod_imagemap«, behandelt.

### 8.2.1 mod_autoindex

Wie bereits erwähnt, besteht die Aufgabe des Apache-Standardmoduls `mod_autoindex` darin, automatisch Indexseiten für Verzeichnisse zu erstellen, die keine Indexdatei enthalten. Dieser automatisch erzeugte Index ist ein HTML-Dokument, in dem die Namen sämtlicher Dateien und Unterverzeichnisse als Hyperlinks aufgelistet werden. Sie können sich zwischen einer einfachen Liste, die nur die Dateinamen enthält, und einem sogenannten *Fancy-Index* mit Icons, Beschreibungen und weiteren Informationen entscheiden. Letzterer kann durch zahlreiche Optionen individuell angepasst werden.

Damit Apache überhaupt Indizes erzeugt, muss das Modul `mod_autoindex` aktiviert werden; außerdem muss die Option `Indexes` gesetzt sein. Dies geschieht in einem beliebigen Unterkontext beispielsweise so:

```
Options +Indexes
```

Näheres über die Direktive `Options` steht in Kapitel 6, »Grundkonfiguration«. Es empfiehlt sich übrigens aus Sicherheitsgründen, die Option nicht für den gesamten Server zu aktivieren. Es gibt viele Verzeichnisse, bei denen es nicht ratsam ist, Besuchern deren Inhalt zu zeigen. Durch `Options None` als Voreinstellung für alle Verzeichnisse, wie in Kapitel 6, »Grundkonfiguration«, empfohlen, lässt sich dies leicht bewerkstelligen.

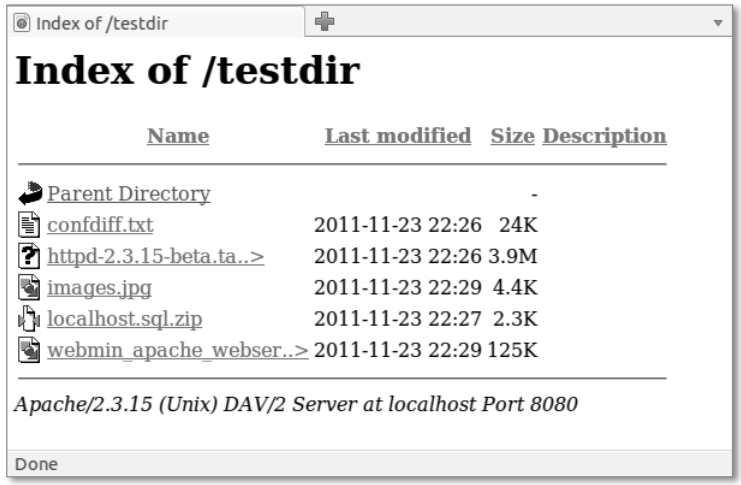



**Abbildung 8.2** Ein ausführlicher Index mithilfe der »FancyIndexing«-Funktion

Neben den einfachen Textlisten beherrscht Apache 2 auch die Option `FancyIndexing` – es handelt sich um ausführliche Indizes mit Icons und Zusatzinformationen über die Dateien. Abbildung 8.2 zeigt ein einfaches Beispiel. Die

nachfolgenden Informationen über FancyIndexing lassen sich leichter nachvollziehen, wenn Sie wissen, wie dieser Index in HTML umgesetzt wird: Standardmäßig steht sein Inhalt in einem <pre>...</pre>-Block; die Aufteilung in Spalten erfolgt also durch vorformatierten Text. Deshalb wird die Breite jeder Spalte in Zeichen statt in Pixeln angegeben. Die in Apache 2.0.23 neu eingeführte Option HTMLTable ändert dies: Sie konstruiert eine einfache HTML-Tabelle zur Darstellung der Spalten.

#### IndexOptions

Optionen für den Index

| | |
|---|---|
| **Modul** | mod_autoindex |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| **Syntax** | IndexOptions [+\|-]Option [[+\|-]Option ...] |
| **Standardwert** | nicht gesetzt |

Mithilfe dieser Direktive wird zunächst einmal das grundlegende Aussehen des Indexes eingestellt. Wenn Sie eine Option ohne eines der Vorzeichen + oder - verwenden, werden *alle* Optionen aus dem übergeordneten Kontext ersetzt. Deshalb sollten Sie Optionen in einem Unterkontext üblicherweise mit +Option hinzufügen oder mit -Option entfernen. Das Verfahren entspricht den in Kapitel 6, »Grundkonfiguration«, beschriebenen Direktiven Options und AllowOverride.

Sie können jede der folgenden Optionen miteinander kombinieren:

- FancyIndexing: Diese Option aktiviert grundsätzlich den Fancy-Index. Wenn Sie -FancyIndexing schreiben, wird dagegen der einfache Index verwendet.
- XHTML: seit Version 2.0.49. Statt des üblicherweise verwendeten HTML 3.2 wird XHTML 1.0 erzeugt.
- Charset=Zeichensatz: seit Version 2.0.61. Gibt den Zeichensatz der erzeugten Indexseite an – z. B. UTF-8 oder ISO-8859-1. Einer dieser beiden Werte ist übrigens je nach zugrunde liegendem Dateisystem Standard.
- Type=MIME-Type: seit Version 2.0.61. Hiermit können Sie den MIME-Type der generierten Seite wählen. Standard: text/html.
- IgnoreCase: Bei der Sortierung nach Dateinamen wird die Groß- und Kleinschreibung nicht beachtet, wenn Sie diese Option angeben. Angenommen, Sie haben Dateien mit den folgenden Namen: A-Datei, a-Datei, b-Datei und C-Datei. Wenn die Option IgnoreCase gesetzt ist, werden die Dateien genau in dieser Reihenfolge aufgelistet. Andernfalls erhalten Sie eine Sortierung in Zeichensatzreihenfolge: A-Datei, C-Datei, a-Datei, b-Datei.

- `IgnoreClient`: seit Version 2.0.23. Ein eventueller Query-String der Client-Anfrage kann normalerweise dazu genutzt werden, eine Reihe von Sortiervorgaben mitzuliefern. Diese werden weiter unten erläutert. Wenn `IgnoreClient` gesetzt ist, wird diese benutzerdefinierte Sortierung abgeschaltet.

Die folgenden Optionen sind nur für Fancy-Indizes geeignet:

- `DescriptionWidth[=n|*]`: verfügbar seit Version 2.0.23. Dies gibt die Zeichenbreite für die Beschreibung in einem Fancy-Index an. Wenn Sie die Option weglassen oder `DescriptionWidth` ohne Wert verwenden, entscheidet `mod_autoindex` automatisch über die beste Breite. Alternativ können Sie einen numerischen Wert angeben, um die Anzahl der Zeichen verbindlich festzulegen, oder `*` für eine beliebige Breite.

  Wenn Sie etwas anderes als `DescriptionWidth=*` angeben, beachten Sie bitte Folgendes: Die Beschreibung darf HTML-Code enthalten. Das kann problematisch werden, wenn Apache sie nach einer bestimmten Zeichenzahl abschneidet – möglicherweise bleibt ein bestimmtes HTML-Tag geöffnet, das Auswirkungen auf den Rest der Liste hat.
- `FoldersFirst`: seit Version 2.0.23. In einem Fancy-Index wird die normale Sortierreihenfolge geändert: Zunächst werden alle Unterverzeichnisse nach dem aktuellen Kriterium sortiert und angezeigt, anschließend alle normalen Dateien. Wenn Sie diese Option weglassen, werden Dateien und Verzeichnisse identisch behandelt und können beim Sortieren vermischt werden.
- `HTMLTable`: seit Version 2.0.23. Diese Option zeigt die Fancy-Index-Liste nicht als `<pre>...</pre>`-Block an, sondern als einfache HTML-Tabelle.
- `IconsAreLinks`: Dies macht die angezeigten Icons zusätzlich zu den Dateinamen zum Teil des jeweiligen Hyperlinks.
- `IconHeight[=Pixel]`: Mit dieser Option können Sie eine Höhenangabe für Icons hinzufügen – das HTML-Tag `<img>` erhält das Attribut `height`. Wenn Sie keinen Pixelwert angeben, wird die Höhe der mit Apache gelieferten Icons eingetragen, was meistens die vernünftigste Einstellung ist.

  Es ist sehr empfehlenswert, diese Option und die im Folgenden beschriebene Option `width` anzugeben: Genau wie bei selbst erstellten HTML-Dokumenten kann ein Browser die Seite sofort korrekt formatieren, wenn er schon vor dem Laden der Bilder weiß, welche Größe dafür freigehalten werden soll.
- `IconWidth[=Pixel]`: Diese Option fügt eine Breitenangabe für die Icons ein, also das HTML-Attribut `width`. Genau wie bei `height` wird die Höhe der Standard-Icons eingetragen, wenn Sie keinen Wert angeben.
- `NameWidth[=n|*]`: Dies ist die Breite für die Anzeige der Dateinamen in Zeichen. Wie bei `DescriptionWidth` wird der Wert automatisch festgelegt, wenn

Sie ihn weglassen. Eine numerische Angabe legt die entsprechende Breite fest; * lässt beliebige Breiten zu.

- ScanHTMLTitles: Bei HTML-Dokumenten ohne Beschreibung wird das <title>-Element ausgelesen und verwendet. Dieser Vorgang ist sehr rechenintensiv und sollte mit viel Bedacht (am besten gar nicht) eingesetzt werden.
- SuppressColumnSorting: Bei FancyIndexing sollen die Spaltentitel nicht zu Hyperlinks für die Sortierung gemacht werden. Vor Version 2.0.23 wurden durch diese Option auch die Query-Strings in Client-Anfragen deaktiviert. Dafür ist nun IgnoreClient zuständig.
- SuppressDescription: Die Beschreibungsspalte in einem Fancy-Index soll weggelassen werden. Dies spart 23 Zeichen, wenn Sie ansonsten die Standardeinstellungen beibehalten.
- SuppressHTMLPreamble: Der HTML-Startabschnitt (<html> bis <body>) wird weggelassen. Sie sollten diese Option aktivieren, wenn Sie mit HeaderName ein eigenes Kopfdokument einbinden.
- SuppressIcon: seit Version 2.0.23. Beim FancyIndexing wird durch diese Option die Anzeige der Icons deaktiviert.
- SuppressLastModified: Die Spalte mit Datum und Uhrzeit der letzten Änderung wird nicht angezeigt.
- SuppressRules: In einem Fancy-Index werden die Trennbalken (<hr>-Tags) weggelassen.
- SuppressSize: Die Spalte mit der Dateigröße wird nicht angezeigt.
- TrackModified: seit Version 2.0.23. Die HTTP-Antwort wird mit Last-Modified- und ETag-Headern für das Verzeichnis ausgestattet, falls diese Informationen verfügbar sind.
- VersionSort: seit Version 2.0a3. Dateinamen mit Versionsnummern sollen korrekt numerisch sortiert werden (andernfalls werden sie falsch »alphabetisch« geordnet). Wenn Ihr Verzeichnis beispielsweise mytool-1.0.1, mytool-1.0.9 und mytool-1.0.10 enthält, werden sie mit VersionSort in dieser Reihenfolge angezeigt. Ist die Option dagegen nicht aktiviert, erhalten Sie die folgende falsche Anordnung: mytool-1.0.1, mytool-1.0.10, mytool-1.0.9.

#### IndexStyleSheet

CSS-Datei für das Styling des Verzeichnisindexes

| Seit Version | 2.2 |
|---|---|
| Modul | mod_autoindex |

| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
|---|---|
| Syntax | IndexStyleSheet URL-Pfad |
| Standardwert | – |

Mit IndexStyleSheet können Sie eine CSS-Datei angeben, die auf den generierten Index angewendet wird und dessen Aussehen beeinflusst. Wenn Sie IndexOptions HTMLTable verwenden, werden die folgenden CSS-Klassen im erzeugten HTML-Code eingesetzt, die Sie zusätzlich stylen können:

- tr.indexhead – Kopfzeile der Verzeichnisliste
- th.indexcolicon, td.indexcolicon – Icon-Spalte
- th.indexcolname, td.indexcolname – Dateinamen-Spalte
- th.indexcollastmod, td.indexcollastmod – Änderungsdatums-Spalte
- th.indexcolsize, td.indexcolsize – Dateigrößen-Spalte
- th.indexcoldesc, td.indexcoldesc – Beschreibungsspalte
- tr.breakrow – horizontale Trennlinie am Fuß der Tabelle
- tr.odd, tr.even – abwechselnde gerade und ungerade Spalten, geeignet zur unterschiedlichen Einfärbung, die eine bessere Orientierung in der Tabelle ermöglicht

#### IndexIgnore

Dateien, die nicht im Index erscheinen sollen

| Modul | mod_autoindex |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| Syntax | IndexIgnore Datei [Datei ...] |
| Standardwert | . (das Verzeichnis selbst) |

Mithilfe dieser Direktive können Sie eine Liste von Dateien oder Dateimustern angeben, die nicht im Index erscheinen sollen. Eine Besonderheit ist das Verhalten, wenn Sie im gleichen oder auch in einem untergeordneten Kontext eine zusätzliche IndexIgnore-Anweisung verwenden: Die bisherigen Einstellungen werden nicht ersetzt, stattdessen werden die neuen hinzugefügt.

Es ist empfehlenswert, zumindest .htaccess und die Muster *~ sowie *.bak und *.old zu sperren, die häufig für Backup-Dateien genutzt werden:

```
IndexIgnore .htaccess *~ *.bak *.old
```

### IndexIgnoreReset

Die Liste der wegzulassenden Dateien zurücksetzen

| Seit Version | 2.3.10 |
|---|---|
| Modul | mod_autoindex |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| Syntax | IndexIgnoreReset On\|Off |
| Standardwert | Off |

Wenn Sie die neue Direktive `IndexIgnoreReset` auf `On` setzen, werden die aus einer übergeordneten Konfiguration geerbten Einstellungen für `IndexIgnore` zurückgesetzt. Somit ist es erstmals möglich, in einem untergeordneten Kontext weniger Dateien oder Dateimuster im Index wegzulassen als in übergeordneten Kontexten.

### IndexOrderDefault

Sortierung der Indexeinträge

| Modul | mod_autoindex |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| Syntax | IndexOrderDefault Ascending \| Descending Name \| Date \| Size \| Description |
| Standardwert | Ascending Name |

Diese Direktive bestimmt, wie Fancy-Indizes standardmäßig sortiert werden sollen. Der erste der beiden Parameter bestimmt die Sortierreihenfolge: `Ascending` sortiert aufsteigend, `Descending` absteigend. Der zweite Parameter legt dagegen das Kriterium fest, nach dem sortiert werden soll: `Name` (Dateiname), `Date` (Änderungsdatum), `Size` (Größe) oder `Description` (Beschreibung).

Beachten Sie, dass die mittels `IndexOptions` gesetzten Optionen `FoldersFirst` und `VersionSort` unabhängig von der `IndexOrderDefault`-Einstellung beachtet werden. Das folgende Beispiel sortiert absteigend nach Datum (neueste Datei zuerst):

```
IndexOrderDefault Descending Date
```

### AddIcon

Index-Icon für eine bestimmte Datei festlegen

| Modul | mod_autoindex |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| Syntax | AddIcon Icon Name [Name ...] |
| Standardwert | nicht gesetzt |

Mit dieser Direktive wird das Icon festgelegt, das für die angegebenen Dateien in einem Fancy-Index angezeigt werden soll. Der erste Parameter ist das Icon; es ist entweder ein URL-Pfad (relativ zur `DocumentRoot`) oder ein in Klammern stehender Ausdruck nach dem Schema `(Alternativtext,RelativeURL)` – ohne Leerzeichen hinter dem Komma.

Das zweite Argument ist eine Datei; eventuelle weitere Argumente ebenfalls. Sie können vollständige Dateinamen oder Muster angeben. Zwei besondere Angaben sind `^^DIRECTORY^^` für Verzeichnisse und `^^BLANKICON^^` für Leerzeilen – für Letztere ist in der Apache-Konfiguration das mitgelieferte Icon `blank.gif` voreingestellt, damit die Formatierung stimmt.

Hier zwei Beispiele aus der vorgefertigten Konfigurationsdatei, für das übergeordnete Verzeichnis (..) beziehungsweise für Verzeichnisse allgemein:

```
AddIcon /icons/back.gif ..
AddIcon /icons/folder.gif ^^DIRECTORY^^
```

In der Regel brauchen Sie sich nicht selbst um die Beschaffung von Icons zu kümmern – Apache wird mit einer großen Auswahl davon ausgeliefert. Sie befinden sich im Verzeichnis *icons*, das je nach Installationslayout an einer anderen Stelle im Dateisystem liegt (siehe Kapitel 4, »Apache kompilieren und installieren«) – bei UNIX-Systemen mit dem GNU-Layout ist es z. B. */usr/local/share/apache2/icons*, in einer Standardinstallation unter Windows *C:/Programme/Apache Group/Apache2/icons*. In der mitgelieferten Konfigurationsdatei wird es durch eine `Alias`-Direktive unter dem URL-Pfad */icons* eingebunden. Alle Icons sind einmal als GIF und einmal als PNG vorhanden. In Abbildung 8.3 sehen Sie sie im Überblick.[4]

4 Einige etwas größere »Powered by Apache«-Banner, die sich ebenfalls in dem Verzeichnis befinden, wurden hier weggelassen.

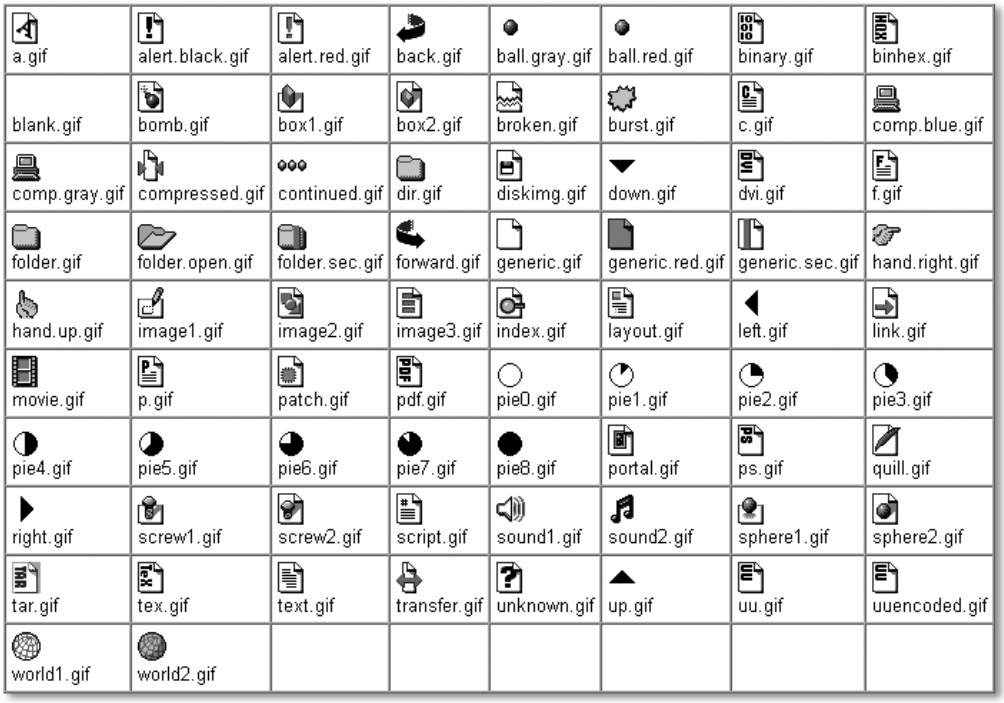



**Abbildung 8.3** Übersicht über die Icons, die mit Apache 2 geliefert werden

### AddIconByEncoding

Index-Icons nach MIME-Codierung festlegen

| | |
|---|---|
| **Modul** | `mod_autoindex` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`Indexes`) |
| **Syntax** | `AddIconByEncoding Icon MIME-Codierung [MIME-Codierung ...]` |
| **Standardwert** | nicht gesetzt |

Diese Direktive setzt ein Icon je nach MIME-Codierung. Auf diese Weise lassen sich komprimierte Dateien kennzeichnen. Das erste Argument ist wie bei `AddIcon` der URL-Pfad des Icons oder eine `(Alternativtext,URL)`-Gruppe; die nachfolgenden Argumente geben die MIME-Codierungen an, für die das Icon festgelegt wird. Beispiel:

```
AddIconByEncoding /icons/compressed.gif x-gzip
```

### AddIconByType

Index-Icon nach MIME-Type

| Modul | mod_autoindex |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| Syntax | AddIconByType Icon MIME-Type [MIME-Type ...] |
| Standardwert | nicht gesetzt |

Mit `AddIconByType` können Sie das Icon für Dateien mit einem bestimmten MIME-Type festlegen. Wie üblich ist das erste Argument die URL- oder `(Alternativtext,URL)`-Liste für das Icon. Die nachfolgenden Argumente sind die gewünschten MIME-Types, die bei Bedarf auch Platzhalter enthalten dürfen – z. B. `audio/*` für alle Sound-Dateiformate.

Die Standard-Konfigurationsdatei enthält bereits einige Zuordnungen für häufig genutzte Dateitypen. Hier zwei Beispiele:

```
AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
```

### DefaultIcon

Icon für Dateien, für die kein spezielles Icon definiert ist

| Modul | mod_autoindex |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| Syntax | DefaultIcon URL-Pfad |
| Standardwert | nicht gesetzt |

Diese Direktive legt fest, welches Symbol angezeigt werden soll, wenn einer Datei kein spezielles Icon zugeordnet wurde. Der einzige Parameter ist der URL-Pfad des gewünschten Icons. In der vorgefertigten *httpd.conf* steht diese Standardeinstellung:

```
DefaultIcon icons/unknown.gif
```

### AddAlt

Alternativtext für ein Icon im Index

| Modul | mod_autoindex |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |

| Syntax | AddAlt String Datei [Datei ...] |
|---|---|
| Standardwert | nicht gesetzt |

Diese Direktive legt den Alternativtext fest, der anstelle eines Icons in Browsern angezeigt werden soll, die keine Bilder anzeigen können oder in denen Bilder deaktiviert sind. Der angegebene String wird als `alt`-Attribut in das jeweilige `<img>`-Tag eingetragen.

Als ersten Parameter müssen Sie den gewünschten Text angeben – falls er Leerzeichen enthält, natürlich in Anführungszeichen. Die nachfolgenden Argumente sind Dateien, Dateimuster oder die beiden Spezialformulierungen `^^BLANKICON^^` und `^^DIRECTORY^^` (siehe `AddIcon`). Beispiele:

```
AddAlt "Nähere Informationen" readme.html
AddAlt "Flash-Movie" *.swf
```

### AddAltByEncoding

Alternativtext für ein Icon nach MIME-Codierung

| Modul | mod_autoindex |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| Syntax | AddAltByEncoding String MIME-Codierung [MIME-Codierung ...] |
| Standardwert | nicht gesetzt |

Diese Direktive legt den Alternativtext für die Icons von Dateien mit der angegebenen MIME-Codierung fest. Als Erstes wird der Text angegeben, die weiteren Argumente sind Codierungen. Beispiel:

```
AddAltByEncoding "GZIP-komprimiert" x-gzip
```

### AddAltByType

Alternativtext für ein Icon nach MIME-Type

| Modul | mod_autoindex |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| Syntax | AddAltByType String MIME-Type [MIME-Type ...] |
| Standardwert | nicht gesetzt |

Die Direktive `AddAltByType` setzt den Alternativtext für die Icons von Dateien mit den entsprechenden MIME-Types. Wie üblich ist der erste Parameter der Text;

dahinter können Sie die MIME-Types angeben, für die dieser Alternativtext festgelegt werden soll. Beispiele:

```
AddAltByType "Textdatei" text/*
AddAltByType "JPEG-Bild" image/jpeg
```

### AddDescription

Beschreibung für eine Datei im Index

| | |
|---|---|
| **Modul** | mod_autoindex |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| **Syntax** | AddDescription String Datei [Datei ...] |
| **Standardwert** | nicht gesetzt |

Mit dieser Direktive wird die Beschreibung für die angegebenen Dateien oder Dateimuster festgelegt. Sie darf HTML-Code enthalten; beachten Sie aber bitte, dass es Probleme geben kann, wenn die Beschreibung vorzeitig abgeschnitten wird – Näheres dazu bei den Erläuterungen zu IndexOptions.

Für die Beschreibung stehen standardmäßig 23 Zeichen zur Verfügung. Wenn Sie mittels IndexOptions SuppressIcon das Icon weglassen, kommen sechs weitere dazu. Noch einmal sieben Zeichen gewinnen Sie mit der Option SuppressSize und weitere 19 mit SuppressLastModified. Sie können IndexOptions aber auch mit der Option DescriptionWidth verwenden, um die Breite anzupassen.

Das erste Argument ist der Beschreibungstext, dahinter können Sie beliebig viele Dateinamen oder -muster angeben, für die dieser Text gelten soll.

Hier einige Beispiele:

```
AddDescription "Logo der MyNet GmbH" logo.gif
AddDescription "JPEG-Bild" *.jpg
AddDescription "Aktuelle Neuigkeiten" \
   jan.html feb.html mrz.html
```

### HeaderName

Name einer Datei, die über dem Index eingefügt wird

| | |
|---|---|
| **Modul** | mod_autoindex |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Indexes) |
| **Syntax** | HeaderName Dateiname |
| **Standardwert** | nicht gesetzt |

Diese Direktive legt den Namen einer HTML-Datei fest, die über dem eigentlichen Verzeichnisinhalt eingefügt werden soll. Ein relativer Pfad wird relativ zum aktuellen Verzeichnis interpretiert. Ein Pfad mit führendem Slash ist dagegen relativ zur `DocumentRoot`. Beispiele:

```
HeaderName localhead.html    # Individuell für das Verzeichnis
HeaderName /globalhead.html  # Datei für alle Indizes
```

Eine korrekte Kopfdatei sollte bereits den Anfangsblock eines HTML-Dokuments von `<html>` bis `<body>` enthalten. Deshalb empfiehlt es sich, die Erzeugung dieses Blocks für den automatisch erzeugten Index selbst zu deaktivieren, damit er nicht doppelt vorkommt:

```
IndexOptions +SuppressHTMLPreamble
```

Wenn für das Verzeichnis die Option `Includes` gesetzt ist, werden im Header-Dokument Server Side Includes (SSI) aktiviert. Auch Content-Negotiation (hier im Sinne einer selektiven Liste gemäß den `Accept-*`-Headern der Anfrage) wird verwendet, und zwar wenn die Option `MultiViews` eingeschaltet ist. CGI ist dagegen nur möglich, wenn der MIME-Type der verwendeten Dateiendung auf einen `text/*`-Typ gesetzt wird, z. B. folgendermaßen:

```
AddType text/html .cgi
HeaderName top.cgi
```

#### ReadmeName

Name der Datei, die unter dem Index eingefügt wird

| | |
|---|---|
| **Modul** | `mod_autoindex` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`Indexes`) |
| **Syntax** | `ReadmeName Dateiname` |
| **Standardwert** | nicht gesetzt |

Diese Direktive legt den Namen einer HTML-Datei fest, die unter dem eigentlichen Index angefügt wird. Die Syntax und eine nähere Beschreibung der Funktionsweise finden Sie unter `HeaderName`. Beispiel:

```
ReadmeName readme.html
```

### 8.2.2 Serverseitige Image Maps mit mod_imagemap

Eine *Image Map* ist ein Bild, das verschiedene anklickbare Bereiche enthält. Ein Klick auf einen dieser Bereiche dient als HTML-Hyperlink. Es gibt zwei Grund-

typen: serverseitige und clientseitige Image Maps. Bei einer serverseitigen Image Map werden die Koordinaten des Mausklicks an den Server übermittelt, der sie daraufhin auswertet. Eine clientseitige Image Map wird dagegen vom Browser selbst bereitgestellt. Da heute so gut wie jeder Browser diese Technologie unterstützt, sind serverseitige Image Maps in den letzten Jahren etwas aus der Mode gekommen. Dennoch stellt Apache sie nach wie vor bereit, und zwar über das Modul `mod_imagemap` (bis Version 2.0.x hieß es irreführenderweise `mod_imap`).

### Image-Map-Dateien

Damit ein Browser bei einem Klick auf ein Bild überhaupt die Koordinaten an den Server übermittelt, benötigt das entsprechende `<img>`-Tag das Attribut `ismap` (ohne speziellen Wert – in klassischem HTML also einfach `ismap`, in XHTML dagegen `ismap="ismap"`). Die serverseitige Image Map selbst ist eine Definitionsdatei in einem speziellen Format, auf die das Bild als Hyperlink verweisen sollte. Hier ein Beispiel für einen solchen Verweis:

```
<a href="menu.map"><img src="map.gif" ismap="ismap" /></a>
```

Die Map-Datei selbst benötigt den Handler `imap-file`, damit Apache sie entsprechend auswertet. Dieser Handler wird normalerweise für die Dateiendung `.map` festgelegt:

```
AddHandler imap-file map
```

Aus Kompatibilitätsgründen kann stattdessen nach wie vor der MIME-Type `application/x-httpd-imap` angegeben werden; in Zukunft wird dies aber wahrscheinlich aufgegeben werden:

```
AddType application/x-httpd-imap map
```

Eine Image-Map-Datei ist eine einfache Textdatei, die einige spezielle Direktiven[5] enthalten kann. Apache 2 erkennt in einer Image-Map-Datei die folgenden Anweisungen:

- `base` oder `base_uri`
  Diese Anweisung entspricht dem HTML-Tag `<base href="..." />`: Sie legt die Basis für relative URLs fest. Beispiel:

  ```
  base http://www.mynet.de/mapfiles/
  ```

- `default`
  Diese Direktive gibt die URL an, auf die die Image Map verweisen soll, wenn sie außerhalb eines speziell definierten Bereiches angeklickt wurde.

5 Verstehen Sie den Begriff »Direktive« hier nicht falsch: Diese speziellen Image-Map-Anweisungen dürfen nicht in der Datei *httpd.conf* stehen, sondern nur in einer Image-Map-Datei.

Beispiel:

```
default standard.html
```

- `circle`
  Definiert einen kreisförmigen, anklickbaren Bereich. Das erste Argument ist die URL; als Koordinaten werden die x- und y-Position des Mittelpunktes sowie die x- und y-Position eines Punktes auf der Außenlinie angegeben. Beispiel:

  ```
  circle kreis.html 200,200 400,200
  # Mittelpunkt 200,200; Radius 200
  ```

- `rect`
  Mit dieser Direktive wird ein rechteckiger Bereich festgelegt. Die Koordinaten sind zwei diagonale Eckpunkte.

  Beispiel:

  ```
  rect rechteck.html 100,100 400,200
  ```

- `poly`
  Mit dieser Anweisung wird ein polygonaler Bereich mit 3 bis 100 Punkten definiert. Beispiel:

  ```
  poly polygon.html 100,100 100,200 200,100
  ```

- `point`
  Dies gibt einen einzelnen Punkt an. Wenn der Cursor in keinem anderweitig definierten Bereich liegt, wird der Link für die nächstgelegene `point`-Definition befolgt. Das bedeutet auch, dass die Einstellung unter `default` bei Verwendung von `point` deaktiviert ist. Beispiel:

  ```
  point punkt.html 200,200
  ```

`default` und die vier Koordinatendefinitionen können zusätzlich Text in Anführungszeichen enthalten. Dieser Text wird als Linktext für den entsprechenden Hyperlink ausgegeben, wenn Apache ein Menü erstellt. Beispiel:

```
rect rechteck.html "Alles &uuml;ber Rechtecke" 100,100 400,200
```

Im Menü würde daraufhin der folgende Hyperlink erstellt werden:

```
<a href="rechteck.html">Alles &uuml;ber Rechtecke</a>
```

Wenn Sie diesen Text weglassen, wird die URL selbst als Linktext eingefügt.

Statt einer URL können Sie für viele der obigen Anweisungen auch folgende Spezialwerte angeben:

- `map` oder `menu`
  Die URL der Map-Datei selbst. Da keine Koordinaten mitgeschickt werden, erzeugen diese speziellen Angaben ein Menü – es sei denn, `ImapMenu` besitzt den Wert `none`. Einige alte Browser können keine Koordinaten übermitteln; deshalb wird das Menü bei Anfragen ohne Koordinaten und im Normalfall auch bei Klicks auf undefinierte Koordinaten angezeigt.
- `Referer`
  URL des Dokuments, das auf die Map-Datei verwiesen hat. Falls die Anfrage keinen `Referer`-Header enthielt, wird *http://servername/* verwendet.
- `Nocontent`
  Dieser Wert kann bei allen Image-Map-Direktiven außer `base` angegeben werden. Er sorgt dafür, dass statt eines Dokuments `204 No Content` zurückgeliefert wird. Normalerweise zeigt ein Browser in diesem Fall weiter die ursprüngliche Seite an; dies ist also ein praktischer Wert für `default`.
- `Error`
  Dies sorgt dafür, dass Apache 2 für alle Anfragen eine Fehlermeldung mit der Statusangabe `500 Internal Server Error` liefert. Sinnvoll ist dieses Verhalten als Einstellung für `default` – mit anderen Worten: Außerhalb der definierten Bereiche soll nicht geklickt werden dürfen.

Im Folgenden finden Sie die drei `httpd.conf`-Direktiven, mit denen Sie die eigentlichen Konfigurationseinstellungen für eine Image Map vornehmen können.

#### ImapBase

Basis-URL für Image-Map-Dateien

| | |
|---|---|
| **Modul** | `mod_imagemap` (vor 2.1: `mod_imap`) |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`Indexes`) |
| **Syntax** | `ImapBase map\|referer\|URL` |
| **Standardwert** | `http://hostname` |

Dies ist die Basis-URL, die Sie mit der Anweisung `base` auch in der Map-Datei selbst angeben können. Relativen URLs in der Map-Definition wird diese URL vorangestellt. Die Verwendung dieser Konfigurationsdirektive empfiehlt sich, wenn Sie in einem Kontext mehrere Image Maps verwenden, deren Basis-URL identisch ist. Die `base`-Direktive in der Map-Datei überschreibt im Einzelfall diese Voreinstellung.

Sie können entweder eine beliebige URL oder aber einen der beiden weiter oben erläuterten Werte `map` oder `menu` angeben.

#### ImapDefault

Aktion, wenn Koordinaten ohne spezielle Definition angeklickt wurden

| | |
|---|---|
| **Modul** | `mod_imagemap` (vor 2.1: `mod_imap`) |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`Indexes`) |
| **Syntax** | `ImapDefault error\|nocontent\|map\|referer\|URL` |
| **Standardwert** | `nocontent` |

Diese Direktive gibt an, wohin ein Klick auf die Image Map verweisen soll, wenn kein definierter Bereich angeklickt wurde. Eine `default`-Anweisung in einer Map-Datei überschreibt diese Einstellung, sodass Sie diese Konfigurationsdirektive vor allem dann verwenden sollten, wenn Sie mehrere Image Maps verwenden. Die möglichen Einstellungen entsprechen den bereits genannten Werten für Image-Map-Dateien.

#### ImapMenu

Anzeigen eines Menüs, wenn der Browser keine Koordinaten übergeben hat

| | |
|---|---|
| **Modul** | `mod_imagemap` (vor 2.1: `mod_imap`) |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`Indexes`) |
| **Syntax** | `ImapMenu none \| formatted \| semiformatted \| unformatted` |
| **Standardwert** | nicht gesetzt |

Wenn ein Client die Image Map selbst ohne Koordinatenangaben als URL anfordert, wertet `mod_imagemap` dies als Aufforderung, ein Menü zu senden. Sie können dieses Verhalten auch für bestimmte Bereiche erzwingen, indem Sie das Linkziel `map` (oder sein Synonym `menu`) angeben. Diese Direktive legt fest, wie das entsprechende Menü beschaffen sein soll. Dazu können Sie einen der vier folgenden Werte angeben:

- `None`
  Es wird kein Menü erzeugt. Der Server befolgt die `default`-Einstellung.
- `Formatted`
  Das Menüdokument erhält eine Hauptüberschrift, die Links werden – durch Trennlinien separiert – untereinandergeschrieben. Kommentare aus der Map-Datei werden ignoriert.

- `Semiformatted`
  Zusätzlich zur Einstellung `formatted` werden die Kommentare aus der Image-Map-Datei angezeigt; darin befindliche Zeilenumbrüche werden in `<br />`-Tags umgewandelt.
- `Unformatted`
  Sie können die Formatierung des Menüs komplett selbst vornehmen, indem Sie die Map-Datei mit HTML-Code anreichern.

## 8.3 Zusammenfassung

Die Direktiven und Module, die in diesem Kapitel vorgestellt wurden, vervollständigen das Handwerkszeug, das Sie für den Betrieb einer vollständigen Website benötigen. Hier noch einmal das Wichtigste im Überblick:

- Mithilfe von `Alias`-Anweisungen lassen sich Dateien und Verzeichnisse, die aus Sicherheits- oder Organisationsgründen eigentlich außerhalb der `DocumentRoot` liegen, mit öffentlich zugänglichen URLs belegen.
- Weiterleitungen (`Redirect`-Optionen) ermöglichen es Ihnen, eine angeforderte URL auf eine andere Website umzuleiten. Anders als bei Aliasen bemerken die User dies, weil sich die Adresse im Browser ändert.
- Das komplexe, aber überaus leistungsfähige Modul `mod_rewrite` ist in der Lage, die vollständige Funktionalität aller `Alias`- und `Redirect`-Anweisungen bereitzustellen. Darüber hinaus bietet es endlos viele weitere Optionen; Sie können die Umleitung von beinahe beliebigen Bedingungen abhängig machen.
- Eine weitere praktische Funktion von Apache ist die Fähigkeit, automatisch einen Verzeichnisindex zu generieren. Sie wird durch das vielseitig konfigurierbare Modul `mod_autoindex` bereitgestellt und bietet sich vor allem für größere Download-Bereiche an.

*»Die beste und sicherste Tarnung ist immer noch die blanke und nackte Wahrheit. Die glaubt niemand.«*
*– Max Frisch*

# 9 Authentifizierung, Autorisierung und Zugriffskontrolle

Während rein informative Webangebote in der Regel ohne Einschränkungen zugänglich sein sollen, ist es für andere Anwendungen erforderlich, den Zugriff auf bestimmte User zu beschränken. In diesem Kapitel geht es deshalb um den Schutz von Verzeichnissen durch die persönliche Anmeldung mit Benutzernamen und Passwort. Apache stellt dazu diverse leicht unterschiedliche Verfahren zur Verfügung, die in den verschiedenen `mod_auth*`-Modulen realisiert wurden. Zahlreiche Drittanbieter-Module ergänzen das Angebot, beispielsweise um den Zugriff auf das Benutzermanagement einiger Betriebssysteme oder die Speicherung der Anmeldedaten in diversen Datenbanksystemen.

## 9.1 Grundlagen

Wahrscheinlich möchten Sie nicht, dass die Öffentlichkeit ohne jegliche Einschränkung auf alle Inhalte Ihrer Website Zugriff hat. Einige Bereiche sind vielleicht für Mitarbeiter reserviert, andere für bestimmte Kunden, und manche dienen sogar ausschließlich der internen Administration. In Kapitel 6, »Grundkonfiguration«, haben Sie bereits das Modul `mod_authz_host` kennengelernt. Es bietet die Möglichkeit, Bereiche Ihrer Site für bestimmte Hosts zu sperren oder sogar nur einzelnen Hosts den Zugriff zu erlauben. Das ist zwar in manchen Fällen praktisch und sollte aus Sicherheitsgründen gemäß den Ratschlägen in Kapitel 6 auch so eingesetzt werden, taugt aber vor allem wegen der folgenden beiden Probleme nicht als alleinige Lösung:

- Die meisten IP-Adressen von Client-Rechnern im Internet werden von Internet-Providern dynamisch zugewiesen. Das bedeutet, dass ein und derselbe

Rechner bei verschiedenen Besuchen auf Ihrer Site jeweils eine andere Adresse besitzt. Eine Identitätsüberprüfung anhand der Adresse ist damit nicht möglich; einen Hostnamen haben solche Rechner in der Regel erst recht nicht – oder er wird ebenfalls temporär zugewiesen und ist so informativ wie `xdsl-client-0815.t-online.de`.

- Ein potenzieller Angreifer kann die IP-Adresse seiner Anfrage fälschen. Dieses Verfahren wird als IP-Spoofing bezeichnet. Da die Adresse falsch ist, kann der Cracker diese Methode zwar nicht dazu benutzen, private Informationen aus geschützten Bereichen Ihres Webservers zu stehlen, aber er könnte sie z. B. für DoS-Attacken einsetzen.

Wirklichen Schutz für bestimmte Verzeichnisse bietet daher nur die persönliche Benutzeranmeldung oder *Authentifizierung* (englisch *authentication*). Bei diesem Verfahren fragt Apache den entfernten Benutzer nach einem Benutzernamen und einem Passwort und liefert die angeforderten Ressourcen nur aus, wenn diese Daten stimmen.

Genau genommen, besteht die Überprüfung vor dem Zugriff auf die Ressource innerhalb einer Website aus drei Schritten:

- Auf Anforderung des Servers sendet der Client Anmeldedaten (Benutzername und Passwort). Dies wird im engeren Sinn als *Authentifizierung* bezeichnet.
- Der Server überprüft anhand des in der Konfiguration gewählten Verfahrens, ob die Anmeldedaten stimmen. Dieser Vorgang heißt *Autorisierung*.
- Schließlich wird festgestellt, ob der erfolgreich angemeldete User auf die ausgewählte Ressource zugreifen darf – dies ist die *Zugriffskontrolle*. Neben der Benutzeridentität gibt es noch diverse andere Kriterien für die Zugriffskontrolle, beispielsweise IP-Adresse und Hostname des Client-Hosts, Dateiberechtigungen und so weiter.

Zusammen werden diese drei Phasen im Apache-Sprachgebrauch auch als *AAA* (*authentication, authorization, and access control*) bezeichnet.

In diesem Abschnitt erhalten Sie zunächst einen Überblick über die Authentifizierungsmodule in Apache seit Version 2.2. Anschließend werden alle Direktiven vorgestellt, die für die persönliche Anmeldung und andere Aspekte der Zugriffskontrolle zuständig sind. Einige von ihnen sind im Core definiert, weil unterschiedliche Arten der Authentifizierung sie gemeinsam verwenden; die eigentlichen Direktiven zur Anmeldungsprüfung gehören dagegen zu den diversen `auth*`-Modulen.

### 9.1.1 Die Organisation der Authentifizierungsmodule seit Apache 2.2

Die beiden grundlegenden Arten der HTTP-Authentifizierung werden in der HTTP/1.1-Definition in RFC 2616 angesprochen (siehe Kapitel 2, »Funktionsweise von Webservern«). Näher ausgeführt werden sie dagegen in RFC 2617, *HTTP Authentication: Basic and Digest Access Authentication*. Die Basic-Authentifizierung überträgt die Anmeldeinformationen im Klartext, während die leider noch nicht von allen Clients unterstützte Digest-Authentifizierung sie verschlüsselt versendet.

Für Apache 2.2 wurden die Authentifizierungsmodule neu organisiert. In Apache 2.0 musste jedes Modul, das die Anmeldedaten auf andere Weise speichert (z.B. in Textdateien, DBM-Dateien, Datenbanken oder LDAP-Verzeichnissen) selbst für die Kommunikation mit dem Client, das heißt für Anmeldeanforderung und -annahme, sorgen. Diese unnötige und fehlerträchtige Codeverdopplung wurde beseitigt: Die Module `mod_auth_basic` und `mod_auth_digest` stellen die beiden Anmeldemethoden zur Verfügung; die verschiedenen Speicherungsmöglichkeiten werden unabhängig davon durch sogenannte *Provider-Module* bereitgestellt. Die `mod_authn_*`-Module sind für die eigentliche Authentifizierung (authentication) zuständig, während sich die `mod_authz_*`-Module (authorization) um die Zugriffskontrolle – unter anderem um die Verwaltung von Gruppenmitgliedschaften – kümmern.

Im Einzelnen stehen seit Apache 2.2 folgende Provider-Module bereit:

- `mod_authn_file` liest Benutzerdaten aus einfachen Textdateien.
- `mod_authz_groupfile` ermittelt entsprechend Gruppendaten aus Textdateien.
- `mod_authn_dbm` verwaltet Benutzerdaten in datenbankähnlichen DBM-Dateien.
- `mod_authz_dbm` ermittelt die Gruppeninformationen aus DBM-Dateien.
- `mod_authnz_ldap` erfragt sowohl Benutzer- als auch Gruppendaten von LDAP-Verzeichnisservern, funktioniert bisher aber nur für die Basic-Authentifizierung mit `mod_auth_basic`.
- `mod_authn_dbd` ermittelt das zu einem Benutzernamen gehörende Passwort per SQL-Abfrage aus einer mittels `mod_dbd` hergestellten Datenbankverbindung.
- `mod_authn_anon` erlaubt die anonyme Anmeldung nach dem Muster von Anonymous FTP.
- `mod_authn_alias` ermöglicht die Definition eigener, komplexerer Authentifizierungsschemata auf der Basis vorhandener Quellen.
- `mod_authn_default` bildet das Fallback-Modul, wenn kein anderes Authentifizierungsschema gültige Anmeldedaten liefern kann. Aufgrund einer weiteren Umstrukturierung wurde das Modul in Apache 2.4 wieder abgeschafft.

- `mod_authz_default` ist das entsprechende Fallback-Modul für Gruppendaten. Auch dieses Modul gibt es in Apache 2.4 nicht mehr.

Daneben gibt es drei spezielle `authz_*`-Module, die nicht in das Provider-Schema passen:

- `mod_authz_owner` regelt Zugriffsberechtigungen anhand von Dateieigentumsrechten.
- `mod_authz_user` sorgt dafür, dass Benutzer mit gültigem Benutzernamen (sofern sie auch das richtige Passwort übermitteln) Zugriff erhalten – dieser Aspekt wurde aus dem früheren Modul `mod_auth` ausgelagert.
- `mod_authz_host` ermöglicht es, Zugriffsrechte auf der Basis von Hostnamen beziehungsweise IP-Adressen zu erteilen. Dafür ist in Apache 2.4 erstmals die Direktive `Require` zuständig, die auch die meisten anderen Arten der Zugriffskontrolle regelt. Die veralteten Direktiven `Order`, `Allow` und `Deny` (siehe Kapitel 6, »Grundkonfiguration«) wurden dagegen in das Modul `mod_access_compat` verschoben und werden demnächst vermutlich verschwinden.

Abbildung 9.1 zeigt einen Ausschnitt des Schemas der Authentifizierungsmodule und deren Aufgaben seit Apache 2.2.

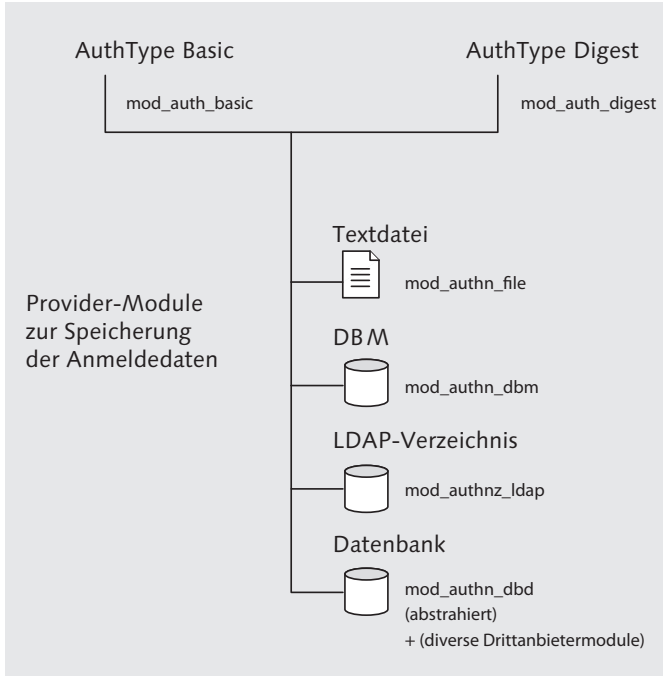



**Abbildung 9.1** Organisation der Authentifizierungsmodule in Apache 2.2

### 9.1.2 Ein erstes Beispiel

Bevor weiter unten die einzelnen Authentifizierungsdirektiven und ihre Optionen besprochen werden, soll hier anhand eines einfachen Beispiels erläutert werden, wie die Authentifizierung grundsätzlich funktioniert. Dabei werden einige Direktiven vorweggenommen.

Als Basis für das Beispiel soll das Verzeichnis `authtest` unterhalb der `DocumentRoot` dienen. Ein User namens `htuser` mit dem Passwort `topSecretX` soll auf dieses Verzeichnis zugreifen dürfen.

Der erste Schritt besteht darin, den Benutzer und sein Passwort einzurichten. Dies geschieht für die einfache Authentifizierung mithilfe des Kommandozeilenprogramms *htpasswd*, das mit Apache geliefert wird. Dieses Programm schreibt eine `Benutzername:Passwort`-Zeile in die angegebene Textdatei. Geben Sie folgenden Befehl ein, um eine neue Benutzerdatei mit dem Namen *.htusers* anzulegen, die sich im Verzeichnis */usr/local/apache2/misc* befindet, und darin den Benutzer `htuser` einzurichten:

```
# htpasswd -c /usr/local/apache2/misc/.htusers htuser
New password: **********
Re-type new password: **********
Adding password for user htuser
```

Auf einem UNIX-Rechner erscheint gar kein visuelles Feedback für die Passworteingabe; unter Windows werden *** angezeigt. Wenn die beiden Eingaben übereinstimmen, enthält die Datei nun eine Zeile wie diese:

```
htuser:uWbgik6e2DqCQ
```

Unter UNIX wurde das Passwort bis Version 2.2 standardmäßig `crypt`-verschlüsselt. Seit 2.3-beta wird dagegen standardmäßig MD5 verwendet (auf Windows- und NetWare-Systemen war dies schon vorher der Fall); dort sieht die entsprechende Zeile deshalb etwa so aus:

```
htuser:$apr1$t3/.....$yVa6oy9gdbyKHIy17WT84/
```

Nachdem der Usereintrag erzeugt wurde, können Sie in Ihrer *httpd.conf* einen entsprechenden `<Location>`-Block erstellen, der für das Verzeichnis *authtest* die Anmeldung des Benutzers `htuser` fordert. Dieser sieht folgendermaßen aus:

```
# Module laden bei DSO-Betrieb
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule authn_file_module modules/mod_authn_file.so
LoadModule authz_user_module modules/mod_authz_user.so
# Bei Apache 2.3/2.4 kommen noch folgende Module hinzu:
```

```
LoadModule authn_core_module modules/mod_authn_core.so
LoadModule authz_core_module modules/mod_authz_core.so

<Location /authtest>
  AuthType Basic
  AuthBasicProvider file
  AuthName "For your eyes only"
  AuthUserFile /usr/local/apache2/misc/.htusers
  Require user htuser
</Location>
```

Grob gesagt, bedeuten die verwendeten Direktiven Folgendes (weiter unten wird jede einzelne von ihnen genauer erläutert):

- `AuthType` – Art der Authentifizierung. `Basic` ist die einfache, klartextbasierte Übertragung von Anmeldedaten.
- `AuthBasicProvider` – Diese Direktive gibt an, von welchem Provider-Modul und damit aus welcher Datenquelle Apache die Benutzerdaten für den Vergleich mit der übermittelten Eingabe beziehen soll. Der Wert `file` steht für einfache Textdateien.
- `AuthName` – Name des Anmeldebereichs
- `AuthUserFile` – Pfad der Textdatei mit den Anmeldeinformationen. Wenn viele verschiedene User Zugriff erhalten sollen, lohnt es sich, diese zu einer Gruppe zusammenzufassen und zusätzlich den Pfad für ein `AuthGroupFile` anzugeben.
- `Require` – Gibt an, wer sich überhaupt anmelden darf beziehungsweise aufgrund welcher Voraussetzungen Benutzer Zugriff auf eine Ressource haben – besonders in Apache 2.4 beschränkt sich dies nicht mehr auf die Authentifizierung.

Anstelle von `<Location /authtest>` können Sie in einem solchen Fall natürlich auch `<Directory /Ihre/DocumentRoot/authtest>` schreiben. Der genaue formale Unterschied zwischen `<Directory>`- und `<Location>`-Containern wurde bereits in Kapitel 6, »Grundkonfiguration«, erläutert.

Sollte nun ein Client auf eine URL unterhalb von */authtest* zugreifen, sendet der Server eine Meldung mit dem Status `401 Unauthorized` und einem `WWW-Authenticate`-Header (siehe Kapitel 2, »Funktionsweise von Webservern«). Daraufhin zeigt jeder normale Browser einen Anmeldedialog an, wie in Abbildung 9.2 dargestellt. Die Anmeldedaten werden an den Server gesendet. Wenn sie mit den geforderten Informationen übereinstimmen, wird die ursprünglich angeforderte Ressource an den Benutzer gesendet. Zukünftige Anfragen für denselben Authentifizierungsbereich beantwortet der Browser während derselben Sitzung

automatisch mit den entsprechenden Anmeldeinformationen. Viele Browser bieten auch die Möglichkeit, sich die Anmeldedaten für zukünftige Besuche zu merken – besonders auf Computern, die von mehreren Personen genutzt werden, kann dies aber zu Sicherheitsproblemen führen.

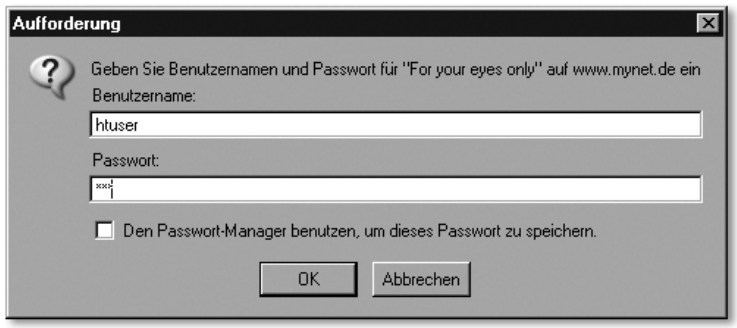



**Abbildung 9.2** Aufforderung zur persönlichen Anmeldung im Browser Mozilla Firefox

**Den ganzen Server schützen?**

Beachten Sie, dass alle Authentifizierungsdirektiven nur in Verzeichnis-Containern wie `<Directory>`, `<Location>` und `<Files>` sowie in *.htaccess*-Dateien stehen dürfen. In der Konfiguration für den Hauptserver oder für einen virtuellen Host sind sie dagegen nicht zulässig. Deshalb spricht man allgemein von »geschützten Verzeichnissen« und nicht von geschützten Servern.

Selbstverständlich steht es Ihnen aber jederzeit frei, sämtliche Verzeichnisse und URLs Ihres Servers oder eines virtuellen Hosts anmeldepflichtig zu machen. Das funktioniert z. B. so:

```
<Directory />
  # Den ganzen Server nur für "admin" zugänglich machen
  AuthType Basic
  AuthBasicProvider file
  AuthName "Die Festung des Apachen"
  AuthUserFile /usr/local/apache2/misc/.htusers
  Require user admin
</Directory>
```

Ratsam ist das im Alltag eigentlich nur, wenn Sie Wartungsarbeiten durchführen, die Schäden eines Absturzes ungestört beseitigen wollen oder nach einem Sicherheitsvorfall Beweise sichern möchten.

### 9.1.3 Grunddirektiven zur Authentifizierung

Da einige Authentifizierungseinstellungen von mehreren konkreten Modulen verwendet werden, wurden sie traditionell im Core definiert. Das bedeutet nicht,

dass Apache die Authentifizierung ohne die betreffenden Module durchführen könnte. Sie müssen für die eigentliche Anmeldung mindestens eines der weiter unten beschriebenen Module aktivieren.

In Apache 2.3-beta wurden diese Direktiven in die beiden neuen Module `mod_authn_core` und `mod_authz_core` ausgelagert, die die gemeinsame Funktionalität aller Authentifizierungs- beziehungsweise Zugriffskontroll-Module beinhalten.

#### AuthName

Angabe des Authentifizierungsbereichs (Realm)

| | |
|---|---|
| **Modul** | `core`; seit 2.3 `mod_authn_core` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthName Name-des-Bereichs` |
| **Standardwert** | nicht gesetzt |

Mithilfe dieser Direktive wird der Bereich für die Authentifizierung (Realm) festgelegt. Diese Information wird den Benutzern im Anmeldedialog gezeigt; auf dieser Basis können sie entscheiden, welcher Benutzername und welches Passwort erforderlich sind. Der Hauptnutzen besteht aber darin, dass die meisten Browser während einer Sitzung automatisch die korrekten Anmeldedaten übermitteln, wenn Domain-Name und Realm mit denjenigen einer bereits erfolgten Anmeldung übereinstimmen.

Wenn der Name des Authentifizierungsbereichs Leerzeichen enthalten soll, müssen Sie ihn in Anführungszeichen setzen. Beispiele:

```
AuthName Kundenbereich
AuthName "Nur fuer Mitarbeiter"
```

#### AuthType

Übertragungsverfahren für Authentifizierungsdaten

| | |
|---|---|
| **Modul** | `core`; seit 2.3 `mod_authn_core` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Synta** | `AuthType Basic\|Digest` |
| **Standardwert** | nicht gesetzt |

Diese Direktive wählt die Art der Authentifizierung aus. Möglich sind die Werte `Basic` (Klartextübertragung der Anmeldedaten) oder `Digest` (für verschlüsselte Übertragung). Entsprechend fordert eines der beiden Module `mod_auth_basic`

beziehungsweise mod_auth_digest die Anmeldedaten vom Client an. Diese werden dann zur Prüfung an das jeweils zuständige Provider-Modul übergeben. Beachten Sie, dass einige Provider-Module nur mit der Basic-Authentifizierung zusammenarbeiten.

#### Require

Angabe der Benutzer beziehungsweise Gruppen, die auf Ressourcen zugreifen dürfen; ab 2.3 auch weitere Zugriffsvoraussetzungen

| Modul | core; seit 2.3 mod_authz_core |
|---|---|
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | Require user UID [UID ...]<br>Require group GID [GID ...]<br>Require valid-user<br>seit 2.3 auch:<br>Require all granted<br>Require all denied<br>Require env Umgebungsvariable [Umgebungsvariable ...]<br>Require method HTTP-Methode [HTTP-Methode ...]<br>Require expr Ausdruck<br>Require ip IP-Adresse [IP-Adresse ...]<br>Require host Hostname [Hostname ...]<br>Require local<br>Require not Require-Bedingung |
| Standardwert | nicht gesetzt |

Mit dieser Direktive wird traditionell angegeben, welche Benutzer auf das geschützte Verzeichnis zugreifen dürfen. Es gibt drei verschiedene Möglichkeiten, die durch die jeweiligen authz-Module bereitgestellt werden:

- Require user Username [Username ...]: Sie können die Namen eines oder mehrerer User angeben, die nach entsprechender Anmeldung Zugriff erhalten sollen. Die Benutzernamen und Passwörter müssen in einer Datei stehen, die Sie – je nach konkreter Authentifizierungsart – durch Direktiven wie AuthUserFile oder AuthDigestFile angeben können.
- Require group Gruppenname [Gruppenname ...]: Wenn viele verschiedene Benutzer auf einen geschützten Bereich zugreifen dürfen, ist es empfehlenswert, sie zu Gruppen zusammenzufassen und hier anstelle der einzelnen Usernamen den oder die Gruppennamen anzugeben. Die Gruppendefinitionen stehen in einer Gruppendatei, die durch eine Direktive wie AuthGroupFile angegeben wird.

- `Require valid-user`: Jeder beliebige Benutzer aus den angegebenen Benutzer- und Gruppendateien darf sich anmelden.

Ab Apache 2.3-beta hat `Require` zusätzliche Aufgaben erhalten – unter anderem diejenigen der veralteten Direktiven `Order`, `Allow` und `Deny` (siehe Kapitel 6, »Grundkonfiguration« – dort finden Sie auch detaillierte Beispiele für die nachfolgenden Einstellungen). Hier gibt es folgende Einstellungen:

- `Require all granted`: Jeder beliebige Benutzer darf von jedem beliebigen Host aus zugreifen. Dies ist die übliche Einstellung für öffentliche Websites, also beispielsweise für den DocumentRoot-Directory-Container.
- `Require all denied`: Niemand darf zugreifen – die passende Voreinstellung für beliebige Verzeichnisse, also für den Container `<Directory /> ... </Directory>` in der Grundkonfiguration.
- `Require env Umgebungsvariable [Umgebungsvariable ...]`: Der Zugriff wird nur erlaubt, wenn die angegebene Umgebungsvariable gesetzt ist.
- `Require method HTTP-Methode [HTTP-Methode ...]`: Der Zugriff wird nur für die angegebenen HTTP-Methoden erlaubt.
- `Require expr Ausdruck`: Der Zugriff wird gewährt, wenn der angegebene Ausdruck zutrifft. Eine detaillierte Beschreibung von ap_expr-Ausdrücken finden Sie in der Beschreibung der Direktive `<If>` in Kapitel 6, »Grundkonfiguration«.

Durch das überarbeitete Modul `mod_authz_host` werden in Apache 2.3/2.4 folgende Einstellungen bereitgestellt (detaillierte Beispiele finden Sie dazu auch in Kapitel 6 bei den Direktiven `Allow` und `Deny`):

- `Require ip IP-Adresse [IP-Adresse ...]`: Nur Clients mit den angegebenen IP-Adressen oder Adressteilen dürfen zugreifen. Sie können sowohl IPv4- als auch IPv6-Adressen und Teile davon angeben, mit oder ohne CIDR-Bitmaske oder Teilnetzmaske.
- `Require host Hostname [Hostname ...]`: Nur Clients mit den angegebenen Hostnamen oder Hostnamensteilen dürfen zugreifen.
- `Require local`: Beschränkt den Zugriff auf den lokalen Host, das heißt, es muss eine der folgenden Vorgaben erfüllt sein: Die Client-Adresse liegt im Bereich `127.0.0.0/8` (IPv4) oder `::1` (IPv6), oder aber Client- und Server-IP-Adresse sind identisch.

Schließlich können Sie eine bestimmte Bedingung auch verneinen, indem Sie ihr das Schlüsselwort `not` voranstellen. Das folgende Beispiel verbietet den Zugriff, wenn die IP-Adresse aus dem angegebenen Teilnetz stammt:

```
Require not ip 177.122.0.0/16
```

Ein `Require not` allein kann den Zugriff nicht explizit erlauben; sein Rückgabewert ist entweder verboten (wenn die ausdrücklich verneinte Bedingung doch erfüllt ist) oder neutral. Damit die gezeigte Direktive tatsächlich eine Wirkung hat, muss sie daher wie folgt in einen Container geschrieben werden, der zunächst allen Usern den Zugriff erlaubt und ihn dann dem Teilnetz verbietet:

```
<RequireAll>
  Require all granted
  Require not ip 177.122.0.0/16
</RequireAll>
```

Beachten Sie, dass andere Module weitere Werte für `Require` hinzufügen können; einige von ihnen werden weiter unten in diesem Kapitel besprochen.

### Satisfy

Bestimmt, ob Host-/IP-Kontrolle und Benutzeranmeldung jeweils allein ausreichen oder ob beide erfüllt sein müssen.

| | |
|---|---|
| **Modul** | `core`; seit 2.3 `mod_access_compat` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` |
| **Syntax** | `Satisfy Any\|All` |
| **Standardwert** | `All` |

*Hinweis*: In Apache 2.3 wurde diese Direktive ins Modul `mod_access_compat` verschoben und für veraltet erklärt; auf Dauer wird sie ganz verschwinden. Verwenden Sie daher in neuen Apache-Versionen besser schon jetzt die im Anschluss beschriebenen, wesentlich leistungsfähigeren `Require`-Container `<RequireAll>`, `<RequireAny>` und `<RequireNone>`.

Natürlich können Sie für ein und dasselbe Verzeichnis sowohl die Host- beziehungsweise IP-basierte Zugriffskontrolle mit `mod_authz_host` (siehe Kapitel 6, »Grundkonfiguration«) als auch persönliche Anmeldungen durchführen. Mit der Direktive `Satisfy` lässt sich für einen solchen Fall festlegen, ob beide Voraussetzungen erfüllt sein müssen oder ob eine von ihnen genügt:

- `Satisfy All`: Beide Bedingungen müssen erfüllt sein. Es dürfen also nur persönlich angemeldete Benutzer von Hosts aus dem angegebenen Netzwerk auf die Ressource zugreifen. Dies ist der Standardfall; hier ein Beispiel:

  ```
  <Directory /usr/local/apache2/htdocs/topsecret>
    Order deny,allow
    Deny from all
  ```

```
  Allow from 192.168.0
  AuthType Basic
  AuthBasicProvider file
  AuthName "Inner Circle"
  AuthUserFile /usr/local/apache2/misc/.htusers
  AuthGroupFile /usr/local/apache2/misc/.htgroups
  Require group admins
  Satisfy All
</Directory>
```

- `Satisfy Any`: Es genügt, wenn eine Voraussetzung erfüllt ist. Auf die Ressource darf also jeder zugreifen, der einen Host innerhalb des angegebenen Netzes verwendet oder sich aus einem anderen Netz persönlich anmeldet. Dazu ebenfalls ein Beispiel:

```
<Directory /usr/local/apache2/htdocs/takeworkhome>
  Order deny,allow
  Deny from all
  Allow from 192.168.0
  AuthType Basic
  AuthBasicProvider file
  AuthName "Staff only"
  AuthUserFile /usr/local/apache2/misc/.htusers
  AuthGroupFile /usr/local/apache2/misc/.htgroups
  Require group staff
  Satisfy Any
</Directory>
```

#### RequireAll

Zusammenfassung beliebig vieler Require-Angaben, von denen eine Erfolg haben muss und keine fehlschlagen darf

| | |
|---|---|
| **Seit Version** | 2.3-beta |
| **Modul** | `mod_authz_core` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess (AuthConfig)` |
| **Syntax** | `<RequireAll> ... </RequireAll>` |

Während in Apache 2.2 die Authentifizierung und die Autorisierung grundlegend überarbeitet wurden, haben die Entwickler sich für 2.4 besonders der Zugriffskontrolle angenommen. Zum einen wurden praktisch alle Zugriffskontrolleinstellungen in der Direktive `Require` zentralisiert, und zum anderen wurde die Direktive `Satisfy` durch die hier vorgestellten Container `<RequireAll>`, `<RequireAny>` und `<RequireNone>` ersetzt.

<RequireAll> funktioniert folgendermaßen: Mindestens eine der dadurch verknüpften Require-Direktiven muss Erfolg haben, und keine von ihnen darf scheitern, damit der Zugriff explizit erlaubt wird. Erfolg bedeutet dabei, dass die überprüfte Bedingung explizit zutrifft. Für das Nicht-Scheitern genügt es dagegen auch schon, wenn es nicht ausdrücklich fehlschlägt, beispielsweise wenn die Bedingung für die aktuelle Anfrage keine Rolle spielt. Wenn auch nur eine Bedingung scheitert, wird der Zugriff verweigert. Wenn keine scheitert, aber auch keine Erfolg hat, wird ein neutrales Ergebnis zurückgegeben, das heißt, an dieser Stelle wird nicht über den Zugriff entschieden.

Alle drei Container-Direktiven können beliebig tief ineinander verschachtelt werden. In diesem Fall gilt das beschriebene Verhalten dann sinngemäß für die Ergebnisse der verschachtelten Container.

Hier ein einfaches Beispiel für <RequireAll>, das den Zugriff nur dann erlaubt, wenn Benutzeranmeldung und IP-Adresse stimmen (es handelt sich um die 2.3-Fassung des Satisfy All-Beispiels):

```
<Directory /usr/local/apache2/htdocs/topsecret>
  AuthType Basic
  AuthBasicProvider file
  AuthName "Inner Circle"
  AuthUserFile /usr/local/apache2/misc/.htusers
  AuthGroupFile /usr/local/apache2/misc/.htgroups
  <RequireAll>
    Require group admins
    Require ip 192.168.0
  </RequireAll>
  Satisfy All
</Directory>
```

#### RequireAny

Zusammenfassung beliebig vieler Require-Angaben, von denen mindestens eine Erfolg haben muss

| | |
|---|---|
| **Seit Version** | 2.3-beta |
| **Modul** | mod_authz_core |
| **Kontext** | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| **Syntax** | <RequireAny> ... </RequireAny> |

<RequireAny> verknüpft Zugriffskontroll-Direktiven per logischem Oder. Wenn eine von ihnen Erfolg hat, wird der Zugriff gewährt; wenn alle fehlschlagen, wird er verweigert. Wenn keine von ihnen Erfolg hat und keine fehlschlägt, wird ein neutrales Ergebnis zurückgegeben.

Das folgende Beispiel ist die Neufassung des weiter oben beschriebenen `Satisfy Any`-Beispiels:

```
<Directory /usr/local/apache2/htdocs/takeworkhome>  AuthType Basic
  AuthBasicProvider file
  AuthName "Staff only"
  AuthUserFile /usr/local/apache2/misc/.htusers
  AuthGroupFile /usr/local/apache2/misc/.htgroups
  <RequireAny>
    Require group staff
    Require ip 192.168.0
  </RequireAny>
</Directory>
```

#### RequireNone

Zusammenfassung beliebig vieler Require-Angaben, von denen keine Erfolg haben darf

| Seit Version | 2.3-beta |
|---|---|
| Modul | `mod_authz_core` |
| Kontext | `<Directory>`, `<Location>`, `<Files>`, `.htaccess (AuthConfig)` |
| Syntax | `<RequireNone> ... </RequireNone>` |

`<RequireNone>` ermöglicht die Verneinung mehrerer `Require`-Angaben. Der Zugriff wird verboten, wenn eine der enthaltenen `Require`-Anweisungen erfolgreich ist. Genau wie `Require not` kann auch ein `<RequireNone>`-Container allein nicht den Zugriff erlauben; Sie müssen den Container mit anderen `Require`-Angaben verknüpfen. Das folgende Beispiel erlaubt grundsätzlich allen Usern den Zugriff, es sei denn, die Anfrage stammt aus einem bestimmten Teilnetz oder erfolgt durch einen User aus der Gruppe `forbidden`:

```
<RequireAll>
  Require all granted
  <RequireNone>
    Require ip 192.168.0.0/8
    Require group forbidden
  </RequireNone>
</RequireAll>
```

#### AuthMerging

Kombination der Zugriffskontroll-Direktiven mit denjenigen aus dem übergeordneten Container

| Seit Version | 2.3-beta |
|---|---|
| Modul | mod_authz_core |
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | AuthMerging Off\|And\|Or |
| Standardwert | Off |

Standardmäßig werden die Require-Direktiven aus dem jeweils übergeordneten Verzeichnis so lange vererbt, bis ein bestimmtes Verzeichnis eigene Einstellungen enthält. Sobald dies der Fall ist, ersetzen die neuen Einstellungen die übergeordneten vollständig. Im folgenden Beispiel dürfen Benutzer aus der Gruppe users auf den URL-Pfad */test* zugreifen, während auf */test/config* nur Benutzer aus dem Teilnetz 196.17.23.0/8, aber ohne Einschränkung bezüglich der Gruppenzugehörigkeit zugreifen dürfen:

```
<Location /test>
  Require group users
</Location>

<Location /test/config>
  Require ip 196.17.23.0/8
</Location>
```

Die Direktive AuthMerging kann dieses Verhalten ändern. Der Wert And erfordert für den untergeordneten Bereich, dass beide Einstellungen beachtet werden (wie <RequireAll>), während Or wie <RequireAny> wirkt und daher verlangt, dass mindestens eine Regel erfüllt wird. Im folgenden geänderten Beispiel können nur noch Benutzer aus der Gruppe users auf den URL-Pfad */test/config* zugreifen, die gleichzeitig aus dem angegebenen Teilnetz stammen:

```
<Location /test>
  Require group users
</Location>

<Location /test/config>
  AuthMerging And
  Require ip 196.17.23.0/8
</Location>
```

#### AuthzSendForbiddenOnFailure

403 Forbidden anstelle von 401 Unauthorized senden, wenn die Authentifzierung Erfolg hat, aber die Autorisierung fehlschlägt

| Seit Version | 2.3-beta |
|---|---|
| Modul | mod_authz_core |
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | AuthzSendForbiddenOnFailure On\|Off |
| Standardwert | Off |

Wenn die Anmeldedaten eines Benutzers stimmen, er aber aus anderen Gründen nicht auf eine Ressource zugreifen darf, sendet Apache standardmäßig erneut den Statuscode `401 Unauthorized`. Daraufhin zeigen die meisten Browser wiederum den Anmeldedialog an. Wenn Sie diese Direktive auf `On` setzen, wird stattdessen `403 Forbidden` gesendet.

Wichtig: Dies kann die Sicherheit Ihrer Websites vermindern, weil die geänderte Nachricht potenzielle Angreifer darauf hinweisen kann, dass sie ein Passwort korrekt erraten haben.

## 9.2 Basic-Authentifizierung

Die einfachste Art der Authentifizierung ist die Kombination aus dem Autorisierungstyp `basic` und der Speicherung der Anmeldedaten in einfachen Textdateien. In dem kleinen Beispiel weiter oben wurde diese Art der Authentifizierung bereits gezeigt; hier wird sie genauer vorgestellt. Zuständig sind folgende Module: `mod_auth_basic` (für die Basic-Authentifizierung), das Provider-Modul `mod_authn_file` (zur Bereitstellung der Textdateien) sowie die Module `mod_authz_user` (Zugriffskontrolle für Benutzer) und `mod_authz_groupfile` (Zugriffskontrolle für Gruppen).

### 9.2.1 Das Programm htpasswd

Wie Sie im Beispiel weiter oben bereits gesehen haben, wird *htpasswd* verwendet, um Username-Passwort-Paare für Benutzer zu erzeugen, die Zugriff auf bestimmte geschützte Bereiche Ihrer Website erhalten sollen. Das Programm wird mit folgender Syntax aufgerufen:

```
# htpasswd [Optionen] Passwortdatei Username [Passwort]
```

Als Optionen können Sie eines oder mehrere der folgenden Flags angeben:

- `-c`: Diese Option erzeugt die Passwortdatei neu, falls sie noch nicht existiert; beim ersten Aufruf ist dieses Flag also erforderlich. Eine vorhandene Datei wird dagegen durch eine neue ersetzt, die nur den neu erstellten Eintrag enthält.

- `-n`: Dies ist eine Testoption: Wenn Sie sie verwenden, wird die Passwortdatei nicht geändert, denn das Ergebnis wird über die Standardausgabe angezeigt.
- `-m`: Das Passwort soll nicht `crypt`-, sondern MD5-verschlüsselt werden. Dies ist das Standardverhalten seit Version 2.3; auf Windows- und NetWare-Systemen schon immer.
- `-d`: Diese Option verlangt explizit die `crypt`-Verschlüsselung des Passworts (nur UNIX). Sinnvoll ist diese Option erst ab Version 2.3, um aus Kompatibilitätsgründen auch weiterhin `crypt` zu verwenden.
- `-p`: Das Passwort soll gar nicht verschlüsselt, sondern im Klartext gespeichert werden. Das klappt ausschließlich auf Nicht-UNIX-Plattformen, ist aber aus Sicherheitsgründen auch dort nicht zu empfehlen.
- `-s`: Mit dieser Option wird das Passwort nicht mit `crypt` oder MD5 verschlüsselt, sondern mit dem SHA1-Algorithmus.
- `-b`: Wenn Sie diese Option verwenden, wird das als letztes Argument angegebene Passwort verwendet, anstatt dass interaktiv danach gefragt wird. Besonders empfehlenswert ist das nicht – erstens könnte Ihnen jemand im falschen Augenblick über die Schulter schauen, und zweitens gibt es keinen Schutz vor Schreibfehlern, weil die Passworteingabe auf diese Weise nicht wiederholt wird.
- `-D`: Diese Option erzeugt keinen neuen Eintrag, sondern löscht den angegebenen Benutzer aus der Datei.

Das folgende Beispiel erzeugt zunächst durch die Option `-c` eine neue Datei namens *.htuserfile* mit dem Benutzer `webadmin`; anschließend wird in derselben Datei ein weiterer User namens `webuser` gespeichert:

```
# htpasswd -c .htuserfile webadmin
New password: **********
Re-type new password: **********
Adding password for user webadmin

# htpasswd .htuserfile webuser
New password: **********
Re-type new password: **********
Adding password for user webuser
```

Die erzeugte Datei wird mithilfe der Direktive `AuthUserFile` zur Authentifizierung herangezogen. Sie sollte nach Möglichkeit nicht innerhalb der `DocumentRoot` liegen, damit kein Unbefugter Zugriff darauf hat. Die einzige Ausnahme muss gemacht werden, wenn die Authentifizierungseinstellungen in einer *.htaccess*-Datei stehen. In diesem Fall sollten Sie die fragliche Datei separat schützen:

```
[...]
AuthUserFile .htuserfile
<Files .htuserfile>
  Order deny,allow
  Deny from all
</Files>
```

Noch praktischer ist es, wenn Sie sich grundsätzlich angewöhnen, solche Userdateien mit der Zeichenfolge *.ht* zu beginnen und den Zugriff auf solche Dateien global in der Hauptkonfigurationsdatei zu sperren:

```
<FilesMatch ^\.ht>
  Order deny,allow
  Deny from all
</FilesMatch>
```

### 9.2.2 Direktiven zur textdateibasierten Basic-Authentifizierung

Nachdem Sie nun grundsätzlich erfahren haben, wie die Basic-Authentifizierung über Textdateien funktioniert, werden hier die entsprechenden Konfigurationsdirektiven der Module `mod_auth_basic`, `mod_authn_file` und `mod_authz_groupfile` im Einzelnen beschrieben.

#### AuthBasicProvider

Auswahl der Provider-Module für die Basic-Authentifizierung

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_auth_basic` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthBasicProvider Provider [Provider ...]` |
| **Standardwert** | `file` |

Mithilfe dieser Direktive wird angegeben, welche Provider-Module als Quelle für Benutzerdaten zur Basic-Authentifizierung herangezogen werden. Die Voreinstellung `file` entspricht dem Modul `mod_authz_file`. Einige andere mögliche Werte sind:

- `dbm` – Anmeldedaten aus DBM-Dateien mit `mod_authn_dbm`
- `ldap` – Benutzerdaten aus einem LDAP-Verzeichnis (`mod_authnz_ldap`)
- `dbd` – Liest die Daten aus einer mittels `mod_dbd` eingebundenen SQL-Datenbank; als Provider dient das Modul `mod_authn_dbd`.

- `anon` – Anonymous-Authentifizierung mit E-Mail-Adresse als Passwort (`mod_authn_anon`)

Das folgende Beispiel legt LDAP als Provider fest:

```
AuthBasicProvider ldap
```

Hier noch ein Beispiel, das zunächst eine DBM-Datei verwendet und notfalls auf eine Textdatei zurückgreift:

```
AuthBasicProvider dbm file
```

#### AuthUserFile

Name der Textdatei mit Benutzernamen und Passwörtern

| | |
|---|---|
| **Modul** | `mod_authz_file` (bis 2.0.x: `mod_auth`) |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthUserFile Dateipfad` |
| **Standardwert** | nicht gesetzt |

Mit dieser Direktive wird der Pfad der Textdatei mit den Benutzernamen und Passwörtern angegeben, aus denen `Require user` schöpfen kann. Der Pfad der Datei kann absolut oder relativ angegeben werden; wie bei vielen Direktiven üblich, werden relative Angaben relativ zur `ServerRoot` interpretiert. Wie bereits erwähnt, sollte diese Datei nicht innerhalb der `DocumentRoot` liegen. Falls sich dies nicht vermeiden lässt, weil Sie auf *.htaccess*-Dateien angewiesen sind und keinen Zugriff auf die Server-Konfiguration haben, muss sie durch eine `Deny`-Anweisung vor sämtlichen Client-Zugriffen geschützt werden, z.B.:

```
AuthUserFile /usr/local/apache2/misc/.htusers
```

Der Aufbau der Datei wurde weiter oben bereits erläutert: Jede Zeile enthält eine `Benutzername:Passwort`-Kombination. Sie können diese Dateien leicht mit dem Kommandozeilen-Tool `htpasswd` verwalten.

#### AuthGroupFile

Name der Textdatei mit Benutzergruppen

| | |
|---|---|
| **Modul** | `mod_authz_groupfile` (bis 2.0.x: `mod_auth`) |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthGroupFile Dateipfad` |
| **Standardwert** | nicht gesetzt |

Die Direktive `AuthGroupFile` gibt den Pfad einer Datei an, in der Gruppenzugehörigkeiten verwaltet werden. Manchmal gibt es Verzeichnisse beziehungsweise URL-Bereiche, auf die viele verschiedene Benutzer zugreifen dürfen, aber eben nicht jeder. Damit Sie in solchen Fällen nicht jedes Mal `Require user` mit unzähligen Benutzern verwenden müssen, besteht die Möglichkeit, die entsprechenden User zu Gruppen zusammenzufassen – das Verfahren dürfte Ihnen aus fast jedem Betriebssystem vertraut sein.

Im Gegensatz zu den Benutzerdateien gibt es kein spezielles Tool, um Gruppendateien anzulegen. Das ist auch nicht nötig, weil das Format einer solchen Datei sehr einfach ist und mit jedem Texteditor erzeugt werden kann. Jede Zeile enthält einen Gruppennamen und – durch einen Doppelpunkt abgetrennt – eine Liste von Benutzern, die der genannten Gruppe angehören sollen. Beispiel:

```
# .htgroups - site access control group list
admins: htadmin webmaster
staff: heinz jutta klaus nicole
```

Jeder Benutzername darf übrigens in beliebig vielen Gruppen vorkommen; für die Anmeldung genügt es, dass `Require group` eine der Gruppen nennt, denen er angehört.

Sollen nun z. B. alle Benutzer der Gruppe `staff` das Recht haben, auf URLs unter `/test` zuzugreifen, lässt sich dies leicht folgendermaßen bewirken:

```
<Location /test>
  AuthType Basic
  AuthUserFile /usr/local/apache2/misc/.htusers
  AuthGroupFile /user/local/apache2/misc/.htgroups
  Require group staff
</Location>
```

Nun können sich die User `heinz`, `jutta`, `klaus` und `nicole` jeweils mit ihrem Benutzernamen und dem ihnen zugewiesenen Passwort anmelden.

#### AuthBasicAuthoritative

Legt fest, ob die Basic-Authentifizierung an andere Module weitergereicht wird.

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_auth_basic` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthAuthoritative On\|Off` |
| **Standardwert** | `On` |

Die Direktive AuthBasicAuthoritative regelt eine eventuelle Weitergabe der Anmeldeinformationen an untergeordnete Module. Dies geschieht, sobald Sie den Wert Off einstellen. Da Sie durch das Provider-Verfahren die Reihenfolge der Datenquellen selbst wählen können, besteht in aller Regel kein Bedarf, dies zu ändern.

#### AuthzUserAuthoritative

Delegation der Benutzerzugriffskontrolle an nachgeordnete Module

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | mod_authz_user |
| **Kontext** | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| **Syntax** | AuthAuthoritative On\|Off |
| **Standardwert** | On |

Neben den Benutzeranmeldedaten lässt sich auch die Benutzerzugriffskontrolle bei Bedarf an weitere Module übergeben. Dazu müssen Sie diese Direktive auf den Wert Off setzen.

#### AuthzGroupFileAuthoritative

Überprüfung der Gruppendaten durch untergeordnete Module

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | mod_authz_groupfile |
| **Kontext** | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| **Syntax** | AuthAuthoritative On\|Off |
| **Standardwert** | On |

Auch die Gruppenzugriffskontrolle lässt sich bei Bedarf durch weitere Module überprüfen, sofern AuthGroupFile keine gültige Gruppe findet. Dazu müssen Sie AuthzGroupFileAuthoritative auf Off setzen. Normalerweise besteht dazu keine Veranlassung.

## 9.3 Digest-Authentifizierung

Eine Schwachstelle der Basic-Authentifizierung ist ganz klar die Tatsache, dass die Anmeldedaten im Klartext über das Internet geschickt werden. Genauer gesagt, werden sie base64-codiert übertragen, damit das Passwort beliebige Sonderzei-

chen enthalten kann – ein Beispiel, wie dies hinter den Kulissen aussieht, finden Sie in Kapitel 2, »Funktionsweise von Webservern«.

Erheblich sicherer ist die *Digest-Authentifizierung*: Bei dieser Variante werden keine Klartextinformationen übertragen, sondern ein MD5-Hash, der aus den Anmeldedaten erzeugt wird. Genauer gesagt, sendet der Server eine genau einmal gültige Zeichenfolge namens *Nonce* an den Client. Dieser verschlüsselt die Anmeldedaten damit und sendet das Ergebnis zurück. Die Originaldaten lassen sich daraus nicht mehr ermitteln, sodass man von einer sicheren Authentifizierungsmethode sprechen kann. Das einzige Problem besteht darin, dass nicht nur der Webserver, sondern auch der Client diese Methode unterstützen muss. Das ist leider noch immer nicht bei allen Browsern der Fall, aber zumindest aktuelle Versionen der folgenden Browser können mit der Digest-Authentifizierung umgehen:

- Mozilla Firefox
- Google Chrome
- Microsoft Internet Explorer
- Opera
- Apple Safari

Der Internet Explorer kommt nur dann mit der Digest-Authentifizierung zurecht, wenn die URL einer `GET`-Anfrage keinen Query-String (Formulardaten nach einem Fragezeichen) enthält. Für echte Webformulare besteht die einfachste Lösungsmöglichkeit in der Verwendung der Versandmethode `POST`. Falls Sie dagegen in einer Webanwendung auf das manuelle Erstellen eines Query-Strings in Zusammenhang mit der Digest-Authentifizierung angewiesen sind, können Sie seit Apache 2.0.51 die spezielle Umgebungsvariable `AuthDigestEnableQueryStringHack` setzen:

```
BrowserMatchNoCase "msie" AuthDigestEnableQueryStringHack=On
```

`BrowserMatchNoCase` vergleicht den Wert des Anfrage-Headers `User-Agent`, also die Selbstidentifikation des Client-Browsers, ohne Berücksichtigung von Groß- und Kleinschreibung mit dem angegebenen String-Wert. Falls dieser in dem Header vorkommt, erfolgt die als Nächstes angegebene Variablendefinition. Näheres über diese und andere Konfigurationsdirektiven für Umgebungsvariablen erfahren Sie in Kapitel 14, »CGI«.

Für die Digest-Authentifizierung wird eine spezielle Datei mit Benutzernamen und Passwörtern verwendet. Sie wird – je nach eingestelltem Provider – mithilfe der bereits behandelten Direktive `AuthUserFile` oder mittels `AuthDBMUserFile`

(siehe Abschnitt 9.4, »Benutzer- und Passwortverwaltung in DBM-Dateien«) eingebunden.

### 9.3.1 Das Tool htdigest

Zum Lieferumfang von Apache 2 gehört ein Programm namens `htdigest`, das zur Erzeugung von Textdateien mit Digest-Benutzerdaten dient. Es funktioniert ähnlich wie `htpasswd` und erzeugt ebenfalls Textdateien, in denen die Anmeldeinformationen gespeichert werden. In einer solchen Datei hat jede Zeile das folgende Format:

```
Username:Realm:Passwort
```

In `htdigest` gibt es erheblich weniger Optionen als bei `htpasswd`, weil es keine unterschiedlichen Arten der Verschlüsselung gibt, in denen die Daten gespeichert werden könnten. Die Syntax des Befehls ist daher folgende:

```
# htdigest [-c] Benutzerdatei Realm Benutzername
```

- `-c`: Wenn Sie diese zusätzliche Option angeben, wird die Datei neu erstellt. Falls sie bereits existiert, werden ihre bisherigen Inhalte vernichtet.
- `Benutzerdatei`: Das ist der Pfad der Datei, in der die Anmeldedaten gespeichert werden. Wenn Sie einen relativen Pfad angeben, wird er zur `ServerRoot` in Beziehung gesetzt. Es sollte noch einmal betont werden, dass auch diese Datei nichts innerhalb der `DocumentRoot` verloren hat.
- `Realm`: Der Authentifizierungsbereich (Realm) muss mit angegeben werden, weil er in die Digest-Berechnung mit einbezogen wird. Soll sich ein Benutzer in verschiedenen Realms anmelden können, müssen Sie mehrere Einträge für ihn erzeugen.
- `Benutzername`: Hier wird wie üblich der Benutzername angegeben. Falls die verwendete Kombination aus Benutzernamen und Realm bereits existiert, wird sie überschrieben. Andernfalls wird ein neuer Eintrag erzeugt.

Das folgende Beispiel zeigt zwei Aufrufe von `htdigest`. Bei der ersten Ausführung wird die Datei *.htdigest* neu angelegt, beim zweiten Mal wird ein weiterer Benutzer darin gespeichert:

```
# htdigest -c /usr/local/apache2/misc/.htdigest \
"Top Secret" htadmin
Adding password for htadmin in realm Top Secret.
New password: ******
Re-type new password: ******

# htdigest /usr/local/apache2/misc/.htdigest \
```

```
"Top Secret" htmaster
Adding password for htmaster in realm Top Secret.
New password: ******
Re-type new password: ******
```

Die entsprechende Datei sieht nach der Ausführung der beiden Befehle etwa so aus:

```
htadmin:Top Secret:
        146715483de2df7c16642ab140c040ebee1222268245e7
htmaster:Top Secret:d6ee3dc100c555835c4c8b3b09f7d3b9
```

Auch für die Digest-Authentifizierung können Sie übrigens Gruppendateien erstellen. Diese werden mit `AuthGroupFile` (oder, je nach Provider-Modul, auch mit anderen Direktiven) eingebunden.

### 9.3.2 Direktiven zur Digest-Authentifizierung

In diesem Abschnitt werden die Konfigurationsdirektiven des Moduls `mod_auth_digest` beschrieben.

#### AuthDigestProvider

Provider-Modul für die Datenquelle der Digest-Authentifizierung

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_auth_digest` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthDigestProvider Provider [Provider ...]` |
| **Standardwert** | `file` |

Diese Direktive wählt das Provider-Modul aus, das die Vergleichsdaten für die Überprüfung der Digest-Authentifizierung liefern soll. Syntax und Funktionalität entsprechen der weiter oben behandelten Direktive für die Basic-Authentifizierung. Als Digest-Provider kommen nur drei der zum Lieferumfang gehörenden Provider-Module infrage:

- `mod_authn_file` – aktiviert durch den Wert `file` (Standard)
- `mod_authn_dbm` – Die zugehörige Provider-Angabe lautet `dbm`.
- `mod_authn_dbd` – mit dem Providernamen `dbd`

Die Dateien mit Benutzerinformationen werden entsprechend mittels `AuthUserFile` beziehungsweise der in Abschnitt 9.4, »Benutzer- und Passwort-

verwaltung in DBM-Dateien«, behandelten Direktive `AuthDBMUserFile` angegeben.

Das folgende Gesamtbeispiel verwendet die weiter oben mittels `htdigest` erzeugte Textdatei:

```
<Directory /usr/local/apache2/htdocs/topsecret>
  AuthType Digest
  AuthDigestProvider file
  AuthName "Top Secrect"
  AuthUserFile /usr/local/apache2/misc/.htdigest
  Require user htadmin htmaster
</Directory>
```

#### AuthDigestAlgorithm

Zu verwendender Algorithmus für Digest-Authentifizierung

| | |
|---|---|
| **Modul** | `mod_auth_digest` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthDigestAlgorithm MD5 \| MD5-sess` |
| **Standardwert** | `MD5` |

Diese Direktive gibt das Verfahren an, nach dem die Digest-Werte für die HTTP-Anfragen und -Antworten berechnet werden. Der in Entwicklung befindliche `MD5-sess`-Algorithmus arbeitet Session-basiert – die Werte aufeinanderfolgender HTTP-Transaktionen beeinflussen einander gegenseitig. Zurzeit müssen Sie noch die Voreinstellung `MD5` beibehalten, da es noch keine korrekt arbeitende `MD5-sess`-Implementierung gibt; das geht durch schlichtes Weglassen der Direktive am einfachsten.

#### AuthDigestDomain

Legt einen gemeinsamen Digest-Authentifizierungsbereich fest.

| | |
|---|---|
| **Modul** | `mod_auth_digest` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthDigestDomain URI [URI ...]` |
| **Standardwert** | nicht gesetzt |

Sie können eine oder mehrere URIs beziehungsweise vollständige URLs angeben, für die dieselben Anmeldedaten (Benutzername und Realm) gelten sollen.

Es ist empfehlenswert, diese Angabe selbst dann zu machen, wenn alle URLs, für die die entsprechenden Authentifizierungsinformationen gelten, unter einer gemeinsamen Verzeichniswurzel liegen. Dies hindert den Client nämlich daran, die Daten auch dann zu berechnen und mitzuschicken, wenn ungeschützte Ressourcen auf demselben Server aufgerufen werden.

Umgekehrt können Sie als Argumente für `AuthDigestDomain` sogar URLs angeben, die auf unterschiedlichen Hosts liegen – manche Browser unterstützen dies und fragen bei einem Wechsel auf einen anderen der angegebenen Server nicht erneut nach Benutzername und Passwort. Beispiele:

```
AuthDigestDomain /topsecret
AuthDigestDomain /topsecret /info/topsecret
AuthDigestDomain http://www.mynet.de/secret \
   http://order.mynet.de/private
```

Im Gesamtzusammenhang sieht die Verwendung dieser Direktive beispielsweise so aus:

```
<Directory /usr/local/apache2/htdocs/topsecret>
  AuthType Digest
  AuthDigestProvider file
  AuthName "Top Secrect"
  AuthDigestDomain /topsecret
  AuthUserFile /usr/local/apache2/misc/.htdigest
  Require user htadmin htmaster
</Directory>
```

#### AuthDigestNonceLifetime

Geltungsdauer des Nonce-Werts

| | |
|---|---|
| **Modul** | `mod_auth_digest` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthDigestNonceLifetime Sekunden` |
| **Standardwert** | `300` |

Diese Direktive legt fest, wie lange der vom Server gesendete Nonce-Wert gültig bleibt. Wenn die Anmeldung des Clients erst nach Ablauf dieser Zeit eintrifft, akzeptiert der Server sie nicht mehr und sendet erneut eine Meldung mit dem Statuscode `401 Authorization Required`. Ein kleinerer Wert verbessert zwar die Sicherheit, kann aber gleichzeitig dazu führen, dass die Anmeldedaten nicht rechtzeitig eintreffen. Deshalb sollten Sie in der Regel keinen kleineren Wert als 10 Sekunden angeben. Ein negativer Wert bedeutet übrigens, dass der Nonce keine maximale Lebensdauer besitzen soll.

### AuthDigestShmemSize

Größe des Shared-Memory-Bereichs für Digest-Authentifizierung

| Modul | mod_auth_digest |
|---|---|
| Kontext | Server |
| Syntax | AuthDigestShmemSize Größe |
| Standardwert | 1000 |

mod_auth_digest verwendet ein Shared-Memory-Segment, um Informationen über einen Client über mehrere Anfragen hinweg zu speichern. Mit der Direktive AuthDigestShmemSize, die im Server-Kontext stehen muss und somit für sämtliche Einstellungen zur Digest-Authentifizierung insgesamt gilt, wird die Größe dieses Speicherbereichs angegeben. Wenn Sie eine einfache Zahl verwenden, wird sie als Byte-Wert interpretiert. Alternativ können Sie einen der Buchstaben K oder M hinter die Zahl schreiben, damit sie als Kilobyte beziehungsweise Megabyte gilt. Beispiele:

```
AuthDigestShmemSize 10000
AuthDigestShmemSize 128K
AuthDigestShmemSize 4M
```

Um herauszufinden, wie groß der Wert auf Ihrem System sein muss, können Sie ihn auf 0 setzen und Apache neu starten – der Start schlägt fehl, aber eine Fehlermeldung gibt die erforderliche Größe für *einen* Client an.

### AuthDigestQop

Quality-of-Protection bei der Digest-Authentifizierung

| Modul | mod_auth_digest |
|---|---|
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | AuthDigestQop none \| auth \| auth-int [auth \| auth-int] |
| Standardwert | auth |

Diese Direktive sorgt dafür, dass Sie die Wahl zwischen verschiedenen konkreten Digest-Authentifizierungsverfahren haben: auth überprüft nur die Anmeldedaten, während auth-int zusätzlich einen MD5-Hash des Body-Inhalts berechnet, um dessen Integrität zu prüfen. Wenn beide Werte angegeben werden, wird die Auswahl dem Browser überlassen. Für ältere, inkompatible Clients gibt es auch noch den Wert none, der für eine Digest-Authentifizierung gemäß RFC 2069 sorgt. Beachten Sie, dass auth-int zurzeit noch nicht implementiert ist.

#### AuthDigestNcCheck

Legt fest, ob der Nonce-Zähler des Servers überprüft wird.

| Modul | mod_auth_digest |
|---|---|
| Kontext | Server |
| Syntax | AuthDigestNcCheck On\|Off |
| Standardwert | Off |

Diese Direktive ist vorgesehen, aber noch nicht implementiert. Sie soll in Zukunft dafür sorgen, dass überprüft wird, ob der Nonce bereits für einen Anmeldeversuch verwendet wurde. Dies schließt aus, dass ein Cracker versuchen könnte, sich mit gestohlenen Anmeldedaten Zugang zu verschaffen.

#### AuthDigestNonceFormat

Bestimmt, wie der Nonce-Count berechnet wird.

| Modul | mod_auth_digest |
|---|---|
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | AuthDigestNonceFormat Format |
| Standardwert | nicht gesetzt |

Auch diese Direktive ist noch nicht implementiert. Da RFC 2617 nicht vorschreibt, welches Format der Nonce haben soll, soll sie in Zukunft die Auswahl eines gewünschten Formats ermöglichen.

## 9.4 Benutzer- und Passwortverwaltung in DBM-Dateien

Wenn Sie sehr viele Benutzerdaten verwalten müssen, wird die Speicherung in Textdateien sehr schnell ineffizient – im schlimmsten Fall muss Apache die gesamte Datei sequenziell durchgehen, um schließlich festzustellen, dass der angegebene Benutzername gar nicht existiert. Erheblich effizienter funktioniert die Verwaltung der Daten in DBM-Dateien, die Sie bereits in Kapitel 8, »Weiterleitungen und Indizes«, im Zusammenhang mit `mod_rewrite` kennengelernt haben: Da es sich um indizierte, datenbankähnliche Dateien handelt, kann das zuständige Provider-Modul `mod_authn_dbm` die Benutzerdaten sofort nachschlagen. Es kann sowohl zur Basic- als auch zur Digest-Authentifizierung herangezogen werden; eine der folgenden Direktiven aktiviert es:

```
# mod_authn_dbm zur Basic-Authentifizierung:
AuthBasicProvider dbm
```

```
# mod_authn_dbm zur Digest-Authentifizierung:
AuthDigestProvider dbm
```

Hinzu kommt das Modul `mod_authz_dbm`, das auch Gruppendateien im DBM-Format verwalten kann.

Wie bereits in Kapitel 8, »Weiterleitungen und Indizes«, im Abschnitt über `mod_rewrite` beschrieben wurde, gibt es unterschiedliche konkrete Typen von DBM-Dateien wie SDBM, GDBM und NDBM. Welche Typen auf Ihrem System zur Verfügung stehen, hängt davon ab, wie Ihr Apache kompiliert wurde – die entsprechenden Optionen werden in Kapitel 4, »Apache kompilieren und installieren«, erläutert.

### 9.4.1 Das Tool dbmmanage

Zur Erstellung von DBM-Benutzerdateien wird mit Apache ein Perl-Skript namens `dbmmanage` geliefert. Ähnlich wie `htpasswd` oder `htdigest` ermöglicht es die einfache Verwaltung der Benutzerdaten. Unter UNIX ist es als ausführbar gekennzeichnet und kann einfach mit seinem Namen aufgerufen werden. Auf Windows-Systemen wird es dagegen mit `perl dbmmanage.pl` ausgeführt. Selbstverständlich muss auf beiden Systemen Perl installiert sein. Das Skript benötigt das Perl-Modul `Crypt::PasswdMD5`, das zumindest unter Windows nicht zum Lieferumfang von Perl gehört. Wenn Sie ActivePerl für Windows verwenden und eine direkte Internetverbindung besteht, lässt es sich einfach folgendermaßen herunterladen und installieren:

```
> ppm install Crypt::PasswdMD5
```

Falls Sie über einen Proxy-Server auf das Web zugreifen, müssen Sie vor diesem Befehl die Umgebungsvariable `HTTP_PROXY` auf dessen IP-Adresse und Port setzen. Beispiel:

```
> set HTTP_PROXY=http://192.168.0.100:3128
```

Bei anderen Perl-Versionen oder auf Nicht-Windows-Systemen müssen Sie das Modul möglicherweise selbst aus dem CPAN herunterladen und manuell installieren.

Das Skript `dbmmanage` kann mit einer der folgenden drei Syntaxvarianten aufgerufen werden:

```
# dbmmanage [Codierung] Dateiname \
add|adduser|check|delete|update Username \
[Passwort [Gruppe[,Gruppe...] [Kommentar]]]
# dbmmanage Dateiname view [Username]
# dbmmanage Dateiname import
```

Die Befehle der ersten Variante (`add`, `adduser` usw.) führen jeweils eine Änderung an einem Datensatz der DBM-Datei durch. Mithilfe des Befehls `view` können Sie den Inhalt der angegebenen Datei betrachten, optional beschränkt auf einen bestimmten Usernamen. `import` ermöglicht das Einlesen von Benutzerdaten über die Standardeingabe.

Im Einzelnen bedeuten die verschiedenen Parameter Folgendes:

- `Dateiname`
  Der Basisdateiname der DBM-Datei – in der Regel ohne Dateiendung, da diese je nach konkreter DBM-Variante anders lauten kann
- `Username`
  Der Benutzer, auf den sich der Befehl bezieht
- `Passwort`
  Wenn Sie in der Kommandozeile ein Passwort eingeben, muss es bereits mit einem externen Hilfsprogramm verschlüsselt worden sein. Andernfalls können Sie ein Minuszeichen (`-`) angeben, um separat nach dem Passwort gefragt zu werden. Zusammen mit dem Befehl `update` kann auch ein Punkt (`.`) verwendet werden, um das ursprüngliche Passwort beizubehalten.
- `Gruppe`
  Sie können eine oder (durch Komma ohne Leerzeichen getrennt) mehrere Gruppen angeben, denen der User angehören soll. Wenn der Benutzer keiner speziellen Gruppe angehören soll, Sie aber dennoch das nächste Feld (Kommentar) ausfüllen möchten, müssen Sie hier den Wert `-` verwenden. Beim `update`-Befehl können Sie einen Punkt (`.`) benutzen, wenn Sie die Gruppenzugehörigkeit nicht ändern möchten.
- `Kommentar`
  Zu Ihrer eigenen Information können Sie an dieser Stelle einen beliebigen Kommentar über den User angeben – z.B. den Realnamen oder die E-Mail-Adresse.

Für die Codierung, das heißt die Verschlüsselungsart, können Sie folgende Werte angeben:

- `-d`: `crypt`-Verschlüsselung (Standard auf UNIX-Systemen)
- `-m`: MD5-Verschlüsselung (Standard auf Windows- und NetWare-Systemen)
- `-s`: SHA1-Verschlüsselung
- `-p`: Klartext (nicht zu empfehlen)

Wie schon bei `htpasswd` empfiehlt sich aus Gründen der einfachen Übertragbarkeit das MD5-Format.

Die einzelnen Befehle bedeuten Folgendes:

- `add`
  Ein neuer Eintrag für den angegebenen User wird erzeugt; dazu wird das verschlüsselte Passwort aus der Kommandozeile benutzt.
- `adduser`
  Auch dieser Befehl fügt einen Datensatz für einen neuen Benutzer hinzu. Im Unterschied zu `add` werden Sie aber nach dem Passwort gefragt.
- `check`
  Sie werden nach dem Passwort gefragt. Anschließend überprüft der Befehl, ob der angegebene User existiert und ob das Passwort korrekt ist.
- `delete`
  Dieser Befehl löscht den angegebenen Benutzer aus der DBM-Datei.
- `import`
  Dieser Befehl liest beliebig viele `Benutzername:Passwort`-Paare aus der Standardeingabe. Die Passwörter müssen bereits verschlüsselt sein. Der eigentliche Zweck dieses Befehls ist allerdings nicht etwa die manuelle Eingabe, sondern der Import von Daten aus einer bestehenden *htpasswd*-Textdatei nach folgendem Schema:

  ```
  # dbmmanage /usr/local/apache2/misc/.htdb \
  </usr/local/apache2/misc/.htusers
  ```

- `update`
  Dieser Befehl ändert die Daten für einen bereits bestehenden Benutzer. Existiert noch kein Eintrag für den Benutzer, wird das Kommando nicht ausgeführt.
- `view`
  Mit diesem Befehl können Sie sich den Inhalt der DBM-Datei anzeigen lassen. Optional können Sie einen Usernamen angeben, um nur einen Datensatz zu sehen.

Das folgende Beispiel verwendet eine Datei namens `.htdb`. Zuerst werden die beiden Benutzer `webadmin` und `webuser` mit MD5-verschlüsselten Passwörtern (Option `-m`) erzeugt, anschließend wird mit dem Befehl `view` das Ergebnis angezeigt:

```
# dbmmanage -m .htdb adduser webadmin
New password:IchDarfDas
Re-type new password:IchDarfDas

User webadmin added with password encrypted to
$apr1$4J3Y1DEP$xstTFQOqnVu.ZWhM5FSyg. using md5
```

```
# dbmmanage -m .htdb adduser webuser
New password:IchDarfAuch
Re-type new password:IchDarfAuch

User webuser added with password encrypted to
$apr1$5Se7H29v$AsOmsq5OM4aMM2.3YrGWK1 using md5

# dbmmanage .htdb view
webadmin:$apr1$4J3Y1DEP$xstTFQOqnVu.ZWhM5FSyg.
webuser:$apr1$5Se7H29v$AsOmsq5OM4aMM2.3YrGWK1
```

Beachten Sie, dass Sie das Programm unbeobachtet verwenden sollten – da ein Perl-Skript nicht ohne Weiteres auf eine erweiterte Terminal-Funktionalität zugreifen kann, ist die Passworteingabe hier sichtbar.

### 9.4.2 Das Programm htdbm

Eine modernere Alternative zu `dbmmanage` ist das mit neueren Apache-Versionen mitgelieferte Hilfsprogramm `htdbm`. Im Gegensatz zu `dbmmanage` ist es aber kein Perl-Skript, sondern ein Binary.

Das Programm wird mit folgender Grundsyntax aufgerufen:

```
# htdbm [Optionen] Datei [Benutzername [Passwort]]
```

Folgende Optionen sind definiert:

- `-b`: Passwort von der Kommandozeile lesen, anstatt danach zu fragen
- `-c`: Mit dieser Option wird eine neue Datenbank angelegt beziehungsweise eine bestehende ersetzt.
- `-n`: Diese Option bestimmt, dass der Befehl nur zu Testzwecken ausgeführt wird: Die Datenbank selbst wird nicht geändert; das Ergebnis wird lediglich angezeigt.
- `-m`: MD5-Verschlüsselung verwenden (Standard unter Windows und NetWare)
- `-p`: Das Passwort soll im Klartext gespeichert und nicht verschlüsselt werden. Dies ist nicht zu empfehlen.
- `-s`: Das Passwort wird SHA1-verschlüsselt.
- `-T<Datenbanktyp>`: Dies gibt den konkreten DBM-Typ an. Mögliche Werte sind `SDBM`, `GDBM`, `DB` und `default` (Systemvorgabe verwenden).
- `-l`: Diese Option gibt die Benutzernamen aus.

- `-t`: Wenn Sie diese Option angeben, wird das letzte Argument als Kommentar gewertet.
- `-v`: Keine Neueingabe, stattdessen sollen Benutzername und Passwort überprüft werden.
- `-x`: Der angegebene Benutzer wird aus der DBM-Datei gelöscht.

Das folgende Beispiel zeigt, wie sich das `dbmmanage`-Beispiel von oben mit `htdbm` realisieren lässt:

```
# htdbm -c -TSDBM .htdb webadmin
Enter password        : **********
Re-type password      : **********
Database .htdb created.

# htdbm .htdb webuser
Enter password        : ***********
Re-type password      : ***********
Database .htdb modified.
```

### 9.4.3 DBM-Direktiven

In diesem Abschnitt finden Sie die gewohnte genaue Beschreibung sämtlicher Direktiven der Module `mod_authn_dbm` und `mod_authz_dbm`.

#### AuthDBMUserFile

Name der DBM-Datei mit Benutzernamen und Passwörtern

| | |
|---|---|
| **Modul** | `mod_authn_dbm` (bis 2.0.x: `mod_auth_dbm`) |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthDBMUserFile Dateipfad` |
| **Standardwert** | nicht gesetzt |

Mit der Direktive `AuthDBMUserFile` wird die Datei angegeben, in der die Benutzernamen und Passwörter der Benutzer stehen, die Zugriff auf das geschützte Verzeichnis erhalten sollen. Es handelt sich in diesem Fall um DBM-Dateien, die mithilfe der oben vorgestellten Tools `dbmmanage` oder `htdbm` erstellt wurden. Ein relativer Pfad bezieht sich wie gehabt auf die `ServerRoot`. Beispiel:

```
AuthDBMUserFile /usr/local/apache2/misc/.htdb
```

Hier ein komplettes Beispiel für die DBM-basierte Basic-Authentifizierung eines Benutzers:

```
<Location /geheim>
  AuthType Basic
  AuthBasicProvider dbm
  AuthName "DB Users Only"
  AuthDBMUserFile /usr/local/apache2/misc/.htdb
  Require user webadmin
</Location>
```

### AuthDBMGroupFile

Name der DBM-Datei mit Benutzergruppen

| | |
|---|---|
| **Modul** | `mod_authz_dbm` (bis 2.0.x: `mod_auth_dbm`) |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthDBMGroupFile Dateipfad` |
| **Standardwert** | nicht gesetzt |

Anders als bei den bisher besprochenen Authentifizierungsmodulen besitzen Benutzer- und Gruppendatei bei `mod_auth_dbm` dasselbe Format – zum Eintrag für einen Benutzer in der Passwortdatei gehören auch die Gruppen, denen er angehört. Wie die Gruppenzugehörigkeit eines Users mithilfe der Tools `dbmmanage` oder `htdbm` angegeben werden kann, wurde weiter oben bereits beschrieben. In aller Regel wird deshalb sogar dieselbe Datei für Benutzer- und Gruppendefinitionen angegeben. Beispiel:

```
AuthDBMGroupFile /usr/local/apache2/misc/.htdb
```

Zur Gänze sieht die Anforderung zur DBM-basierten Authentifizierung einer Gruppe beispielsweise so aus:

```
<Location /geheim>
  AuthType Basic
  AuthBasic Provider dbm
  AuthName "DB Users Only"
  AuthDBMUserFile /usr/local/apache2/misc/.htdb
  AuthDBMGroupFile /usr/local/apache2/misc/.htdb
  Require group webusers
</Location>
```

### AuthDBMType

Art der DBM-Datei für Passwörter

| | |
|---|---|
| **Seit Version** | 2.0.30 |
| **Modul** | `mod_authn_dbm` (bis 2.0.x: `mod_auth_dbm`) |

| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
|---|---|
| Syntax | AuthDBMType Default \| SDBM \| GDBM \| NDBM \| DB |
| Standardwert | Default |

Mit dieser Direktive können Sie bei Bedarf den Typ der DBM-Datei festlegen, die durch AuthDBMUserFile (in Apache 2.0 auch AuthDBMGroupFile) eingebunden wurde. Das ist in der Regel nur erforderlich, wenn Ihre Apache-Installation mehr als eine Sorte von DBM-Dateien unterstützt. Dies wird beim Kompilieren festgelegt (siehe Kapitel 4, »Apache kompilieren und installieren«). Solange die DBM-Authentifizierung ohne diese Direktive funktioniert, können Sie sie natürlich weglassen. *Wenn* Sie sie angeben, müssen Sie allerdings sicherstellen, dass der Wert auch tatsächlich der verwendeten DBM-Datei entspricht. Beispiel:

```
AuthDBMType SDBM
```

#### AuthzDBMType

Art der DBM-Datei für Gruppendaten

| Seit Version | 2.1-beta |
|---|---|
| Modul | mod_authz_dbm |
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | AuthzDBMType Default \| SDBM \| GDBM \| NDBM \| DB |
| Standardwert | Default |

Da Apache 2.2 die Funktionen für DBM-Gruppendateien durch ein eigenes Modul implementiert, wurde auch diese separate Direktive definiert, die das Format dieser Dateien festlegt. Syntax und Funktionsweise entsprechen AuthDBMType. Hier ein Beispiel:

```
AuthzDBMType GDBM
```

#### AuthzDBMAuthoritative

Legt fest, ob Gruppendaten durch andere Module überprüft werden.

| Seit Version | 2.1-beta |
|---|---|
| Modul | mod_authz_dbm |
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | AuthzDBMAuthoritative On\|Off |
| Standardwert | On |

Diese Direktive funktioniert analog zu den anderen Auth*Authoritative-Direktiven: Gruppenanmeldedaten werden von mod_authz_dbm an nachgeordnete

Module weitergereicht, sofern Sie explizit den Wert `Off` einstellen und in den verfügbaren Gruppen kein gültiger Benutzer gefunden wird.

Wenn `mod_authn_dbm` als Provider-Modul für `mod_auth_basic` dient, wird die entsprechende Funktion auf Userebene von `AuthBasicAuthoritative` übernommen.

## 9.5 LDAP-Authentifizierung

Alle bisher gezeigten Authentifizierungsmethoden haben einen gemeinsamen Nachteil: Sie müssen völlig neue Benutzer- und Gruppendefinitionen erzeugen, um sie zu nutzen. In vielen Unternehmen mit Netzwerken existiert aber bereits eine Infrastruktur mit solchen Definitionen, und es ist aufwendig und fehlerträchtig, diese noch einmal für den Website-Zugriff nachzuahmen.

Heutzutage verwenden immer mehr Firmen eine Lösung für ihre unternehmensweite Authentifizierungsverwaltung, die man als *Verzeichnisdienst* bezeichnet. Dabei handelt es sich um einen Server-Dienst, der die Daten sämtlicher Netzwerke, Rechner, Benutzer und Gruppen des Unternehmens speichert und bei jeder Netzwerkanmeldung überprüft, ob die aktuelle Rechner-Benutzer-Kombination die erforderliche Berechtigung besitzt. Ein Verzeichnisdienst ist praktisch, weil er einmal zentral eingerichtet wird und mehr Leistungsfähigkeit und Sicherheit bietet als ein »Gestrüpp« aus lokalen Berechtigungen für einzelne Rechner.

Konkrete Verzeichnisdienste gibt es wie Sand am Meer: NIS beziehungsweise NIS(+) von Sun (früher yp oder Yellow Pages genannt), Novell Directory Services (NDS) beziehungsweise eDirectory, OpenLDAP und Microsoft Active Directory sind nur einige von ihnen. Bis auf die Sun-Eigenentwicklung NIS(+) basieren all diese Verzeichnisdienste auf einer abgespeckten, aber TCP/IP-fähigen Variante der X.500-Spezifikation, auf die mithilfe von LDAP (Lightweight Directory Access Protocol) zugegriffen werden kann. Genau hier kommt `mod_authnz_ldap` ins Spiel: Es überlässt die Überprüfung der Anmeldedaten einem separaten LDAP-fähigen Verzeichnisdienstserver. Geschrieben wurde das Modul für die Zusammenarbeit mit OpenLDAP, es arbeitet aber im Grunde mit jedem LDAP-standardkonformen Server zusammen.

### 9.5.1 LDAP-Grundwissen

LDAP-Verzeichnisse speichern die Daten in einer DNS-ähnlichen, hierarchischen Baumstruktur, die als DIT (Directory Information Tree) bezeichnet wird. Die Speicherung selbst erfolgt binär, aber für den Datenaustausch mit Benutzern oder Programmen unterstützen LDAP-Server ein textbasiertes Format namens LDIF

(LDAP Interchange Format). LDIF-Einträge haben stets die Struktur `Name=Wert`; hinzu kommen fast immer Unterobjekte im selben Format sowie Attribute in der Schreibweise `Attributname: Wert`.

Die Namen der LDAP-Objekte und -Attribute bilden das standardisierte LDAP-Schema. Einträge in diesem Schema besitzen einen Namen und eine eindeutige Nummer; auch die erlaubten Arten der Verschachtelung sind im Schema festgelegt. Auf Ihren eigenen LDAP-Servern können Sie das Schema bei Bedarf erweitern.

Die Wurzel des LDAP-Baums bilden `dc`-Einträge; `dc` steht für *Domain Component*. Diese Einträge stellen die einzelnen Bestandteile des Domain-Namens dar, die normalerweise durch Punkte voneinander getrennt werden. So wäre die Wurzel eines LDAP-Verzeichnisses für die Domain *mynet.de* beispielsweise `dc=mynet`, `dc=de`. Alle `dc`-Einträge zusammen bilden den DN (Distinguished Name) des Wurzelknotens.

Die erste Unterteilungsstufe unterhalb der Wurzel sind die Organisationseinheiten (LDAP-Name `ou` für Organizational Units). Bekannte Organisationseinheiten sind etwa `ou=devices` für Computer und andere Netzwerkgeräte oder `ou=people` für Personen, insbesondere Benutzer des Netzwerks. Organisationseinheiten können beliebig weiter unterteilt werden. Unterhalb von `ou=devices` könnte es beispielsweise `ou=computers` und `ou=printers` geben; `ou=people` könnte dagegen zum Beispiel in die Unterorganisationseinheiten `ou=sales`, `ou=marketing` und `ou=accounting` für die entsprechenden Abteilungen unterteilt werden.

Benutzer werden durch `cn`-Knoten gekennzeichnet; `cn` steht für Common Name. Der Inhalt des Knotens sind meist Vor- und Nachname der Person, aber im Bedarfsfall (etwa bei Namensgleichheit oder verschiedenen Standorten) können weitere Unterscheidungsmerkmale hinzukommen. Der DN eines Benutzers in der Abteilung Entwicklung (`developers`) unter der Organisationseinheit `people` im LDAP-Verzeichnis der Domain *mynet.de* könnte z.B. so aussehen:

```
cn=Sascha Kersken,ou=developers,ou=people,dc=mynet,dc=de
```

Im klassischen LDAP-Schema gehören Personendaten zur Objektklasse `person`. Eine Objektklasse legt die zulässigen und/oder erforderlichen Attribute für jeden LDAP-Eintrag fest. In modernen Netzwerken ist die von `person` abgeleitete Objektklasse `inetOrgPerson` besser geeignet, da sie beispielsweise ein Standardattribut für die E-Mail-Adresse enthält.

LDAP-Verzeichnisse werden sehr häufig zur netzwerkweiten Verwaltung von Benutzerkonten und -logins eingesetzt. Auch für diesen Zweck gibt es unterschiedliche Objektklassen. In UNIX-Netzwerken kommt beispielsweise die Objektklasse `posixAccount` zum Einsatz, deren Attribute alle Einträge von */etc/*

*passwd* abbilden; entsprechend gibt es noch die Klasse `shadowAccount` für die */etc/shadow*-Felder. Für Logins von Windows-Rechnern aus gibt es schließlich die Objektklasse `sambaAccount`.

Zur Verwaltung von User-Logins im Netzwerk oder auch in einem geschützten Apache-Verzeichnis liegt es nahe, im DN der Userknoten `uid` (den Benutzernamen) anstelle von `cn` zu verwenden. Beispiel:

```
uid=sascha,ou=developers,ou=people,dc=mynet,dc=de
```

Jeder Knoten in einem LDAP-Verzeichnisbaum kann übrigens beliebig vielen Objektklassen gleichzeitig anhören. Sie müssen dann allerdings darauf achten, dass die Pflichtattribute aller beteiligten Klassen verwendet werden – diese überschneiden sich mitunter, besonders dann, wenn die Klassen voneinander abgeleitet werden.

Hier sehen Sie ein komplettes Beispiel für einen Usereintrag; er gehört den drei Objektklassen `person`, `inetOrgPerson` und `posixAccount` an:

```
dn: uid=sascha,ou=developers,ou=people,dc=mynet,dc=de
objectClass: person
objectClass: inetOrgPerson
objectClass: posixAccount
cn: Sascha Kersken
sn: Kersken
givenName: Sascha
l: Köln
o: MyNet GmbH
telephoneNumber: 0221-1234567
facsimileTelephoneNumber: 0221-1234568
mail: sk@lingoworld.de
uid: sascha
uidNumber: 1001
gidNumber: 100
userPassword: {MD5}WY1MIARhuBUiozKFZcJffA==
homeDirectory: /home/sascha
loginShell: /bin/bash
```

Hier die Bedeutung der einzelnen Attribute im Detail:

- `dn` (distinguished name): der komplette Pfad des Eintrags im LDAP-Verzeichnisbaum
- `objectClass`: Die Kategorie des Eintrags, die festlegt, welche Felder vorgeschrieben beziehungsweise zulässig sind; der Beispieleintrag gehört wie bereits erwähnt zu drei verschiedenen Klassen.

- `cn` (common name): Der eindeutige Eigenname des Knotens; bei Personen werden meist Vor- und Nachname verwendet.
- `sn` (surname): Nachname
- `givenName`: Vorname
- `l` (location): Wohnort
- `o` (organization): Firmen- beziehungsweise Institutionsname
- `telephoneNumber`: Telefonnummer der Person
- `facsimileTelephoneNumber`: Faxnummer der Person
- `mail`: E-Mail-Adresse
- `uid`: User-ID für die Benutzeranmeldung
- `uidNumber`: die numerische User-ID
- `gidNumber`: numerische ID der primären Gruppe dieses Benutzers
- `userPassword`: Zunächst steht in geschweiften Klammern der verwendete Verschlüsselungsalgorithmus, dahinter das entsprechend verschlüsselte Passwort selbst.
- `homeDirectory`: das Home-Verzeichnis des Users
- `loginShell`: die Shell, die dem Benutzer nach der Anmeldung zur Verfügung steht

Jeder Knoten kann beliebig viele untergeordnete Knoten enthalten. Der DN wird dazu um den `cn` (oder ein anderes eindeutiges Merkmal) des übergeordneten Objekts ergänzt. Wenn Sie den `inetOrgPerson`-Eintrag beispielsweise lieber als Unterknoten des User-Accounts speichern möchten, dann lautet dessen DN wie folgt:

```
dn: cn=Sascha Kersken,uid=sascha,ou=people,dc=mynet,dc=de
```

### 9.5.2 OpenLDAP einrichten und verwalten

Die meisten Linux- und UNIX-Versionen enthalten den OpenLDAP-Server bereits ab Werk, zumindest als optionales Paket. Falls er bei Ihnen nicht vorhanden ist, müssen Sie ihn von *http://www.openldap.org* herunterladen und selbst kompilieren. Für Windows-Systeme ist mittlerweile ein binärer Installer verfügbar. Es handelt sich hierbei zwar um die nicht mehr aktuelle Version 2.2, und der Installer ist als experimentell gekennzeichnet – in meinen Praxistests für dieses Buch und für das »IT-Handbuch für Fachinformatiker« (Galileo Press, Bonn 2011) machte er jedoch einen recht stabilen Eindruck. Sie können diese Version unter *http://download.bergmans.us/openldap/* herunterladen. Wenn Sie den Installer mit Standardeinstellungen ausführen, wird OpenLDAP automatisch als Dienst

eingerichtet. Nach der Installation sollten Sie das OpenLDAP-Verzeichnis zu Ihrem `PATH` hinzufügen, um von überall auf die LDAP-Kommandozeilen-Tools zugreifen zu können.

Der OpenLDAP-Server-Dienst heißt auf allen Systemen *slapd*. Seine Hauptkonfigurationsdatei *slapd.conf* befindet sich auf UNIX-Systemen im Verzeichnis */etc/openldap*; bei Windows liegt diese Datei im Verzeichnis der OpenLDAP-Installation selbst.

Um den Zugriff auf den LDAP-Server zu ermöglichen, müssen Sie die beiden Einträge `suffix` (die Wurzel des LDAP-Baums) und `rootdn` (DN des Verwaltungs-Users) anpassen. Üblicherweise erhält der `rootdn` den Common Name `Manager` oder `ldapadmin`; darauf folgen dieselben Domain-Komponenten wie beim `suffix`. Beispiel:

```
suffix        "dc=mynet,dc=de"
rootdn        "cn=Manager,dc=mynet,dc=de"
```

Außerdem muss ein Passwort für den `rootdn` angegeben werden. Im Auslieferungszustand von OpenLDAP lautet die Zeile:

```
rootpw        secret
```

Selbstverständlich sollten Sie `secret` durch ein anderes Passwort ersetzen; für den Produktiveinsatz ist zudem dringend zu empfehlen, das Passwort zu verschlüsseln. Dazu wird der Name des Verschlüsselungsalgorithmus in geschweiften Klammern angegeben, dahinter das Base64-codierte Passwort in der entsprechenden Verschlüsselung. Hier ein Beispiel für das MD5-verschlüsselte Passwort »geheim« (in der Praxis natürlich ebenso indiskutabel wie »secret«):

```
rootpw        {MD5}6GNuoBPmgvr2H1bOHLGrXA==
```

Um solche Passwörter zu erzeugen, können Sie das Konsolen-Tool `slappasswd` verwenden. Standardmäßig verschlüsselt es Passwörter mit dem Algorithmus, der in der *slapd.conf*-Direktive `password-hash` angegeben wird. Alternativ können Sie die Option `-h` verwenden, gefolgt vom gewünschten Algorithmus in geschweiften Klammern. Nach dem Aufruf müssen Sie zweimal das zu verschlüsselnde Passwort eingeben und erhalten als Ausgabe die `{Algorithmus}Passwort`-Kombination, die Sie dann in die Datei *slapd.conf* einfügen können. Beispiele:

```
> slappasswd
New password:
Re-enter new password:
{SSHA}0/6NOGhwu2Lk2sl/63wNnIpH7mbmhk9Q
> slappasswd -h {MD5}
New password:
```

```
Re-enter new password:
{MD5}6GNuoBPmgvr2H1bOHLGrXA==
```

Gegebenenfalls müssen Sie mithilfe von `include`-Direktiven zusätzliche Schema-Dateien einbinden, z.B. *nis.schema* für die Objektklasse `posixAccount`.

Nach Änderungen der Konfigurationsdatei müssen Sie den OpenLDAP-Server mit der zu Ihrem Betriebssystem passenden Methode neu starten. OpenLDAP-Versionen ab 2.3 werten allerdings auch die Konfigurationsdateien im Verzeichnis */etc/openldap/slapd.d* zur Laufzeit aus, was Konfigurationsänderungen ohne Neustart ermöglicht.

Sobald der Server läuft, können Sie Verzeichniseinträge erzeugen und ihn anderweitig nutzen. Am einfachsten ist es, die gewünschten Einträge im LDIF-Format in Textdateien zu schreiben und dann mithilfe der OpenLDAP-Kommandozeilen-Tools einzulesen. Das folgende Beispiel legt die LDAP-Wurzel für *mynet.de*, die Organisationseinheiten `people` und `developers` sowie den Beispiel-User-Account `sascha` an:

```
dn: dc=mynet,dc=de
dc: mynet
objectClass: dcObject
objectClass: organizationalUnit
ou: lingoworld.de

dn: ou=people,dc=mynet,dc=de
ou: people
objectClass: organizationalUnit
dn: ou=developers,ou=people,dc=mynet,dc=de

ou: developers
objectClass: organizationalUnit

dn: uid=sascha,ou=developers,ou=people,dc=mynet,dc=de
objectClass: posixAccount
objectClass: person
uid: sascha
cn: Sascha Kersken
sn: Kersken
uidNumber: 1001
gidNumber: 100
gecos: Sascha Kersken
userPassword: {MD5}uRymnwhjHoXhkrCqRfpiUg==
homeDirectory: /home/sascha
loginShell: /bin/bash
```

Das noch nicht besprochene Attribut `gecos` steht für das gleichnamige */etc/passwd*-Feld mit dem ausführlichen Beschreibungstext (in der Regel wird dort der vollständige Name eingetragen).

Angenommen, die Datei heißt *data.ldif*. Dann können Sie Folgendes eingeben, um sie zu Ihrem LDAP-Verzeichnis hinzuzufügen:

```
# ldapadd -x -f data.ldif \
-D "cn=Manager,dc=mynet,dc=de" -W
```

Die Option `-x` steht für Klartext-Authentifizierung; Sie können sie weglassen, wenn Sie das `rootdn`-Passwort gemäß der obigen Empfehlung verschlüsselt haben. Mit `-f Dateiname` geben Sie die Datei an, aus der `ldapadd` lesen soll. `-D` gibt den `rootdn` an, und `-W` sorgt dafür, dass in der nächsten Zeile das Passwort angefordert wird.

Weitere interessante LDAP-Kommandozeilen-Tools, die Sie sich ansehen sollten, sind `ldapmodify` zur Änderung eines existierenden Eintrags (`ldapadd` ist in Wirklichkeit ein `ldapmodify`-Aufruf mit der Option `-a` für `add`) sowie `ldapsearch`, das im LDAP-Verzeichnis nach Einträgen sucht.

Geben Sie beispielsweise Folgendes ein, wenn Sie alle Einträge Ihres Verzeichnisses lesen möchten:

```
# ldapsearch -x -D "cn=Manager,dc=mynet,dc=de" \
-b "dc=mynet,dc=de" "(objectclass=*)" -W
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <dc=mynet,dc=de> with scope sub
# filter: (objectClass=*)
# requesting: ALL
#

# mynet.de
dn: dc=mynet,dc=de
dc: mynet
objectClass: dcObject
objectClass: organizationalUnit
ou: mynet.de
[...]
```

Die Optionen `-x`, `-D` und `-W` wurden bereits erläutert. Die Option `-b` gibt den DN der Basis an, unter der gesucht werden soll. `"(objectclass=*)"` ist ein LDAP-Filter. Da jeder LDAP-Eintrag ein `objectClass`-Attribut besitzen muss, trifft dieser

Filter auf jeden Knoten des Verzeichnisses zu. Näheres über LDAP-Filter erfahren Sie weiter unten bei der Definition der `mod_authnz_ldap`-Direktiven.

Mehr Komfort als diese Konsolenkommandos bieten grafische LDAP-Clients. Einer der empfehlenswertesten ist *phpLDAPadmin*, der unter *http://phpldapadmin.sourceforge.net* zum Download bereitsteht. Er wird als PHP-Anwendung auf einem Webserver installiert. Nachdem PHP auf Ihrem Apache läuft (siehe dazu Kapitel 15, »Technologien zur Webprogrammierung«), müssen Sie das Paket in ein Verzeichnis außerhalb Ihrer `DocumentRoot` kopieren, das betreffende Verzeichnis mithilfe einer `Alias`-Direktive und eines Containers für die nötigen Rechte einbinden, die Konfigurationsdatei *conf/config.php* anpassen (z.B. die URL Ihres LDAP-Servers eintragen) und können dann die betreffende URL im Browser öffnen.

### 9.5.3 LDAP-Authentifizierungs-Direktiven

Nachdem Sie jetzt in Grundzügen erfahren haben, wie LDAP funktioniert und wie man einen OpenLDAP-Server einrichtet, kommen nun die verschiedenen Direktiven von `mod_authnz_ldap` zur Sprache, die verwendet werden, um das Verzeichnis als Provider für die Anmeldeverwaltung einzusetzen.

**Require-Erweiterungen**

`mod_authnz_ldap` nimmt Ergänzungen an den zulässigen Angaben für die Core-Direktive `Require` vor: Neben `Require user`, `Require group` und `Require valid-user` können Sie mit `Require` angeben, dass die Anmeldedaten bestimmte LDAP-Kriterien erfüllen müssen:

- Require ldap-user
  Ein Benutzer mit einem angegebenen Benutzernamen erhält Zugriff auf den geschützten Bereich. Wenn ein Benutzername Leerzeichen enthält, muss er in Anführungszeichen stehen (das gilt auch für die Werte der meisten anderen `Require ldap-*`-Varianten). Welches LDAP-Attribut als Username betrachtet wird (z.B. cn oder uid), regelt die weiter unten besprochene Direktive AuthLDAPUrl. Beispiele:

  ```
  Require ldap-user peter
  Require ldap-user "Peter Schmitz"
  ```

- `Require ldap-group`
  Gewährt einer bestimmten Gruppe Zugriff auf die geschützte Ressource. Der DN der Gruppe darf dabei nicht in Anführungszeichen stehen. Beispiel:

  ```
  Require ldap-group cn=Users, o=MyNet
  ```

- `Require ldap-dn`
  Der Zugriff wird Benutzern mit einem bestimmten DN gewährt. Beispiel:

  ```
  Require ldap-dn \
  "cn=Peter Schmitz,ou=Sales,o=MyNet"
  ```

- `Require ldap-attribute`
  Der Zugriff wird all denjenigen Usern gestattet, für die ein Attribut einen bestimmten Wert besitzt; wenn Sie mehrere Attribute angeben, werden diese mit einem Oder verknüpft, sodass jeweils eine Übereinstimmung genügt. Beispiel:
  ```
  Require ldap-attribute city=Köln ou=developers
  ```
- `Require ldap-filter`
  Diese Variante ermöglicht die Angabe eines komplexen Ausdrucks (Näheres siehe unter `AuthLDAPUrl`), mit dem die potenziellen Zugriffskandidaten verglichen werden. Das folgende Beispiel erlaubt allen Benutzern den Zugriff, bei denen das Attribut `city` den Wert `Köln` hat *und* die eine Telefaxnummer besitzen:
  ```
  Require ldap-filter \
     &(city=Köln)(facsimileTelephoneNumber=*)
  ```

### AuthLDAPUrl

URL, die die LDAP-Suchparameter angibt

| | |
|---|---|
| **Modul** | `mod_authnz_ldap` (bis 2.0.x: `mod_auth_ldap`) |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthLDAPUrl URL [NONE\|SSL\|TLS\|STARTTLS]` |
| **Standardwert** | nicht gesetzt |

Dies ist die wichtigste Direktive für die LDAP-Authentifizierung: Sie gibt die RFC-2255-konforme URL des LDAP-Servers an, der befragt werden soll, und kann zahlreiche Präzisierungen bezüglich des Teilbaums enthalten, der durchsucht werden soll. Die grundlegende Syntax der URL sieht so aus:

```
ldap://Host:Port/Basis-DN?Attribut?Suchtiefe?Filter
```

Die einzelnen Bestandteile bedeuten Folgendes:

- `Schema`: Das Schema ist entweder `ldap` (wie in der Übersicht) oder `ldaps`, falls Sie über eine SSL-Verbindung auf LDAP zugreifen. Letzteres funktioniert nur, wenn Sie Apache beim Kompilieren gegen eine LDAP-Bibliothek gelinkt haben, die SSL unterstützt. Dies ist z. B. bei neueren OpenLDAP-Versionen der Fall.
- `Host:Port`: Dieser URL-Bestandteil dürfte sich von selbst erklären: Es handelt sich um den Hostnamen und die Portnummer des LDAP-Servers, der die Anmeldung bestätigen soll. Sie können die Angabe ganz weglassen, wenn der LDAP-Server auf `localhost` läuft. Falls Sie keinen Port angeben, wird die Standardeinstellung 389 für `ldap` beziehungsweise 636 für `ldaps` verwendet.
- Sie können durch Leerzeichen getrennt mehrere Hosts angeben; `mod_authnz_ldap` verwendet den ersten, zu dem erfolgreich eine Verbindung aufgebaut

werden kann. Diese Verbindung bleibt bestehen, bis Apache oder der LDAP-Server-Dienst beendet wird.

- `Basis-DN`: Über diesen Parameter können Sie den DN (Distinguished Name) der Wurzel des Bereichs angeben, in dem auf dem LDAP-Server nach der Anmeldebestätigung gesucht werden soll.
- `Attribut`: Dies ist das Attribut, mit dem die Nutzerdaten auf dem LDAP-Server verglichen werden sollen – es bildet auch den Benutzernamen für `Require ldap-user`. Der Standardwert ist `uid`; häufig wird auch `cn` angegeben.
- `Suchtiefe`: Der Wert `one` beschränkt die Suche auf eine Ebene, während `sub` auch in untergeordneten Bereichen sucht. Der dritte in RFC 2255 definierte Wert, `base`, wird von `mod_authnz_ldap` nicht unterstützt.
- `Filter`: Dies ist ein LDAP-Suchfilter mit maximal 8.192 Zeichen[1] Länge, der auf den durchsuchten Bereich angewendet wird. Filter haben Formen wie `Attribut=Wert` oder auch `Attribut=*`, um zu verlangen, dass das entsprechende Attribut vorhanden ist (beispielsweise beschränkt `facsimileTelephoneNumber=*` die Suche auf User, die eine Faxnummer besitzen). Der Standardwert ist `objectClass=*` – es werden sämtliche Datensätze durchsucht.

Hier einige Beispiele:

- `AuthLDAPUrl "ldap://ldap.mynet.de/ou=Sales, o=MyNet?cn"`
  Diese LDAP-URL durchsucht den vom LDAP-Server `ldap.mynet.de` bereitgestellten Bereich `ou=Sales, o=MyNet` nach Usernamen (`cn`). Dies lässt sich beispielsweise sinnvoll mit `Require: valid-user` kombinieren – alle Mitarbeiter der Verkaufsabteilung dürften auf den entsprechenden Bereich zugreifen.
- `AuthLDAPUrl "ldap://ldap.mynet.de ldap-emerg.mynet.de /o=MyNet?cn"`
  Dies spricht den Server `ldap.mynet.de` oder notfalls den Ersatzserver `ldap-emerg.mynet.de` an und durchsucht die gesamte `MyNet`-Domain nach Usernamen.
- ```
  AuthLDAPUrl \
  "ldap://ldap.mynet.de/
  o=MyNet?uid?sub?(&(facsimileTelephoneNumber=*)(cn=Admins))"
  ```
  Die URL verwendet einen Filter:
  `(&(facsimileTelephoneNumber=*)(cn=Admins))`

  Diese mit »Und« verknüpften Bedingungen bedeuten: User, die eine Faxnummer haben und zur Gruppe `Admins` gehören. »Gruppe« kann in einem LDAP-

1 Das ist der Wert der symbolischen Konstanten `MAX_STRING_LEN` in der Datei *includes/httpd.h* im Apache-Quellcode.
  ```

Verzeichnis mehr als eine Bedeutung haben. Beispielsweise könnte die Gruppe Admins auf dem LDAP-Server folgendermaßen definiert sein:

```
dn: cn=Admins, ou=Sales, o=MyNet
objectClass: groupOfUniqueNames
uniqueName: cn=Peter Schmitz, ou=Sales, o=MyNet
uniqueName: cn=Heinz Mueller, ou=Sales, o=MyNet
```

Der optionale zweite Parameter von `AuthLDAPUrl` kann verwendet werden, um das in der eigentlichen URL angegebene URL-Schema durch eine bestimmte Verbindungsart zu überschreiben. Die möglichen Werte bedeuten Folgendes:

- `NONE` (Standard): ungesicherte Verbindung über den Standard-`ldap`-Port (389)
- `SSL`: gesicherte Verbindung über den Standard-`ldaps`-Port (636)
- `TLS` oder `STARTTLS`: Startet zunächst eine ungesicherte Verbindung auf Port 389 und initiiert danach eine gesicherte Verbindung auf demselben Port.

Damit Sie sich vorstellen können, wie die LDAP-Authentifizierung in der Praxis funktioniert, sehen Sie hier noch ein kleines Komplettbeispiel:

```
<Location /ldap-secured>
  AuthType Basic
  AuthBasicProvider ldap
  AuthName "Sales dpt. only"
  AuthLDAPUrl "ldap://ldap.mynet.de/ou=Sales, o=MyNet?cn"
  Require ldap-user "Peter Schmitz"
</Location>
```

#### AuthLDAPBindDN

Optionaler DN für die LDAP-Server-Bindung

| | |
|---|---|
| **Modul** | `mod_authnz_ldap` (bis 2.0.x: `mod_auth_ldap`) |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthLDAPBindDN Distinguished-Name` |
| **Standardwert** | nicht gesetzt |

Optional können Sie mit dieser Direktive einen DN (Anmeldenamen) angeben, der bei der Bindung an den LDAP-Server übergeben werden soll. Normalerweise wird eine anonyme Bindung verwendet, was zu bevorzugen ist, weil dann insbesondere das zugehörige Passwort (`AuthLDAPBindPassword`) nicht offen an den LDAP-Server übertragen werden muss.

Beim Einsatz von Microsoft Active Directory als LDAP-Server scheint es ohne dieses Verfahren nicht zu funktionieren.

### AuthLDAPBindPassword

Passwort für den DN der LDAP-Server-Bindung

| | |
|---|---|
| **Modul** | `mod_authnz_ldap` (bis 2.0.x: `mod_auth_ldap`) |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthLDAPBindPassword Passwort` |
| **Standardwert** | nicht gesetzt |

Wenn Sie mit `AuthLDAPBindDN` den DN eines berechtigten Benutzers für den LDAP-Server angeben, wird mit dieser Direktive das entsprechende Passwort übertragen. Wie bereits erwähnt, sollte beides unter normalen Umständen unterbleiben.

### AuthLDAPCharsetConfig

Name der Datei für die Sprachkürzel-Zeichensatz-Zuordnung

| | |
|---|---|
| **Modul** | `mod_authnz_ldap` (bis 2.0.x: `mod_auth_ldap`) |
| **Kontext** | Server |
| **Syntax** | `AuthLDAPCharsetConfig Dateipfad` |
| **Standardwert** | nicht gesetzt |

Mithilfe dieser Direktive können Sie den Pfad einer Datei angeben, die Sprachkürzel (die als Dateiendungen fungieren) den passenden Zeichensätzen zuordnet. Wie üblich kann der Pfad relativ zur `ServerRoot` oder absolut angegeben werden.

Die angegebene Datei ist eine einfache Textdatei, die Einträge im folgenden Format enthält:

```
Sprachkürzel Zeichensatz [Beschreibung]
```

Beispiel:

```
de iso-8859-1 Deutsch
tr iso-8859-9 Türkisch
```

Sowohl die Sprachkürzel als auch die Zeichensätze finden Sie in den Tabellen in Anhang E.

Mit `mod_authnz_ldap` wird eine Datei namens *charset.conv* geliefert, die für die meisten Zwecke ausreichend sein sollte.

### AuthLDAPCompareDNOnServer

LDAP-Server für den Vergleich der DN verwenden

| Modul | mod_authnz_ldap (bis 2.0.x: mod_auth_ldap) |
|---|---|
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | AuthLDAPCompareDNOnServer On\|Off |
| Standardwert | On |

Wenn diese Direktive auf `On` gesetzt wird – was standardmäßig der Fall ist –, findet der Vergleich eines mittels `Require ldap-dn` angegebenen DN auf dem LDAP-Server statt. Bei `Off` führt `mod_authnz_ldap` dagegen selbst einen einfachen Stringvergleich durch – dies ist schneller, kann aber korrekte Angaben irrtümlich ausschließen. Wenn Sie die Performance verbessern möchten, sollten Sie deshalb zunächst den Einsatz von `mod_ldap` in Betracht ziehen; dieses im nächsten Abschnitt beschriebene Modul stellt unter anderem einen Cache bereit, der LDAP-Zugriffe beschleunigen kann.

#### AuthLDAPDereferenceAliases

Bestimmt, wann LDAP-Aliase aufgelöst werden.

| Modul | mod_authnz_ldap (bis 2.0.x: mod_auth_ldap) |
|---|---|
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | AuthLDAPDereferenceAliases never \| searching \| finding \| always |
| Standardwert | always |

Diese Direktive legt fest, wann `mod_auth_ldap` Aliase auflöst, das heißt, wann überprüft wird, ob es Alternativeinträge im LDAP-Verzeichnis gibt, die einen gegebenen Wert gültig machen. Es gibt vier mögliche Werte:

- `always` – Aliase werden immer sofort aufgelöst. Dies ist der Standardfall, der in den meisten Fällen keine Probleme bereitet.
- `searching` – Die Aliase werden beim Durchsuchen des LDAP-Verzeichnisses aufgelöst.
- `finding` – Es wird erst dann versucht, Aliase aufzulösen, wenn der angegebene Eintrag nicht gefunden wird.
- `never` – Aliase werden nie aufgelöst; nur der tatsächlich angegebene Eintrag gilt.

#### AuthLDAPGroupAttribute

LDAP-Attribute, die zur Prüfung der Gruppenmitgliedschaft verwendet werden

| Modul | mod_authnz_ldap (bis 2.0.x: mod_auth_ldap) |
|---|---|
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |

| Syntax | AuthLDAPGroupAttribute Attribut |
|---|---|
| Standardwert | nicht gesetzt |

Diese Direktive gibt die LDAP-Attribute an, nach denen zur Überprüfung der Gruppenmitgliedschaft gesucht werden soll. Wenn Sie keine Angaben machen, werden `member` und `uniquemember` verwendet. Um mehrere Attribute festzulegen, können Sie die Direktive mehrfach untereinanderschreiben. Beispiel:

```
AuthLDAPGroupAttribute member
```

#### AuthLDAPGroupAttributeIsDN

DN des Client-Benutzernamens zur Prüfung der Gruppenmitgliedschaft verwenden

| Modul | mod_authnz_ldap (bis 2.0.x: mod_auth_ldap) |
|---|---|
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | AuthLDAPGroupAttributeIsDN On\|Off |
| Standardwert | On |

`AuthLDAPGroupAttributeIsDN` ist standardmäßig eingeschaltet und sorgt dafür, dass der Distinguished Name (DN) des Benutzernamens zur Überprüfung der Gruppenmitgliedschaft verwendet wird. Standardmäßig wird also nach einer Information wie `cn=Peter Schmitz, ou=Sales, o=MyNet` gesucht; wenn Sie die Direktive auf `Off` setzen, wird stattdessen nur der einfache Username (z.B. `pschmitz`) verwendet.

#### AuthLDAPRemoteUserIsDN

DN des Client-Benutzernamens als `REMOTE_USER` setzen

| Modul | mod_authnz_ldap (bis 2.0.x: mod_auth_ldap) |
|---|---|
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | AuthLDAPRemoteUserIsDN On\|Off |
| Standardwert | Off |

Normalerweise enthält die Umgebungsvariable `REMOTE_USER` den einfachen Benutzernamen, der bei der Client-Anfrage übertragen wurde, beispielsweise `hmueller`. Wenn Sie in der Variablen stattdessen den vollständigen DN haben möchten, damit Sie ihn z.B. für CGI-Skripte oder `Rewrite`-Aktivitäten auswerten können, müssen Sie die Direktive explizit auf `On` setzen:

```
AuthLDAPRemoteUserIsDN On
```

#### AuthLDAPRemoteUserAttribute

Das angegebene LDAP-Attribut als `REMOTE_USER` setzen

| | |
|---|---|
| **Seit Version** | 2.2 |
| **Modul** | `mod_authnz_ldap` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthLDAPRemoteUserIsDN Attribut` |
| **Standardwert** | Nicht gesetzt |

Mithilfe dieser Direktive können Sie ein beliebiges LDAP-Attribut festlegen, das als Wert für die Umgebungsvariable `REMOTE_USER` gesetzt wird – beispielsweise `uid` oder `cn`. Wenn diese Direktive gesetzt ist, hat sie Vorrang vor `AuthLDAP-RemoteUserIsDN`. Wichtig ist allerdings, dass das angegebene Attribut in `AuthLDAPUrl` enthalten ist, sonst hat diese Direktive gar keine Wirkung.

#### AuthzLDAPAuthoritative

Bestimmt, ob Nicht-LDAP-Authentifizierungsdaten an andere Module weitergereicht werden.

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_authnz_ldap` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthLDAPAuthoritative On\|Off` |
| **Standardwert** | `On` |

Wie alle `Auth*Authoritative`-Direktiven ermöglicht es `AuthzLDAP Authoritative`, die Anmeldedaten gegebenenfalls an weitere Authentifizierungsmechanismen durchzureichen. Für die Kompatibilität mit Microsoft FrontPage müssen Sie `Authz-LDAPAuthoritative` auf `Off` setzen.

### 9.5.4 LDAP-Performance-Verbesserung mit mod_ldap

Wenn Ihre Website zu großen Teilen von der LDAP-basierten Authentifizierung abhängt oder wenn Sie andere Module einsetzen, die mit einem LDAP-Server zusammenarbeiten, sollten Sie das Modul `mod_ldap` aktivieren; je nach konkretem LDAP-Server kann es sogar erforderlich sein, das Modul grundsätzlich zu aktivieren. Es ergänzt die grundlegende LDAP-Funktionalität um zwei wichtige Faktoren:

- **Connection-Pooling**
  Eine Verbindung zum LDAP-Server wird nach einmaligem Gebrauch nicht wieder abgebaut, sondern aufrechterhalten. Dies bringt einen ähnlichen Performance-Gewinn wie persistente HTTP-Verbindungen.
- **LDAP-Cache**
  `mod_ldap` richtet in einem allgemein zugänglichen Shared-Memory-Segment einen Cache ein.[2] Der Cache erledigt zwei verschiedene Teilaufgaben: Der Search/Bind-Cache speichert Suchergebnisse, die zu einer erfolgreichen Bindung an das LDAP-Verzeichnis geführt haben, während der Operations-Cache Vergleichsoperationen und ihre Ergebnisse zwischenspeichert.

Nachdem Sie `mod_ldap` aktiviert haben, brauchen Sie nichts Besonderes mehr zu veranlassen, damit `mod_auth_ldap` und andere LDAP-basierte Module von dessen Fähigkeiten Gebrauch machen; die `mod_ldap`-Direktiven dienen lediglich der genauen Steuerung des Cache-Verhaltens und ermöglichen zusätzlich die Konfiguration von SSL-Verbindungen zum LDAP-Server.

### Der Handler ldap-status

Das Modul `mod_ldap` definiert einen eigenen Handler, der Statusinformationen über den Cache anzeigen kann. Er heißt `ldap-status` und funktioniert genau wie die in Kapitel 7, »Header und MIME-Einstellungen«, vorgestellten Handler `server-info` und `server-status`. Die folgende Beispielkonfiguration ermöglicht den Zugriff auf den LDAP-Status über eine URL nach dem Muster *http://www.mynet.de/ldap-status*:

```
<Location /ldap-status>
  SetHandler ldap-status
</Location>
```

Selbstverständlich sollten Sie diese URL mit Authentifizierungsdirektiven versehen, damit die Öffentlichkeit nicht auf diese Informationen zugreifen kann.

### LDAPCacheEntries

Maximale Anzahl von Einträgen im LDAP-Cache

| | |
|---|---|
| **Seit Version** | 2.0.41 |
| **Modul** | `mod_ldap` |
| **Kontext** | Server |

2 Auf Plattformen, die kein Shared Memory unterstützen, wird für jeden Child-Prozess ein eigener LPAP-Cache eingerichtet. In der Regel verbessert aber auch das noch die Performance.

| Syntax | `LDAPCacheEntries Anzahl` |
|---|---|
| Standardwert | `1024` |

Diese Direktive bestimmt die Höchstzahl der Einträge im Search/Bind-Cache. Der Vorgabewert 1.024 sollte normalerweise ausreichen; über eine Erhöhung brauchen Sie nur nachzudenken, wenn Sie im `ErrorLog` eine entsprechende Fehlermeldung finden sollten. Wenn Sie den Wert 0 verwenden, wird der Search/Bind-Cache deaktiviert.

#### LDAPCacheTTL

Gültigkeitsdauer für Elemente im LDAP-Cache

| Seit Version | 2.0.41 |
|---|---|
| Modul | `mod_ldap` |
| Kontext | Server |
| Syntax | `LDAPCacheTTL Sekunden` |
| Standardwert | `600` |

Mit dieser Direktive wird die Gültigkeitsdauer für Elemente im Search/Bind-Cache festgelegt. Wenn sich in Ihrem LDAP-Verzeichnis nur selten Änderungen ergeben, könnten Sie in Erwägung ziehen, den Wert von 600 Sekunden (zehn Minuten) zu erhöhen. Andererseits gibt es Situationen, in denen Zugriffsrechte sehr schnell geändert werden müssen – beispielsweise, wenn ein ehemaliger Mitarbeiter bei der bedeutendsten Konkurrenzfirma angefangen hat. Insofern ist eine wesentliche Steigerung ein gewisses Sicherheitsrisiko.

#### LDAPConnectionTimeout

Maximale Dauer eines LDAP-Verbindungsversuchs

| Seit Version | 2.0.41 |
|---|---|
| Modul | `mod_ldap` |
| Kontext | Server |
| Syntax | `LDAPConnectionTimeout Sekunden` |
| Standardwert | `10` |

Diese Direktive legt fest, wie lange Apache versuchen soll, eine Verbindung zu einem LDAP-Server herzustellen. Die Standardwartezeit beträgt zehn Sekunden.

#### LDAPOpCacheEntries

Anzahl der LDAP-Vergleichsoperationen im LDAP-Cache

| Seit Version | 2.0.41 |
|---|---|
| Modul | mod_ldap |
| Kontext | Server |
| Syntax | LDAPOpCacheEntries Anzahl |
| Standardwert | 1024 |

Dies ist die maximale Menge von Einträgen im Operations-Cache. Der Wert 0 deaktiviert diesen Teil des Caches ganz; die Vorgabe 1.024 reicht normalerweise aus.

#### LDAPOpCacheTTL

Gültigkeitsdauer für Elemente im Operations-Cache

| Seit Version | 2.0.41 |
|---|---|
| Modul | mod_ldap |
| Kontext | Server |
| Syntax | LDAPOpCacheTTL Sekunden |
| Standardwert | 600 |

Diese Direktive bestimmt die maximale Gültigkeitsdauer von Objekten im Operations-Cache. Auch hier sind Sie mit dem Standardwert 600 in der Regel gut beraten (Näheres siehe unter `LDAPCacheTTL`).

#### LDAPSharedCacheFile

Pfad und Dateiname des Shared-Memory-Bereichs für den LDAP-Cache

| Seit Version | 2.0.41 |
|---|---|
| Modul | mod_ldap |
| Kontext | Server |
| Syntax | LDAPSharedCacheFile Verzeichnispfad/Dateiname |
| Standardwert | nicht gesetzt |

Mit dieser Direktive werden der Pfad und der Dateiname für den Shared-Memory-Bereich festgelegt, in dem der LDAP-Cache platziert wird. Wenn Sie die Direktive weglassen, verwendet jeder Child-Prozess seinen eigenen Cache. Selbstverständlich darf die Datei aus Sicherheitsgründen nicht innerhalb der `DocumentRoot` oder in einem mittels `Alias` eingebundenen Verzeichnis liegen. Beispiel:

```
LDAPSharedCacheFile /usr/local/apache2/ldap_cache
```

### LDAPSharedCacheSize

Größe des Shared-Memory-Bereichs für den LDAP-Cache

| | |
|---|---|
| **Seit Version** | 2.0.41 |
| **Modul** | `mod_ldap` |
| **Kontext** | Server |
| **Syntax** | `LDAPSharedCacheSize Bytes` |
| **Standardwert** | `102400` |

Dies ist die maximale Größe des Shared-Memory-Bereichs für den Cache. Der Vorgabewert ist 100 Kilobyte, was in der Regel ausreicht. Der Wert 0 deaktiviert den Shared-Memory-Cache.

### LDAPTrustedClientCert

Dateiname oder Nickname eines LDAP-Client-Zertifikats

| | |
|---|---|
| **Seit Version** | 2.2 |
| **Modul** | `mod_ldap` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`All`) |
| **Syntax** | `LDAPTrustedClientCert Typ Pfad\|Dateiname\|Nickname [Passwort]` |
| **Standardwert** | nicht gesetzt |

Diese Direktive ermöglicht die Angabe von Pfad, Dateiname oder Nickname eines Zertifikats, das bei der jeweiligen Verbindung zum LDAP-Server eingesetzt wird, um die Identität des Clients (in diesem Fall also des Webservers) zu verifizieren. Ob Ihr spezifischer LDAP-Server Client-Zertifikate unterstützt – und, falls ja, welchen Typ –, entnehmen Sie bitte seiner Dokumentation. Novell-Verzeichnisdienstserver unterstützen beispielsweise keine Zertifikate für einzelne Verbindungen; hier können Sie nur die anschließend beschriebene Direktive `LDAPTrustedGlobal Cert` verwenden.

Die unterstützten Typen sind:

- `CERT_DER` – binäres Client-Zertifikat in DER-Codierung
- `CERT_BASE64` – PEM-codiertes Client-Zertifikat
- `CERT_NICKNAME` – Client-Zertifikat-»Nickname« (Netscape SDK)
- `KEY_DER` – binärer Private Key in DER-Codierung
- `KEY_BASE64` – PEM-codierter Private Key

### LDAPTrustedGlobalCert

Datei oder Datenbank mit Informationen über vertrauenswürdige Zertifizierungsstellen oder globale Client-Zertifikate

| Seit Version | 2.2 |
|---|---|
| Modul | mod_ldap |
| Kontext | Server |
| Syntax | LDAPTrustedGlobalCert Typ Pfad\|Dateiname [Passwort] |
| Standardwert | nicht gesetzt |

Mithilfe dieser Direktive können Sie eine Datei oder eine Datenbank angeben, in der sich globale, also systemweit gültige, Client-Zertifikate befinden, die beim Aufbau von LDAP-Verbindungen verwendet werden sollen. Auch Dateien mit den Zertifikaten vertrauenswürdiger Zertifizierungsstellen (Certificate Authority oder CA) können hier angegeben werden. Die unterstützten Zertifikats- und Dateitypen hängen auch hier vom konkreten LDAP-Server ab. Grundsätzlich werden folgende Typen unterstützt:

- CA_DER – binäres CA-Zertifikat in DER-Codierung
- CA_BASE64 – PEM-codiertes CA-Zertifikat
- CA_CERT7_DB – Netscape-CA-Zertifikat-Datenbankdatei cert7.db
- CA_SECMOD – Netscape-secmod-Datenbankdatei
- CERT_DER – binäres Client-Zertifikat in DER-Codierung
- CERT_BASE64 – PEM-codiertes Client-Zertifikat
- CERT_KEY3_DB – Netscape-Client-Zertifikat-Datenbankdatei key3.db
- CERT_NICKNAME – Client-Zertifikat-»Nickname« (Netscape SDK)
- CERT_PFX – PKCS#12-codiertes Client-Zertifikat (Novell SDK)
- KEY_DER – binärer Private Key in DER-Codierung
- KEY_BASE64 – PEM-codierter Private Key
- KEY_PFX – PKCS#12-codierter Private Key (Novell SDK)

### LDAPTrustedMode

Voreinstellung für die Art der LDAP-Server-Verbindungen

| Seit Version | 2.2 |
|---|---|
| Modul | mod_ldap |
| Kontext | Server, <VirtualHost> |

| Syntax | LDAPTrustedMode NONE\|SSL\|TLS\|STARTTLS |
|---|---|
| Standardwert | nicht gesetzt |

Die möglichen Werte für diese Direktive und ihre Bedeutung wurden bereits für AuthLDAPUrl beschrieben. Diese Direktive stellt die globale Voreinstellung für den Server beziehungsweise für den jeweiligen virtuellen Host bereit. Wenn Sie in LDAP-URLs das Schema *ldaps://* verwenden, wird sie überschrieben.

## 9.6 Anonymous-Authentifizierung

Das Standard-Provider-Modul mod_authn_anon ermöglicht eine Art »Authentifizierung light«, wie sie auch bei Anonymous-FTP-Servern verwendet wird: Als Benutzername wird Anonymous (oder ein frei konfigurierbarer Name) angegeben, als Passwort wird die eigene E-Mail-Adresse des Benutzers erwartet. Da sich diese nicht überprüfen lässt, verwenden viele User bei solchen Gelegenheiten Nonsens wie geht@dich-nix.an, zumal sie nicht sicher sein können, ob sie nicht an einen potenziellen Spammer geraten sind. Der Nutzen dieser Nicht-Authentifizierung ist also eher zweifelhaft. Sie könnten sie beispielsweise benutzen, um ein Download-Archiv vor allzu vielen Zugriffen zu bewahren, weil die Aufforderung zur anonymen Authentifizierung einige Besucher abschrecken könnte.

Hier sehen Sie ein vollständiges Beispiel. Soweit die Direktiven nicht zum Core gehören und bereits erläutert wurden, finden Sie ihre ausführlichen Beschreibungen unten:

```
<Directory /usr/local/apache2/htdocs>
  AuthType Basic
  AuthBasicProvider anon
  AuthName "Benutzername: 'anonymous', Passwort: Ihre E-Mail"
  Anonymous anonymous
  Anonymous_NoUserID Off
  Anonymous_MustGiveEmail On
  Anonymous_VerifyEmail On
  Require valid-user
</Directory>
```

#### Anonymous

Usernamen, die Zugriff ohne Passwortüberprüfung erhalten

| Modul | mod_authn_anon (bis 2.0.x: mod_auth_anon) |
|---|---|
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |

| Syntax | `Anonymous User [User ...]` |
|---|---|
| Standardwert | nicht gesetzt |

Mit dieser Direktive können Sie einen oder mehrere Benutzernamen angeben, die sich anonym anmelden dürfen. Groß- und Kleinschreibung spielen keine Rolle. `anonymous` sollte immer unter den angegebenen Namen sein, weil dies den Anonymous-FTP-Gepflogenheiten entspricht. Beispiel:

```
Anonymous anonymous gast user
```

#### Anonymous_NoUserID

Bestimmt, ob ein Benutzername erforderlich ist.

| Modul | `mod_authn_anon` (bis 2.0.x: `mod_auth_anon`) |
|---|---|
| Kontext | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| Syntax | `Anonymous_NoUserID On\|Off` |
| Standardwert | `Off` |

Normalerweise muss einer der mittels `Anonymous` festgelegten Benutzernamen eingegeben werden. Wenn Sie dagegen auf jegliche Eingabe verzichten können, kann diese Direktive explizit auf `On` gesetzt werden:

```
Anonymous_NoUserID On
```

#### Anonymous_MustGiveEmail

Bestimmt, ob eine (beliebige) Eingabe in das Passwortfeld erforderlich ist.

| Modul | `mod_authn_anon` (bis 2.0.x: `mod_auth_anon`) |
|---|---|
| Kontext | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| Syntax | `Anonymous_MustGiveEmail On\|Off` |
| Standardwert | `On` |

Analog zu `Anonymous_NoUserID` können Sie mit dieser Direktive festlegen, ob auch ein völlig leeres Passwortfeld akzeptiert wird: Der Standardwert `On` erwartet eine wie auch immer geartete Eingabe; wenn Sie es zulassen möchten, dass keine Eingabe stattfindet, müssen Sie den Wert `Off` ausdrücklich angeben:

```
Anonymous_MustGiveEmail Off
```

#### Anonymous_VerifyEmail

Legt fest, ob das Passwort dem Schema einer E-Mail-Adresse genügen muss.

| Modul | mod_authn_anon (bis 2.0.x: mod_auth_anon) |
|---|---|
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | Anonymous_VerifyEmail On\|Off |
| Standardwert | Off |

Diese Direktive ermöglicht eine etwas verschärfte Überprüfung der Eingabe in das Passwortfeld. Wenn Sie möchten, dass der eingegebene Wert zumindest formal einer E-Mail-Adresse entspricht (xxx@yyy.zz), dann müssen Sie Folgendes angeben:

```
Anonymous_VerifyEmail On
```

Natürlich lassen sich auch Webanwendungen programmieren, bei denen Sie ein zufällig generiertes Passwort an die angegebene E-Mail-Adresse senden, das der Benutzer eingeben muss, um fortfahren zu können. Damit haben Sie die Gewissheit, dass er unter dieser Adresse erreichbar ist. Allerdings müssen Sie tatsächlich nützliche Dienste anbieten, damit sich überhaupt jemand darauf einlässt.

#### Anonymous_Authoritative

Bestimmt, ob nicht anonyme Anmeldedaten an weitere Authentifizierungsmethoden durchgereicht werden.

| Modul | mod_authn_anon (bis 2.0.x: mod_auth_anon) |
|---|---|
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | Anonymous_Authoritative On\|Off |
| Standardwert | Off |

Weiter oben wurden bereits andere Auth*Authoritative-Direktiven beschrieben. Anonymous_Authoritative funktioniert genauso: Wenn Sie dieser Direktive den Wert On geben, werden die Anmeldedaten bei Bedarf an weitere Authentifizierungsmodule durchgereicht.

#### Anonymous_LogEmail

Legt fest, ob das eingegebene Passwort im ErrorLog landet.

| Modul | mod_authn_anon (bis 2.0.x: mod_auth_anon) |
|---|---|
| Kontext | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | Anonymous_LogEmail On\|Off |
| Standardwert | On |

Wenn diese Direktive ihren Standardwert On hat, werden die eingegebenen Passwörter – das heißt, wenn Sie Glück haben, die E-Mail-Adressen der Besucher – in die ErrorLog-Datei aufgenommen.

## 9.7 Datenbankbasierte Authentifizierung mit mod_authn_dbd

Eine der interessantesten Neuerungen von Apache 2.2 war die eingebaute Unterstützung für Verbindungen zu SQL-Datenbanken. Dafür ist das Modul `mod_dbd` zuständig. Es erleichtert allen Modulautoren das Leben, die Informationen aus Datenbanken laden beziehungsweise darin speichern möchten; Anwendungsbeispiele wären Logging, Authentifizierung oder Content-Management. Die erste mitgelieferte Nutzanwendung ist der Authentifizierungs-Provider `mod_authn_dbd`, der zusammen mit dem Datenbankzugriffsmodul hier behandelt wird. Die Schnittstelle selbst, `apr_dbd`, wird genauer auf den Seiten ihres Autors Nick Kew beschrieben; die Adresse lautet *http://people.apache. org/~niq/dbd.html*.

Für Apache 2.0 gab es zahlreiche Authentifizierungsmodule für einzelne Datenbanken, etwa `mod_auth_mysql` sowie `mod_auth_oracle`. Nach und nach werden einige dieser Module auch als Provider für Apache 2.2 oder neuer portiert; beispielsweise existieren bereits `mod_authn_mysql` und `mod_authn_oracle`. Unter *http://mod-auth.sourceforge.net* befindet sich eine Projekt-Website zur Entwicklung diverser Provider-Module.

### 9.7.1 Datenbankverbindungen mit mod_dbd

Das Modul `mod_dbd` bietet eine einheitliche Schnittstelle für Datenbankverbindungen und -abfragen durch Apache-Module. Da die verfügbare Online-Dokumentation bisher recht spärlich ist, werden die verfügbaren Direktiven nur kurz angesprochen.

Beachten Sie, dass einige (entsprechend gekennzeichnete) Direktiven nur in Thread-basierten MPMs wie `worker` funktionieren; hier stellt `mod_dbd` Connection-Pooling-Dienste zur Verfügung, die die Effizienz der Datenbankverbindungen verbessern.

#### DBDriver

Einbinden eines Treibers für eine konkrete Datenbank

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_dbd` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `DBDriver Treibername` |
| **Standardwert** | nicht gesetzt |

Wie es bei Datenbank-Frameworks üblich ist, verwendet auch `apr_dbd` Treiber, um mit den verschiedenen Datenbanktypen zu kommunizieren. Apache 2.2 wird zurzeit mit Treibern für PostgreSQL, SQLite und MySQL geliefert. Da MySQL nicht unter der Apache-Lizenz, sondern unter der GNU GPL steht, müssen Sie die Unterstützung für GNU GPL explizit mittels `--with-mysql` in Apache beziehungsweise APR einkompilieren (siehe Kapitel 4, »Apache kompilieren und installieren«).

Das folgende Beispiel stellt den MySQL-Treiber ein:

```
DBDriver mysql
```

#### DBDParams

Zusätzliche Parameter für die Datenbankverbindung

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_dbd` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `DBDParams Param1=Wert1[,Param2=Wert2...]` |
| **Standardwert** | nicht gesetzt |

Diese Direktive kann zusätzliche Verbindungsparameter übergeben, falls der zugrunde liegende Treiber diese benötigt. Hier ein typisches Beispiel, das die gewünschte Datenbank sowie den Benutzernamen und das Passwort des Datenbankbenutzers festlegt:

```
DBDParams "dbname=htusers user=htadmin pass=geheim"
```

#### DBDPersist

Legt fest, ob persistente Datenbankverbindungen verwendet werden.

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_dbd` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `DBDPersist 0\|1` |
| **Standardwert** | nicht gesetzt |

Wenn diese Direktive den Wert 1 erhält, was üblicherweise der Fall sein sollte, werden persistente Datenbankverbindungen und – falls verfügbar – Connection-Pooling verwendet. Falls Sie 0 einstellen, wird dagegen für jede Datenbankab-

frage eine neue Verbindung hergestellt und sofort wieder geschlossen. Dies sollten Sie nur in Ausnahmefällen tun, beispielsweise zu Debugging-Zwecken.

#### DBDPrepareSQL

Ein SQL Prepared Statement erstellen

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_dbd` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `DBDPrepareSQL "Abfrage" Label` |
| **Standardwert** | nicht gesetzt |

Prepared Statements sind ein praktisches Feature vieler SQL-Datenbanken; aushilfsweise werden sie auch manchmal von den entsprechenden APIs zur Datenbankprogrammierung bereitgestellt. Es handelt sich um ein Verfahren, Abfragen dauerhaft zu speichern und bei Bedarf aufzurufen, was schneller und praktischer ist, als jedes Mal die gesamte Abfrage neu zu senden.

Der erste Parameter dieser Direktive ist die SQL-Abfrage, der zweite eine beliebige Bezeichnung, unter der die Abfrage später wieder aufgerufen werden kann. Das folgende Beispiel definiert eine Abfrage:

```
DBDPrepareSQL "SELECT pw FROM htusers WHERE user=%s" pwquery
```

Beachten Sie aber bitte, dass die Passwortabfrage-Direktiven von `mod_authn_dbd` zurzeit noch nicht mit Prepared Statements funktionieren.

#### DBDMin

Mindestanzahl der Datenbankverbindungen

| | |
|---|---|
| **Seit Version** | 2.1-beta, nur Thread-MPMs |
| **Modul** | `mod_dbd` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `DBDMin Anzahl` |
| **Standardwert** | nicht gesetzt |

Diese Direktive stellt die Mindestanzahl der Datenbankverbindungen pro Prozess in Thread-MPMs ein (bei den meisten MPMs also die Gesamtmindestanzahl). Beispiel:

```
DBDMin 1
```

### DBDMax

Höchstanzahl der Datenbankverbindungen

| | |
|---|---|
| **Seit Version** | 2.1-beta, nur Thread-MPMs |
| **Modul** | `mod_dbd` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `DBDMax Anzahl` |
| **Standardwert** | nicht gesetzt |

Analog zu `DBDMin` setzt diese Direktive die absolute Höchstzahl der Verbindungen für jeden Prozess. Beispiel:

```
DBDMax 8
```

### DBDKeep

Höchstanzahl aufrechterhaltener Datenbankverbindungen

| | |
|---|---|
| **Seit Version** | 2.1-beta, nur Thread-MPMs |
| **Modul** | `mod_dbd` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `DBDKeep Anzahl` |
| **Standardwert** | nicht gesetzt |

Diese Direktive legt die Höchstzahl der Datenbankverbindungen fest, die dauerhaft offen gehalten werden. Hier ein Beispiel:

```
DBDKeep 2
```

### DBDExpTime

Maximale Lebensdauer unbeschäftigter Datenbankverbindungen

| | |
|---|---|
| **Seit Version** | 2.1-beta, nur Thread-MPMs |
| **Modul** | `mod_dbd` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `DBDExpTime Sekunden` |
| **Standardwert** | nicht gesetzt |

Mithilfe dieser Direktive können Sie einstellen, wie lange eine Datenbankverbindung maximal aufrechterhalten werden soll, obwohl sie gerade nicht verwendet wird. Beispiel:

```
DBDExpTime 120
```

### 9.7.2 mod_authn_dbd-Direktiven

Bevor die beiden Direktiven zur DBD-Authentifizierung erläutert werden, hier ein Komplettbeispiel. Es verwendet Basic-Authentifizierung und den Datenbanktreiber für MySQL:

```
DBDriver mysql
DBDParams "dbname=htusers user=htadmin pass=geheim"

# Connection-Pooling
DBDMin  1
DBDKeep 2
DBDMax  8
DBDExptime 120

<Directory /usr/local/apache2/htdocs/privat>
  AuthType Basic
  AuthBasicProvider dbd
  AuthName "Privater Bereich"
  Require valid-user
  AuthDBDUserPWQuery "SELECT pw FROM users WHERE user=%s"
</Directory>
```

Dieses Beispiel setzt voraus, dass Ihr MySQL-Server eine Datenbank namens `htusers` mit einer Tabelle namens `users` enthält. In dieser Tabelle muss die Spalte `user` die Benutzernamen und die Spalte `pw` die zugehörigen Passwörter enthalten. Der MySQL-User `htadmin` benötigt `SELECT`-Privilegien für diese Tabelle; weitergehende Rechte sollte er aus Sicherheitsgründen nicht erhalten.

#### AuthDBDUserPWQuery

SQL-Abfrage zur Ermittlung des Passworts eines gegebenen Benutzers

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_authn_dbd` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthDBDUserPWQuery Abfrage` |
| **Standardwert** | nicht gesetzt |

Mithilfe dieser Direktive wird die SQL-Abfrage formuliert, die das Passwort eines gegebenen Benutzers aus der Datenbank liest. Der vom Client gelieferte Benutzername wird mithilfe des `printf()`-String-Platzhalters `%s` angegeben. Das folgende Beispiel wurde oben bereits im Zusammenhang erläutert:

```
AuthDBDUserPWQuery "SELECT pw FROM users WHERE user=%s"
```

**AuthDBDUserRealmQuery**

SQL-Abfrage zur Ermittlung des Passworts eines gegebenen Benutzers mit Realm

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | mod_authn_dbd |
| **Kontext** | <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| **Syntax** | AuthDBDUserRealmQuery Abfrage |
| **Standardwert** | nicht gesetzt |

Alternativ können Sie auch diese Direktive zur Übergabe der SQL-Abfrage verwenden. Sie kann zwei %s-Platzhalter enthalten, nämlich den Benutzernamen und den durch AuthName festgelegten Authentifizierungsbereich (Realm) – in genau dieser Reihenfolge. Hier ein Beispiel, wobei davon ausgegangen wird, dass die zusätzliche Spalte realm für den Authentifizierungsbereich existiert:

```
AuthDBDUserRealmQuery \
"SELECT pw FROM users WHERE user=%s AND realm=%s"
```

```
AuthDBDUserRealmQuery \
"SELECT pw FROM users WHERE user=%s AND realm=%s"
```

## 9.8 Formularbasierte Authentifizierung mit mod_auth_form

In Apache 2.3/2.4 wurde mit mod_auth_form eine Möglichkeit hinzugefügt, die Authentifizierung mit Hilfe selbst gestalteter HTML-Formulare abzufragen. Dazu wird der zusätzliche AuthType-Wert form bereitgestellt. Diese passen besser zum Layout einer Website als die vom Browser bereitgestellten Standard-Authentifizierungsdialoge. Für die formularbasierte Authentifizierung werden die Dienste des ebenfalls neuen Moduls mod_session benötigt, das in Kapitel 14, »CGI«, beschrieben wird.

### 9.8.1 Formular- und Konfigurations-Beispiel

Das Login-Formular selbst benötigt die beiden Eingabefelder httpd_username für den Benutzernamen und httpd_password für das Passwort. Hier ein einfaches Beispiel:

```
<form method="POST" action="/private.html">
Benutzername: <input type="text" name="httpd_username" value="" />
Passwort: <input type="password" name="httpd_password" value="" />
```

```
<input type="submit" name="login" value="Login" />
</form>
```

Die Konfiguration, um dieses Formular für die Authentifizierung zu verwenden, sieht beispielsweise so aus:

```
<Location /private.html>
  SetHandler form-login-handler
  AuthFormLoginRequiredLocation http://mynet.de/login.html
  AuthFormLoginSuccessLocation http://mynet.de/success.html
  AuthFormProvider file
  AuthUserFile conf/.htpasswd
  AuthType form
  AuthName Privat
  Session On
  SessionCookieName session path=/
  SessionCryptoPassphrase secret
</Location>
```

Der Handler `form-login-handler` sorgt dafür, dass die angegebene Location als Login-Formular betrachtet wird. `AuthFormLoginRequiredLocation` gibt die URL an, die aufgerufen wird, wenn die Authentifizierung fehlschlägt – hier sollte eine Information stehen, dass ein Login erforderlich ist und dass die Benutzer es erneut versuchen sollten. `AuthFormLoginSuccessLocation` ist die URL, zu der bei erfolgreichem Login weitergeleitet wird. Die Vergleichsdaten für Benutzername und Passwort befinden sich in diesem Beispiel in einer gewöhnlichen `htpasswd`-Textdatei, es kann aber auch jeder andere Provider verwendet werden. Schließlich wird noch die Session konfiguriert, damit der Login-Status zwischengespeichert werden kann.

### 9.8.2 mod_auth_form-Direktiven

Im Folgenden werden alle Direktiven des Moduls `mod_auth_form` beschrieben. Die benötigten `mod_session`-Direktiven werden in Kapitel 14, »CGI«, besprochen.

#### AuthFormBody

Formularfeld mit dem Body der Anfrage, die bei erfolgreichem Login versandt wird

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_auth_form` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `AuthFormBody Feldname` |
| **Standardwert** | `httpd_body` |

Wenn Sie die durch `AuthFormMethod`, `AuthFormMimetype` und `AuthFormBody` festgelegten Formularfelder in das Login-Formular einfügen, wird bei erfolgreichem Login eine HTTP-Anfrage mit der angegebenen Anfragemethode (`AuthFormMethod`), dem gewünschten MIME-Type (`AuthFormMimetype`) und dem entsprechenden Body (`AuthFormBody`, die vorliegende Direktive) versandt.

#### AuthFormDisableNoStore

Den Header CacheControl no-store für die Loginseite deakivieren

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_auth_form` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `AuthFormDisableNoStore On\|Off` |
| **Standardwert** | `Off` |

Wenn ein Benutzer nicht eingeloggt ist, wird mit dem Login-Formular wie üblich der Status 401 gesendet. Normalerweise wird zusätzlich der Header `CacheControl` mit dem Wert `no-store` gesetzt, um standardkonformen Clients das Speichern zu verbieten (siehe Kapitel 2, »Funktionsweise von Webservern«, und 13, »Proxy- und Cache-Funktionen«, für Details). Wenn Sie diese Direktive auf `On` setzen, wird dies verhindert, was Sie aus Sicherheitsgründen nur tun sollten, wenn es notwendig ist – zum Beispiel, wenn Sie JavaScript verwenden, das das Formular unter bestimmten Umständen erneut versendet.

#### AuthFormFakeBasicAuth

Einen gefälschten Basic-Auth-Header hinzufügen

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_auth_form` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `AuthFormFakeBasicAuth On\|Off` |
| **Standardwert** | `Off` |

Wenn Sie diese Direktive auf `On` setzen, wird ein Basic-Authentication-Header zur Anfrage hinzugefügt. Dieser kann genutzt werden, um die Anmeldedaten an eine Webanwendung zu übergeben, ohne dass diese das Formular auswerten oder auch nur von diesem wissen muss.

#### AuthFormLocation

Formularfeld mit der URL, zu der bei erfolgreichem Login weitergeleitet wird

| Seit Version | 2.3 |
|---|---|
| Modul | mod_auth_form |
| Kontext | <Directory>, <Location>, <Files> |
| Syntax | AuthFormLocation Feldname |
| Standardwert | httpd_location |

Diese Direktive gibt den Namen eines Formularfelds an, in dem sich die URL befindet, zu der bei erfolgreichem Login weitergeleitet wird. In aller Regel wird es sich dabei um ein `hidden`-Feld handeln, das im Formular vorgegeben ist und nicht vom Benutzer eingegeben wird.

#### AuthFormLoginRequiredLocation

URL der Seite, zu der weitergeleitet wird, wenn ein Login erforderlich ist

| Seit Version | 2.3 |
|---|---|
| Modul | mod_auth_form |
| Kontext | <Directory>, <Location>, <Files> |
| Syntax | AuthFormLoginRequiredLocation URL |
| Standardwert | nicht gesetzt |

Wenn Sie eine dedizierte Login-Seite haben, können Sie mit Hilfe dieser Direktive deren URL angeben. Andernfalls wählt Apache die `ErrorDocument`-Einstellung (siehe Kapitel 8, »Weiterleitungen und Indizes«).

#### AuthFormLoginSuccessLocation

URL, zu der nach erfolgreichem Login weitergeleitet wird

| Seit Version | 2.3 |
|---|---|
| Modul | mod_auth_form |
| Kontext | <Directory>, <Location>, <Files> |
| Syntax | AuthFormLoginSuccessLocation URL |
| Standardwert | nicht gesetzt |

Diese Direktive gibt die URL an, zu der nach einem erfolgreichen Formular-Login weitergeleitet wird. Ihr Wert kann durch das in `AuthFormLocation` angegebene Formularfeld überschrieben werden.

#### AuthFormLogoutLocation

URL, zu der nach dem Logout weitergeleitet wird

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_auth_form` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `AuthFormLogoutLocation URL` |
| **Standardwert** | nicht gesetzt |

Sie können neben einem Login-Formular auch ein Logout-Formular erstellen. Dabei kommt der Handler `form-logout-handler` zum Einsatz, und diese Direktive gibt die URL an, zu der nach dem Logout weitergeleitet wird.

#### AuthFormMethod

Formularfeld mit der HTTP-Methode der Anfrage, die bei erfolgreichem Login versandt wird

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_auth_form` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `AuthFormMethod Feldname` |
| **Standardwert** | `httpd_method` |

Diese Direktive wurde bereits zusammen mit `AuthFormBody` beschrieben.

#### AuthFormMimetype

Formularfeld mit dem MIME-Type der Anfrage, die bei erfolgreichem Login versandt wird

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_auth_form` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `AuthFormMimetype Feldname` |
| **Standardwert** | `httpd_mimetype` |

Diese Direktive wurde bereits zusammen mit `AuthFormBody` beschrieben.

#### AuthFormPassword

Name des Formularfelds, in dem das Anmeldepasswort steht

| Seit Version | 2.3 |
|---|---|
| Modul | mod_auth_form |
| Kontext | <Directory>, <Location>, <Files> |
| Syntax | AuthFormPassword Feldname |
| Standardwert | httpd_password |

Diese Direktive gibt den Namen des Formularfelds an, in dem das Passwort erwartet wird.

#### AuthFormProvider

Provider für die Formular-Authentifizierung

| Seit Version | 2.3 |
|---|---|
| Modul | mod_auth_form |
| Kontext | <Directory>, <Location>, <Files> |
| Syntax | AuthFormProvider provider [provider ...] |
| Standardwert | file |

Analog zu `AuthBasicProvider` und `AuthDigestProvider` gibt diese Direktive die Provider an, mit deren Hilfe Apache die Authentifizierungs-Vergleichsdaten finden soll. Das folgende Beispiel wählt den Provider `dbd`:

```
AuthFormProvider dbd
```

#### AuthFormSitePassphrase

Passphrase zum Umgehen der Authentifizierung auf Sites mit viel Traffic

| Seit Version | 2.3 |
|---|---|
| Modul | mod_auth_form |
| Kontext | <Directory>, <Location>, <Files> |
| Syntax | AuthFormSitePassphrase Passphrase |
| Standardwert | none |

Wenn die User-Session die angegebene Passphrase enthält, umgeht Apache die Authentifizierungsüberprüfung. Wenn Sie diese Direktive in die Konfiguration des form-login-handlers einfügen, wird die Passphrase in die Session geschrieben.

**Wichtig**

Wenn die Session mittels `mod_session_cookie` in einem clientseitigen Cookie gespeichert wird, ist es sehr empfehlenswert, die Session mittels `mod_session_crypto` zu verschlüsseln, da die Passphrase ansonsten unter Umständen durch einen Dictionary-Angriff ausgelesen werden kann.

#### AuthFormSize

Maximale Größe der Formulardaten

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_auth_form` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `AuthFormSize Bytes` |
| **Standardwert** | `8192` |

Diese Direktive gibt die maximal zulässige Größe der gesammelten Anmeldeformular-Daten an. Wenn Sie nur Benutzername und Passwort verwenden, genügt die Standardgröße von 8192 Byte für gewöhnlich. Verwenden Sie dagegen `AuthFormBody` und Konsorten, dann müssen Sie den Wert möglicherweise erhöhen.

#### AuthFormUsername

Name des Formularfelds, in dem der Benutzername steht

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_auth_form` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `AuthFormUsername Feldname` |
| **Standardwert** | `httpd_username` |

Diese Direktive gibt den Namen des Formularfelds an, in dem der Anmeldename erwartet wird.

#### AuthFormAuthoritative

Bestimmt, ob die Authentifizierung durch weitere Module überprüft wird

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_auth_form` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>` |

| Syntax | AuthFormAuthoritative On\|Off |
|---|---|
| Standardwert | On |

Wie alle Auth*Authoritative-Direktiven bestimmt auch diese, ob die Formular-Authentifizierung das »letzte Wort« bei der Entscheidung über die Anmeldung hat (Standardwert On), oder ob gegebenenfalls weitere Module befragt werden (Off).

## 9.9 Weitere Authentifizierungseinstellungen

In diesem Abschnitt erhalten Sie Informationen über einige zusätzliche Module und Konfigurationsmöglichkeiten zur Authentifizierung.

### 9.9.1 Provider-Aliase

Dieses Apache-2.2-Modul stellt die Container-Direktive <AuthnProviderAlias> bereit. Diese ermöglicht die Speicherung von Authentifizierungseinstellungen als benutzerdefinierte Provider. Seit Version 2.3-beta befindet die Direktive sich im Modul mod_authn_core; mod_authn_alias gibt es nicht mehr.

#### <AuthnProviderAlias> ... </AuthnProviderAlias>

Definition alternativer Authentifizierungs-Provider

| Seit Version | 2.1-beta |
|---|---|
| Modul | mod_authn_alias; seit 2.3 mod_authn_core |
| Kontext | Server |
| Syntax | <AuthnProviderAlias Provider Aliasname> ...<br></AuthnProviderAlias> |
| Standardwert | nicht gesetzt |

Als Parameter des Containers müssen Sie den Namen eines vorhandenen Providers wie etwa file oder ldap sowie einen beliebigen selbst definierten Aliasnamen angeben. Innerhalb des Containers können Sie die Logik des gewünschten Authentifizierungsablaufs genauer definieren.

Das folgende Komplettbeispiel definiert zwei Aliase auf der Basis des Providers file. In der Praxis ermöglicht dies das Nachschlagen von Anmeldedaten in zwei verschiedenen Dateien:

```
<AuthnProviderAlias file file1>
  AuthUserFile /usr/local/apache2/misc/.htusers1
</AuthnProviderAlias>
```

```
<AuthnProviderAlias file file2>
  AuthUserFile /usr/local/apache2/misc/.htusers2
</AuthnProviderAlias>

# Verzeichnisschutz auf der Grundlage der neuen Provider:
<Directory /usr/local/apache2/htdocs/privat>
  AuthType Basic
  AuthBasicProvider file1 file2
  AuthName "Private Dateien"
  Require valid-user
</Directory>
```

**<AuthzProviderAlias> ... </AuthzProviderAlias>**

Definition alternativer Autorisierungs-Provider

| Seit Version | 2.3-beta |
|---|---|
| Modul | mod_authz_core |
| Kontext | Server |
| Syntax | <AuthzProviderAlias Provider Aliasname> ...<br></AuthzProviderAlias> |
| Standardwert | nicht gesetzt |

Analog zu `<AuthnProviderAlias>` wurde in der neuen Apache-Version die Container-Direktive `<AuthzProviderAlias>` hinzugefügt, die beliebig viele `Require`-Direktiven oder -Container unter einem Aliasnamen zusammenfassen kann. In einem konkreten Verzeichnis kann der Aliasname dann als neuer Wert für `Require` verwendet werden.

Beispiel:

```
<AuthzProviderAlias group_and_ip>
  <RequireAll>
    Require group admins
    Require host 192.168.0.0/8
  </RequireAll>
</AuthzProviderAlias>

<Location /admin>
  Require group_and_ip
</Location>
```

### 9.9.2 mod_authz_owner

Dieses Modul vergleicht den zur Authentifizierung verwendeten Benutzernamen mit der User-ID einer Ressource und den Gruppennamen mit der Group-ID. Entsprechend definiert das Modul folgende Erweiterungen für die weiter oben beschriebene Direktive `Require`:

```
# User-ID muss übereinstimmen
Require file-owner
# Group-ID muss übereinstimmen
Require file-group
```

Hier ein Gesamtbeispiel für die Verwendung von `Require file-owner`:

```
<Directory /usr/local/apache2/htdocs/owner-files>
  AuthType Basic
  AuthBasicProvider file
  AuthName "Private Benutzerdateien"
  AuthUserFile /usr/local/apache2/misc/.htusers
  Require file-owner
  # Wichtig, damit ALLE Voraussetzungen geprüft werden:
  Satisfy All
</Directory>
```

#### AuthzOwnerAuthoritative

Bestimmt, ob die eigentümerbasierte Zugriffsberechtigung das letzte Wort hat.

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_authz_owner` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthzOwnerAuthoritative On\|Off` |
| **Standardwert** | `On` |

Diese Direktive bestimmt nach dem üblichen Schema, ob die Anmeldedaten an nachgeordnete Module weitergereicht werden. Soll dies geschehen, müssen Sie den Wert `Off` einstellen.

### 9.9.3 mod_authn_default und mod_authz_default

Wenn Sie keinen Authentifizierungs-Provider gewählt haben, dient `mod_authn_default` als Fallback-Modul; seine einzige Aufgabe besteht darin, sämtliche Anmeldedaten zurückzuweisen. Ebenso wird `mod_authz_default` als Rettungsanker eingesetzt, wenn kein Modul zur Zugriffskontrolle wie `mod_authz_user` oder `mod_authz_groupfile` ausgewählt wurde. Obwohl diese Module also eigentlich

schon den äußersten Rand der Authentifizierung beziehungsweise Autorisierung bilden, werden die beiden nachfolgend vorgestellten Direktiven definiert, um die Anmelde- beziehungsweise Zugriffskontrolldaten theoretisch an untergeordnete Module zu delegieren.

Seit Apache 2.3-beta gibt es diese Module und Direktiven nicht mehr.

#### AuthDefaultAuthoritative

Weitergabe der Anmeldedaten an untergeordnete Module

| | |
|---|---|
| **Version** | Nur 2.1-beta und 2.2 |
| **Modul** | `mod_authn_default` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthnDefaultAuthoritative On\|Off` |
| **Standardwert** | `On` |

Wenn Sie diese Direktive auf `Off` setzen, dann werden die durch `mod_authn_default` pauschal abgelehnten Anmeldedaten weiterdelegiert. Nach menschlichem Ermessen gibt es hinter diesem Fallback-Modul allerdings kein nachgeordnetes Modul mehr.

#### AuthzDefaultAuthoritative

Delegation der Zugriffskontrolle an untergeordnete Module

| | |
|---|---|
| **Version** | Nur 2.1-beta und 2.2 |
| **Modul** | `mod_authz_default` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthnDefaultAuthoritative On\|Off` |
| **Standardwert** | `On` |

Dies ist das entsprechende Modul für die Weitergabe der Zugriffskontrolle. In der Praxis ist es ebenfalls nicht nützlich.

## 9.10 Zusammenfassung

Für die Authentifizierung sorgen zahlreiche eingebaute und optional noch erheblich mehr externe Module. Zunächst einmal geht es um die Auswahl der eigentlichen Authentifizierungsmethode: Die Basic-Authentifizierung wird von so gut wie allen bekannten Clients unterstützt, birgt aber das Sicherheitsrisiko, dass sie

Benutzernamen und Passwörter unverschlüsselt überträgt. Die Digest-Authentifizierung verschlüsselt die Anmeldedaten, wird aber nicht von jedem Client verstanden. Auf der anderen Seite können Sie sich aussuchen, aus welcher Quelle Apache die Anmeldedaten für die Anmeldungsüberprüfung beziehen soll. Standardmodule bringen die Unterstützung für einfache Textdateien, DBM-Dateien und LDAP-Verzeichnisse mit; für andere Speichermethoden – beispielsweise für fast jede existierende Datenbank – gibt es zahlreiche Drittanbieter-Module.

Die in Apache 2.2 eingeführte Aufteilung in Authentifizierungs- und Provider-Module erhöht die Flexibilität der möglichen Lösungen. Als Administrator können Sie davon zunächst durch die größer gewordene Auswahl an Speichermethoden für Digest-Anmeldedaten profitieren. In Zukunft sind zudem noch mehr Speicherlösungen zu erwarten, da Provider-Module keine Client-Kommunikation mehr implementieren müssen.

In Version 2.3/2.4 wurde dann entsprechend die Autorisierung aufgebohrt – besonders die Zusammenfassung aller Aspekte der Autorisierung und Zugriffskontrolle in der Direktive `Require` sowie die Schaffung der Autorisierungs-Container wie `<RequireAll>` erhöhen nochmals die Leistungsfähigkeit und Flexibilität von Apache.

*»Durch einen dichten Zaun gelangt kein Hund.«*
*– Chinesisches Sprichwort*

# 10 Gesicherte Verbindungen

Bei besonders sensiblen Web-Transaktionen wie der Übertragung von Bezahlinformationen im E-Commerce ist nicht nur die persönliche Anmeldung wünschenswert. Darüber hinaus sollten die Inhalte selbst verschlüsselt übertragen werden, damit Angreifer, denen es gelingt, diese von einem Router abzufangen, sie nicht lesen können.

Die passende Lösung für gesicherte Verbindungen heißt SSL (Secure Sockets Layer). Inzwischen gibt es einen Nachfolger namens TLS (Transport Layer Security). Wenn in diesem Kapitel von SSL die Rede ist, dann ist immer auch TLS gemeint.

SSL ist nicht auf den Einsatz mit einem Webserver beschränkt, sondern kann im Grunde jede beliebige TCP-Verbindung absichern. Die grundlegende Funktionalität wird nämlich von einer unabhängigen Bibliothek namens OpenSSL bereitgestellt, die sich sowohl in eigenen als auch in bestehenden Programmen einsetzen lässt. Übrigens ist OpenSSL nicht die einzige Implementierung einer Verschlüsselungsbibliothek – die Tatsache, dass sie frei verfügbar ist, macht sie aber zur ersten Wahl. Sie besteht aus Funktionen, Programmierschnittstellen und Kommandozeilen-Tools.

In Apache wird die SSL-Unterstützung durch das Modul `mod_ssl` bereitgestellt; in Version 2 gehört es sogar zum Lieferumfang. Die erste SSL-Implementierung in Apache war die im Core modifizierte Variante Apache-SSL von Ben Laurie. Der entsprechende Code wurde dann von Ralf S. Engelschall, dem Autor von `mod_rewrite`, in ein unabhängiges Modul übernommen und seitdem weiterentwickelt.

**Grundbegriffe der Kryptografie**

In diesem Abschnitt werden zahlreiche Begriffe aus dem Bereich der Kryptografie verwendet. Zwar geht es hier vornehmlich um die praktische Konfiguration von `mod_ssl`, aber mit einigen Grundlagen sollten Sie vertraut sein. Die wichtigsten von ihnen werden hier kurz eingeführt:

- **Symmetrische Verschlüsselung**
  Bei dieser klassischen Form der Verschlüsselung wird zur Verschlüsselung und zur Entschlüsselung derselbe Schlüssel verwendet. Damit dies sicher ist, muss der Schlüssel

selbst vor Außenstehenden verborgen werden. Für Internetverbindungen ist dieses Verfahren deshalb nicht geeignet.

- **Asymmetrische Verschlüsselung**
  Diese Form der Kryptografie wird auch als Public-Key-Verfahren bezeichnet. Für die Verschlüsselung wird ein anderer Schlüssel verwendet als zur Entschlüsselung. Einer der beiden Schlüssel ist öffentlich (Public Key), der andere geheim (Private Key). Jeder, der an einem solchen Verschlüsselungsverfahren teilnehmen möchte, kann seinen öffentlichen Schlüssel verteilen. Wer diesem Teilnehmer eine Nachricht senden möchte, kann sie mit dessen öffentlichem Schlüssel verschlüsseln. Der andere Teilnehmer kann sie daraufhin mit seinem privaten Schlüssel dechiffrieren. Genau dieses Verfahren liegt Internet-Kryptografielösungen wie SSL oder PGP zugrunde.
- **Einwegverschlüsselung**
  Bei dieser besonderen Form der Verschlüsselung ist die Dechiffrierung nicht vorgesehen und (sofern der Algorithmus korrekt arbeitet) auch nicht möglich. Einwegverschlüsselung wird beispielsweise für Passwörter verwendet: Anstatt das Passwort für den Vergleich mit der Benutzereingabe zu entschlüsseln (was ein erhebliches Sicherheitsrisiko wäre), wird die Eingabe nach demselben Verfahren chiffriert wie das verschlüsselt gespeicherte Passwort.

## 10.1 SSL-Grundlagen

Das SSL-Protokoll wurde ursprünglich von Netscape entwickelt. Die Version 1.0 wurde kaum in der Praxis eingesetzt. Bei Version 2.0 war das anders, weil sie 1995 in den Netscape Navigator 1.0 eingebaut wurde. Dies hat zu einer recht schnellen Verbreitung von SSL geführt – gesicherte Verbindungen erfordern nämlich, dass sich Server und Client auf dasselbe Protokoll einigen. Inzwischen beherrschen so gut wie alle wichtigen Browser sowohl SSL 1 und 2 als auch TLS.

Im Browser ist die verschlüsselte Verbindung vor allem daran erkennbar, dass die URLs mit dem Schema `https:` (`s` für »secure«) statt mit `http:` beginnen. Der Standard-TCP-Port für diese Verbindungen ist nicht der HTTP-Port 80, sondern 443.

`mod_ssl` erledigt in Zusammenarbeit mit dem Browser die folgenden Aufgaben:

- **Identitätsgarantie**
  SSL verwendet digitale Zertifikate, die bestätigen, dass der Anbieter einer Information tatsächlich er selbst ist. Das Zertifikat an sich wäre eigentlich nichts wert – weiter unten wird gezeigt, wie leicht Sie selbst eines erstellen können. Vielmehr geht es darum, dass ein vernünftiges Zertifikat von einer vertrauenswürdigen Zertifizierungsstelle mit einer digitalen Signatur versehen wird. Solche Zertifizierungsstellen (Certification Authorities) signieren Zertifikate gegen eine Gebühr und überprüfen dabei genau die Identität des

Antragstellers. Die Daten der wichtigsten Zertifizierungsstellen sind bereits in viele Browser eingebaut.

- **Integritätsprüfung**
  `mod_ssl` verwendet Message-Digest-Algorithmen wie MD5, um sicherzustellen, dass der Client genau die Daten erhält, die der Server geliefert hat. Verschlüsselung allein schützt nämlich unter Umständen nicht davor, dass ein Angreifer die Daten auf dem Weg austauscht.
- **Verschlüsselung**
  Dies ist natürlich die bekannteste Funktion von SSL. Client und Server vereinbaren einen Session Key, mit dem die Daten verschlüsselt werden. Wie der Name schon sagt, ist dieser Schlüssel nur für die aktuelle Sitzung gültig. Ein Cracker könnte also zu einem späteren Zeitpunkt nichts mehr damit anfangen.

Der Aufbau der gesicherten Verbindung erfolgt nach diesem Schema:

1. Der Client fordert eine URL mit dem Schema `https:` an.
2. Der Server sendet dem Client sein Zertifikat und seinen öffentlichen Schlüssel.
3. Der Client überprüft das Zertifikat. Wenn es nicht von einer bekannten, vertrauenswürdigen Zertifizierungsstelle signiert wurde, warnt der Browser den Benutzer und fragt, ob er mit der Anfrage fortfahren soll. Ähnliches geschieht, wenn das Zertifikat abgelaufen ist oder zurückgezogen wurde. Abbildung 10.1 zeigt, wie eine solche Meldung in Firefox 3 aussieht.

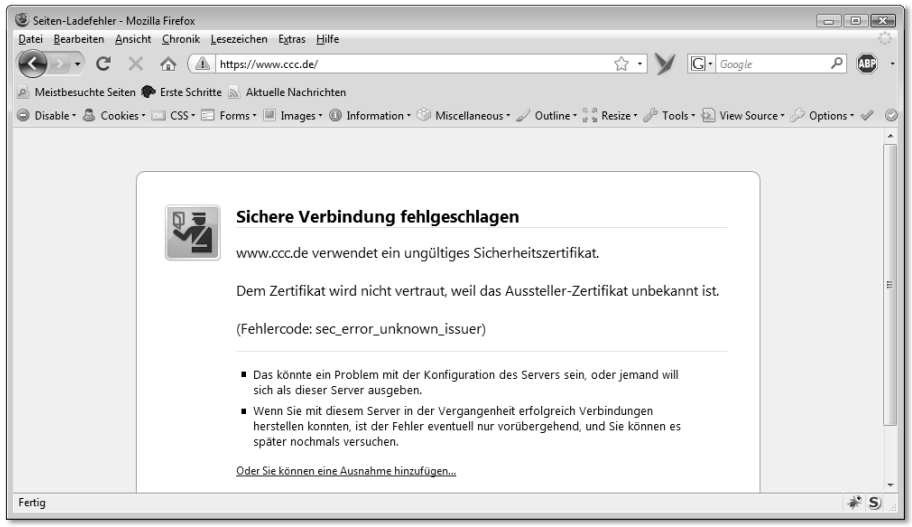



**Abbildung 10.1** Ausgabe von Firefox 3, wenn eine SSL-Verbindung angefordert wurde, die ein ungültiges (in diesem Fall von einer Zertifizierungsstelle, der der Browser nicht vertraut), signiertes Zertifikat liefert.

4. Nachdem der Client (oder der Benutzer) das Zertifikat anerkannt hat, sendet der Server den Session Key, der zur Verschlüsselung des gesamten Datenverkehrs während der Sitzung verwendet wird.

### 10.1.1 SSL einrichten

Bevor Sie `mod_ssl` einsetzen können, müssen Sie OpenSSL einrichten und ein Zertifikat erzeugen. Auf UNIX-Systemen ist die Wahrscheinlichkeit groß, dass OpenSSL-Bibliothek und -Tools bereits installiert sind. Falls nicht, werden sie wahrscheinlich in einer speziellen Variante für Ihre Distribution angeboten. Unter *http://www.openssl.org* finden Sie alternativ die Quellcode-Pakete zum Selbstkompilieren. In jedem Fall ist OpenSSL unter UNIX Standardsoftware und die Beschaffung sowie die Installation bereiten keine Probleme. Deshalb braucht hier auch nicht näher darauf eingegangen zu werden.

#### OpenSSL unter Windows installieren

Wie Sie in Kapitel 4, »Apache kompilieren und installieren«, bereits erfahren haben, gibt es Apache-Binärdistributionen für Windows mit und ohne SSL-Unterstützung. Wenn Sie gesicherte Verbindungen verwenden möchten, sollten Sie die Version mit eingebautem OpenSSL wählen, denn die Beschaffung separater OpenSSL-Installationspakete erfordert eine längere Suchmaschinensitzung. Selbstverständlich finden Sie die benötigten Pakete auf der DVD zum Buch.

Falls Sie eine ältere Apache-Version ohne SSL haben oder OpenSSL aus einem anderen Grund separat installieren möchten, doppelklicken Sie zur Installation auf die Datei *Win32OpenSSL-1_0_0e.exe*. Nun müssen Sie sich nacheinander mit Next durch folgende Screens klicken:

- kurze Info, dass OpenSSL installiert wird
- Zustimmung zum Lizenzvertrag. Dieser weist Sie unter anderem darauf hin, dass die Verwendung starker Kryptografie in einigen Ländern gesetzlichen Einschränkungen unterworfen ist und dass ihre Verwendung in anderen Ländern durch US-Exportbeschränkungen behindert wird.
- Auswahl des Installationsverzeichnisses (z. B. *C:\OpenSSL*)
- Name des Eintrags im Startmenü (standardmäßig *OpenSSL*)
- Zusammenfassung der Installationsoptionen. Wenn alles in Ordnung ist, können Sie Install anklicken.

Es gibt neben diesem Installer übrigens auch einfache ZIP-Archive mit OpenSSL für Windows. Diese können Sie einfach in ein Verzeichnis wie *C:\OpenSSL* entpacken; sie sind dann ebenfalls einsatzbereit – mit einer Ausnahme: Sie müssen die

mitgelieferten Dateien *ssleay32.dll*, *libeay32.dll* und *libssl32.dll* manuell in Ihr *%SystemRoot%\System32*-Verzeichnis kopieren.

Im *bin*-Verzeichnis Ihrer OpenSSL-Installation benötigen Sie zu guter Letzt noch eine Datei namens *openssl.cnf*, die unter anderem Einstellungen für die Zertifikatsanforderung enthält. Bei der Binärinstallation wird sie automatisch installiert und funktioniert ohne Änderungen; wenn Sie eine andere Version installiert haben, können Sie die Version von der DVD zum Buch in das Verzeichnis *OpenSSL\bin* kopieren. Beachten Sie, dass die Datei auf manchen Windows-Rechnern aufgrund der Dateiendung *.cnf* als Verknüpfungs-Icon dargestellt wird – auch die Endung wird nicht angezeigt.

#### Ein Zertifikat erzeugen

Nachdem OpenSSL zur Verfügung steht, können Sie ein (Test-)Zertifikat erstellen. Dieser Schritt funktioniert unter UNIX und Windows identisch. Als Erstes müssen Sie in das Verzeichnis *bin* Ihrer OpenSSL-Installation wechseln; bei der Windows-Version von Apache mit integriertem OpenSSL entspricht es dem Apache-*bin*-Verzeichnis. Alternativ können Sie dieses Verzeichnis auch zu Ihrem `PATH` hinzufügen und die Umgebungsvariable `OPENSSL_CONF` auf den Pfad Ihrer *openssl.cnf*-Datei setzen. Geben Sie danach folgenden Befehl ein, um mit dem Erzeugen des Zertifikats zu beginnen:

```
# openssl req -config openssl.cnf -new -out mynet.csr
```

Statt *mynet.csr* sollten Sie natürlich einen Dateinamen wählen, der Ihrem eigenen Domain-Namen entspricht. Nun müssen Sie eine Reihe von Fragen beantworten – die fett gedruckten Angaben in der nachfolgenden Beispielausgabe müssen Sie durch die für Sie zutreffenden Werte ersetzen. Sehr wichtig ist allerdings, dass Sie bei der Frage `Common Name (eg, YOUR name)` den vollständigen Hostnamen Ihrer Website (z. B. `www.mynet.de`) angeben, weil die Browser das Zertifikat sonst nicht akzeptieren und eine Meldung wie in Abbildung 10.1 anzeigen.

```
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
...................++++++
.....................++++++
writing new private key to 'privkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished
```

```
Name or a DN. There are quite a few fields but you can leave
some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:de
State or Province Name (full name) [Some-State]:NRW
Locality Name (eg, city) []:Koeln
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MyNet GmbH
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:www.mynet.de
Email Address []:info@mynet.de

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Als PEM-Passphrase für den privaten Schlüssel müssen Sie zweimal denselben Text eingeben (der hier nicht angezeigt wird). »Passphrase« bedeutet nicht, dass es ein sinnvoller Satz sein sollte, sondern weist nur darauf hin, dass der Text aus Sicherheitsgründen erheblich länger sein sollte als ein normales Passwort. Möglicherweise kennen Sie das Verfahren von PGP – dort müssen Sie sich die Passphrase gut einprägen, weil sie zur Entschlüsselung eingegeben werden muss. Im vorliegenden Fall wird sie aber eigentlich nur noch einmal benötigt – nämlich zur Erzeugung des RSA-Schlüssels für das Zertifikat. Geben Sie dazu Folgendes ein (`mynet` können Sie wieder durch Ihren selbst gewählten Namen ersetzen):

```
# openssl rsa -in privkey.pem -out mynet.key
Enter pass phrase for privkey.pem:
writing RSA key
```

Optional können Sie nun noch die Datei `.rnd` löschen, da sie nicht mehr benötigt wird und potenziellen Angreifern nützlich sein könnte:

```
# rm .rnd
```

Unter Windows müssen Sie natürlich `del` statt `rm` verwenden – es sei denn, bei Ihnen ist CygWin installiert.

Wenn Sie alle Schritte bis zu diesem Punkt ausgeführt haben, besitzen Sie noch kein fertiges Zertifikat, sondern eine Anforderung zur Signatur des Zertifikats. Wenn Sie das Zertifikat für einen öffentlichen Webserver einsetzen möchten, müssen Sie diese Anforderung nun an eine vertrauenswürdige Zertifizierungsstelle wie VeriSign oder Thawte senden und von ihr signieren lassen.

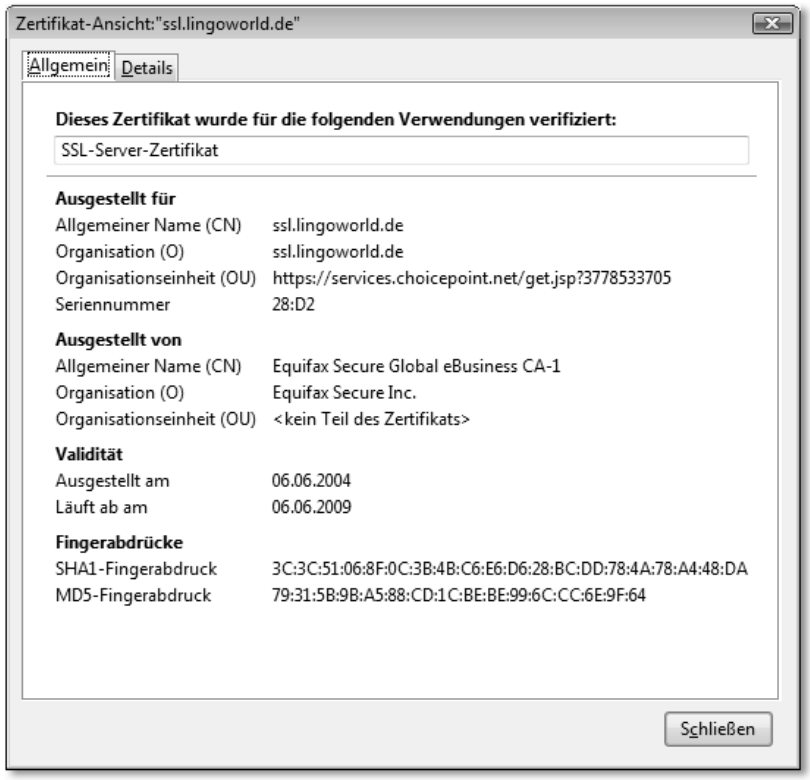



**Abbildung 10.2** Darstellung eines zweigliedrigen Zertifizierungspfades im Firefox

Ein Browser, der das Zertifikat überprüft, geht dabei nach einem sogenannten *Zertifizierungspfad* (Certification Chain) vor: Er überprüft zunächst den Aussteller des gelieferten Zertifikats. Wenn dieser unbekannt ist, wird wiederum der Aussteller von dessen Zertifikat gesucht usw., bis eine vertrauenswürdige Zertifizierungsstelle gefunden wird (oder bis keine weiteren Aussteller mehr verfügbar sind). Abbildung 10.2 zeigt, wie ein solcher Zertifizierungspfad im Firefox dargestellt wird.

Zu Testzwecken oder für einen geschlossenen Nutzerkreis – z. B. im Intranet – können Sie das Zertifikat aber auch einfach selbst signieren. Das funktioniert folgendermaßen:

```
# openssl x509 -in mynet.csr -out mynet.cert -req -signkey \
mynet.key -days 365
Loading 'screen' into random state - done
Signature ok
subject=/C=de/ST=NRW/L=Koeln/O=MyNet
```

```
GmbH/CN=www.mynet.de/emailAddress=info@mynet.de
Getting Private key
```

Wenn Sie möchten, dass Benutzer mit dem Internet Explorer 4.x auf Ihre gesicherte Site zugreifen können, müssen Sie den Schlüssel nun DER-codieren. Dies geschieht mit folgendem Befehl:

```
# openssl x509 -in mynet.cert -out mynet.der.crt -outform DER
```

Damit haben Sie die beiden Dateien *mynet.cert* (das Zertifikat) und *mynet.key* (den öffentlichen Schlüssel) beziehungsweise entsprechende Dateien mit Ihrem eigenen Domain-Namen erstellt. Diese müssen mithilfe der Apache-Konfigurationsdirektiven `SSLCertificateFile` (Zertifikat) beziehungsweise `SSLCACertificateKeyFile` (Schlüssel) eingebunden werden.

### 10.1.2 SSL-Grundkonfiguration

Eine ausführliche Referenz aller `mod_ssl`-Direktiven finden Sie in Abschnitt 10.1.3, »mod_ssl-Umgebungsvariablen«. Hier geht es dagegen um eine vollständige Beispielkonfiguration, die es Ihnen ermöglicht, auf ein und demselben Server eine ungesicherte Website auf Port 80 und eine verschlüsselte auf Port 443 zu betreiben.

Zunächst einmal müssen Sie natürlich das Modul `mod_ssl` laden. Auf UNIX-Systemen wird es grundsätzlich mitgeliefert; bei Windows kommt es darauf an, dass Sie die richtige Version installieren.

Mit der üblichen `LoadModule`-Direktive wird `mod_ssl` nun eingebunden:

```
LoadModule ssl_module modules/mod_ssl.so
```

Die Standardversion der *httpd.conf* ab Apache 2.2 enthält die (standardmäßig auskommentierte) Zeile:

```
Include extra/httpd-ssl.conf
```

Sobald Sie das Kommentarzeichen der `Include`-Direktive entfernen und gegebenenfalls auch die passende `LoadModule`-Direktive freischalten, ist `mod_ssl` aktiv.

Das Erste, was Sie für eine eigene SSL-Konfiguration hinzufügen müssen, ist eine zusätzliche `Listen`-Direktive – in aller Regel für den HTTPS-Standard-Port 443:

```
Listen 443
```

Als Nächstes wird der folgende Mindestsatz an Direktiven zur allgemeinen SSL-Grundkonfiguration benötigt:

```
SSLMutex default
SSLRandomSeed startup builtin
SSLSessionCache none
```

Die einzelnen Direktiven werden im Folgenden genau erläutert; grob gesprochen handelt es sich um die Definition eines Sperrverfahrens zur Serialisierung bestimmter SSL-Prozesse (`SSLMutex`), die Methode zur Bestimmung eines Random-Seeds und der Startzahl für die Berechnung von Zufallszahlen (`SSLRandomSeed`) sowie um die Einstellung eines Caches für SSL-Sessions (`SSLSessionCache`).

Nun wird für Port 443 ein virtueller Host definiert; innerhalb des `<VirtualHost>`-Containers befinden sich die Direktiven für die SSL-Funktionalität. Der entsprechende Abschnitt sieht beispielsweise so aus:

```
<VirtualHost *:443>
  SSLEngine On
  SSLCertificateFile conf/ssl/mynet.cert
  SSLCertificateKeyFile conf/ssl/mynet.key
  DocumentRoot /usr/local/apache2/sichere_seiten
</VirtualHost>
```

Statt `*:443` können Sie übrigens auch `_default_:443` schreiben. Beide Varianten sorgen dafür, dass die `<VirtualHost>`-Definition für Port 443 unter sämtlichen Hostnamen und IP-Adressen gilt, unter denen der Server erreichbar ist. Näheres zur Definition virtueller Hosts finden Sie in einem eigenen Abschnitt in Kapitel 12, »Skalierung und Performance-Tuning«.

**Server Name Indication (SNI)**

`mod_ssl` funktionierte in früheren Versionen von Apache nicht mit namensbasierten virtuellen Hosts (siehe Kapitel 12, »Skalierung und Performance-Tuning«): Da der Hostname des Servers einen Teil des Server-Zertifikats bildet, kann pro IP-Adresse nur ein Zertifikat mit einem bestimmten Hostnamen verwendet werden.

Es gibt jedoch ein (von den meisten aktuellen Browsern unterstütztes) Verfahren namens *Server Name Indication* (SNI). Es ist in RFC 3546 dokumentiert und erlaubt Ihnen, für jeden einzelnen namensbasierten virtuellen Host ein eigenes Zertifikat einzurichten, sodass die lange geltende Beschränkung auf einen Hostnamen pro IP-Adresse aufgehoben wird.

Seit Version 2.2.12 ist SNI standardmäßig in Apache enthalten, sofern mindestens OpenSSL 0.9.8i verwendet wird. Für ältere Versionen gibt es einen Patch, der SNI nachrüstet; in künftigen Versionen soll es fest eingebaut werden. Den Patch finden Sie unter *https://issues.apache.org/bugzilla/attachment.cgi?id=19676*. Was die Konfiguration angeht, brauchen Sie eigentlich nur nach der obigen Anleitung mehrere Zertifikate zu erstellen und dann in jedem `<VirtualHost>`-Container das zum `ServerName` passende Zertifikat anzugeben.

Die Direktive SSLEngine schaltet die SSL-Funktion ein. SSLCertificateFile und SSLCertificateKeyFile wurden bereits erwähnt; sie binden das Zertifikat beziehungsweise den öffentlichen Schlüssel ein. In diesem Beispiel wurden die entsprechenden Dateien in das Verzeichnis ssl unterhalb des *conf*-Verzeichnisses kopiert.

Wenn Sie die DocumentRoot-Direktive weglassen, werden über die gesicherte Verbindung dieselben Inhalte veröffentlicht wie über die ungesicherte. Dies ist in den meisten Fällen nicht besonders sinnvoll.

Starten Sie nun Ihren Webserver neu, und geben Sie in Ihren Browser eine Adresse wie *https://www.mynet.de* ein. Nachdem der Browser sich über das ungültige (da selbst signierte) Zertifikat beschwert hat, sollte die Startseite der DocumentRoot erscheinen, die Sie im <VirtualHost>-Container definiert haben. Damit ist Ihre SSL-Konfiguration im Prinzip beendet; die Beschreibung aller Direktiven im nächsten Abschnitt dient dem Fine-Tuning.

### 10.1.3 mod_ssl-Umgebungsvariablen

Sobald Sie mit mod_ssl arbeiten, werden zusätzlich zu den CGI-Umgebungsvariablen (siehe Kapitel 14, »CGI«) noch einige weitere definiert. Sie finden sie in Tabelle 10.1. Viele der Variablen beziehen sich auf das Client-Zertifikat, das heißt das Zertifikat, das der Browser an den Server sendet. In aller Regel besitzen Clients aber gar kein Zertifikat, sodass diese Variablen oft leer sind.

Bei vier Variablen in der Liste enthält der Name ein Sternchen (*). Sie müssen es durch eine der X.509-Verzeichniskomponenten ersetzen. Dies sind die folgenden Werte: C, ST, L, O, OU, CN, T, I, G, S, D, UID und Email. Die Variable SSL_SERVER_S_DN_C liefert also beispielsweise das Land des Server-Zertifikats-Inhabers.

| Variable | Beschreibung |
|---|---|
| HTTPS | Wird HTTPS verwendet? |
| SSL_PROTOCOL | Version des SSL-Protokolls (SSLv2, SSLv3, TLSv1) |
| SSL_SESSION_ID | hexadezimale SSL-Session-ID |
| SSL_CIPHER | Verschlüsselungsverfahren |
| SSL_CIPHER_EXPORT | Ist das Verschlüsselungsverfahren ein Exportschlüssel? |
| SSL_CIPHER_USEKEYSIZE | tatsächliche Schlüsselbreite in Bit |
| SSL_CIPHER_ALGKEYSIZE | mögliche Schlüsselbreite in Bit |
| SSL_VERSION_INTERFACE | mod_ssl-Version |

**Tabelle 10.1** Zusätzliche SSL-Umgebungsvariablen

| Variable | Beschreibung |
|---|---|
| SSL_VERSION_LIBRARY | OpenSSL-Version |
| SSL_CLIENT_M_VERSION | Version des Client-Zertifikats |
| SSL_CLIENT_M_SERIAL | Seriennummer des Client-Zertifikats |
| SSL_CLIENT_S_DN | Anbieter-DN des Client-Zertifikats |
| SSL_CLIENT_S_DN_* | Komponente der Anbieter-DN des Client-Zertifikats |
| SSL_CLIENT_I_DN | Aussteller-DN des Client-Zertifikats |
| SSL_CLIENT_I_DN_* | Komponente der Aussteller-DN des Client-Zertifikats |
| SSL_CLIENT_V_START | Gültigkeitsbeginn des Client-Zertifikats |
| SSL_CLIENT_V_END | Gültigkeitsende des Client-Zertifikats |
| SSL_CLIENT_A_SIG | Signatur-Algorithmus des Client-Zertifikats |
| SSL_CLIENT_A_KEY | Public-Key-Algorithmus des Client-Zertifikats |
| SSL_CLIENT_CERT | PEM-codiertes Client-Zertifikat |
| SSL_CLIENT_CERT_CHAIN*n* | PEM-codiertes Zertifikat (Folgenummer *n*) im Zertifizierungspfad |
| SSL_CLIENT_VERIFY | NONE, SUCCESS, GENEROUS, FAILED: Grund |
| SSL_SERVER_M_VERSION | Version des Server-Zertifikats |
| SSL_SERVER_M_SERIAL | Seriennummer des Server-Zertifikats |
| SSL_SERVER_S_DN | Anbieter-DN des Server-Zertifikats |
| SSL_SERVER_S_DN_* | Komponente im Anbieter-DN des Server-Zertifikats |
| SSL_SERVER_I_DN | Aussteller-DN des Server-Zertifikats |
| SSL_SERVER_I_DN_* | Komponente im Aussteller-DN des Server-Zertifikats |
| SSL_SERVER_V_START | Gültigkeitsbeginn des Server-Zertifikats |
| SSL_SERVER_V_END | Gültigkeitsende des Server-Zertifikats |
| SSL_SERVER_A_SIG | Signatur-Algorithmus des Server-Zertifikats |
| SSL_SERVER_A_KEY | Public-Key-Algorithmus des Server-Zertifikats |
| SSL_SERVER_CERT | PEM-codiertes Server-Zertifikat |

**Tabelle 10.1** Zusätzliche SSL-Umgebungsvariablen (Forts.)

## 10.2 mod_ssl-Direktiven

Hier finden Sie die vollständige Referenz sämtlicher Direktiven des Moduls `mod_ssl`. Wie Sie im vorigen Abschnitt bemerkt haben werden, brauchen Sie für die grundlegende SSL-Einrichtung nicht allzu viele von ihnen zu benutzen. Die meisten dienen der Lösung sehr spezieller Probleme.

### 10.2.1 Standard-Direktiven

Hier werden zunächst alle `mod_ssl`-Direktiven beschrieben, die dem Betrieb des eigenen SSL gesicherten Webservers dienen. Die speziellen Direktiven für den SSL-Proxy-Betrieb finden Sie in Abschnitt 10.2.2, »mod_ssl-Proxy-Direktiven«.

#### SSLCACertificateFile

Datei mit CA-Zertifikaten zur Client-Authentifizierung

| | |
|---|---|
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSLCACertificateFile Dateipfad` |
| **Standardwert** | nicht gesetzt |

Diese Direktive gibt eine Datei an, in der Zertifikate zur Überprüfung der Clients abgelegt werden sollen. In der Datei stehen die PEM-codierten Zertifikate einfach untereinander. Da Clients, wie bereits erwähnt, selten Zertifikate einsetzen, kommt diese Direktive in der Praxis nicht allzu häufig zum Einsatz. Beispiel:

```
SSLCACertificateFile /usr/local/apache2/conf/ssl/client.crt
```

#### SSLCACertificatePath

Verzeichnispfad für CA-Zertifikate zur Client-Authentifizierung

| | |
|---|---|
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSLCACertificatePath Verzeichnispfad` |
| **Standardwert** | nicht gesetzt |

Zusätzlich oder alternativ zu der einzelnen Datei für Client-Zertifikate, die mit `SSLCACertificateFile` eingebunden wird, können Sie mithilfe dieser Direktive ein ganzes Verzeichnis bestimmen, in dem solche Zertifikate liegen sollen. Beispiel:

```
SSLCACertificatePath /usr/local/apache2/conf/client-crts
```

Die Zertifikate müssen PEM-codiert sein. Sie werden nicht über ihren eigentlichen Dateinamen (`NAME.crt`) angesprochen, sondern es wird für jede Zertifikatsdatei ein Symlink erwartet, dessen Name der Hash-Wert des Zertifikats ist. Diesen Wert können Sie (hier für das Beispielzertifikat `client.crt`) folgendermaßen erzeugen:

```
# openssl x509 -noout -hash <client.crt
7147702e
```

Den Wert können Sie nun verwenden, um den Namen des symbolischen Links zu bilden, wobei Sie noch `.0` anhängen müssen:

```
# ln -s client.crt 7147702e.0
```

Unter Windows funktioniert diese Direktive nicht, weil das Betriebssystem keine Symlinks kennt. Hier müssen Sie sich mit `SSLCACertificateFile` behelfen.

#### SSLCADNRequestFile

Liste von CAs, die dem Client zur Zertifikatsanforderung übermittelt wird

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSLCADNRequestFile Dateipfad` |
| **Standardwert** | nicht gesetzt |

Diese optionale Direktive verweist auf eine Datei, die eine Liste PEM-codierter CAs enthält. Diese Liste wird an Clients gesendet, damit diese aus ihren Zertifikaten eines auswählen können, das von einer vom Server akzeptierten CA unterzeichnet wurde.

Wenn Sie `SSLCADNRequestFile` nicht setzen, wird dem Client der Wert von `SSLCACertificateFile` beziehungsweise `SSLCACertificatePath` gesendet. Das heißt, dass diese Einstellung nur benötigt wird, wenn den Clients CAs genannt werden sollen, die in Wirklichkeit nicht zur Zertifikatsüberprüfung herangezogen werden. Dies ist beispielsweise bei mehrstufigen Zertifizierungspfaden der Fall. Hier ein Beispiel:

```
SSLCADNRequestFile /usr/local/apache2/conf/client-ca.crt
```

#### SSLCADNRequestPath

Verzeichnis mit CAs, die dem Client zur Zertifikatsanforderung übermittelt werden

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSLCADNRequestFile Verzeichnispfad` |
| **Standardwert** | nicht gesetzt |

Auch mithilfe dieser Direktive können Sie alternative CAs angeben, die dem Client bei der Zertifikatsanforderung übermittelt werden sollen. Als Wert müssen Sie den Pfad eines Verzeichnisses mit Symlinks angeben, die nach demselben Verfahren wie bei `SSLCACertificatePath` gebildet werden. Beispiel:

```
SSLCADNRequestPath /usr/local/apache2/conf/client-ca/
```

#### SSLCARevocationCheck

CRL-basierte Zertifikatsüberprüfung aktivieren

| | |
|---|---|
| **Seit Version** | 2.3-beta |
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSLCARevocationCheck chain\|leaf\|none` |
| **Standardwert** | `none` |

Wenn Sie einen der Werte `chain` oder `leaf` einstellen, wird die Überprüfung von Client-Zertifikaten anhand von Certificate Revocation Lists (CRL), also Listen zurückgezogener Zertifikate, aktiviert. Dazu muss mindestens eine der Direktiven `SSLCARevocationFile` oder `SSLCARevocationPath` gesetzt sein. Seit Version 2.3.15 müssen an den angegebenen Stellen auch tatsächlich CRLs vorhanden sein, andernfalls scheitert die Validierung der Zertifikate mit der Fehlermeldung »unable to get certificate CRL«.

Der Wert `chain` bedeutet, dass die gesamte Kette von Zertifikaten bis zur CA untersucht wird, während `leaf` nur das verwendete Zertifikat selbst, aber nicht dessen Vorfahren prüft. Die empfehlenswertere Einstellung ist daher `chain`.

#### SSLCARevocationFile

Datei mit ungültigen CA-CRLs

| | |
|---|---|
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSLCARevocationFile Dateipfad` |
| **Standardwert** | nicht gesetzt |

In der Datei, die Sie mithilfe dieser Direktive angeben, können Sie Certificate Revocation Lists (CRLs) für Client-Zertifikate speichern. Dabei handelt es sich um Listen ungültiger oder zurückgezogener Zertifikate. Die Datei enthält einfach untereinander die PEM-codierten Inhalte der CRL-Dateien. Beispiel:

```
SSLCARevocationFile /usr/local/apache2/conf/ssl/clients.crl
```

### SSLCARevocationPath

Verzeichnispfad für ungültige CA-CRLs

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | SSLCARevocationPath Verzeichnispfad |
| Standardwert | nicht gesetzt |

Diese Direktive können Sie alternativ oder zusätzlich zu SSLCACertificateFile verwenden. Sie definiert ein Verzeichnis für CRL-Dateien. Beispiel:

```
SSLCARevocationPath /usr/local/apache2/conf/ssl/client-crls/
```

Ähnlich wie bei SSLCACertificatePath müssen Sie hier die Hash-Werte der CRL-Dateien berechnen. Anschließend können Sie nach dem weiter oben beschriebenen Verfahren Symlinks erzeugen: Ihr Name ist der Hash-Wert, gefolgt von .0.

### SSLCertificateChainFile

Datei mit dem Server-CA-Zertifizierungspfad

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | SSLCertificateChainFile Dateipfad |
| Standardwert | nicht gesetzt |

Diese Direktive ermöglicht die Angabe einer Datei, die den gesamten Zertifizierungspfad des Server-Zertifikats enthalten kann. Dies ist eine Alternative zur Platzierung dieser Datei in dem Verzeichnis, das mittels SSLCACertficatePath (siehe oben) angegeben wurde. Letzteres führt nämlich dazu, dass auch Client-Zertifikate mit demselben Zertifizierungspfad automatisch akzeptiert werden – dies ist nicht immer wünschenswert. Beispiel:

```
SSLCertificateChainFile /usr/local/apache2/conf/ssl/ca.crt
```

### SSLCertificateFile

Datei mit X.509-Zertifikaten

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | SSLCertificateFile Dateipfad |
| Standardwert | nicht gesetzt |

Mit dieser Direktive wird das PEM-codierte Server-Zertifikat angegeben. Optional kann dieselbe Datei auch den DSA- oder RSA-codierten Private Key für dieses Zertifikat enthalten – in diesem Fall wird beim Start von Apache die Passphrase abgefragt. Wenn Sie sowohl ein DSA- als auch ein RSA-Zertifikat haben, können Sie diese Direktive zweimal hintereinander verwenden. Beispiel:

```
SSLCertificateFile /usr/local/apache2/conf/ssl/server.crt
```

#### SSLCertificateKeyFile

Datei mit privatem Schlüssel

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | SSLCertificateKeyFile Dateipfad |
| Standardwert | nicht gesetzt |

Diese Direktive verweist auf den privaten Schlüssel, der zum Server-Zertifikat gehört. Sie brauchen diese Direktive nur zu verwenden, wenn der Schlüssel nicht in der Zertifikatsdatei (SSLCertificateFile) enthalten ist. Die Verwendung getrennter Dateien ist aber aus Sicherheitsgründen zu bevorzugen. Sie können dieselbe Direktive zweimal benutzen, wenn Sie sowohl einen DSA- als auch einen RSA-Schlüssel besitzen. Beispiel:

```
SSLCertificateFile /usr/local/apache2/conf/ssl/server.key
```

#### SSLCipherSuite

Verfügbare Verschlüsselungsalgorithmen für SSL-Handshake

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SSLCipherSuite Algorithmenliste |
| Standardwert | ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:<br>+LOW:+SSLv2:+EXP |

Diese Direktive enthält in einer durch Doppelpunkte getrennten Liste alle Verschlüsselungsalgorithmen, auf die sich Server und Client während des SSL-Handshakes einigen können. Die Liste besteht aus folgenden Komponenten:

- **Schlüsselaustausch-Algorithmus**
  Bestimmt die mehr oder weniger sichere Methode, mit der sich Client und Server auf einen Schlüssel einigen. Folgende Werte sind möglich:

- `kRSA`: RSA-Schlüsselaustausch
- `kDHr`: Diffie-Hellman-Schlüsselaustausch mit RSA-Schlüssel
- `kDHd`: Diffie-Hellman-Schlüsselaustausch mit DSA-Schlüssel
- `kEDH`: Ephemeral-Diffie-Hellman-Schlüsselaustausch (temporäre Schlüssel ohne Zertifikat)

- **Authentifizierungsalgorithmus**
  Legt den Algorithmus zur Übermittlung verschlüsselter Authentifizierungsdaten fest. Mögliche Werte:
  - `aNULL`: keine Authentifizierung
  - `aRSA`: RSA-Authentifizierung
  - `aDSS`: DSS-Authentifizierung
  - `aDH`: Diffie-Hellman-Authentifizierung
- **Verschlüsselungsalgorithmus**
  Dies ist die Methode, mit der der Schlüssel codiert wird:
  - `eNULL`: keine Codierung
  - `DES`: DES-Codierung
  - `3DES`: Triple-DES-Codierung
  - `RC4`: RC4-Codierung
  - `RC2`: RC2-Codierung
  - `IDEA`: IDEA-Codierung
- **MAC-Digest-Algorithmus**
  Der Algorithmus, der für Message-Digests zur Integritätsprüfung eingesetzt wird:
  - `MD5`: MD5-Hash-Funktion
  - `SHA1`: SHA1-Hash-Funktion
  - `SHA`: SHA-Hash-Funktion
- **Alias-Bezeichnungen**
  Sie können folgende Werte benutzen, um die Namen mehrerer Algorithmen nach verschiedenen Kriterien zusammenzufassen:
  - `SSLv2`: alle Algorithmen, die SSL 2.0 unterstützt
  - `SSLv3`: alle Algorithmen, die SSL 3.0 unterstützt
  - `TLSv1`: alle Algorithmen, die TLS 1.0 unterstützt
  - `EXP`: alle exportfähigen Algorithmen (schwache Verschlüsselung; der Begriff »exportfähig« bezieht sich auf Exportbeschränkungen für Verschlüsselungsverfahren mit hoher Bit-Breite)

- `EXPORT40`: nur 40-Bit-Exportalgorithmen
- `EXPORT56`: nur 56-Bit-Exportalgorithmen
- `LOW`: alle schwachen, aber nicht exportfähigen Algorithmen (Single-DES)
- `MEDIUM`: alle Algorithmen mit 128-Bit-Verschlüsselung
- `HIGH`: alle Triple-DES-Algorithmen
- `RSA`: alle Algorithmen mit RSA-Schlüsselaustausch
- `DH`: alle Algorithmen mit Diffie-Hellman-Schlüsselaustausch
- `EDH`: alle Algorithmen mit Ephemeral-Diffie-Hellman-Schlüsselaustausch
- `ADH`: alle Algorithmen mit Anonymous-Diffie-Hellman-Schlüsselaustausch
- `DSS`: alle Algorithmen mit DSS-Authentifizierung
- `NULL`: alle Algorithmen ohne Verschlüsselung

Die Liste kann aus all diesen Elementen zusammengefügt werden. Für jede Komponente sind folgende Optionen möglich:

- **Name ohne Präfix**
  Der Algorithmus wird zur Liste hinzugefügt.
- **Präfix** `+`
  Der Algorithmus wird an der aktuellen Position (die seine Priorität bestimmt) zur Liste hinzugefügt.
- **Präfix** `-`
  Der Algorithmus wird aus der Liste entfernt.
- **Präfix** `!`
  Der Algorithmus wird unwiderruflich aus der Liste entfernt – erneutes Hinzufügen ist nicht mehr möglich.

Betrachten Sie als Beispiel den Standardwert der Direktive:

```
SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP
```

Zunächst werden über das Schlüsselwort `ALL` sämtliche Algorithmen zugelassen; die Anonymous-Diffie-Hellman-Verfahren werden anschließend mittels `!ADH` ausgeschlossen. Anschließend werden mit `+` einige Algorithmengruppen hinzugefügt. Dies legt die Reihenfolge fest, in der sie der Server dem Client vorschlägt. Gewählt wird jeweils der erste Algorithmus einer Kategorie, auf den sich Server und Client einigen können.

In Tabelle 10.2 sehen Sie eine vollständige Liste der kombinierten RSA- und Diffie-Hellman-Algorithmen, die in der Praxis von SSL unterstützt werden. »Auth.« steht natürlich für den Authentifizierungsalgorithmus, MAC ist das Message-Digest-Verfahren. Die Zahlen in Klammern geben die Bit-Breiten der Schlüssel an.

| Name | Protokoll | Schlüssel | Auth. | Codierung | MAC | Export |
|---|---|---|---|---|---|---|
| DES-CBC3-SHA | SSLv3 | RSA | RSA | 3DES(168) | SHA1 | |
| DES-CBC3-MD5 | SSLv2 | RSA | RSA | 3DES(168) | MD5 | |
| IDEA-CBC-SHA | SSLv3 | RSA | RSA | IDEA(128) | SHA1 | |
| RC4-SHA | SSLv3 | RSA | RSA | RC4(128) | SHA1 | |
| RC4-MD5 | SSLv3 | RSA | RSA | RC4(128) | MD5 | |
| IDEA-CBC-MD5 | SSLv2 | RSA | RSA | IDEA(128) | MD5 | |
| RC2-CBC-MD5 | SSLv2 | RSA | RSA | RC2(128) | MD5 | |
| RC4-MD5 | SSLv2 | RSA | RSA | RC4(128) | MD5 | |
| DES-CBC-SHA | SSLv3 | RSA | RSA | DES(56) | SHA1 | |
| RC4-64-MD5 | SSLv2 | RSA | RSA | RC4(64) | MD5 | |
| DES-CBC-MD5 | SSLv2 | RSA | RSA | DES(56) | MD5 | |
| EXP-DES-CBC-SHA | SSLv3 | RSA(512) | RSA | DES(40) | SHA1 | X |
| EXP-RC2-CBC-MD5 | SSLv3 | RSA(512) | RSA | RC2(40) | MD5 | X |
| EXP-RC4-MD5 | SSLv3 | RSA(512) | RSA | RC4(40) | MD5 | X |
| EXP-RC2-CBC-MD5 | SSLv2 | RSA(512) | RSA | RC2(40) | MD5 | X |
| EXP-RC4-MD5 | SSLv2 | RSA(512) | RSA | RC4(40) | MD5 | X |
| NULL-SHA | SSLv3 | RSA | RSA | None | SHA1 | |
| NULL-MD5 | SSLv3 | RSA | RSA | None | MD5 | |
| ADH-DES-CBC3-SHA | SSLv3 | DH | None | 3DES(168) | SHA1 | |
| ADH-DES-CBC-SHA | SSLv3 | DH | None | DES(56) | SHA1 | |
| ADH-RC4-MD5 | SSLv3 | DH | None | RC4(128) | MD5 | |
| EDH-RSA-DES-CBC3-SHA | SSLv3 | DH | RSA | 3DES(168) | SHA1 | |
| EDH-DSS-DES-CBC3-SHA | SSLv3 | DH | DSS | 3DES(168) | SHA1 | |
| EDH-RSA-DES-CBC-SHA | SSLv3 | DH | RSA | DES(56) | SHA1 | |
| EDH-DSS-DES-CBC-SHA | SSLv3 | DH | DSS | DES(56) | SHA1 | |
| EXP-EDH-RSA-DES-CBC-SHA | SSLv3 | DH(512) | RSA | DES(40) | SHA1 | X |
| EXP-EDH-DSS-DES-CBC-SHA | SSLv3 | DH(512) | DSS | DES(40) | SHA1 | X |
| EXP-ADH-DES-CBC-SHA | SSLv3 | DH(512) | None | DES(40) | SHA1 | X |
| EXP-ADH-RC4-MD5 | SSLv3 | DH(512) | None | RC4(40) | MD5 | X |

**Tabelle 10.2** SSL-Verschlüsselungsalgorithmen

### SSLCryptoDevice

Einen Kryptografie-Hardwarebeschleuniger für SSL einsetzen

| Seit Version | 2.1-beta |
|---|---|
| Modul | mod_ssl |

| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
|---|---|
| Syntax | SSLCryptoDevice Gerätename |
| Standardwert | Builtin |

Mithilfe dieser Direktive lässt sich die Verwendung eines Hardwarebeschleunigers für kryptografische Aufgaben steuern. Ihre OpenSSL-Version muss den Einsatz solcher Geräte unterstützen. Seit Version 0.9.7 ist dies automatisch der Fall, während es von der Vorversion 0.9.6 spezielle »Engine«-Varianten gibt. Außerdem muss mod_ssl mit dem speziellen Parameter -DSSL_ENGINE_EXPERIMENTAL kompiliert werden, damit diese Option zur Verfügung steht.

Welche Gerätenamen zur Verfügung stehen, können Sie mithilfe der folgenden Kommandozeilenanweisung herausfinden:

```
$ openssl engine
(dynamic) Dynamic engine loading support
(cswift) CryptoSwift hardware engine support
(chil) nCipher hardware engine support
(atalla) Atalla hardware engine support
(nuron) Nuron hardware engine support
(ubsec) UBSEC hardware engine support
(aep) Aep hardware engine support
(sureware) SureWare hardware engine support
(4758cca) IBM 4758 CCA hardware engine support
```

Das folgende Beispiel stellt die Nuron-Hardware-Engine ein:

```
SSLCryptoDevice nuron
```

### SSLEngine

SSL-Funktionalität ein-/ausschalten

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | SSLEngine On\|Off |
| Standardwert | Off |

Auch wenn Sie mod_ssl eingebunden haben, ist seine Funktionalität zunächst einmal standardmäßig deaktiviert. Wenn Sie SSL für den ganzen Server oder einen bestimmten virtuellen Host einschalten möchten, müssen Sie im entsprechenden Kontext folgende Konfigurationsanweisung verwenden:

```
SSLEngine On
```

### SSLFIPS

SSL-FIPS-Modus ein-/ausschalten

| Modul | mod_ssl |
|---|---|
| Kontext | Server |
| Syntax | SSLFIPS On\|Off |
| Standardwert | Off |

FIPS 140-2 (Federal Information Processing Standard) ist ein Computersicherheitsstandard der US-Regierung. Wenn eine OpenSSL-Version mit FIPS-Unterstützung verwendet wird, können Sie `mod_ssl` mithilfe der Angabe `SSLFIPS On` in einem FIPS-konformen Modus betreiben, der die Vorgaben des Standards einhält. Näheres zu FIPS 140-2 erfahren Sie unter *http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf* (offizielle Spezifikation im PDF-Format). Das Dokument finden Sie auch auf der DVD zum Buch.

### SSLHonorCipherOrder

Verschlüsselungsalgorithmen nach Präferenzreihenfolge des Servers auswählen

| Seit Version | 2.1-beta |
|---|---|
| Modul | mod_ssl |
| Kontext | Server, <VirtualHost> |
| Syntax | SSLHonorCipherOrder On\|Off |
| Standardwert | nicht gesetzt |

Wenn während eines SSLv3- oder TLSv1-Handshakes mehrere Verschlüsselungsalgorithmen verfügbar sind, erfolgt deren Auswahl normalerweise nach der Präferenz des Clients. Wenn Sie diese Direktive explizit auf `On` setzen, erfolgt die Auswahl dagegen nach den Vorlieben des Servers:

```
SSLHonorCipherOrder On
```

Beachten Sie, dass diese Option nur mit OpenSSL ab Version 0.9.7 verfügbar ist.

### SSLMutex

Semaphore (Lock) zum gegenseitigen Sperren von SSL-Operationen

| Version | 2.0 bis 2.2 |
|---|---|
| Modul | mod_ssl |
| Kontext | Server |

| Syntax | SSLMutex Typ |
|---|---|
| Standardwert | none |

Einige SSL-Operationen können nur nacheinander ausgeführt werden. Bei einem nebenläufigen Programm wie Apache ist die Gefahr natürlich groß, dass zwei verschiedene Child-Prozesse oder Threads dennoch versuchen, gleichzeitig auf dieselbe Funktion zuzugreifen – die Folge ist eine Race-Condition, sodass beide Anfragen blockieren. Mithilfe der Direktive `SSLMutex` können Sie das Verfahren festlegen, mit dem eine Sperre errichtet werden kann, die solche Zugriffe serialisiert. Die Optionen ähneln denjenigen der Direktive `AcceptMutex` (siehe Kapitel 6, »Grundkonfiguration«), die ein ähnliches Problem mit `Accept`-Aufrufen für Sockets löst. Nähere Erläuterungen zur Funktionsweise und Kompatibilität der verschiedenen Optionen finden Sie in der Beschreibung dieser Direktive. Auch die notwendigen APR-Compiler-Optionen sind übrigens identisch.

Im Einzelnen können folgende Einstellungen vorgenommen werden:

- `none | no`. Keine Sperre verwenden (nicht empfehlenswert).
- `posixsem`. Als Sperre wird eine POSIX-Semaphore verwendet.
- `sysvsem`. Eine System-V-Semaphore wird benutzt.
- `sem`. `mod_ssl` wählt die besser zur jeweiligen Plattform passende Semaphoren-Art. Wenn Ihr System weder System-V- noch POSIX-Semaphoren unterstützt, verhindert diese Einstellung den Start von Apache und führt zu einer Fehlermeldung. Dies ist beispielsweise unter Windows der Fall.
- `fcntl:/Dateipfad`. Als Sperre wird die angegebene Datei benutzt; die Steuerung erfolgt über den Systemaufruf `fcntl()`.
- `flock:/Dateipfad`. Hier ist die Sperre ebenfalls eine Datei, wird aber mittels `flock()` angesprochen.
- `file:/Dateipfad`. `mod_ssl` verwendet auf jeden Fall eine Datei als Sperre, trifft aber selbst die Entscheidung, ob `flock()` oder `fcntl()` besser geeignet ist.
- `default | yes`. Diese Einstellung verwendet auf jeden Fall eine Sperre, überlässt die Auswahl der Implementierung aber vollständig dem Modul. Dies ist die empfohlene Einstellung unter Windows.

In Apache 2.4 existiert diese Direktive nicht mehr. Stattdessen wird für den SSL-Mutex die Einstellung der Core-Direktive `Mutex` verwendet (siehe Kapitel 6, »Grundkonfiguration«).

### SSLOCSPEnable

OCSP-Validierung von Client-Zertifikaten aktivieren

| Seit Version | 2.3-beta mit OpenSSL ab Version 0.9.7 |
|---|---|
| Modul | mod_ssl |
| Kontext | Server, <VirtualHost> |
| Syntax | SSLOCSPEnable On\|Off |
| Standardwert | Off |

Das Online Certificate Status Protocol (OCSP) ist ein in RFC 2560 spezifiziertes Protokoll, mit dem SSL-fähige Software den Status von Zertifikaten abfragen kann, insbesondere bezüglich ihrer Ungültigkeit (Revocation). Das Protokoll ist in OpenSSL ab Version 0.9.7 implementiert. Wenn Sie mindestens diese Version verwenden und die Direktive SSLOCSPEnable auf On setzen, werden Client-Zertifikate nach Abschluss aller anderen Überprüfungen (einschließlich CRL) auch per OCSP geprüft. Als Gegenstelle dient entweder der Aussteller des Zertifikats selbst, oder sie wird per Konfiguration angegeben (siehe SSLOCSPDefaultResponder und SSLOCSPOverrideResponder).

### SSLOCSPDefaultResponder

Standard-Gegenstelle für OCSP-Überprüfungen

| Seit Version | 2.3-beta mit OpenSSL ab Version 0.9.7 |
|---|---|
| Modul | mod_ssl |
| Kontext | Server, <VirtualHost> |
| Syntax | SSLOCSPDefaultResponder URI |
| Standardwert | nicht gesetzt |

Die durch diese Direktive angegebene URI wird als Standard-Gegenstelle bei der OCSP-Überprüfung verwendet. Dies geschieht im Normalfall nur, wenn im zu überprüfenden Zertifikat keine OCSP-URI angegeben ist, es sei denn, Sie aktivieren SSLOCSPOverrideResponder.

### SSLOCSPOverrideResponder

Verwendung der Default-Gegenstelle für OCSP-Überprüfungen erzwingen

| Seit Version | 2.3-beta mit OpenSSL ab Version 0.9.7 |
|---|---|
| Modul | mod_ssl |
| Kontext | Server, <VirtualHost> |
| Syntax | SSLOCSPOverrideResponder On\|Off |
| Standardwert | Off |

Normalerweise wird die durch `SSLOCSPDefaultResponder` angegebene Standard-Gegenstelle nur dann verwendet, wenn das überprüfte Zertifikat keine eigene OCSP-URI enthält. Wenn Sie `SSLOCSPOverrideResponder` auf `On` setzen, wird dagegen immer die Default-Gegenstelle verwendet.

#### SSLOCSPResponderTimeout

Timeout für OCSP-Abfragen

| | |
|---|---|
| **Seit Version** | 2.3-beta mit OpenSSL ab Version 0.9.7 |
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSLOCSPResponderTimeout Sekunden` |
| **Standardwert** | `10` |

Diese Direktive legt die maximale Wartezeit auf die Antwort einer OCSP-Gegenstelle in Sekunden fest. Beispiel:

```
SSLOCSPResponderTimeout 5
```

#### SSLOCSPResponseMaxAge

Akzeptiertes Höchstalter für OCSP-Antworten

| | |
|---|---|
| **Seit Version** | 2.3-beta mit OpenSSL ab Version 0.9.7 |
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSLOCSPResponseMaxAge Sekunden` |
| **Standardwert** | `-1` |

Diese Direktive bestimmt das zulässige Höchstalter einer OCSP-Antwort. Der Standardwert `-1` besagt, dass beliebig alte OCSP-Antworten akzeptiert werden, solange ihr `nextUpdate`-Feld in der Zukunft liegt.

#### SSLOCSPResponseTimeSkew

Maximaler Zeitversatz für OCSP-Antworten

| | |
|---|---|
| **Seit Version** | 2.3-beta mit OpenSSL ab Version 0.9.7 |
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSLOCSPResponseTimeSkew Sekunden` |
| **Standardwert** | `300` |

Diese Direktive bestimmt den maximalen akzeptierten Zeitversatz für OCSP-Antworten, gemessen anhand ihrer `thisUpdate`- und `nextUpdate`-Felder.

#### SSLUseStapling

Stapling von OCSP-Antworten in TLS-Handshakes aktivieren

| | |
|---|---|
| **Seit Version** | 2.3.3 mit OpenSSL ab Version 0.9.8h |
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSLUseStapling On\|Off` |
| **Standardwert** | `Off` |

Wenn Sie diese Direktive auf `On` setzen, schickt Apache beim TLS-Handshake eine OCSP-Antwort für sein eigenes Zertifikat mit. Dies wird als *OCSP-Stapling* gemäß RFC 6066 bezeichnet und befreit den Client von der Last, eine zusätzliche OCSP-Anfrage stellen zu müssen. Es funktioniert bisher allerdings nur für ein einzelnes Zertifikat, nicht für eine Chain, sodass der Nutzen des Staplings in seiner bisherigen Form begrenzt ist. Beachten Sie, dass Sie auch `SSLStaplingCache` konfigurieren müssen, um Stapling verwenden zu können.

#### SSLStaplingCache

Cache für OCSP-Stapling

| | |
|---|---|
| **Seit Version** | 2.3.3 mit OpenSSL ab Version 0.9.8h |
| **Modul** | `mod_ssl` |
| **Kontext** | Server |
| **Syntax** | `SSLStaplingCache Typ` |
| **Standardwert** | nicht gesetzt |

Mit dieser Direktive konfigurieren Sie den Cache, der für OCSP-Stapling zwingend benötigt wird. Die möglichen Werte entsprechen denjenigen für `SSLSessionCache`.

#### SSLStaplingErrorCacheTimeout

Sekunden, bis ungültige Antworten aus dem OCSP-Stapling-Cache entfernt werden

| | |
|---|---|
| **Seit Version** | 2.3.3 mit OpenSSL ab Version 0.9.8h |
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |

| Syntax | SSLStaplingErrorCacheTimeout Sekunden |
|---|---|
| Standardwert | 600 |

Diese Direktive gibt die Zeit in Sekunden an, bis fehlerhafte Antworten endgültig aus dem OCSP-Stapling-Cache gelöscht werden. Für die Cache-Dauer gültiger Antworten wird die Direktive `SSLStaplingErrorCacheTimeout` verwendet.

#### SSLStaplingFakeTryLater

Auf fehlgeschlagene OCSP-Stapling-Anfragen mit vermeintlichen »tryLater«-Antworten reagieren

| Seit Version | 2.3.3 mit OpenSSL ab Version 0.9.8h |
|---|---|
| Modul | mod_ssl |
| Kontext | Server, <VirtualHost> |
| Syntax | SSLStaplingFakeTryLater On\|Off |
| Standardwert | On |

Wenn diese Direktive eingeschaltet ist (Standard), beantwortet `mod_ssl` fehlgeschlagene OCSP-Stapling-Anfragen mit »tryLater«. Dies funktioniert nur, wenn auch `SSLStaplingReturnResponderErrors` aktiv ist.

#### SSLStaplingForceURL

OCSP-Responder-URI aus der AIA-Erweiterung des Zertifikats überschreiben

| Seit Version | 2.3.3 mit OpenSSL ab Version 0.9.8h |
|---|---|
| Modul | mod_ssl |
| Kontext | Server, <VirtualHost> |
| Syntax | SSLStaplingForceURL URI |
| Standardwert | nicht gesetzt |

Mit dieser Direktive können Sie die URI eines OCSP-Responders überschreiben, die in der authorityInfoAccess-Erweiterung (AIA) des Zertifikats steht. Dies kann nützlich sein, wenn die OCSP-Anfragen über einen Proxy empfangen werden.

#### SSLStaplingResponderTimeout

Timeout für OCSP-Stapling-Anfragen

| Seit Version | 2.3.3 mit OpenSSL ab Version 0.9.8h |
|---|---|
| Modul | mod_ssl |
| Kontext | Server, <VirtualHost> |

| Syntax | SSLStaplingResponseTimeout Sekunden |
|---|---|
| Standardwert | 10 |

Wenn mod_ssl andere OCSP-Responder für Stapling-Zwecke befragen muss, legt diese Direktive das Timeout für diesen Vorgang fest.

#### SSLStaplingResponseMaxAge

Maximales Alter von OCSP-Stapling-Antworten

| Seit Version | 2.3.3 mit OpenSSL ab Version 0.9.8h |
|---|---|
| Modul | mod_ssl |
| Kontext | Server, <VirtualHost> |
| Syntax | SSLStaplingResponseMaxAge Sekunden |
| Standardwert | -1 |

Diese Direktive bestimmt das zulässige Höchstalter einer OCSP-Stapling-Antwort. Der Standardwert -1 besagt, dass beliebig alte OCSP-Stapling-Antworten akzeptiert werden, solange ihr nextUpdate-Feld in der Zukunft liegt.

#### SSLStaplingResponseTimeSkew

Maximaler Zeitversatz für OCSP-Stapling-Antworten

| Seit Version | 2.3.3 mit OpenSSL ab Version 0.9.8h |
|---|---|
| Modul | mod_ssl |
| Kontext | Server, <VirtualHost> |
| Syntax | SSLStaplingResponseTimeSkew Sekunden |
| Standardwert | 300 |

Diese Direktive bestimmt den maximalen akzeptierten Zeitversatz für OCSP-Stapling-Antworten, gemessen anhand ihrer thisUpdate- und nextUpdate-Felder.

#### SSLStaplingReturnResponderErrors

OCSP-Stapling-Fehler an den Client weiterreichen

| Seit Version | 2.3.3 mit OpenSSL ab Version 0.9.8h |
|---|---|
| Modul | mod_ssl |
| Kontext | Server, <VirtualHost> |

| Syntax | SSLStaplingReturnResponderErrors On\|Off |
|---|---|
| Standardwert | On |

Wenn diese Direktive aktiviert ist (Standardwert `On`), dann werden Antworten auf fehlgeschlagene OCSP-Stapling-Anfragen an den Client weitergegeben. Dies betrifft beispielsweise Statusfehler oder abgelaufene Antworten. Wenn Sie den Wert `Off` wählen, werden keine Stapling-Informationen in den TLS-Handshake eingefügt, falls die Anfrage fehlschlägt.

#### SSLStaplingStandardCacheTimeout

Sekunden, bis Antworten im OCSP-Stapling-Cache ungültig werden

| Seit Version | 2.3.3 mit OpenSSL ab Version 0.9.8h |
|---|---|
| Modul | mod_ssl |
| Kontext | Server, <VirtualHost> |
| Syntax | SSLStaplingStandardCacheTimeout Sekunden |
| Standardwert | 3600 |

Diese Direktive setzt ein Timeout in Sekunden, nach dessen Ablauf Antworten im OCSP-Stapling-Cache ungültig werden. Diese Direktive kümmert sich um gültige Antworten; für fehlerhafte ist `SSLStaplingErrorCacheTimeout` zuständig.

#### SSLOptions

Spezielle SSL-Optionen

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Options) |
| Syntax | SSLOptions [+\|-]Option [[+\|-]Option ...] |
| Standardwert | nicht gesetzt |

Diese Direktive legt Optionen für die SSL-Verarbeitung fest. Ihre Aufgabe und ihre Syntax ähneln der Direktive `Options`, die normale Verzeichnisoptionen festlegt (siehe Kapitel 6, »Grundkonfiguration«).

Der Wert von `SSLOptions` ist eine durch Leerzeichen getrennte Liste verschiedener Optionen. Wenn Sie Optionen ohne Präfix verwenden, werden *alle* Optionen aus dem übergeordneten Kontext durch diese ersetzt. Wenn Sie einer Option das Präfix + voranstellen, wird sie zu den aus dem übergeordneten Kontext geerbten Optionen hinzugefügt; ein - entfernt sie entsprechend aus der bisherigen Optionsliste. Die einzelnen Optionen sehen Sie in Tabelle 10.3.

| Option | Erläuterung |
|---|---|
| StdEnvVars | Die Standard-SSL-Umgebungsvariablen (siehe oben) sollen exportiert werden. Standardmäßig werden sie nicht erzeugt, weil dies viel Rechenzeit benötigt. Deshalb sollte die Option nur in Verzeichnissen gesetzt werden, in denen CGI-Skripte, SSI oder Rewrite-Einstellungen davon Gebrauch machen. |
| CompatEnvVars | Wenn diese Option aktiv ist, werden zusätzliche (in der Regel nur anders benannte) SSL-Umgebungsvariablen erzeugt, die die Kompatibilität mit älteren Apache-SSL-Implementierungen sicherstellen. Eine Liste dieser Variablen finden Sie in der Apache-Online-Dokumentation. |
| ExportCertData | Erzeugt die Umgebungsvariablen SSL_SERVER_CERT, SSL_CLIENT_CERT und SSL_CLIENT_CERT_CHAIN. Damit haben CGI-Skripte Zugriff auf das Server-Zertifikat, das Client-Zertifikat und den Client-Zertifizierungspfad. |
| FakeBasicAuth | Aus dem Anbieter-DN wird ein einfacher Username extrahiert, damit die Direktiven zur Basic-Authentifizierung eingesetzt werden können. Das Passwort wird nicht vom Benutzer erfragt, sondern lautet immer password. In der Userdatei können Sie dafür die Werte xxj31ZMTZzkVA (crypt, UNIX) beziehungsweise $1$OXLyS... $Owx8s2/m9/gfkcRVXzgoE/ (MD5 für Windows oder NetWare) eintragen. |
| StrictRequire | Hindert die Einstellung Satisfy Any daran, Zugriffsbeschränkungen durch SSLRequire oder SSLRequireSSL zu umgehen. |
| OptRenegotiate | Wenn Sie SSL-Direktiven in Verzeichniskontexten einsetzen, wird beim Zugriff einer Anfrage auf das entsprechende Verzeichnis normalerweise stets ein neuer SSL-Handshake zur Protokollaushandlung durchgeführt. Diese Option verhindert das, solange die vorherige Protokollvereinbarung noch verwendet werden kann. |

**Tabelle 10.3** Erlaubte Einstellungen für SSL-Optionen

Hier ein Beispiel, das in einem *cgi-bin*-Verzeichnis verschiedene Umgebungsvariablen aktiviert:

```
<Location /cgi-bin>
  SSLOptions +StdEnvVars +ExportCertData
  # ... weitere Optionen
</Location>
```

### SSLPassPhraseDialog

Passphrasen-Dialog für private Schlüssel

| | |
|---|---|
| **Modul** | mod_ssl |
| **Kontext** | Server |
| **Syntax** | SSLPassPhraseDialog builtin \| exec:/Programmpfad |
| **Standardwert** | builtin |

Beim Start von Apache müssen Sie unter bestimmten, weiter oben beschriebenen Umständen die Passphrase Ihres privaten Schlüssels eingeben. `mod_ssl` stellt dazu zwei Möglichkeiten bereit:

- `builtin`. `mod_ssl` verwendet sein eigenes Verfahren; in der Praxis ist dies meist die beste Lösung. Nach jeder Eingabe einer Passphrase wird versucht, diese auf den privaten Schlüssel jedes virtuellen SSL-Hosts anzuwenden. Sie brauchen also unter Umständen nicht für jeden Schlüssel die Passphrase einzugeben, sondern nur jede unterschiedliche Passphrase.
- `exec:/Programmpfad`. Hier können Sie ein beliebiges externes Programm angeben, das von `mod_ssl` aufgerufen wird, um die gewünschte Passphrase zurückzuliefern. Dabei werden zwei Kommandozeilenargumente übermittelt: `Host:Port` sowie der benötigte Algorithmus (`RSA` oder `DSA`). Es wird von dem Programm erwartet, dass es die entsprechende Passphrase in die Standardausgabe schreibt. Natürlich sollten Sie aus Sicherheitsgründen kein Programm schreiben, in dem alle Passphrasen im Klartext gespeichert sind und nach Bedarf ausgegeben werden. Aber möglicherweise gibt es andere Automatisierungsmöglichkeiten.

### SSLProtocol

Bevorzugtes SSL-Protokoll

| | |
|---|---|
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSLProtocol [+\|-]Protokoll1 [[+\|-]Protokoll2 ...]` |
| **Standardwert** | `all` |

Diese Direktive legt die konkreten SSL-Protokolle fest, mit denen Clients auf den SSL-Server zugreifen dürfen. Die folgenden vier Werte sind definiert:

- `SSLv2` – das ursprüngliche SSL-Protokoll in der Netscape-Urversion 2.0
- `SSLv3` – die erweiterte Version 3.0 des SSL-Protokolls
- `TLSv1` – Transport Layer Security 1.0, die IETF-Weiterentwicklung von SSL
- `All` – Kurzfassung für `+SSLv2 +SSLv3 +TLSv1`. Sie können einzelne Protokolle durch Voranstellen eines Minuszeichens daraus entfernen.

In aller Regel sollten Sie die Voreinstellung `All` beibehalten, um mit der maximalen Anzahl von Clients kommunizieren zu können.

### SSLRandomSeed

Quelle für die Basiszahl (RandomSeed) des SSL-Zufallsgenerators

| | |
|---|---|
| **Modul** | `mod_ssl` |
| **Kontext** | Server |
| **Syntax** | `SSLRandomSeed Kontext Quelle [Bytes]` |
| **Standardwert** | nicht gesetzt |

Die Sicherheit kryptografischer Verfahren hängt zu einem gewissen Teil von der Qualität der verfügbaren Zufallsgeneratoren ab. Diese wiederum beginnen ihre Arbeit stets mit einer – möglichst ebenfalls zufällig ermittelten – Basiszahl, dem sogenannten *RandomSeed*. Mit dieser Direktive können Sie angeben, woher der RandomSeed für zwei verschiedene Aspekte stammen soll: für den Start des Servers (Kontext `startup`) und für den Aufbau einer Verbindung (Kontext `connect`). Sie können die folgenden Quellen für den RandomSeed angeben:

- `builtin` – das eingebaute Verfahren. Es wählt als RandomSeed eine Byte-Folge von einer zufälligen Stelle des Scoreboard-Speichers, den Apache zur Interprozesskommunikation einsetzt. Dies ist das ressourcenschonendste Verfahren, aber insbesondere für `startup` keine gute Quelle, weil das Scoreboard beim Start möglicherweise noch gar nicht verfügbar ist.
- `file:/Dateipfad` – die Quelle ist eine Datei. Natürlich sollten Sie auf keinen Fall eine statische Datei wählen; damit würde bei jedem Start beziehungsweise jedem Verbindungsaufbau derselbe RandomSeed gewählt! Die Option existiert für den Einsatz der UNIX-Gerätedatei `/dev/random`, aus der Zufallszahlen gelesen werden können.
- `exec:/Programmpfad` – ein Programm, das den RandomSeed auf der Standardausgabe liefern muss. Wenn Sie ein Programm haben, das einen besseren Zufallsgenerator implementiert als */dev/random* (oder wenn Sie mit einer Plattform wie Windows arbeiten, die kein */dev/random* besitzt), können Sie hier dieses Programm angeben. Ein gutes Beispiel ist das Programm *truerand*, das zur `mod_ssl`-Distribution gehört. Diese Lösung sollte aus Performance-Gründen nur für `startup` eingesetzt werden.
- `egd:/EGD-Socket-Pfad` – Pfad zum UNIX-Domain-Socket des externen Zufallszahlenservers EGD (Entropy Gathering Daemon). Sie finden dieses Perl-Programm auf der DVD zum Buch oder unter *http://egd.sourceforge.net*.

Als drittes Argument können Sie optional die Anzahl der Bits angeben, die aus der angegebenen Quelle gelesen werden sollen. Seien Sie aber vorsichtig: Unter Umständen blockiert der Aufruf, bis die gewünschte Bit-Anzahl verfügbar ist.

Beispiele:

```
SSLRandomSeed file:/dev/random
SSLRandomSeed startup exec:/usr/local/bin/truerand 128
SSLRandomSeed connect builtin
SSLRandomSeed connect egd:/var/run/egd.sock
```

#### SSLRequire

Zugriff von einer booleschen Bedingung abhängig machen

| | |
|---|---|
| **Modul** | mod_ssl |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `SSLRequire Ausdruck` |
| **Standardwert** | nicht gesetzt |

Diese komplexe Direktive ermöglicht beliebige Zugriffsbeschränkungen auf Ihre SSL-Site auf der Basis von CGI- und SSL-Umgebungsvariablen. Listen der entsprechenden Variablen finden Sie weiter oben in diesem Kapitel beziehungsweise in Kapitel 14, »CGI«.

In dem Ausdruck, der ausgewertet werden soll, können Sie folgende Elemente verwenden:

- Logische Verknüpfungen:
  - `&&` oder `and`: logisches Und; beide Teilausdrücke müssen zutreffen.
  - `||` oder `or`: logisches Oder; mindestens ein Teilausdruck muss zutreffen.
  - `!`: logisches Nicht; Verneinung des nachfolgenden Ausdrucks
  - `()`: Klammern zur Zusammenfassung
- Vergleichsoperatoren:
  - `==` oder `eq`: Gleichheit
  - `!=` oder `ne`: Ungleichheit
  - `<` oder `lt`: kleiner als (bei Strings: weiter vorn im Zeichensatz)
  - `>` oder `gt`: größer als (bei Strings: weiter hinten im Zeichensatz)
  - `<=` oder `le`: kleiner oder gleich
  - `>=` oder `ge`: größer oder gleich
  - `in {Element1, [Element2, ...]}`: in der Liste vorhanden
  - `=~ RegExp`: entspricht dem angegebenen (Perl-kompatiblen) regulären Ausdruck.
  - `!~ RegExp`: entspricht nicht dem angegebenen regulären Ausdruck.

- Elemente:
  - Zahlen
  - `"Text"`: Anführungszeichen umschließen einen beliebigen String.
  - `%{VARIABLE}`: Der Wert der angegebenen CGI- oder SSL-Umgebungsvariablen wird ermittelt.
  - `file(Dateiname)`: Der Inhalt der angegebenen Datei wird verwendet (praktisch beispielsweise für lange, komplexe reguläre Ausdrücke).

Die Direktive bietet unzählige Möglichkeiten der Zugriffsbeschränkung; es würde zu weit führen, hier ein eigenes Tutorial zu diesem Thema anzubieten. Vieles erinnert an die Fähigkeiten von `mod_rewrite`; Näheres zu diesem mächtigen Modul finden Sie in Kapitel 8, »Weiterleitung und Indizes«.

Das folgende schöne Beispiel entstammt der Originaldokumentation von `mod_ssl`:

```
SSLRequire ( %{SSL_CIPHER} !~ m/^(EXP|NULL)-/ \
and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20 ) \
or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
```

Dieses »Ungetüm« besagt, dass nur Mitarbeiter einer bestimmten Abteilung der Firma Snake Oil, Ltd. montags bis freitags zwischen 8 und 20 Uhr zugreifen dürfen. Beachten Sie die Klammersetzung – Clients aus dem Netzwerk 192.76.162.0/24 haben immer Zugriff. Es geht darum, bei externen Zugriffen eine sehr hohe Sicherheitsstufe zu verwenden, die für Zugriffe aus dem lokalen Netzwerk nicht nötig ist.

#### SSLRequireSSL

Zugriff ausschließlich über SSL zulassen

| | |
|---|---|
| **Modul** | `mod_ssl` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `SSLRequireSSL` |
| **Standardwert** | nicht gesetzt |

Diese Direktive besitzt keinen Wert. Wenn sie gesetzt wird, bedeutet dies, dass auf die aktuelle Ressource nur über SSL, nicht aber über eine normale HTTP-Verbindung zugegriffen werden darf.

### SSLSessionCache

Art des SSL-Session-Caches

| | |
|---|---|
| **Modul** | mod_ssl |
| **Kontext** | Server |
| **Syntax** | SSLSessionCache Typ |
| **Standardwert** | none |

Mithilfe der Direktive SSLSessionCache können Sie optional einen Session-Cache für die SSL-Verarbeitung einrichten. Dieser beschleunigt die Verarbeitung paralleler Anfragen desselben Clients (beispielsweise zum Laden von eingebetteten Bildern) und mindert die Ressourcenbelastung des Servers.

Als Typ können Sie einen der drei folgenden Werte angeben:

- none. Dies ist der Standardfall: Es wird kein Session-Cache verwendet.
- dbm:/Dateipfad. Die angegebene DBM-Datei wird benutzt.
- shm:/Dateipfad. In einem Shared-Memory-Bereich wird eine Hash-Tabelle eingerichtet. Diese ist schneller als dbm, funktioniert aber nur unter einigen UNIX-Varianten.

### SSLSessionCacheTimeout

Dauer bis zum Timeout einer SSL-Session

| | |
|---|---|
| **Modul** | mod_ssl |
| **Kontext** | Server, <VirtualHost> |
| **Syntax** | SSLSessionCacheTimeout Sekunden |
| **Standardwert** | 300 |

Diese Direktive legt einen Timeout-Wert für den Session-Cache und den internen SSL-Speicher-Cache in Sekunden fest. Der Standardwert 300 ist in den meisten Fällen eine gute Wahl; niedrigere Werte können zulasten der Performance gehen.

### SSLUserName

Variablenname zur Bestimmung des Benutzernamens

| | |
|---|---|
| **Seit Version** | 2.0.51 |
| **Modul** | mod_ssl |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |

| Syntax | SSLUserName Variable |
|---|---|
| Standardwert | nicht gesetzt |

Mithilfe dieser Direktive können Sie eine Variable angeben, die das Userfeld im Anfrageobjekt setzt. Der Wert der Variablen, die Sie hier angeben, wird für die Umgebungsvariable REMOTE_USER verwendet und kann auch durch Authentifizierungsmodule ausgewertet werden. Sie können jede der oben beschriebenen SSL-Umgebungsvariablen als Wert verwenden. Falls Sie mittels SSLOptions die Option FakeBasicAuth gesetzt haben, wird diese Direktive wirkungslos. Beispiel:

```
SSLUserName SSL_CLIENT_S_DN_CN
```

#### SSLVerifyClient

Art der Überprüfung von Client-Zertifikaten

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SSLVerifyClient Level |
| Standardwert | none |

Diese Direktive bestimmt, ob und wie ein eventuelles Client-Zertifikat überprüft werden soll. Wie bereits erwähnt, werden Client-Zertifikate in der Praxis nicht allzu oft eingesetzt, zumindest spielen sie für öffentlich zugängliche Websites keine nennenswerte Rolle. Sie sind aber ein praktisches Mittel, eine Site auf einen bestimmten, über das Internet verteilten Nutzerkreis zu beschränken (ein sogenanntes *Extranet* als Gegenbegriff zum lokalen Intranet).

Es sind vier verschiedene Einstellungen möglich:

- none. Der Client braucht kein Zertifikat; falls er eines mitschickt, wird es nicht beachtet.
- optional. Der Client kann mit seiner Anfrage ein Zertifikat schicken; in diesem Fall wird es untersucht.
- require. Der Client muss ein gültiges Zertifikat senden.
- require_no_ca. Der Client muss ein Zertifikat senden, aber es braucht nicht von einer vertrauenswürdigen Zertifizierungsstelle (CA) unterzeichnet worden zu sein.

Die wichtigsten Werte sind none und require, weil optional und require_no_ca keinen besonderen praktischen Nutzen bringen.

#### SSLVerifyDepth

Maximale Tiefe der Überprüfung von Client-Zertifikaten

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SSLVerifyDepth Anzahl |
| Standardwert | 1 |

Mit `SSLVerifyDepth` können Sie bestimmen, wie viele Stufen eines Zertifizierungspfades auf der Suche nach einem vertrauenswürdigen Unterzeichner von Client-Zertifikaten untersucht werden sollen. Ein zu hoher Wert verlangsamt den Server, ein zu niedriger führt eventuell zur unberechtigten Ablehnung von Clients. Beispiel:

```
SSLVerifyDepth 10
```

### 10.2.2 mod_ssl-Proxy-Direktiven

Auch wenn Apache als Proxy eingesetzt wird (siehe Kapitel 13, »Proxy- und Cache-Funktionen«), kann die entsprechende Verbindung über SSL abgesichert werden. Zu diesem Zweck stellt `mod_ssl` eine Reihe spezieller Proxy-Direktiven zur Verfügung. Viele von ihnen besitzen ein Pendant für gewöhnliche Webserver-Dienste, das bereits weiter oben vorgestellt wurde. In diesen Fällen wird nicht noch einmal auf die Syntax eingegangen, sondern es wird auf die entsprechende Nicht-Proxy-Direktive verwiesen.

#### SSLProxyCACertificateFile

Datei mit CA-Zertifikaten zur Remote-Authentifizierung

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | SSLProxyCACertificateFile Dateipfad |
| Standardwert | nicht gesetzt |

Dies ist die Proxy-Variante von `SSLCACertificateFile`.

#### SSLProxyCACertificatePath

Verzeichnispfad für CA-Zertifikate zur Remote-Authentifizierung

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost> |

| Syntax | SSLProxyCACertificatePath Verzeichnispfad |
|---|---|
| Standardwert | nicht gesetzt |

Näheres finden Sie unter der Beschreibung der Nicht-Proxy-Version SSLCACertificatePath.

#### SSLProxyCARevocationFile

Datei mit CA-CRLs zur Remote-Authentifizierung

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | SSLProxyCARevocationFile Dateipfad |
| Standardwert | nicht gesetzt |

Diese Direktive stellt die Proxy-Entsprechung von SSLCARevocationFile dar; Näheres siehe dort.

#### SSLProxyCARevocationPath

Verzeichnispfad für CA-CRLs zur Remote-Authentifizierung

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | SSLProxyCARevocationPath Verzeichnispfad |
| Standardwert | nicht gesetzt |

Dies ist die für den Proxy-Betrieb benötigte Version der bereits besprochenen Direktive SSLCARevocationPath.

#### SSLProxyCipherSuite

Verfügbare Verschlüsselungsalgorithmen für SSL-Proxy-Handshake

| Modul | mod_ssl |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SSLProxyCipherSuite chipher-spec |
| Standardwert | ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:<br>+LOW:+SSLv2:+EXP |

Näheres finden Sie unter der Beschreibung der Nicht-Proxy-Version dieser Direktive, SSLCipherSuite.

### SSLProxyEngine

SSL-Proxy ein-/ausschalten

| | |
|---|---|
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSLProxyEngine On\|Off` |
| **Standardwert** | `Off` |

`SSLProxyEngine On` wird verwendet, um die SSL-Proxy-Funktionalität explizit einzuschalten. Näheres siehe `SSLEngine`.

### SSLProxyMachineCertificateFile

Datei mit Client-Zertifikaten und Schlüsseln für den SSL-Proxy

| | |
|---|---|
| **Modul** | `mod_ssl` |
| **Kontext** | Server |
| **Syntax** | `SSLProxyMachineCertificateFile Dateiname` |
| **Standardwert** | nicht gesetzt |

Dies ist die Proxy-Version von `SSLCertificateFile`; weitere Informationen erhalten Sie dort.

### SSLProxyMachineCertificatePath

Verzeichnispfad mit Client-Zertifikaten und Schlüsseln für den SSL-Proxy

| | |
|---|---|
| **Modul** | `mod_ssl` |
| **Kontext** | Server |
| **Syntax** | `SSLProxyMachineCertificatePath Verzeichnis` |
| **Standardwert** | nicht gesetzt |

Die Aufgabe dieser Direktive entspricht derjenigen von `SSLCertificatePath` im Nicht-Proxy-Bereich.

### SSLProxyProtocol

Bevorzugtes Proxy-SSL-Protokoll

| | |
|---|---|
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSLProxyProtocol [+\|-]Protokoll [[+\|-]Protokoll ...]` |
| **Standardwert** | `all` |

Diese Direktive legt die SSL-Protokolle fest, die der Server im Proxy-Betrieb akzeptiert. Näheres finden Sie unter `SSLProtocol`.

#### SSLProxyVerify

Art der Überprüfung des Remote-Zertifikats

| | |
|---|---|
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `SSLProxyVerify Level` |
| **Standardwert** | `none` |

Analog zu `SSLVerifyClient` überprüft diese Direktive das Zertifikat eines entfernten Proxy-Servers.

#### SSLProxyVerifyDepth

Maximale Tiefe der Überprüfung von Remote-Zertifikaten

| | |
|---|---|
| **Modul** | `mod_ssl` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `SSLProxyVerifyDepth Anzahl` |
| **Standardwert** | `1` |

Diese Direktive legt fest, wie viele Stufen des Zertifizierungspfades eines entfernten Proxys überprüft werden sollen. Syntax und Funktionsweise entsprechen `SSLVerifyDepth`.

### 10.2.3 mod_nw_ssl für NetWare

Das Modul `mod_nw_ssl` ermöglicht die Verwendung der SSL-Verschlüsselung unter Novell NetWare. Dieses Betriebssystem enthält bereits ab Werk SSL-Unterstützung, sodass kein Bedarf besteht, eine externe Bibliothek wie OpenSSL zu bemühen. Das Modul definiert nur zwei Direktiven, die hier kurz vorgestellt werden.

#### NWSSLTrustedCerts

Liste zusätzlicher Client-Zertifikate (NetWare)

| | |
|---|---|
| **Modul** | `mod_nw_ssl` |
| **Kontext** | Server |

| Syntax | NWSSLTrustedCerts Dateiname [Dateiname ...] |
|---|---|
| Standardwert | nicht gesetzt |

Mithilfe dieser Direktive können Sie eine Liste von Dateien angeben, in denen DER-codierte Client-Zertifikate gespeichert sind. Diese Zertifikate werden für SSL-Proxy-Verbindungen verwendet.

#### SecureListen

SSL-Verschlüsselung für den angegebenen TCP-Port aktivieren (NetWare)

| Modul | mod_nw_ssl |
|---|---|
| Kontext | Server |
| Syntax | SecureListen [IP-Adresse:]Port Zertifikat [MUTUAL] |
| Standardwert | nicht gesetzt |

Diese Direktive bestimmt einen Port und optional eine IP-Adresse für SSL-verschlüsselte Verbindungen. Damit entspricht die Syntax des ersten Parameters der Direktive `Listen` (siehe Kapitel 6, »Grundkonfiguration«). Als zweiten Parameter müssen Sie den eDirectory-Pfad des Server-Zertifikats angeben. Die Anwesenheit des optionalen Argumentes `MUTUAL` legt schließlich fest, dass auch vom Client ein Zertifikat erwartet wird.

## 10.3 Zusammenfassung

Die SSL-Unterstützung wird in Apache durch das Standardmodul `mod_ssl` bereitgestellt. Es greift auf die Bibliothek OpenSSL zu, mit deren Kommandozeilenfunktionalität Sie auch Ihre Zertifikate erzeugen können. Eine grundlegende SSL-Site lässt sich schnell und einfach einrichten. Dennoch lohnt es sich, die zahlreichen optionalen Direktiven von `mod_ssl` genau zu studieren und auszuprobieren, um die Leistungsfähigkeit, Sicherheit und Performance Ihrer abgesicherten Verbindungen weiter zu verbessern.

Bevor Apache mit SSL-Verbindungen umgehen kann, benötigen Sie ein digital signiertes Server-Zertifikat. Im Grunde können Sie es mithilfe von OpenSSL selbst erstellen; allerdings akzeptieren Browser aus Sicherheitsgründen nur Zertifikate, die von einer vertrauenswürdigen Zertifizierungsautorität (CA) signiert wurden. Solche öffentlich einsetzbaren Zertifikate sind in jedem Fall kostenpflichtig, dafür garantiert die CA die Identität eines Anbieters.

*»Wenn du eine weise Antwort willst, musst du vernünftig fragen.«*
*– Johann Wolfgang von Goethe*

# 11 Logging

*Log-Dateien* (»logfiles«, Protokolldateien) sind für den ernsthaften Betrieb einer Website sehr wichtig: Sie geben sowohl über das Besucherverhalten als auch über Fehler und sogar Angriffsversuche Auskunft. Apache kann deshalb nicht nur einfache Log-Dateien im klassischen CLF-Format oder im NCSA-Stil erstellen, sondern bietet zahlreiche zusätzliche Möglichkeiten. In diesem Kapitel werden zunächst die eingebauten Logging-Fähigkeiten und -Tools von Apache 2 vorgestellt; anschließend lernen Sie noch einige »selbst gestrickte« Perl-Skripte sowie Hilfsprogramme von Drittanbietern kennen.

Traditionellerweise werden mindestens zwei Log-Dateien geführt: die Fehler-Log-Datei (*ErrorLog*), in der sämtliche unerwünschten Vorkommnisse dokumentiert werden, und die Zugriffs-Log-Datei (*TransferLog* oder auch *AccessLog*), die alle Client-Zugriffe protokolliert.

Die *TransferLog*-Datei besitzt üblicherweise entweder das *Common Log Format* (CLF) oder das erweiterte Log-Format des NCSA HTTPd (*Combined Log Format*). Ein Eintrag im Common Log Format sieht beispielsweise so aus:

```
163.15.41.28 - - [29/Aug/2011:15:35:07 +0200] "GET / HTTP/1.1" 200 2263
```

Die Bedeutung der einzelnen Bestandteile wurde bereits in Kapitel 2, »Funktionsweise von Webservern«, erläutert. Kurz gesagt handelt es sich um die IP-Adresse des Clients, die RFC-1413-Identität des entfernten Users (in aller Regel »–«, also nicht verfügbar), den Benutzernamen, das Datum und die Uhrzeit, die erste Zeile der HTTP-Anfrage, den HTTP-Statuscode sowie die Anzahl der Bytes im Antwort-Body.

Das Combined Log Format enthält noch zwei zusätzliche Felder: den `Referer` (die URL der zuvor besuchten Seite) sowie den `User-Agent` (den Namen des Client-Programms). Ein typischer Eintrag hat diese Form:

```
163.15.41.28 - - [29/Aug/2011:15:36:24 +0200] "GET /seite2.html HTTP/
1.1" 200
2326 "http://www.mynet.de/" "Mozilla/4.0 (compatible; MSIE 6.0;
Windows NT 4.0; .NET CLR 1.0.3705)"
```

In Apache können Sie das Format der Log-Einträge völlig frei definieren; Sie sind also nicht auf diese beiden Standardformate angewiesen. Da zahlreiche Auswertungs- und Weiterverarbeitungsprogramme allerdings diese Formate erwarten, lohnt es sich, auf jeden Fall eine `TransferLog`-Datei in einem dieser Formate zu führen und zusätzlich eine weitere Datei für Ihre Spezialanwendungen anzulegen.

Das Format der *ErrorLog*-Datei entspricht dagegen dem UNIX-Systemstandard `syslog`: Jede Zeile enthält das Datum, die Dringlichkeit der Meldung (siehe weiter unten bei der Beschreibung der Direktive `LogLevel`) und eine Beschreibung des Fehlers. Beispiel:

```
[Mon Aug 29 15:34:57 2011] [error] [client 145.82.41.93] File
does not exist: /usr/local/apache2/htdocs/unavail.html
```

Seit Apache 2.4 existiert allerdings die neue Direktive `ErrorLogFormat`, mit der genau festgelegt werden kann, welche Informationen im *ErrorLog* abgelegt werden sollen.

## 11.1 Logging-Direktiven und -Module

In diesem Abschnitt werden die Logging-Bordmittel des Apache-Webservers vorgestellt. Die Einstellungen für die *ErrorLog*-Datei sind aufgrund ihrer besonderen Wichtigkeit im Core definiert. Die `TransferLog`-Optionen befinden sich im Modul `mod_log_config`, weitere im Modul `mod_logio`. Recht neue Entwicklungen sind die Module `mod_log_forensic` für forensische Log-Dateien sowie `mod_dumpio` zur Protokollierung der gesamten Ein- und Ausgabe des Servers. Daneben enthält das URL-Umwandlungsmodul `mod_rewrite` (siehe Kapitel 8, »Weiterleitungen und Indizes«) seine eigenen Logging-Direktiven, die in Abschnitt 11.1.6, »Logging-Direktiven in mod_rewrite«, vorgestellt werden.

Ein etwas anderes Ziel verfolgt `mod_usertrack`: Es ermöglicht die Verfolgung einer Client-Session über Cookies. Auch die Direktiven dieses Moduls werden in einem eigenen Abschnitt, und zwar in Abschnitt 11.1.5, »mod_usertrack«, behandelt.

### 11.1.1 Core-Direktiven

Neben den bereits erwähnten `ErrorLog`-Einstellungen enthält der Core noch die beiden indirekt mit dem Logging verbundenen Direktiven `HostnameLookups` und `IdentityCheck` (Letztere wird seit Version 2.1-beta in ein eigenes Modul ausgelagert).

### ErrorLog

Verzeichnis für die Fehler-Log-Datei

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | ErrorLog Dateiname\|syslog[:Quelle] |
| Standardwert | UNIX: logs/error_log; Windows: logs/error.log |

Die Direktive ErrorLog legt den Namen der Datei fest, in der die verschiedenen Fehler protokolliert werden. Unter UNIX ist die Voreinstellung *logs/error_log*, unter Windows *logs/error.log*. Wie üblich ist der Pfad relativ zur ServerRoot.

Statt einer Datei können Sie unter Betriebssystemen, die dies unterstützen, auch eine Pipe zu einem Programm angeben, das die Log-Meldungen unmittelbar weiterverarbeitet. Dies ist beispielsweise für das Apache-Hilfsprogramm rotatelogs interessant, das in regelmäßigen Abständen neue Log-Dateien anlegt. Das Verfahren wird weiter unten im Zusammenhang mit der Direktive CustomLog genauer erläutert. Hier nur ein Beispiel, das jeden Tag eine neue Log-Datei mit einem Namen wie access_log.2011-09-07 (Jahr-Monat-Tag) anlegt:

```
ErrorLog "|/usr/local/apache2/bin/rotatelogs \
    /usr/local/apache2/logs/access_log.%Y-%m-%d 86400"
```

Unter UNIX besteht darüber hinaus die alternative Möglichkeit, die Fehlermeldungen an das syslog zu übergeben, anstatt sie in eine einfache Datei zu schreiben. Dazu wird statt eines Pfades die Syntax syslog oder syslog:Fehlerquelle verwendet. Die Fehlerquelle (facility) beschreibt die Art des Programms oder Prozesses, von dem die Meldung stammt. Die folgenden Quellen sind in syslog definiert:

- auth: Authentifizierung
- authpriv: Authentifizierung privilegierter Benutzer
- cron: cron-Daemon
- daemon: sonstige Daemons
- ftp: FTP-Server
- kern: Kernel
- local0 bis local7: zur freien Verwendung durch beliebige Programme
- lpr: Drucker-Subsystem
- mail: Mail-Subsystem
- news: News-Subsystem

- `syslog`: interne Meldungen von `syslog` selbst
- `uucp`: UUCP-Subsystem
- `user`: Anwendungsprogramme

Beispiel:

```
ErrorLog syslog:daemon
```

### ErrorLogFormat

Formatangabe für zusätzliche Informationen in den Einträgen im `ErrorLog`

| Seit Version | 2.3.9 |
|---|---|
| Modul | `core` |
| Kontext | Server, `<VirtualHost>` |
| Syntax | `ErrorLogFormt [connection\|request] Format` |

Traditionell war das Format der Einträge im `ErrorLog` stets dasselbe. Erst in neueren 2.3-Beta-Versionen kann das Format modifiziert werden, ähnlich wie es bei Access- und anderen Log-Dateien schon seit langer Zeit der Fall ist. Genauer gesagt können Sie zusätzliche Informationen hinzufügen.

Wenn Sie `connection` oder `request` vor dem eigentlichen Format angeben, werden die zusätzlichen Daten nur bei der ersten Log-Meldung einer Verbindung beziehungsweise Anfrage protokolliert, ansonsten gelten sie für jeden Log-Eintrag.

Für die Formatangabe selbst ist eine Reihe verschiedener Platzhalter verfügbar, die in Tabelle 11.1 zusammengefasst werden. Sie ähneln den Platzhaltern für Access-Log-Dateien (siehe »Kürzel für Log-Formate«, Seite 524), sind aber nicht identisch.

| Kürzel | Beschreibung |
|---|---|
| `%%` | Prozentzeichen |
| `%a` | Remote-IP-Adresse und Port |
| `%A` | lokale IP-Adresse und Port |
| `%{Name}e` | Umgebungsvariable *Name* |
| `%E` | Fehlerstatuscode und -String der Apache Portable Runtime bzw. des Betriebssystems |
| `%F` | Quelldateiname und Zeilennummer des Protokollaufrufs |
| `%{Name}i` | HTTP-Anfrage-Header *Name* |

**Tabelle 11.1** 11.1 Formatplatzhalter für ErrorLogFormat

| Kürzel | Beschreibung |
|---|---|
| `%k` | Anzahl der Keepalive-Anfragen über die aktuelle Verbindung |
| `%l` | LogLevel der Meldung (siehe »LogLevel«, Seite 520) |
| `%L` | Log-ID der Anfrage |
| `%{c}L` | Log-ID der Verbindung |
| `%{C}L` | Log-ID, falls die Meldung zu einer Verbindung gehört, ansonsten leer |
| `%m` | Name des Moduls, das die Log-Meldung ausgibt |
| `%M` | die Log-Meldung selbst |
| `%{Name}n` | Anfrage-Hinweis *Name* |
| `%P` | Prozess-ID des aktuellen Prozesses |
| `%T` | Thread-ID des aktuellen Threads |
| `%{g}T` | systemweit eindeutige ID des aktuellen Threads (wie in der Ausgabe von `top`); zurzeit nur Linux |
| `%t` | aktuelle Zeit |
| `%{u}t` | aktuelle Zeit einschließlich Mikrosekunden |
| `%{cu}t` | aktuelle Zeit im ISO-8601-Format einschließlich Mikrosekunden |
| `%v` | kanonischer Servername des aktuellen Servers |
| `%V` | Servername desjenigen Servers, der die Anfrage bearbeitet, gemäß `UseCanonicalName`-Einstellung |
| `\` (Backslash-Leerzeichen) | Leerzeichen, das keine Feldtrennung verursacht |
| `%` (Prozent-Leerzeichen) | Feldtrennzeichen (wird nicht ausgegeben) |

**Tabelle 11.1** 11.1 Formatplatzhalter für ErrorLogFormat (Forts.)

Die einzelnen Felder werden durch Leerzeichen voneinander getrennt. Wenn ein Feld keine Ausgabe erzeugt – z. B. `%{Referer}i`, in dem es keinen Referer gibt –, werden auch eventuelle zusätzliche Zeichen des Feldes nicht ausgegeben. Wenn Sie z. B. `(%{Referer}i)` schreiben, um den Referer in runde Klammern zu setzen, werden auch die runden Klammern nicht ausgegeben, wenn der Referer fehlt. Wenn Sie innerhalb eines Feldes ein Leerzeichen ausgeben möchten, können Sie es durch einen Backslash escapen – Beispiel:

```
Referer:\ %{Referer}i
```

Möchten Sie umgekehrt zwei Felder ohne sichtbaren Abstand hintereinander ausgeben, können Sie ein Prozentzeichen, gefolgt von einem Leerzeichen, als Feldtrenner verwenden.

Hinter dem Prozentzeichen der meisten Kürzel kann noch ein Modifier stehen:

- - (Minus): Wenn das Element keine Ausgabe erzeugt, wird stattdessen ein Minuszeichen in die Ausgabe geschrieben. Beispiel: `%-{Accept}i` gibt, falls vorhanden, die vom Client akzeptierten MIME-Types aus oder `-`, falls der Header `Accept` in der Anfrage fehlt.
- + (Plus): Wenn Sie bei einem Feld den Modifier + verwenden, wird die gesamte Zeile nicht ausgegeben, falls das entsprechende Element keinen Inhalt hat. Zeilen, in denen `%+{Referer}i` vorkommt, werden also beispielsweise nur protokolliert, wenn die entsprechende Anfrage einen Referer enthielt. Dies ist nur in Anfragen mit der Zusatzangabe `request` sinnvoll.
- Numerischer Wert (LogLevel): Das Element wird nur protokolliert, wenn die Dringlichkeitsstufe (LogLevel) mindestens dem angegebenen Wert entspricht. Mögliche Werte reichen von 1 (`alert`) über 4 (`warn`) und 7 (`debug`) bis hin zu 15 (`trace8`). Einzelheiten über LogLevels erfahren Sie in der Beschreibung der im Anschluss beschriebenen gleichnamigen Direktive. `%4{Host}i` loggt den Anfrage-Header `Host` nur dann, wenn die Dringlichkeit mindestens `warn` ist.

Beispiel:

```
ErrorLogFormat "[%{u}t] [%-m:%l] [pid %P:tid %T] %7F: %E: [client\
%a] %M% ,\ referer\ %{Referer}i"
```

Dies entspricht dem Standardformat in Apache 2.3 und erzeugt eine Ausgabe wie diese, die auf einen fehlerhaften Link hindeutet:

```
[Thu Oct 13 15:26:31.145241 2011] [core:error] [pid 6921:tid
3241673914] [client ::1:41963] File does not exist: /usr/local/
apache2/htdocs/idex.html, referer http://www.mynet.de/about.html
```

#### LogLevel

Dringlichkeitsstufe, ab der Fehlermeldungen in das `ErrorLog` geschrieben werden

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>`; seit 2.3.6 auch `<Directory>` |
| **Syntax** | `LogLevel Stufe`<br>seit 2.3.6: `Loglevel [Modul:]Stufe [Modul:Stufe] ...` |
| **Standardwert** | `warn` |

Diese Direktive bestimmt, ab welcher Dringlichkeitsstufe Meldungen in die *ErrorLog*-Datei geschrieben werden. Die Werte entsprechen den Stufen, die für `syslog` definiert sind (in Klammern sind die zugehörigen Werte für numerische `ErrorLogFormat`-Modifier angegeben):

- `emerg`: Notfall; das System ist unbrauchbar
- `alert`: Alarm – sofortiger Eingriff erforderlich (1)
- `crit`: kritischer Fehler (2)
- `error`: normaler Fehler (3)
- `warn`: Warnung (4)
- `notice`: Hinweis (5)
- `info`: normale Information (6)
- `debug`: Debugging-Information (7)
- `trace1` bis `trace8`: Zusatzinformationen bestimmter Module (8–15; seit Version 2.3.6)

Die Voreinstellung ist `warn`; zu dieser Stufe gehören Meldungen wie die folgende:

```
[Wed Aug 31 09:28:41 2011] [warn] Parent: child process 16526 did
not exit, sending another SIGHUP
```

Warnungen sind also Ausnahmezustände, die sich in der Regel ohne Eingriff des Administrators beseitigen lassen. Deshalb genügt für einen Produktionsserver meistens die Einstellung `error`. Werte, die mehr Einträge erzeugen – vor allem `debug` –, sind aufgrund des zusätzlichen Ressourcenverbrauchs eigentlich nur zu Testzwecken empfehlenswert, etwa wenn Sie eine neue Apache-Version oder ein experimentelles Modul in Betrieb nehmen möchten.

Ab Version 2.3.6-beta wurden die Fähigkeiten von `LogLevel` stark erweitert. Seit dieser Version ist es zum einen möglich, pro Verzeichnis ein anderes `LogLevel` zu wählen. Angenommen, Sie möchten im Allgemeinen das `LogLevel warn` beibehalten, in einem bestimmten Unterverzeichnis jedoch zur Untersuchung eines Fehlverhaltens auf `debug` umschalten. Dann funktioniert dies beispielsweise folgendermaßen:

```
# Serverkonfiguration
LogLevel warn
# Problematisches Verzeichnis
<Directory /usr/local/apache2/htdocs/scripts>
LogLevel debug
</Directory>
```

Außerdem wurde die Möglichkeit eingeführt, das `LogLevel` für einzelne Module zu verändern. Dazu wird die erweiterte Syntax `Modul:Stufe` verwendet, wobei mehrere durch Leerzeichen getrennte Angaben möglich sind. Das Modul kann auf drei Arten angegeben werden (hier jeweils am Beispiel von `mod_rewrite`):

- der Modul-Identifier, der beispielsweise auch bei `LoadModule` verwendet wird – `rewrite_module`
- der Module-Identifier ohne das Suffix `_module` – `rewrite`
- der Name der Haupt-Quellcodedatei des Moduls – `mod_rewrite.c`

Dieses Verfahren ersetzt einige in älteren Apache-Versionen verfügbare Spezialdirektiven für das Logging der Ausgabe bestimmter Module, insbesondere die in Abschnitt 11.1.6, »Logging-Direktiven in mod_rewrite«, ab Seite 537 vorgestellten Rewrite-Log-Direktiven.

Das folgende Beispiel setzt das `LogLevel` für `mod_rewrite` im aktuellen Konfigurationskontext auf den maximalen Wert `trace8`:

```
LogLevel rewrite:trace8
```

**Warnung:**

Ein zu hoher Trace-Wert kann die Performance von Apache deutlich beeinträchtigen. Ein höherer Wert als `trace2` sollte nur vorübergehend zu Debugging-Zwecken verwendet werden.

Wenn Sie mit ein und derselben Direktive einen allgemeinen und einen oder mehrere modulspezifische `LogLevel`-Werte einstellen möchten, muss der allgemeine an erster Stelle stehen. Das folgende Beispiel stellt `debug` als allgemeinen Wert und `trace2` für `mod_rewrite` ein:

```
LogLevel debug rewrite:trace2
```

#### HostnameLookups

DNS-Lookups von Hostnamen aktivieren

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `HostnameLookups On\|Off\|Double` |
| **Standardwert** | `Off` |

Diese Direktive kann dafür sorgen, dass Apache durch einen Reverse-DNS-Lookup die Hostnamen der anfragenden Clients ermittelt. Diese können in Log-Dateien geschrieben und außerdem über die Umgebungsvariable `REMOTE_HOST` an CGI-Skripte oder Server Side Includes weitergereicht werden. Sie können drei mögliche Werte angeben:

- `Off`: Die Hostnamen werden nicht ermittelt; es werden nur IP-Adressen protokolliert. Dies ist die Standardeinstellung.

- `On`: Der Reverse-Lookup wird durchgeführt, Apache versucht also, die Hostnamen zu ermitteln.
- `Double`: Diese Einstellung sorgt für einen Double-Reverse-Lookup. Zunächst wird ein gewöhnlicher Reverse-Lookup durchgeführt. Mit dem Ergebnis wird wiederum ein Forward-Lookup ausgeführt. Die dabei gefundene IP-Adresse muss der ursprünglich nachgeschlagenen entsprechen. Im `tcpwrappers`-Sprachgebrauch wird das Verfahren auch als `PARANOID` bezeichnet. Es stellt nämlich durch die Gegenprobe sicher, dass die Information korrekt ist.

Bei stark beanspruchten Servern sollten Sie besser `Off` beibehalten und die Hostnamen später mithilfe des weiter unten vorgestellten Tools `logresolve` ermitteln.

#### IdentityCheck

Protokollierung entfernter Benutzernamen gemäß RFC 1413

| | |
|---|---|
| **Modul** | `mod_ident` (bis Version 2.0.x im `core`) |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `IdentityCheck On\|Off` |
| **Standardwert** | `Off` |

Wenn diese Direktive eingeschaltet wird, versucht Apache, einen auf dem Client-Host laufenden `identd`-Dienst zu befragen und von ihm Benutzerinformationen zu erhalten. Falls dies funktioniert, wird die gefundene Information bei der Verwendung der Standard-Log-Formate in die *TransferLog*-Datei aufgenommen.

Normalerweise sollte diese Möglichkeit nicht verwendet werden. Zum einen leidet die Performance stark, und zum anderen brauchen eventuell ermittelte Daten überhaupt nicht der Wahrheit zu entsprechen. Darüber hinaus läuft auf den meisten Systemen ohnehin kein `identd`-Server.

#### IdentityCheckTimeout

Maximale Wartezeit für die `identd`-Abfrage

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_ident` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `IdentityCheckTimeout Sekunden` |
| **Standardwert** | `30` |

Diese Direktive gibt an, wie lange Apache maximal auf die Antwort eines `identd`-Servers warten soll. Dies gilt natürlich nur dann, wenn IdentityCheck überhaupt

eingeschaltet ist. Der Vorgabewert von 30 Sekunden wird in RFC 1413 empfohlen; Sie brauchen ihn nur dann zu verlängern, wenn Sie es mit bekanntermaßen langsamen `identd`-Servern zu tun haben.

### 11.1.2 mod_log_config

Dieses Modul stellt die Direktiven zur Einstellung der gewöhnlichen *TransferLog*-Datei sowie beliebiger anderer Log-Dateien zur Verfügung. Alle diese Direktiven werden hier ausführlich vorgestellt.

#### Kürzel für Log-Formate

Apache verwendet ein äußerst flexibles Formatverfahren, mit dessen Hilfe Sie sich die Inhalte Ihrer Log-Dateien aus zahlreichen möglichen Komponenten beliebig zusammenstellen können. Da diese Elemente in mehreren Direktiven benutzt werden können, zeigt Tabelle 11.2 zunächst einen Überblick.

| Kürzel | Beschreibung |
|---|---|
| `%%` | Prozentzeichen (seit Apache 2.0.44) |
| `%...a` | IP-Adresse des Clients |
| `%...A` | IP-Adresse des Servers |
| `%...B` | gesendete Bytes des HTTP-Bodys |
| `%...b` | gesendete Bytes des HTTP-Bodys im Common Log Format: mit – statt 0 |
| `%...{Cookie-name}C` | Inhalt des angegebenen Cookies |
| `%...D` | Dauer der Anfrageverarbeitung in Mikrosekunden |
| `%...{Variable}e` | Wert der angegebenen Umgebungsvariablen |
| `%...f` | Dateiname |
| `%...h` | Client-Hostname |
| `%...H` | Protokoll der Anfrage |
| `%...{Header}i` | Wert des angegebenen Anfrage-Headers |
| `%...l` | vom Client gesendeter `identd`-Benutzername |
| `%...m` | HTTP-Anfragemethode |
| `%...{Modul-text}n` | Wert des angegebenen Feldes aus einem angegebenen Modul |
| `%...{Header}o` | Wert des angegebenen Antwort-Headers |
| `%...p` | Server-Port |
| `%...P` | Prozess-ID des Child-Prozesses, der die Anfrage beantwortet |

**Tabelle 11.2** Mögliche Kürzel für Log-Formate

| Kürzel | Beschreibung |
|---|---|
| %...{pid\|tid}P | Prozess-ID (pid) oder Thread-ID (tid) des Child-Prozesses beziehungsweise Threads, der die Anfrage beantwortet (seit Apache 2.0.46) |
| %...q | Query-String (mit führendem Fragezeichen) |
| %...r | Startzeile der Anfrage |
| %...s | Status der ursprünglichen Anfrage |
| %...t | Datum/Uhrzeit im Common Log Format |
| %...{Format}t | Datum/Uhrzeit im angegebenen Format nach strftime()-Syntax |
| %...T | Dauer der Anfrageverarbeitung in Sekunden |
| %...u | User-ID gemäß Authentifizierung |
| %...U | angeforderter URL-Pfad ohne Query-String |
| %...v | Server-Name des antwortenden Servers |
| %...V | Server-Name gemäß UseCanonicalName |
| %...X | Verbindungsstatus:<br>X: vorzeitig abgebrochene Verbindung<br>+: persistente Verbindung<br>-: Verbindung wird geschlossen |
| %...I | Anzahl empfangener Bytes (Header und Body); erfordert mod_logio |
| %...O | Anzahl gesendeter Bytes (Header und Body); erfordert ebenfalls mod_logio |

**Tabelle 11.2** Mögliche Kürzel für Log-Formate (Forts.)

Wie Sie sehen, enthält jedes Kürzel in der Tabelle drei Punkte (z. B. %...T). Diese Platzhalter repräsentieren eine optionale, durch Komma getrennte Liste numerischer HTTP-Statuscodes ohne Leerzeichen. Das bedeutet, dass der jeweilige Wert nur in den Log-Eintrag aufgenommen wird, wenn die Anfrage mit einem der genannten Statuscodes beantwortet wurde. Das Gegenteil ist der Fall, wenn Sie der Liste ein Ausrufezeichen voranstellen: Das Feld wird demzufolge nur in die Log-Datei geschrieben, wenn die Antwort keinen der angegebenen Codes hatte.

Das folgende Beispiel schreibt nur dann den Referer in die Log-Zeile, wenn die Antwort mit einem der Statuscodes 200 OK oder 301 Moved Permanently oder 302 Found versehen war:

```
%200,301{Referer}i
```

Im nächsten Beispiel wird der Dateiname der angeforderten Ressource geschrieben, falls der Status nicht 404 Not Found lautete:

```
%!404f
```

Außerdem können Sie ein Feld durch die folgenden beiden Zeichen modifizieren:

- `<` bezieht sich auch dann auf die ursprüngliche Client-Anfrage, wenn sie intern weitergeleitet wurde.
- `>` verwendet bei einer intern weitergeleiteten Anfrage die letzte Weiterleitung, also diejenige Anfrage, die letztendlich tatsächlich beantwortet wurde.

Standardmäßig verwenden die Felder `%s`, `%U`, `%T`, `%D` und `%r` die Originalanfrage, alle anderen die letzte Weiterleitung. `%>s` ist also z. B. der Statuscode der letztendlichen Antwort, während `%<m` die HTTP-Methode der ursprünglichen Anfrage angibt.

Seit Apache 2.0.46 werden Sonderzeichen aus der Anfrage aus Sicherheitsgründen in Escape-Sequenzen umgewandelt – dies ist besonders für Piped Logs, also die Weiterleitung der Log-Daten an externe Programme, wichtig. Die betroffenen Zeichen sind: `\"` (Anführungszeichen), `\\` (Backslash), `\n` (Zeilenumbruch), `\t` (Tabulator) sowie `\xHH` (hexadezimaler Code eines beliebigen Sonderzeichens, z. B. `\xFC` für »ü« in iso-8859-1).

Mithilfe dieses Formatierungsverfahrens sieht die Definition des Common Log Formats so aus:

```
"%h %l %u %t \"%r\" %>s %b"
```

Das Combined Log Format der NCSA ist dagegen folgendermaßen definiert:

```
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
```

#### strftime( )-Zeitangaben

Das oben genannte Feld `%{Format}t` und einige andere Bereiche von `mod_log_config` sind in der Lage, `strftime()`-kompatible Zeitformate zu verarbeiten. Auch weiter unten vorgestellte Hilfsprogramme wie `rotatelogs` machen davon Gebrauch. Tabelle 11.3 zeigt deshalb gewissermaßen den kleinsten gemeinsamen Nenner aller `strftime()`-Implementierungen, das heißt diejenigen Angaben, die Sie über alle (UNIX-)Betriebssysteme hinweg sicher verwenden können.

| Format | Beschreibung |
|---|---|
| `%A` | Wochentagname (lokalisiert) |
| `%a` | abgekürzter Wochentag (lokalisiert) |
| `%B` | Monatsname (lokalisiert) |
| `%b` | abgekürzter Monat (lokalisiert) |

**Tabelle 11.3** Allgemein kompatible »strftime()«-Formate

| Format | Beschreibung |
|---|---|
| %c | Datum und Uhrzeit (lokalisiert) |
| %d | Tag des Monats, zwei Ziffern |
| %H | Stunde 00–23 |
| %I | Stunde 00–11 |
| %j | Tag des Jahres 000–366 |
| %M | Minute 00–59 |
| %m | Monat 01–12 |
| %p | AM oder PM für 12-Stunden-Anzeige |
| %S | Sekunde 00–59 |
| %U | Woche des Jahres (Starttag Sonntag) |
| %W | Woche des Jahres (Starttag Montag) |
| %w | Wochentag 0–6 (Starttag Sonntag) |
| %X | Uhrzeit (lokalisiert) |
| %x | Datum (lokalisiert) |
| %Y | Jahr (vierstellig) |
| %y | Jahr (zweistellig) |
| %Z | Name der Zeitzone |
| %% | Prozentzeichen |

**Tabelle 11.3** Allgemein kompatible »strftime()«-Formate (Forts.)

### BufferedLogs

Logs vor dem Speichern im Arbeitsspeicher sammeln

| | |
|---|---|
| **Seit Version** | 2.0.41 |
| **Modul** | mod_log_config |
| **Kontext** | Server |
| **Syntax** | BufferedLogs On\|Off |
| **Standardwert** | Off |

Normalerweise werden Log-Einträge nach jeder einzelnen Anfrage sofort in die Log-Dateien geschrieben. Wenn Sie dagegen eine Pufferung von Log-Einträgen im Arbeitsspeicher wünschen, müssen Sie auf Server-Ebene folgende Einstellung vornehmen:

```
BufferedLogs On
```

Dies verbessert auf manchen Systemen die Performance, ist aber noch nicht ausreichend getestet, um als stabil betrachtet zu werden.

### CookieLog

Log-Datei für Cookies

| Seit Version | 1.x; in 2.x veraltet; ab 2.3 nicht mehr unterstützt |
|---|---|
| Modul | mod_log_config |
| Kontext | Server, <VirtualHost> |
| Syntax | CookieLog Dateiname |
| Standardwert | nicht gesetzt |

CookieLog stellt eine spezielle Log-Datei für Usertracking-Cookies bereit. Die Direktive ist veraltet und wird nur aus Gründen der Kompatibilität zu dem veralteten Modul mod_cookies zur Verfügung gestellt. Die entsprechende Funktionalität wird inzwischen von dem weiter unten behandelten Modul mod_usertrack bereitgestellt; das Logging der Tracking-Cookies können Sie nach eigenem Gutdünken durch eine CustomLog-Direktive vornehmen. Der Dateiname kann relativ zur ServerRoot sein.

### CustomLog

Dateiname und Format für eine Log-Datei einstellen

| Modul | mod_log_config |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | CustomLog<br>Datei\|Pipe Format\|Nickname [env=[!]Umgebungsvariable] |
| Standardwert | nicht gesetzt |

Mithilfe dieser Direktive lassen sich völlig frei definierte Log-Dateien anlegen. CustomLog kann bis zu drei Argumente annehmen:

- **Log-Datei oder Pipe**
  Das erste Argument ist der Name der Datei, in die die Daten geschrieben werden sollen.

  Genau wie bei ErrorLog können Sie statt einer Datei auch eine Pipe zu einem Programm Ihrer Wahl angeben – das wichtigste Beispiel ist auch hier rotatelogs für den regelmäßigen Austausch der Datei. Die Syntax des Arguments ist "|Programmname [Argumente]"; zu den Argumenten eines typischen Log-Verarbeitungsprogramms gehört üblicherweise der Name oder das Namensmuster für die Datei, in der die Einträge letztendlich erscheinen sollen. Für rotatelogs sieht das beispielsweise so aus:

```
"|/usr/local/apache2/bin/rotatelogs \
/usr/local/apache2/logs/access_log.%Y-%m-%d"
```

Der Anhang des Dateinamenmusters ist eine Zeitangabe im `strftime()`-Format. Genauere Erläuterungen zu Piped Logs finden Sie weiter unten im Zusammenhang mit dem Tool `rotatelogs`.

Beachten Sie, dass das entsprechende Programm unter der User-ID ausgeführt wird, unter der Apache gestartet wurde. Dies ist in der Regel `root`, sodass das Programm hohen Sicherheitsanforderungen genügen muss.

- **Format**
  Als zweiten Parameter müssen Sie das Format angeben, in dem die Log-Einträge vorgenommen werden sollen. Sie können hier entweder nach dem weiter oben beschriebenen Verfahren an Ort und Stelle ein Format definieren oder aber den Nickname eines mittels `LogFormat` festgelegten Formats angeben.
- **Umgebungsvariable** (optional)
  Es besteht die Möglichkeit, das Logging von einer Bedingung abhängig zu machen. Geben Sie dazu als drittes Argument `env=Umgebungsvariable` an – der Log-Eintrag wird nur geschrieben, wenn die angegebene Variable definiert ist. Umgekehrt können Sie `env=!Umgebungsvariable` verwenden, um den Eintrag nur zu erstellen, wenn die Variable undefiniert ist.

  Beachten Sie, dass es hier in aller Regel nicht darum geht, die Existenz automatisch generierter CGI-Umgebungsvariablen zu testen. Es handelt sich vielmehr um eine Möglichkeit, in Zusammenarbeit mit Direktiven zur Variablendefinition selbst zu bestimmen, welche Einträge Sie in Ihren Log-Dateien haben möchten. Wie Sie Variablen mithilfe von `mod_rewrite` setzen können, wurde bereits in Kapitel 8, »Weiterleitungen und Indizes«, angesprochen. Ähnliche Möglichkeiten bieten die darauf spezialisierten Module `mod_env` und `mod_setenvif`, die Sie in Kapitel 14, »CGI«, kennenlernen werden.

Dieses Beispiel erstellt eine Log-Datei namens *access_log* im Common Log Format unter direkter Angabe des Formats:

```
CustomLog logs/access_log "%h %l %u %t \"%r\" %>s %b"
```

Das nächste Beispiel definiert zunächst mithilfe einer `LogFormat`-Direktive das Combined Log Format unter dem Namen `combined`, wodurch es wiederverwendbar gemacht wird. Anschließend wird die Datei *verbose_log* definiert, die diese Formatdefinition verwendet:

```
LogFormat combined \
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
CustomLog logs/verbose_log combined
```

Das folgende Beispiel definiert mithilfe von `SetEnvIf` die Variable `nolog`, wenn eine Anfrage von `localhost` selbst stammt, und schließt das Logging entsprechender Zugriffe aus:

```
# localhost ausschließen
SetEnvIf Remote_Addr "127\.0\.0\.1" nolog
# Log-Format common definieren
LogFormat common "%h %l %u %t \"%r\" %>s %b"
# Alles andere in die Log-Datei schreiben
CustomLog logs/access_log common env=!nolog
```

Das letzte Beispiel erstellt eine Combined-Log-Format-Datei sämtlicher Anfragen, die eine webtypische Bilddatei (JPEG, GIF oder PNG) angefordert haben:

```
# imgreq definieren, falls Bild angefordert
SetEnvIf Request_URI "(\.jpg)|(\.gif)|(\.png)$" imgreq
# Combined Log Format definieren
LogFormat combined \
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
# Logdatei img_log anlegen
CustomLog logs/img_log combined env=imgreq
```

### LogFormat

Format für Einträge in Log-Dateien

| | |
|---|---|
| **Modul** | `mod_log_config` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `LogFormat Format\|Nickname [Nickname]` |
| **Standardwert** | `"%h %l %u %t \"%r\" %>s %b"` |

Die Direktive `LogFormat` dient der Definition von Formaten für Log-Einträge. Es gibt drei verschiedene Schreibweisen dieser Direktive:

- Wenn Sie als einzigen Parameter ein Format angeben, wird dieses Format für nachfolgende `TransferLog`-Direktiven verwendet. Das folgende Beispiel definiert das Combined Log Format mit vorangestelltem virtuellem Host. Anschließend wird mithilfe von `TransferLog` die Standard-Log-Datei *access_log* eingerichtet:

  ```
  LogFormat "%v %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \
  \"%{User-agent}i\""
  TransferLog logs/access_log
  ```

- Sie können ein Format und einen Nickname angeben. Dies weist dem angegebenen Format einen beliebigen Namen zu, den Sie anschließend in `CustomLog`-

Direktiven verwenden können. Die Standard-Konfigurationsdatei der Apache-Distribution enthält die folgenden wichtigen Formatdefinitionen:

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \
\"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

Die Formate `combined` und `common` haben Sie bereits kennengelernt. `referer` kann verwendet werden, um eine separate *Referer*-Log-Datei anzulegen, die Einträge im Format `Referer -> URL` enthält. Ein solcher Eintrag sieht etwa so aus:

```
http://www.mynet.de/manual/license.html
-> /manual/style/css/manual.css
```

Eine *agent*-Log-Datei enthält dagegen eine einfache Liste der Browser-Kennungen, mit denen Clients auf die Site zugreifen. Beispieleintrag:

```
Mozilla/5.0 (Windows; U; WinNT4.0; de-DE; rv:1.6) Gecko/20040206
Firefox/0.8
```

Die vorgefertigte Konfigurationsdatei macht auch gleich Gebrauch von diesen Formatdefinitionen: Sie definiert die Datei *access_log* über eine `CustomLog`-Direktive im Format `common` und enthält darüber hinaus die (standardmäßig auskommentierten) Einstellungen für die Dateien *referer_log* und *agent_log*:

```
CustomLog logs/access_log common
CustomLog logs/referer_log referer
CustomLog logs/agent_log agent
```

- Eine dritte mögliche Schreibweise verwendet schließlich *nur* einen Nickname. Dies bedeutet, dass ein zuvor mithilfe einer anderen `LogFormat`-Anweisung definierter Nickname das Format für die nächste `TransferLog`-Direktive bilden soll.

### TransferLog

Pfad für eine Standard-Log-Datei

| | |
|---|---|
| **Modul** | `mod_log_config` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `TransferLog Dateiname\|Pipe` |
| **Standardwert** | nicht gesetzt |

Dies ist die klassische Direktive zur Definition einer *access_log*-Datei. Das einzige Argument ist die Datei oder Pipe, an die die Log-Einträge gesendet werden sollen; beide Möglichkeiten wurden weiter oben bereits für Direktiven wie `ErrorLog` und `CustomLog` erläutert. Das Format für die Log-Datei wird der letzten zuvor definierten `LogFormat`-Direktive entnommen oder ist ansonsten das Common Log Format (`"%h %l %u %t \"%r\" %>s %b"`), da dies der Standardwert von `LogFormat` ist.

In der vorgefertigten Apache-Konfigurationsdatei wird für die Standard-Log-Datei übrigens keine `TransferLog`-Direktive mehr verwendet, sondern `CustomLog`. In aller Regel sollten Sie es ebenso halten, weil `CustomLog` erheblich mehr Flexibilität und Kontrolle ermöglicht.

### 11.1.3 mod_log_forensic

Das Apache-Modul `mod_log_forensic` von Ben Laurie erstellt sogenannte *forensische Log-Dateien*, die eine genaue Untersuchung fehlgeschlagener Anfragen ermöglichen: Jede einzelne HTTP-Anfrage, die der Server entgegennimmt, wird mit einer eindeutigen ID versehen (dazu musste in Apache 2.0 auch `mod_unique_id` installiert sein; seit Version 2.1-beta ist dies nicht mehr erforderlich) und mit den Werten sämtlicher HTTP-Anfrage-Header, getrennt durch Pipe-Zeichen (`|`), protokolliert. Nach erfolgreichem Abschluss der Anfrage wird die ID noch einmal wiederholt. Der Starteintrag beginnt dabei mit einem Pluszeichen (`+`), der Endeintrag mit einem Minuszeichen (`-`). Das zugehörige Skript *check_ forensic* beschwert sich, wenn die Wiederholung fehlt, und findet auf diese Weise sämtliche fehlgeschlagenen Anfragen. Hier ein Beispiel für einen solchen Eintrag:

```
+yQtJf8CoAB4AAFNXBIEAAAAA|GET /manual/de/images/down.gif HTTP/
1.1|Host:localhost%3a8080|User-Agent:Mozilla/5.0 (X11; U; Linux
i686; en-US; rv%3a1.6) Gecko/20040216 Firefox/0.8
-yQtJf8CoAB4AAFNXBIEAAAAA
```

#### ForensicLog

Pfad der forensischen Log-Datei

| | |
|---|---|
| **Seit Version** | 1.3.30 und 2.0.50 |
| **Modul** | `mod_log_forensic` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ForensicLog Dateiname\|Pipe` |
| **Standardwert** | nicht gesetzt |

Dies ist die einzige Direktive des Moduls `mod_log_forensic`. Sie legt den Dateinamen für die Forensic-Log-Datei (gegebenenfalls relativ zur `ServerRoot`) oder den Namen eines externen Programms zur Weiterverarbeitung (`|Programmname`) fest. Dies funktioniert genau wie bei `CustomLog`.

Das folgende Beispiel definiert eine Forensic-Log-Datei namens *forensic_log* im Verzeichnis *logs*:

```
ForensicLog logs/forensic_log
```

### 11.1.4 mod_dumpio

Das Modul `mod_dumpio` hat die Aufgabe, *sämtliche* Eingabe- und/oder Ausgabedaten von Apache in die *ErrorLog*-Datei zu schreiben. Dies führt zu erheblichem Datenaufkommen in der Log-Datei und bremst den Server daher stark aus. Deshalb sollten die Direktiven `DumpIOInput` beziehungsweise `DumpIOOutput` ausschließlich zur Behebung spezifischer Probleme aktiviert und danach wieder abgeschaltet werden.

#### DumpIOInput

Sämtliche Eingabedaten in die *ErrorLog*-Datei schreiben

| | |
|---|---|
| **Seit Version** | 2.0.53 und 2.1.3 |
| **Modul** | `mod_dumpio` |
| **Kontext** | Server |
| **Syntax** | `DumpIOInput On\|Off` |
| **Standardwert** | `Off` |

Diese Direktive übernimmt sämtliche Eingabedaten in das `ErrorLog`, wenn sie wie folgt aktiviert wird:

```
DumpIOInput On
```

#### DumpIOOutput

Sämtliche Eingabedaten in die *ErrorLog*-Datei schreiben

| | |
|---|---|
| **Seit Version** | 2.0.53 und 2.1.3 |
| **Modul** | `mod_dumpio` |
| **Kontext** | Server |
| **Syntax** | `DumpIOOutput On\|Off` |
| **Standardwert** | `Off` |

Die Direktive `DumpIOOutput` ist für die Ausgabedaten zuständig. Um die Protokollierung aller Ausgabedaten zu aktivieren, müssen Sie folgende Zeile in den Server-Kontext einfügen:

```
DumpIOOutput On
```

#### DumpIOLogLevel

Sämtliche Eingabedaten in die *ErrorLog*-Datei schreiben

| | |
|---|---|
| **Seit Version** | 2.2.4; ab 2.3 nicht mehr unterstützt |
| **Modul** | `mod_dumpio` |
| **Kontext** | Server |
| **Syntax** | `DumpIOLogLevel Level` |
| **Standardwert** | `debug` |

Mithilfe dieser Direktive können Sie in Apache 2.2 das `LogLevel` festlegen, ab dem das Dumping der Ein- oder Ausgabe ins `ErrorLog` stattfinden soll. Die möglichen Werte finden Sie in der Beschreibung der Direktive `LogLevel` in Abschnitt 10.4.1. Vor Apache 2.2.4 war das Level nicht konfigurierbar; `mod_dumpio` wurde nur im `LogLevel debug` aktiv. Beispiel:

```
DumpIOLogLevel warning
```

In Apache 2.4 steht die Direktive nicht mehr zur Verfügung. Seit Version 2.3.6 wird ihre Funktionalität durch eine modulspezifische LogLevel-Angabe bereitgestellt. Das folgende Beispiel ist äquivalent zum vorherigen 2.2-Beispiel:

```
LogLevel dumpio:warning
```

### 11.1.5 mod_usertrack

Das Modul `mod_usertrack` ermöglicht ein einfaches Nachvollziehen von Benutzer-Sessions über Cookies. Der Nutzen ist nicht allzu groß, weil viele User Cookies deaktivieren – nicht zuletzt gerade deshalb, weil viele Site- und vor allem Bannerwerbungsbetreiber sie zum Usertracking missbrauchen. Sie sollten also auf keinen Fall Webanwendungen schreiben, die sich auf dieses Feature verlassen. Wie Sie Sessions sinnvoll einsetzen können, wird in Kapitel 15, »Technologien zur Webprogrammierung«, am Beispiel PHP gezeigt. Dafür brauchen Sie dieses Modul übrigens nicht.

Um die Informationen dieses Moduls auszuwerten, können Sie das Cookie mit der Formatangabe `%{Cookiename}C` in eine Log-Datei übernehmen, wobei

»Cookiename« die Bezeichnung ist, die Sie mithilfe der Direktive `CookieName` festgelegt haben. Alternativ steht auch die veraltete Direktive `CookieLog` (siehe oben) zur Verfügung.

#### CookieDomain

Domain für das Usertracking-Cookie

| Modul | mod_usertrack |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | CookieDomain Domain |
| Standardwert | nicht gesetzt |

Diese Direktive bestimmt das Domain-Suffix, für das das Usertracking-Cookie gelten soll. Es muss mit einem Punkt beginnen und mindestens eine Second-Level-Domain enthalten: *.sales.mynet.de* und *.mynet.de* sind also zulässig, *sales.mynet.de* oder *.de* dagegen nicht. Beispiel:

```
CookieDomain .mynet.de
```

#### CookieExpires

Verfallsdatum für das Usertracking-Cookie

| Modul | mod_usertrack |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | CookieExpires Verfallsdatum |
| Standardwert | nicht gesetzt |

Mithilfe dieser Direktive können Sie die Gültigkeitsdauer für das Usertracking-Cookie festlegen. Wenn Sie eine einfache Zahl ohne Maßeinheit verwenden, gilt dieser Wert für die Sekundenanzahl. Ansonsten können Sie die Einheiten `year[s]`, `month[s]`, `week[s]`, `day[s]`, `hour[s]`, `minute[s]` und `second[s]` verwenden, und zwar auch kombiniert. Beispiele:

```
CookieExpires 86400    # 1 Tag
CookieExpires 2 weeks
```

Wenn Sie gar keinen Wert angeben, ist das Cookie nur für die aktuelle Client-Sitzung gültig (Session-Cookie).

### CookieName

Name des Usertracking-Cookies

| Modul | mod_usertrack |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | CookieName Name |
| Standardwert | Apache (Bug in 2.0.48 und 1.3.29: muss ausdrücklich angegeben werden, da ansonsten nicht gesetzt) |

Mit dieser Direktive wird der Name für die Usertracking-Cookies festgelegt. Der Standardname sollte eigentlich `Apache` sein. In Version 2.0.48 (und übrigens auch 1.3.29) gab es allerdings einen Bug, der das explizite Angeben des Namens erforderlich machte. Beispiel:

```
CookieName UserSpyFly
```

### CookieStyle

Header-Format des Usertracking-Cookies

| Modul | mod_usertrack |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | CookieStyle Netscape \| Cookie \| Cookie2 \| RFC2109 \| RFC2965 |
| Standardwert | Netscape |

Mit `CookieStyle` wird die Syntax für den Set-Cookie-Header festgelegt, mit dem das Usertracking-Cookie an den Client übertragen wird. Sie haben die drei folgenden Möglichkeiten:

- `Netscape`: Cookie nach der ursprünglichen Netscape-Syntax
- `Cookie` oder `RFC2109`: klassisches Cookie
- `Cookie2` oder `RFC2965`: aktuelle Cookie-Syntax

In den meisten Fällen ist heutzutage `Cookie2` zu empfehlen, da dieses Format inzwischen von praktisch allen Browsern unterstützt wird.

### CookieTracking

Ein-/Ausschalten des Usertracking-Cookies

| Modul | mod_usertrack |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | CookieTracking On\|Off |
| Standardwert | Off |

Usertracking wird in einem Kontext und in dessen Unterkontexten nur ausgeführt, wenn Sie die Direktive explizit auf On setzen:

```
CookieTracking On
```

### 11.1.6 Logging-Direktiven in mod_rewrite

Das Modul mod_rewrite (siehe Kapitel 8, »Weiterleitungen und Indizes«) besaß bis zur Apache-Version 2.2 seine eigenen Logging-Direktiven. Da die Funktionalität dieses Moduls sehr komplex ist, lohnt es sich, diese einzuschalten – Fehler sind während der Entwicklung von Rewrite-Anweisungsfolgen an der Tagesordnung. Im Live-Betrieb sollte das Logging allerdings wieder ausgeschaltet werden, um Ressourcen zu schonen.

In Apache 2.3-beta und 2.4 stehen die hier beschriebenen Direktiven nicht mehr zur Verfügung, weil sie durch den modulspezifischen Einsatz von LogLevel ersetzt wurden. Dies wurde bereits im Abschnitt zu LogLevel ab Seite 520 beschrieben.

#### RewriteLog

Log-Datei für die Protokollierung von Rewrite-Vorgängen

| Version | 1.3 bis 2.2 |
|---|---|
| Modul | mod_rewrite |
| Kontext | Server, <VirtualHost> |
| Syntax | RewriteLog Dateipfad |
| Standardwert | nicht gesetzt |

Diese Direktive legt die Datei für die Protokollierung von Rewrite-Aktionen fest; der Pfad kann entweder absolut sein oder ist relativ zur ServerRoot. Eine *Rewrite-Log*-Datei sollte in der Regel nur für das Debugging neuer Rewrite-Einstellungen verwendet werden, da sie den Server verlangsamen kann. Deaktivieren lässt sie sich durch die Einstellung RewriteLogLevel 0.

Hier sehen Sie ein Beispiel:

```
RewriteLog /usr/local/apache2/logs/rewrite_log
```

#### RewriteLogLevel

Dringlichkeitsstufe, ab der Rewrite-Vorgänge in der Log-Datei erfasst werden

| | |
|---|---|
| **Version** | 1.3 bis 2.2 |
| **Modul** | `mod_rewrite` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `RewriteLogLevel Level` |
| **Standardwert** | `0` |

Diese Direktive bestimmt die Ausführlichkeit der `RewriteLog`-Datei. Der Wert ist numerisch. Bei der Einstellung 0 wird nichts protokolliert; 9 bedeutet, dass jede Kleinigkeit in die Log-Datei geschrieben wird. Beispiel:

```
RewriteLogLevel 4
```

### 11.1.7 Benutzerdefinierte Debug-Log-Meldungen

In Apache 2.3.14-beta wurde das Modul `mod_log_debug` hinzugefügt. Es ermöglicht die gezielte Ausgabe benutzerdefinierter Meldungen ins `ErrorLog`, optional unter einer bestimmten Bedingung. Damit lässt sich beispielsweise prüfen, ob und wann bestimmte Zweige der Konfiguration überhaupt gelesen werden oder ob bestimmte Bedingungen jemals eintreffen. Das Modul stellt nur eine Direktive bereit: `LogMessage`.

#### LogMessage

Eine Debug-Log-Meldung ins ErrorLog schreiben

| | |
|---|---|
| **Seit Version** | 2.3.14 |
| **Modul** | `mod_log_debug` |
| **Kontext** | `<Directory>` |
| **Syntax** | `LogMessage Meldung [hook=Hook] [if=Ausdruck]` |
| **Standardwert** | nicht gesetzt |

Wenn diese Direktive interpretiert wird, schreibt Apache einen frei definierbaren Meldungstext in die ErrorLog-Datei. Der Meldungstext selbst kann Ausdrücke in der `ap_expr`-Syntax enthalten, die in Kapitel 6, »Grundkonfiguration«, im Zusammenhang mit den Direktiven `<If>`, `<ElseIf>` und `<Else>` beschrieben wurde. Das LogLevel der Meldungen ist `info`.

Optional können Sie mit `hook=Hook` einen der folgenden Hooks angeben; dies bestimmt, in welcher Phase der Anfrage-Verarbeitung die Meldung geschrieben

wird – und damit indirekt, welche Variablen und Ausdrücke zu diesem Zeitpunkt überhaupt verfügbar sind:

- `translate_name`
- `type_checker`
- `quick_handler`
- `map_to_storage`
- `check_access`
- `check_access_ex`
- `insert_filter`
- `check_authn`
- `check_authz`
- `fixups`
- `handler`
- `log_transaction`

Die Bedeutung der einzelnen Hooks wird in Kapitel 17, »Apache erweitern«, beschrieben. Wenn Sie keinen Hook angeben, wird `log_transaction` verwendet.

Wenn Sie ein Argument vom Typ `if=Ausdruck` hinzufügen, wird die Meldung nur protokolliert, falls der Ausdruck zutrifft. Die Syntax dieses Ausdrucks entspricht ebenfalls `ap_expr`.

Das folgende Beispiel gibt immer dann eine Log-Meldung aus, wenn der URL-Pfad */check/* angefordert wird:

```
<Location /check/>
LogMessage "URL-Pfad /check/ angefordert."
</Location>
```

Hier ein weiteres Beispiel, das nur für Unterabfragen auf */check/* ausgeführt wird (hier muss explizit ein Hook angegeben werden, da `log_transaction` bei Unterabfragen nicht ausgeführt wird):

```
<Location /check/>
LogMessage "URL-Pfad /check/ in Unterabfrage angefordert." \
  hook=type_checker if=%{IS_SUBREQ}
</Location>
```

## 11.2 Auswertung von Log-Dateien

Log-Dateien sind ein sehr wichtiges und praktisches Hilfsmittel. Allerdings nützen sie nur dann wirklich etwas, wenn sie effizient ausgewertet werden können. Deshalb werden hier kurz die mit Apache mitgelieferten und darüber hinaus

einige externe Tools vorgestellt. Wenn Sie wissen möchten, was man noch alles mit Log-Dateien anfangen kann, empfehle ich Ihnen darüber hinaus das Buch [HEIN 2003].

### 11.2.1 Apache-Hilfsprogramme

Einige Hilfsprogramme zur Verarbeitung von Log-Dateien werden automatisch mit Apache installiert. Ihr grundsätzlicher Einsatz wurde weiter oben bereits erwähnt; hier finden Sie eine Kurzreferenz.

#### rotatelogs

Das Programm `rotatelogs` ermöglicht den regelmäßigen Wechsel der Log-Datei nach Ablauf einer bestimmten Zeitspanne. Dabei kann der Name der Datei mit `strftime()`-konformen Datums- und Uhrzeitkürzeln versehen werden. Die Syntax des Programms ist folgende:

```
rotatelogs Dateimuster \
[Sekunden [Zeitzonendifferenz]] | [DateigrößeM]
```

Das »Dateimuster« ist der mit `strftime()`-Formatangaben angereicherte Dateiname. Eine Tabelle der allgemein verfügbaren Kürzel finden Sie weiter oben in diesem Kapitel. Das folgende Beispiel verwendet Tag, Monat und Jahr: `access_log.%d.%m.%Y`. Sie erhalten also Dateinamen wie beispielsweise `access_log.22.08.2011`.

Das zweite (optionale) Argument ist die Wechselfrequenz in Sekunden. Geben Sie beispielsweise `86400` für einen Tag oder `604800` für eine Woche an.

Die Zeitzonendifferenz können Sie verwenden, um den Abstand Ihrer Zeitzone zur Greenwich Mean Time (oder UTC) anzugeben, damit der Wechsel auch zum passenden Zeitpunkt erfolgt. Der Wert wird mit vorangestelltem `+/–` in Minuten angegeben. In Deutschland müssen Sie also z. B. `+60` verwenden.

Eine Alternative zu den Zeitangaben ist die Dateigröße: Sie können eine Zahl mit nachgestelltem `M` verwenden, um festzulegen, dass Ihre Log-Datei nicht größer werden darf als die angegebene Megabyte-Zahl. Beispiel: `3M` für drei Megabyte.

`rotatelogs` wird so gut wie immer als Piped-Log-Programm aufgerufen. Das bedeutet, dass Sie in einer Logging-Direktive statt des einfachen Namens einer Log-Datei ein Konstrukt nach dem Schema `"|rotatelogs Dateiname [Optionen]"` verwenden können. Das folgende Beispiel legt wöchentlich eine neue Zugriffs-Log-Datei an, deren Name die Wochennummer und die Jahreszahl enthält:

```
CustomLog \
"|rotatelogs /usr/local/apache2/logs/access_log.%W-%Y 604800" \
common
```

Weitere Beispiele wurden bereits bei der Vorstellung der Logging-Direktiven gegeben.

#### logresolve

Dieses kleine Hilfsprogramm wird in der Regel nicht mittels Piped Logging, sondern über die Kommandozeile aufgerufen. Es ermittelt per DNS Hostnamen, die zu den IP-Adressen in Ihren Log-Dateien gehören. Das Programm liest von der Standardeingabe und schreibt auf die Standardausgabe, sodass Sie normalerweise die Umleitungen `< Original-Logdatei` und `> Hostnamen-Logdatei` verwenden sollten.

Die Syntax des Tools ist demnach folgende:

```
logresolve [-s Statistikdatei] [-c] < Original > Neue_Datei
```

Die Option `-s` erlaubt die Festlegung eines Dateinamens; in die Datei werden statistische Informationen geschrieben. `-c` bestimmt dagegen, dass ein Double-Reverse-DNS-Lookup durchgeführt wird (siehe Kapitel 6, »Grundkonfiguration«; dort bei der Beschreibung der Direktive `Allow`).

Wenn Sie möchten, können Sie `logresolve` zu einer Uhrzeit als Cronjob ausführen, zu der auf Ihrer Website wenig los ist. Das ist auf jeden Fall sparsamer als die Verwendung der Direktive `HostnameLookups`, zumal `logresolve` einen eingebauten DNS-Cache besitzt und somit jede Adresse nur einmal pro Durchgang ermitteln muss.

#### split-logfile

Das Perl-Skript *split-logfile* zerlegt Log-Dateien nach verschiedenen virtuellen Hosts. Dazu müssen Sie ein Log-Format verwenden, das den Namen des jeweiligen virtuellen Hosts (`%v`) an die erste Stelle setzt. Das Skript teilt die Log-Datei daraufhin in einzelne Dateien auf, die jeweils *VIRTUELLER_HOST.log* heißen.

#### log_server_status

Dieses kleine Perl-Skript dient nicht der Bearbeitung gewöhnlicher Log-Dateien. Seine einzige Aufgabe besteht darin, den Status Ihres Webservers abzufragen, zu einer einzelnen Zeile zusammenzufassen und auf die Standardausgabe zu schreiben. Damit es funktionieren kann, müssen Sie `mod_status` aktivieren und einer URL den Handler `server-status` zuordnen (siehe Kapitel 8, »Weiterleitungen und Indizes«). Anschließend können Sie `log_server_status` regelmäßig per Cronjob aufrufen und erhalten auf diese Weise eine Status-Log-Datei.

### 11.2.2 Log-Datei-Auswertung durch eigene Skripte

Für angepasste Einsatzzwecke lohnt es sich, eigene Skripte für die Auswertung von Log-Dateien zu schreiben. Die ideale Sprache dafür ist Perl, weil sich die Regeln zur Zerlegung der Dateien am besten durch reguläre Ausdrücke beschreiben lassen; diese sind bekanntlich in Perl perfekt integriert.

In diesem Unterabschnitt werden drei kleine Skriptbeispiele betrachtet, die Ihnen einen Einblick in die vielfältigen Möglichkeiten geben. Davor werden hier noch die beiden regulären Ausdrücke vorgestellt, die das Common beziehungsweise das Combined Log Format erkennt. Für das CLF gilt das folgende Ungetüm:

```
^([^\s]+)\s+([^\s]+)\s+([^\s]+)\s+\[([^\]]+)\]\s+"([^"]+)"\s+
(\d+)\s+([\d\-]+)$
```

Die Teilausdrücke in den runden Klammern bilden die einzelnen Komponenten, die tatsächlich gefunden werden sollen. Viele von ihnen werden einfach durch `[^\s]+` – ein oder mehrere Nicht-Leerzeichen – beschrieben, da sie jeweils durch Leerzeichen voneinander getrennt werden. Hinzu kommen die Sonderzeichen, die manche Bestandteile umschließen: Die eckigen Klammern um das Datum sowie die Anführungszeichen um die Anfrage (genau wie Referer und User-Agent beim Combined Log Format) stehen außerhalb der runden Klammern, werden also nicht mitgespeichert. Vorsichtshalber werden die Felder jeweils durch »ein oder mehrere Leerzeichen« (`\s+`) getrennt. Wichtig ist das Dollarzeichen als Abschlussmarkierung, da das Regexp ansonsten auch auf das Combined Log Format passen würde. Dessen regulärer Ausdruck sieht entsprechend so aus:

```
^([^\s]+)\s+([^\s]+)\s+([^\s]+)\s+\[([^\]]+)\]\s+"([^"]+)"\s+
(\d+)\s+([\d\-]+)\s+"([^"]+)"\s+"([^"]+)"
```

Der Vollständigkeit halber sei noch angemerkt, dass der beliebte deutsche Hoster 1&1 ein eigenes Log-Format verwendet, das sich mithilfe der nachfolgenden Skripte nicht verarbeiten lässt – vor dem Referer steht hier noch der Name des virtuellen Hosts; ganz am Ende kommt eine eventuelle Proxy-Liste (oder `"-"`) hinzu. Der reguläre Ausdruck müsste entsprechend erweitert werden, was hier allerdings aus Platzgründen unterbleibt.

#### Formatierte Ausgabe

Das erste Beispiel dient eher der Erläuterung der Grundlagen als der praktischen Nutzeranwendung. Es gibt die Komponenten jeder Log-Zeile in einem Format wie diesem aus:

```
1. Format:         Combined Log Format
   Remote Host:    84.165.169.1
```

```
RFC 1413 ID:    -
Remote User:    -
Date/Time:      24/Aug/2011:00:00:53 +0200
HTTP Request:   GET /it_komp/fragen.html HTTP/1.1
HTTP Status:    200
Content Length: 6337
Referer:       http://buecher.lingoworld.de/it_komp/aus  -wert.php
User Agent:     Mozilla/4.0 (compatible; MSIE 6.0;
                Windows NT 5.1; .NET CLR 1.1.4322)
```

Aufgerufen wird es mithilfe der folgenden Syntax:

```
$ ./format_log.pl Logdatei [n]
```

Wenn Sie den zweiten Parameter (eine ganze Zahl) angeben, werden nur die ersten *n* Zeilen betrachtet. Die meisten echten Log-Dateien sind zu lang, um auf diese Weise behandelt zu werden.

Listing 11.1 zeigt den vollständigen Code des Skripts. Sie finden es im Verzeichnis *logscripts* der beiliegenden DVD-ROM.

```
#!/usr/bin/perl -w
use strict;

# ****************************************
# * Skript: log_format.pl                *
# * Apache-Logdateien formatiert ausgeben *
# *                                      *
# * (C) 2005-2011, Sascha Kersken        *
# *                                      *
# * Free Software under the terms of the  *
# * GNU General Public License, V2.0:    *
# * www.gnu.org/licenses/gpl-2.0.html    *
# ****************************************

# Namen der Logdatei von der Kommandozeile lesen
my $logfile = $ARGV[0] || die "Verwendung: $0 Logdatei [n]\n";
# Eventuelle Höchstzahl lesen
my $maxlines = $ARGV[1] || 0;

# Versuchen, die Logdatei zu öffnen
open (LOG, "<$logfile") || die "Kann $logfile nicht öffnen\n";

# Variablen für die Logdatei-Bestandteile
my ($host, $id1413, $uid, $date, $request, $status, $size, $referer,
$useragent);
```

```
# Zähler, Gültigkeit der Zeile
my $count = 1;
my $valid;

# Zeilenweise einlesen und auswerten
while (my $line = <LOG>) {
   chomp $line;
   if ($line =~ /^([^\s]+)\s+([^\s]+)\s+([^\s]+)\s+
         \[([^\]]+)\]\s+"([^"]+)"\s+(\d+)\s+([\d\-]+)$/) {
      $valid = 1;
      # Common Log Format zerlegen
      ($host, $id1413, $uid, $date, $request, $status, $size) =
      ($1, $2, $3, $4, $5, $6, $7);
      # Fehlende Felder setzen
      $referer = "[ohne]";
      $useragent = "[ohne]";
      print "$count. Format: Common Log Format\n";
   } elsif ($line =~ /^([^\s]+)\s+([^\s]+)\s+([^\s]+)\s+
        \[([^\]]+)\]\s+"([^"]+)"\s+(\d+)\s+([\d\-]+)\s+
        "([^"]+)"\s+"([^"]+)"/) {
      $valid = 1;
      # Combined Log Format zerlegen
      ($host, $id1413, $uid, $date, $request, $status, $size,
           $referer, $useragent) =
           ($1, $2, $3, $4, $5, $6, $7, $8, $9);
      print "$count. Format: Combined Log Format\n";
   } else {
      $valid = 0;
      print "$count. Unrecognized line: \"$line\"\n\n";
   }
   # Ausgeben, falls gültig
   if ($valid) {
      print "   Remote Host:    $host\n";
      print "   RFC 1413 ID:    $id1413\n";
      print "   Remote User:    $uid\n";
      print "   Date/Time:      $date\n";
      print "   HTTP Request:   $request\n";
      print "   HTTP Status:    $status\n";
      print "   Content Length: $size\n";
      print "   Referer:        $referer\n";
      print "   User Agent:     $useragent\n\n";
   }
   # Zähler erhöhen
   $count++;
   # Beenden, falls begrenzt
```

```
   last if $maxlines && $count > $maxlines;
}
```

**Listing 11.1** log_format.pl; formatierte Ausgabe von Log-Dateien

### Einfache Statistik

Interessanter als die einfache Formatierung ist es, die erhaltenen Daten statistisch auszuwerten. Das hier vorgestellte Perl-Skript beschränkt sich beispielhaft auf zwei typische Werte solcher Statistiken: die treuesten Besucher(-URLs) und die am häufigsten besuchten URLs, jeweils mit Angabe ihrer Anzahl. Als Pflichtparameter müssen Sie wieder den Namen einer Log-Datei angeben; optional können Sie zusätzlich die gewünschte Höchstzahl der Ergebnisse setzen (ansonsten wird 20 verwendet). Ein typischer Aufruf dieses Skripts mit seinem Ergebnis sieht so aus:

```
$ ./log_stat.pl access_log 10
Die (bis zu) 10 haeufigsten Besucher:

1. 66.246.218.107: 696
2. 213.160.11.114: 683
3. 66.249.65.67: 386
4. 65.54.188.71: 379
5. 81.173.174.49: 306
6. 213.196.248.98: 271
7. 66.249.65.133: 270
8. 213.196.241.196: 237
9. 66.249.65.79: 157
10. 87.78.46.0: 151

Die (bis zu) 10 haeufigsten erfolgreichen Anfragen:

1. /main.css: 1791
2. /pixel.gif: 1771
3. /galileo.gif: 1593
4. /apache2/c_168_240.jpg: 1158
5. /apache2/noepatents.jpg: 1090
6. /it_komp/c_180_240.jpg: 409
7. /it_komp/noepatents.jpg: 345
8. /: 261
9. /fachinfo/c_190_240.jpg: 184
10. /apache2/mod_ssl.html: 170
```

In Listing 11.2 sehen Sie den Quellcode dieses Beispiels. Die Funktionsweise dürfte durch die relativ ausführlichen Kommentare einigermaßen klar werden.

```
#!/usr/bin/perl -w
use strict;
# ****************************************
# * Skript: log_stat.pl                  *
# * Einfache Statistik für Web-Logdateien *
# *                                      *
# * (C) 2005-2011, Sascha Kersken        *
# *                                      *
# * Free Software under the terms of the  *
# * GNU General Public License, V2.0:     *
# * www.gnu.org/licenses/gpl-2.0.html     *
# ****************************************

# Namen der Logdatei von der Kommandozeile lesen
my $logfile = $ARGV[0] || die "Verwendung: $0 Logdatei [Max]\n";
# Gewünschte Höchst-Zeilenzahl lesen oder auf 20 setzen
my $maxlines = $ARGV[1] || 20;

# Versuchen, die Logdatei zu öffnen
open (LOG, "<$logfile") || die "Kann $logfile nicht öffnen\n";

# Arrays für die Logdatei-Bestandteile
my (@hosts, @ids, @uids, @dates, @requests, @status, @sizes,
@referers, @useragents);

# Zeilenweise einlesen und auswerten
while (my $line = <LOG>) {
   chomp $line;
   if ($line =~ /^([^\s]+)\s+([^\s]+)\s+([^\s]+)\s+
            \[([^\]]+)\]\s+"([^"]+)"\s+(\d+)\s+([\d\-]+)$/) {
      push @hosts, $1;
      push @ids, $2;
      push @uids, $3;
      push @dates, $4;
      push @requests, $5;
      push @status, $6;
      push @sizes, $7;
      push @referers, "-";
      push @useragents, "-";
   } elsif ($line =~ /^([^\s]+)\s+([^\s]+)\s+([^\s]+)\s+
             \[([^\]]+)\]\s+"([^"]+)"\s+(\d+)\s+([\d\-]+)\s+
             "([^"]+)"\s+"([^"]+)"/) {
      push @hosts, $1;
      push @ids, $2;
      push @uids, $3;
      push @dates, $4;
```

```
        push @requests, $5;
        push @status, $6;
        push @sizes, $7;
        push @referers, $8;
        push @useragents, $9;
    }
}

# Anzahlen unterschiedlicher Hosts ermitteln
my %sorthosts;
foreach my $host(@hosts) {
    # Im Hash %sorthosts ist der Host der Schlüssel;
    # der Wert dient als Zähler und wird um 1 erhöht
    $sorthosts{$host}++;
}
print "Die (bis zu) $maxlines haeufigsten Besucher:\n\n";
my $c = 1;
foreach my $host
    (sort {$sorthosts{$b} <=> $sorthosts{$a}} (keys %sorthosts)) {
    print "$c. $host: ${sorthosts{$host}}\n";
    $c++;
    last if $c > $maxlines;
}
print "\n";

# Anzahlen unterschiedlicher erfolgreich angeforderter URLs
my %sorturls;
my $i = 0;
foreach my $request(@requests) {
    # URL aus dem Request ermitteln
    my $url;
    if ($request =~ /[A-Z]+\s+([^\s]+)/) {
        $url = $1;
    }
    # War diese Anfrage erfolgreich?
    if ($status[$i] == 200) {
        # URL zählen
        $sorturls{$url}++;
    }
    $i++;
}
print "Die (bis zu) $maxlines haeufigsten erfolgreichen Anfragen:\n\
n";
$c = 1;
foreach my $url
    (sort {$sorturls{$b} <=> $sorturls{$a}} (keys %sorturls)) {
```

```
    print "$c. $url: ${sorturls{$url}}\n";
    $c++;
    last if $c > $maxlines;
}
```

**Listing 11.2** log_stat.pl; einfache statistische Auswertung von Log-Dateien

Ein wenig kompliziert ist in diesem Beispiel das Verfahren zur sortierten Ausgabe der Hashes – diese sollen nämlich nicht nach ihren Schlüsseln (den IP-Adressen beziehungsweise URL-Pfaden), sondern nach den Werten sortiert werden. Die Perl-Funktion `sort()` erlaubt für solche Zwecke die Angabe einer Sortiervorschrift in Form einer eingebetteten Funktion. Die stilisierten Variablen `$a` und `$b` geben die Reihenfolge an. Für numerische Vergleiche wird der spezielle Operator `<=>` verwendet, während die String-Sortierung durch `cmp` geregelt wird. Beispielsweise werden die URL-Pfade wie folgt absteigend nach ihrer Anzahl sortiert:

```
sort {$sorturls{$b} <=> $sorturls{$a}} (keys %sorturls)
```

### Die Log-Daten in eine Datenbank übernehmen

Das letzte Beispielskript erfüllt eine besonders nützliche Aufgabe: Es schreibt die Inhalte der angegebenen Log-Datei in eine MySQL-Datenbank. Zu diesem Zweck müssen Sie eine Datenbank namens `log` mit einer Tabelle namens `logs` anlegen. Im Kommandozeilen-Client `mysql` (siehe Kapitel 15, »Technologien zur Webprogrammierung«) sehen die entsprechenden Anweisungen so aus:

```
mysql> CREATE DATABASE log;
mysql> USE log
mysql> CREATE TABLE logs (
    ->   id INT AUTO_INCREMENT,
    ->   host VARCHAR(50),
    ->   rid VARCHAR(30),
    ->   uid VARCHAR(30),
    ->   rdate DATETIME,
    ->   request VARCHAR(400),
    ->   status INT,
    ->   size INT,
    ->   referer VARCHAR(400),
    ->   useragent VARCHAR(200),
    ->   PRIMARY KEY (id)
    -> );
```

Näheres über die Perl-Datenbankschnittstelle DBI, die für die Datenbankverbindung eingesetzt wird, erfahren Sie ebenfalls in Kapitel 15, »Technologien zur Webprogrammierung«. Wichtig ist hier, dass Sie `username` und `password` durch die korrekten Werte für Ihren eigenen MySQL-Server ersetzen.

```
#!/usr/bin/perl -w
use strict;
use DBI;

# ****************************************
# * Skript: log2db.pl                    *
# * Web-Logdateien in MySQL-DB schreiben  *
# *                                      *
# * (C) 2005-2011, Sascha Kersken        *
# *                                      *
# * Free Software under the terms of the  *
# * GNU General Public License, V2.0:    *
# * www.gnu.org/licenses/gpl-2.0.html    *
# ****************************************

# Namen der Logdatei von der Kommandozeile lesen
my $logfile = $ARGV[0] || die "Verwendung: $0 Logdatei\n";

# Verbindung zur MySQL-Datenbank herstellen
my $conn = DBI->connect ("dbi:mysql:log;localhost",
     "username", "password") ||
     die "Kann Datenbankverbindung nicht herstellen\n";

# Versuchen, die Logdatei zu öffnen
open (LOG, "<$logfile") || die "Kann $logfile nicht öffnen\n";

# Variablen für die Logdatei-Bestandteile
my ($host, $id, $uid, $date, $request, $status, $size,
     $referer, $useragent);

# Gültigkeit der Zeile
my $valid;

# Monate für die Datumsumwandlung
my %months = (
   Jan => '01',
   Feb => '02',
   Mar => '03',
   Apr => '04',
   May => '05',
   Jun => '06',
   Jul => '07',
   Aug => '08',
   Sep => '09',
   Oct => '10',
```

```
   Nov => '11',
   Dec => '12'
);

# Zeilenweise einlesen und in die DB schreiben
while (my $line = <LOG>) {
   chomp $line;
   if ($line =~ /^([^\s]+)\s+([^\s]+)\s+([^\s]+)\s+
         \[([^\]]+)\]\s+"([^"]+)"\s+(\d+)\s+([\d\-]+)$/) {
      # Gültig
      $valid = 1;
      # Bestandteile übernehmen
      $host = $1;
      $id = $2;
      $uid = $3;
      $date = $4;
      $request = $5;
      $status = $6;
      $size = $7;
      $referer = "-";
      $useragent = "-";
   } elsif ($line =~ /^([^\s]+)\s+([^\s]+)\s+([^\s]+)\s+
         \[([^\]]+)\]\s+"([^"]+)"\s+(\d+)\s+
         ([\d\-]+)\s+"([^"]+)"\s+"([^"]+)"/) {
      # Gültig
      $valid = 1;
      # Bestandteile übernehmen
      $host = $1;
      $id = $2;
      $uid = $3;
      $date = $4;
      $request = $5;
      $status = $6;
      $size = $7;
      $referer = $8;
      $useragent = $9;
   } else {
      # Ungültig
      $valid = 0;
   }
   # Daten in die Datenbank schreiben, falls gültig
   if ($valid) {
      # Datum in SQL-Format umwandeln, z.B.:
      # 24/Oct/2005:00:06:58 +0200  -> "2005-10-24 00:06:58"
      my $sqldate;
      if ($date =~
```

```
            m|(\d+)/([A-Za-z]+)/(\d+):(\d+):(\d+):(\d+)|) {
        $sqldate = "$3-".$months{$2}."-$1 $4:$5:$6";
    }
    # "-" in Content-Length durch 0 ersetzen
    $size =~ s/\-/0/;
    # Die SQL-Abfrage
    my $q = "INSERT INTO logs
         (host, rid, uid, rdate, request, status, size,
          referer, useragent)
         VALUES (\"$host\", \"$id\", \"$uid\", \"$sqldate\",
         \"$request\", \"$status\", \"$size\", \"$referer\",
         \"$useragent\")";
    # Abfrage durchführen
    $conn->do ($q);
  }
}
```

**Listing 11.3** log2db.pl; Log-Datei-Inhalte in eine MySQL-Datenbank schreiben

Befinden sich die Daten erst einmal in der Datenbank, können Sie beliebige SQL-Operationen damit ausführen – die Erstellung eigener Statistiken wird so noch leichter. Abbildung 11.1 zeigt einen kleinen Ausschnitt der Daten in der webbasierten MySQL-Arbeitsoberfläche »phpMyAdmin«.

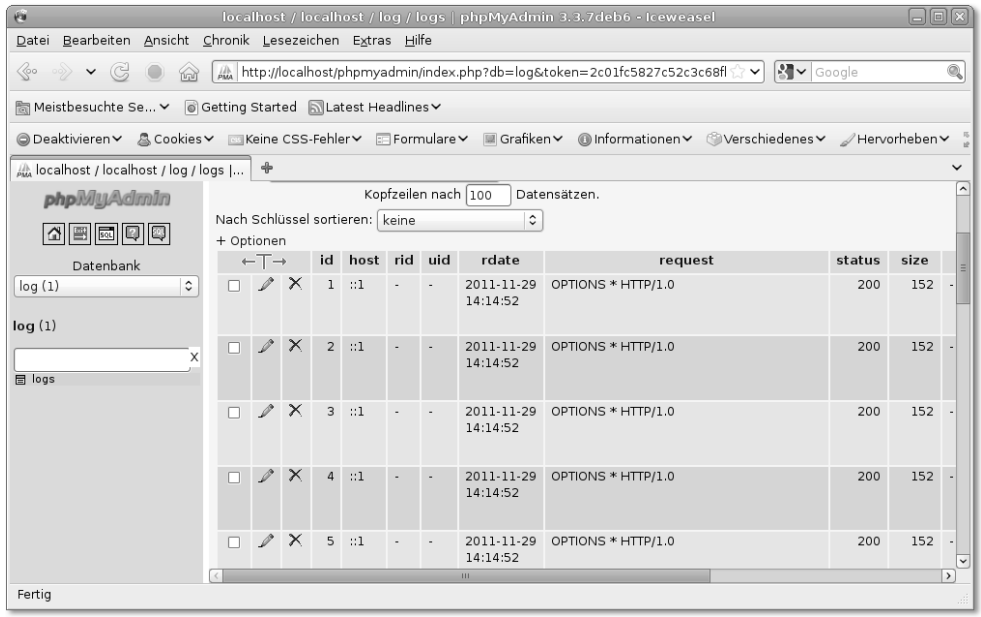



**Abbildung 11.1** Ein Ausschnitt einer in die Datenbank übernommenen Log-Datei in »phpMyAdmin«

Hier noch ein Beispiel für eine SQL-Abfrage zur statistischen Auswertung. Sie liefert die zehn Hosts mit den meisten Seitenaufrufen, genau wie das weiter oben vorgestellte Statistikskript. Sie können die Abfrage sowohl im Kommandozeilen-Client als auch in »phpMyAdmin« eingeben, sobald Sie `log` als Standarddatenbank ausgewählt haben:

```
mysql> SELECT host, COUNT(*) AS Besuche FROM logs GROUP BY host
    -> ORDER BY Besuche DESC LIMIT 0,10;
+-----------------+---------+
| host            | Besuche |
+-----------------+---------+
| 66.246.218.107  |     696 |
| 213.160.11.114  |     683 |
| 66.249.65.67    |     386 |
| 65.54.188.71    |     379 |
| 81.173.174.49   |     306 |
| 213.196.248.98  |     271 |
| 66.249.65.133   |     270 |
| 213.196.241.196 |     237 |
| 66.249.65.79    |     157 |
| 87.78.46.0      |     151 |
+-----------------+---------+
```

Weitere Informationen zur MySQL-Syntax würden den Rahmen dieses Buches sprengen. Wenn Sie sich für MySQL und seine Möglichkeiten interessieren, kann ich Ihnen mein Buch *Praktischer Einstieg in MySQL mit PHP* (2. Auflage; Köln 2007, O'Reilly Verlag) empfehlen.[1]

Wenn Sie die Verwaltung von Log-Daten in einer Datenbank nützlich finden, können Sie alternativ auch auf diverse Drittanbieter-Module zurückgreifen, beispielsweise `mod_log_mysql` für MySQL-Datenbanken. Einige dieser Module werden in Kapitel 17, »Apache erweitern«, erwähnt.

### 11.2.3 Externe Tools

Es gibt unzählige Hilfsprogramme von Drittanbietern zur Verwaltung, Analyse oder Visualisierung von Log-Dateien. Eine Auswahl davon finden Sie auf der DVD zum Buch. Unter anderem sind dort folgende Programme zu finden:

- *cronolog*
  Dieses Programm von Andrew Ford ist eine erheblich leistungsfähigere Alter-

1 Im Handel ausverkauft; kostenloser Download (PDF) unter: *http://www.oreilly.de/german/freebooks/einmysql2ger/*

native zu `rotatelogs`. Download-Möglichkeiten und zusätzliche Informationen finden Sie unter *http://www.cronolog.org*.

- *Webalizer*
  Der Webalizer ist das bekannteste Hilfsmittel zur Erzeugung grafischer Darstellungen aus Log-Dateien. Er erzeugt HTML-Dateien und Grafiken; über Konfigurationsdateien und Stylesheets können Sie ihn sehr flexibel an Ihre eigenen Bedürfnisse anpassen. Im Übrigen gibt es zahlreiche Erweiterungen und Plug-ins; interessant ist etwa der GeoLizer, der die geografische Verteilung nach IP-Adressen (per GeoIP-Datenbank) ermittelt. Die Website des Projekts ist *http://www.webalizer.com*.
- *Wusage*
  Dieses Tool arbeitet browserbasiert durch seinen eigenen kleinen Webserver auf einem ungenutzten Port. Es erstellt die bei Weitem detailliertesten Statistiken im Testfeld und lässt sich sehr intuitiv nutzen. Die Buch-DVD enthält eine Trial-Version. Die Lizenz für einen Host kostet 25 $; für 75 $ ist eine Lizenz für fünf Hosts zu haben. Download und weitere Informationen unter *http://www.boutell.com/wusage*.
- *Mescalero*
  Dies ist ein kommerzielles, aber recht kostengünstiges (Preis: 99 €) Programm für Windows. Es enthält zahlreiche Tools zur Auswertung von Log-Dateien. Die Website für dieses Programm ist *http://www.rendle.de/software/mescalero.html*; auf der DVD finden Sie eine 30-Tage-Testversion.

## 11.3 Zusammenfassung

Log-Dateien sind ein wichtiges Hilfsmittel für Sicherheit, Webdesign beziehungsweise Webentwicklung und sogar für das Marketing. Apache verfügt über ein äußerst flexibles Verfahren zum Erstellen von Log-Dateien.

Die erste Anlaufstelle bei Fehlern und Problemen ist die *ErrorLog*-Datei. Wenn Apache sich merkwürdig verhält, sollten Sie das `LogLevel` herabsetzen und sie genau beobachten. Besonders bei der Einstellung `Debug` gibt Apache bereitwillig über jeden Arbeitsschritt Auskunft. Das ist wertvoll bei der Fehlersuche sowie bei der Modulprogrammierung, sollte aber im Alltagsbetrieb natürlich nicht verwendet werden.

Die Nutzer- und Erfolgsanalyse Ihrer Website können Sie mithilfe von Zugriffsstatistiken vornehmen. Apache macht es Ihnen leicht, genau die Informationen zu sammeln, die Sie benötigen: Die Definition eigener Log-Formate kann zahlreiche Felder enthalten, die weit über das Common Log Format hinausgehen.

Die Auswertung der fertigen Log-Dateien können Sie leicht mithilfe eigener Skripte vornehmen; die Perl-Beispiele in diesem Kapitel geben einen ersten Einblick in die verschiedenen Möglichkeiten. Darüber hinaus gibt es noch zahlreiche vorgefertigte Hilfsmittel: Zum einen wird Apache mit Hilfsmitteln wie `rotatelogs` oder `logresolve` geliefert, zum anderen gibt es viele Drittanbieter-Tools, die besonders die Aufbereitung und Analyse Ihrer Logs erleichtern. Eine Auswahl davon finden Sie auf der beiliegenden DVD-ROM.

*»Persönlichkeiten werden nicht durch schöne Reden geformt, sondern durch Arbeit und eigene Leistung.«*
*– Albert Einstein*

# 12 Skalierung und Performance-Tuning

In diesem Kapitel werden verschiedene Verfahren vorgestellt, die für den Betrieb größerer Websites nützlich sind. Diese Maßnahmen lassen sich im Wesentlichen unter dem Stichwort *Skalierung* zusammenfassen. Im Einzelnen geht es um folgende Aspekte:

- **Virtuelle Hosts**
  Wie in diesem Buch bereits mehrfach erwähnt wurde, können Sie Apache 2 so einrichten, dass er für unterschiedliche IP-Adressen, Hostnamen oder Ports verschiedene Inhalte liefert. Dies wird als *Virtual Hosting* bezeichnet und ist für größere Firmen mit umfangreichen Websites nützlich. Insbesondere für Internet-Service-Provider gibt es schließlich das Modul `mod_vhost_alias`, das die dynamische Einrichtung zahlreicher virtueller Hosts ermöglicht.
- **Performance-Tuning**
  Es gibt verschiedene Möglichkeiten, die Leistungsfähigkeit des Apache-Webservers zu steigern. Hier erhalten Sie einige Praxistipps zur Performance; außerdem werden einige relevante Direktiven vorgestellt.
- **Load-Balancing**
  Wenn Ihre Site sehr populär wird und Millionen von Anfragen ausgesetzt ist, kommt ein einzelner Server-Rechner auch mit dem besten Performance-Tuning irgendwann an seine Leistungsgrenze. Dann wird es Zeit, sich über eine Lösung Gedanken zu machen, die die Last der Anfragen möglichst gleichmäßig auf mehrere physikalische Server verteilt.

## 12.1 Virtuelle Hosts

Wenn Sie auf ein und demselben Rechner mehrere Websites betreiben, werden diese als *virtuelle Hosts* bezeichnet. Apache bietet vier verschiedene grundlegende Möglichkeiten für die Einrichtung virtueller Hosts:

- **IP-basierte virtuelle Hosts**
  Wenn ein Rechner über mehrere IP-Adressen verfügt, können Sie einen virtuellen Host für jede dieser Adressen einrichten. Diese Voraussetzung ist erfüllt, wenn mehrere Netzwerkschnittstellen vorhanden sind oder wenn Sie spezielle Hilfsmittel Ihres Betriebssystems eingesetzt haben, um einer Schnittstelle mehrere Adressen zuzuweisen.

  Dasselbe können Sie übrigens auch durch zwei völlig voneinander getrennte Apache-Installationen erreichen. Sie brauchen dann lediglich `Listen`-Direktiven mit unterschiedlichen IP-Adressen zu verwenden. Beachten Sie in diesem Zusammenhang, dass Sie unter Windows nur für eine der beiden Versionen den Binär-Installer verwenden können; eine zweite müssten Sie durch Kopieren erstellen. Darüber hinaus benötigt eine der beiden Installationen einen anderen Dienstnamen. Näheres über diese Anpassung finden Sie in Kapitel 5, »Apache in Betrieb nehmen«.

- **Namensbasierte virtuelle Hosts**
  Dies ist heutzutage die gängigste Methode zur Einrichtung virtueller Hosts. Namensbasierte virtuelle Hosts werden für eine gemeinsame IP-Adresse eingerichtet; Apache entscheidet anhand des HTTP/1.1-Pflicht-Headers `Host`, welche Site ausgeliefert wird.

- **Port-basierte virtuelle Hosts**
  Wie Sie wissen, können Sie Apache so konfigurieren, dass er an mehreren TCP-Ports lauscht. Für jeden dieser Ports können Sie anschließend einen anderen virtuellen Host einrichten. Dies wird besonders häufig für spezielle Protokolle wie SSL oder für einen Proxy-Server genutzt; entsprechende Beispiele wurden in diesem Buch bereits behandelt.

- **Massen-Hosting mit mod_vhost_alias**
  Dieses Modul ist besonders für Webhoster interessant: Es erlaubt die automatisierte Einrichtung virtueller Hosts anhand von Verzeichnisstrukturen.

### 12.1.1 Konfigurationsbeispiele

Die Container `<VirtualHost>` beziehungsweise `<VirtualHostMatch>` wurden bereits in Kapitel 6, »Grundkonfiguration«, vorgestellt. Sie enthalten jeweils die Direktiven, die für einen bestimmten virtuellen Host (beziehungsweise eine durch reguläre Ausdrücke definierte Gruppe virtueller Hosts) gelten. Daneben gibt es noch einige andere Core-Direktiven für die Einrichtung und Anpassung virtueller Hosts. Diese werden weiter unten ausführlich vorgestellt. Hier finden Sie zunächst einmal einige Konfigurationsbeispiele für die unterschiedlichen Arten virtueller Hosts.

### Einrichtung IP-basierter virtueller Hosts

Für den Einsatz IP-basierter virtueller Hosts benötigen Sie lediglich `<VirtualHost>`-Container. Jeder dieser Container benötigt mindestens eine `DocumentRoot`-Direktive, damit der entsprechende virtuelle Host eine andere Website ausliefert. Das folgende Beispiel definiert IP-basierte virtuelle Hosts für zwei unterschiedliche IP-Adressen des Rechners:

```
<VirtualHost 196.23.17.51>
  ServerName www1.mynet.de
  ServerAdmin webguy@mynet.de
  DocumentRoot /var/www1/htdocs
  ErrorLog logs/www1.error_log
  CustomLog logs/www1.access_log common
</VirtualHost>

<VirtualHost 196.23.17.52>
  ServerName www2.mynet.de
  ServerAdmin webgirl@mynet.de
  DocumentRoot /var/www2/htdocs
  ErrorLog logs/www2.error_log
  CustomLog logs/www2.access_log common
</VirtualHost>
```

Es ist dringend zu empfehlen, eine `ServerName`-Direktive zu verwenden. Andernfalls muss Apache nämlich einen Reverse-DNS-Lookup durchführen, um den Namen zu ermitteln. Außerdem sollten Sie für die `<VirtualHost>`-Container die IP-Adresse und nicht den Hostnamen verwenden – sonst müsste sie ebenfalls per DNS ermittelt werden. DNS-Operationen können den Start von Apache erheblich verzögern – möglicherweise startet der Server sogar gar nicht, wenn der DNS-Zugriff gerade nicht möglich ist.

Wie Sie sehen, lassen sich auch für jeden virtuellen Host eigene Log-Dateien anlegen. Wie Sie spezifische Informationen über die virtuellen Hosts selbst in Log-Formate integrieren können, erfahren Sie in Kapitel 11, »Logging«. Wenn Sie mehrere unterschiedliche Sites betreiben, ist es außerdem ratsam, je eine `ServerAdmin`-Einstellung mit unterschiedlichen E-Mail-Adressen zu wählen, um eventuelle Problemberichte von Usern sofort der richtigen Site zuordnen zu können.

**Tipp zum Ausprobieren**

Angenommen, Sie möchten das Prinzip IP-basierter virtueller Hosts gern ausprobieren, aber Ihr System verfügt nur über eine Netzwerkschnittstelle. Wenn Ihr Betriebssystem keine Möglichkeit bietet, dieser Schnittstelle mehrere Adressen zuzuweisen, oder wenn Sie diese nicht kennen, können Sie noch immer folgendes Verfahren anwenden: Richten

Sie nach dem oben gezeigten Schema je einen virtuellen Host für die IP-Adresse Ihrer Netzwerkkarte und einen für die Loopback-Schnittstelle (127.0.0.1) ein. Nun können Sie einen Browser öffnen und nacheinander beide Adressen eingeben – Sie sollten die beiden unterschiedlichen Startseiten sehen, die Sie mittels `DocumentRoot` eingerichtet haben.

**Einrichtung namensbasierter virtueller Hosts**

Für einen namensbasierten virtuellen Host benötigten Sie bis einschließlich Apache 2.3.10 die Direktive `NameVirtualHost`. Sie legt die gemeinsame IP-Adresse fest, unter der die verschiedenen virtuellen Hosts betrieben werden. Sie können auch ein `*` verwenden, um die IP-Adressen aller Netzwerkschnittstellen für das namensbasierte Virtual Hosting freizuschalten – optional gefolgt von einem durch Doppelpunkt getrennten TCP-Port.

In neueren Versionen von Apache 2.3-beta wurde die Direktive `NameVirtualHost` ersatzlos gestrichen. Apache erkennt nun anhand des Vorhandenseins mehrerer `<VirtualHost>`-Konfigurationsblöcke mit derselben IP- oder Hostnamens- und Portkombination, dass es sich um namensbasierte virtuelle Hosts handelt. Die Direktive ist zwar aus Kompatibilitätsgründen noch vorhanden, hat jedoch keinen Effekt mehr.

Die `ServerName`-Header innerhalb der einzelnen `<VirtualHost>`-Container bestimmen, welche Anfragen durch den jeweiligen virtuellen Host bedient werden. Beachten Sie bitte, dass es nach der Einschaltung des namensbasierten Virtual Hosting für alle IP-Adressen keinen »Hauptserver« mehr gibt. Alle Anfragen werden durch die verschiedenen virtuellen Hosts beantwortet, sodass Sie die Konfigurationseinstellungen des bisherigen Hauptservers in einen `<VirtualHost>`-Container kopieren müssen.

Das folgende Beispiel zeigt, wie Sie zwei namensbasierte virtuelle Hosts für die Adressen *www.mynet.de* und *order.mynet.de* einrichten können:

```
# Nur für Apache bis 2.3.10:
NameVirtualHost *:80

<VirtualHost *:80>
  ServerName www.mynet.de
  ServerAdmin webmaster@mynet.de
  DocumentRoot /usr/local/apache2/htdocs
  ErrorLog logs/error_log
  CustomLog logs/access_log common
</VirtualHost>

<VirtualHost *:80>
```

```
  ServerName order.mynet.de
  ServerAdmin webmaster@order.mynet.de
  DocumentRoot /var/order/htdocs
  ErrorLog logs/order.error_log
  CustomLog logs/order.access_log common
</VirtualHost>
```

Anstelle des Ports können Sie auch die (gleiche) IP-Adresse angeben. Das ist besonders nützlich, wenn Sie namens- und IP-basiertes virtuelles Hosting mischen möchten. Das folgende Beispiel definiert zwei namensbasierte virtuelle Hosts unter derselben IP-Adresse und einen weiteren unter einer anderen:

```
# Nur für Apache bis 2.3.10:
NameVirtualHost 196.17.23.52

# Der bisherige Hauptserver
<VirtualHost 196.17.23.51>
  ServerName www.mynet.de
  ServerAdmin webmaster@mynet.de
  DocumentRoot /usr/local/apache2/htdocs
  ErrorLog logs/error_log
  CustomLog logs/access_log common
</VirtualHost>

# Die beiden namensbasierten virtuellen Hosts
<VirtualHost 196.17.23.52>
  ServerName order.mynet.de
  ServerAdmin webmaster@order.mynet.de
  DocumentRoot /var/order/htdocs
  ErrorLog logs/order.error_log
  CustomLog logs/order.access_log
</VirtualHost>

<VirtualHost 196.17.23.52>
  ServerName advertising.mynet.de
  ServerAdmin webmaster@advertising.mynet.de
  DocumentRoot /var/advertising/htdocs
  ErrorLog logs/advertising.error_log
  CustomLog logs/advertising.access_log
</VirtualHost>
```

#### Einrichtung Port-basierter virtueller Hosts

Port-basierte virtuelle Hosts wurden in diesem Buch bereits an verschiedenen Stellen verwendet: Bei der SSL-Konfiguration (siehe Kapitel 10, »Gesicherte Ver-

bindungen«) ist es üblich, für einzelne TCP-Ports spezielle Einstellungen vorzunehmen; bei der Proxy-Konfiguration ist es auf jeden Fall nützlich.

Beachten Sie, dass die `Listen`-Direktiven, die dafür sorgen, dass Apache an den verschiedenen Ports lauscht, nicht innerhalb der jeweiligen `<VirtualHost>`-Container stehen dürfen, sondern im Server-Kontext stehen müssen. Deshalb brauchen Sie auch nicht für jeden Port einen virtuellen Host einzurichten; bei zwei Ports genügt es beispielweise, einen von ihnen in einem `<VirtualHost>`-Container zu konfigurieren. Im folgenden Beispiel lauscht der Server an den Ports 80 und 8000. Es werden die relevanten Direktiven des Hauptservers (Port 80) und des virtuellen Hosts für das Intranet (Port 8000) gezeigt:

```
Listen 80
Listen 8000

# Hauptserver: Port 80
ServerName www.mynet.de
ServerAdmin webmaster@mynet.de
DocumentRoot /usr/local/apache2/htdocs
ErrorLog logs/error_log
CustomLog logs/access_log common

# Intranet-Server: Port 8000
<VirtualHost *:8000>
  ServerName intranet.local
  ServerAdmin it-support@mynet.de
  DocumentRoot /var/intranet/htdocs
  ErrorLog logs/intranet.error_log
  CustomLog logs/intranet.access_log
</VirtualHost>

# Intranetverzeichnis vor der Außenwelt schützen
<Directory /var/intranet/htdocs>
  Order deny,allow
  Deny from all
  Allow from 192.168.0
</Directory>
```

### 12.1.2 Core-Direktiven für virtuelle Hosts

In diesem Abschnitt werden die einzelnen Core-Direktiven für die Einrichtung und Anpassung virtueller Hosts vorgestellt. Die Syntax des Containers `<VirtualHost>` wurde allerdings bereits im Rahmen der Konfigurationskontexte in Kapitel 6, »Grundkonfiguration«, vorgestellt.

### NameVirtualHost

Festlegung eines namensbasierten virtuellen Hosts

| | |
|---|---|
| **Version** | seit 2.3.11 veraltet; wird abgeschafft |
| **Modul** | `core` |
| **Kontext** | Server |
| **Syntax** | `NameVirtualHost Adresse[:Port]` |
| **Standardwert** | nicht gesetzt |

Die Direktive `NameVirtualHost` konfiguriert eine Adresse und optional einen TCP-Port für den Einsatz virtueller Ports. Als Adresse sollten Sie bevorzugt eine IP-Adresse angeben; wenn Sie einen Hostnamen verwenden, kann es zu den oben erwähnten Problemen bei DNS-Ausfällen kommen. Wenn alle Netzwerkschnittstellen angesprochen werden sollen, können Sie anstelle einer konkreten Adresse `*` verwenden.

Beachten Sie, dass es für die angegebene IP-Adresse keinen Hauptserver mehr gibt – Sie müssen die bisherige Hauptserver-Konfiguration ebenfalls in einen `<VirtualHost>`-Container verlagern.

Hier sehen Sie einige Beispiele:

```
# IPv4-Adresse
NameVirtualHost 196.17.23.52

# IPv6-Adresse
NameVirtualHost [9AC:7B03:CC3D:BD88::37:6293:CCF1]

# Port; beliebige Adresse
NameVirtualHost *:80

# Adresse und Port
NameVirtualHost 196.17.23.52:80
```

Seit Version 2.3.11 erkennt Apache anhand der `<VirtualHost>`-Container selbstständig, dass es sich um namensbasierte virtuelle Hosts handelt. Daher ist diese Direktive veraltet und sollte aus der Konfiguration entfernt werden. Zurzeit wird sie zwar noch stillschweigend ignoriert, aber es ist damit zu rechnen, dass sie auf Dauer komplett ungültig wird und ihr Verbleib in der Konfiguration zu einer Fehlermeldung führen wird.

### ServerAlias

Alternativer Hostname für einen namensbasierten virtuellen Host

| Modul | core |
|---|---|
| Kontext | <VirtualHost> |
| Syntax | ServerAlias Hostname [Hostname] |
| Standardwert | nicht gesetzt |

Wenn ein namensbasierter virtueller Host über mehrere Hostnamen angesprochen werden soll, können Sie mit dieser Direktive einen oder mehrere Aliasnamen festlegen. Hier ein Beispiel:

```
# Nur bis Version 2.3.10:
NameVirtualHost *:80

<VirtualHost *:80>
  ServerName www.mynet.de
  ServerAlias mynet.de web.mynet.de
  # ...
</VirtualHost>
```

### ServerPath

Alternativer URL-Pfad eines namensbasierten virtuellen Hosts für HTTP/1.0-Clients, die keinen `Host`-Header senden

| Modul | core |
|---|---|
| Kontext | <VirtualHost> |
| Syntax | ServerPath URL-Pfad |
| Standardwert | nicht gesetzt |

Wie bereits erwähnt, beruht die Funktion namensbasierter virtueller Hosts auf der Auswertung des Anfrage-Headers `Host`. Dieser ist bei HTTP/1.1 vorgeschrieben und wird sogar von den meisten HTTP/1.0-Browsern gesendet. Nur einige wirklich alte Clients senden diesen Header nicht. Wenn Sie die Benutzer solcher Browser nicht ausschließen möchten, können Sie die Direktive `ServerPath` setzen. Sie ermöglicht es, die URLs eines virtuellen Hosts in ein Unterverzeichnis eines anderen virtuellen Hosts abzubilden. Hier sehen Sie ein Beispiel:

```
<VirtualHost *:80>
  ServerName test.mynet.de
  ServerPath /seiten
  DocumentRoot /usr/local/apache2/htdocs/seiten
  # ...
</VirtualHost>
```

URL-Pfade, die mit */seiten* beginnen, liefern auf diese Weise – unabhängig vom angeforderten Domain-Namen – dieselbe Site wie *test.mynet.de* bei einer Anfrage mit `Host`-Header. Vom ersten virtuellen Host aus können Sie nun einen Link auf *www.mynet.de/seiten* setzen, damit auch die veralteten Clients ohne Probleme bedient werden können.

### 12.1.3 mod_vhost_alias

Das Modul `mod_vhost_alias` ist für die umfangreichen Websites großer Unternehmen und für Webhoster interessant: Es ermöglicht die Konfiguration beliebig vieler gleichartiger virtueller Hosts; die unterschiedlichen Adressen werden dabei anhand von Mustern in verschiedene Unterverzeichnisse umgesetzt. Übrigens lässt sich eine ähnliche Funktionalität auch mit `mod_rewrite` erreichen (siehe Kapitel 8, »Weiterleitungen und Indizes«).

Hier ein erstes kurzes Praxisbeispiel für namensbasierte virtuelle Hosts:

```
# Nur bis 2.3.10:
NameVirtualHost *:80

<VirtualHost *:80>
  ServerName www.mynet.de
  UseCanonicalName Off
  VirtualDocumentRoot /var/htdocs/%0
  # ...
</VirtualHost>
```

`UseCanonicalName` muss auf `Off` gesetzt werden, damit der vom Client angeforderte Domain-Name auch bei Weiterleitungen beibehalten wird. Die eigentlich wichtige Konfigurationsanweisung ist aber `VirtualDocumentRoot`: Durch das Kürzel `%0` wird der gesamte Domain-Name zum Teil des Verzeichnisnamens. Falls ein Benutzer nun *sales.mynet.de* anfordert, versucht Apache, die Dateien aus dem Verzeichnis */var/htdocs/sales.mynet.de* zu lesen; für *www.othernet.de* gilt entsprechend */var/htdocs/www.othernet.de*.

In `VirtualDocumentRoot` und den anderen Direktiven von `mod_vhost_alias` können Sie aber nicht nur `%0` verwenden, sondern jedes der folgenden Kürzel, die ihrerseits verschiedene Teile der Adresse bezeichnen:

- **`%%`** – ein Prozentzeichen (wird wahrscheinlich niemals benötigt)
- **`%p`** – die Portnummer aus der angeforderten Adresse
- **`%N`** – ein Teil des Domain-Namens oder der IP-Adresse:
  - Wie bereits erwähnt, ist `%0` die gesamte Adresse.

  - Eine positive Zahl steht für das jeweils durch einen Punkt abgetrennte Teilstück der Adresse. Bei dem Namen `www.mynet.de` steht `%1` also für `www` und `%3` für `de`.
  - Negative Zahlen bezeichnen ebenfalls die einzelnen Teilstücke des Domain-Namens, zählen diese allerdings von rechts nach links: `%-1` ist ebenfalls `de`, `%-2` wäre `mynet`.
  - Schließlich können Sie noch ein nachgestelltes `+` verwenden; ein solcher Name steht für das angegebene Element sowie für alle weiteren. `%2+` ist demnach `mynet.de`, `%-2+` dagegen `www.mynet`.
- `%N.M` – Beschreibt die einzelnen Zeichen eines Bestandteils. `%N` ist dabei wiederum ein Bestandteil der Adresse; `%M` bezeichnet nach denselben Regeln dessen einzelne Zeichen: Bei `www.mynet.de` wäre `%2.2` der Buchstabe `y`; `%2.-1` ist `t`. Auch das nachgestellte `+` ist wieder möglich: `%2.3+` steht für `net`, während `%2.-4+` die Zeichenfolge `my` darstellt.

Das folgende Beispiel könnte also z.B. verwendet werden, um die Sites *www1.mynet.de* bis *www99.mynet.de* in virtuelle Hosts mit den Stammverzeichnissen */var/sites/docs1* bis */var/sites/docs99* umzusetzen:

```
NameVirtualHost *:80

<VirtualHost *:80>
  UseCanonicalName Off
  VirtualDocumentRoot /var/sites/docs%2.4+
  # ...
</VirtualHost>
```

#### VirtualDocumentRoot

Stammverzeichnisse für die Websites virtueller Hosts nach Hostnamen

| | |
|---|---|
| **Modul** | `mod_vhost_alias` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `VirtualDocumentRoot Verzeichnis \| none` |
| **Standardwert** | `none` |

Diese Direktive hat dieselbe Aufgabe wie `DocumentRoot`. Die Besonderheit besteht darin, dass Sie nach diesem Schema die Stammverzeichnisse für mehrere virtuelle Hosts dynamisch angeben können. Wenn Sie den speziellen Wert `none` verwenden, wird das Feature deaktiviert – dies ist nützlich, um neben vielen gleichartigen virtuellen Hosts einen einzelnen mit einer speziellen Konfiguration zu betreiben.

### VirtualDocumentRootIP

Stammverzeichnisse für die Websites virtueller Hosts nach IP-Adressen

| Modul | mod_vhost_alias |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | VirtualDocumentRootIP Verzeichnis \| none |
| Standardwert | none |

`VirtualDocumentRootIP` ist das Pendant zu `VirtualDocumentRoot` für IP-basierte virtuelle Hosts: Sie können nach diesem Schema die angeforderte IP-Adresse zerlegen, um die verschiedenen Stammverzeichnisse zu bilden. Das folgende Beispiel definiert virtuelle Hosts für verschiedene IP-Adressen im Netzwerk 196.17.23.0/24; die Verzeichnisnamen sind */var/sites/ip1* bis */var/sites/ip254*:

```
VirtualDocumentRootIP /var/sites/ip%4
```

### VirtualScriptAlias

CGI-Verzeichnisse für die Websites virtueller Hosts nach Hostnamen

| Modul | mod_vhost_alias |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | VirtualScriptAlias Verzeichnis \| none |
| Standardwert | none |

Diese Direktive ermöglicht die dynamische Definition von CGI-Verzeichnissen für namensbasierte virtuelle Hosts. Anders als bei `ScriptAlias` (siehe Kapitel 14, »CGI«) ist der Pfad, unter dem das Verzeichnis in das URL-Schema des virtuellen Hosts eingebunden wird, nicht frei wählbar; es ist immer */cgi-bin*. Das folgende Beispiel definiert die Stammverzeichnisse unter */var/sites/** und die CGI-Verzeichnisse unter */var/sites/cgi/**, sodass ein Zugriff auf *www.othernet.de/cgi-bin/* beispielsweise im Verzeichnis */var/sites/cgi/ www.othernet.de* landet:

```
VirtualDocumentRoot /var/sites/%0
VirtualScriptAlias /var/sites/cgi/%0
```

### VirtualScriptAliasIP

CGI-Verzeichnisse für die Websites virtueller Hosts nach IP-Adressen

| Modul | mod_vhost_alias |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | VirtualScriptAliasIP Verzeichnis \| none |
| Standardwert | none |

`VirtualScriptAliasIP` ist die Variante von `VirtualScriptAlias` für IP-basierte virtuelle Hosts.

## 12.2 Performance-Tuning

Apache 2 bietet durch seine gute Anpassung an die Fähigkeiten des jeweiligen Betriebssystems bereits ab Werk eine hohe Performance. Die wichtigsten Techniken, mit denen dies erreicht wird, sind die verschiedenen Multiprocessing-Module (MPMs) sowie die Apache Portable Runtime (APR), die die speziellen Fähigkeiten und Optimierungen der verschiedenen Betriebssysteme verallgemeinert und für die Apache-Programmierung verfügbar macht. Beide Themen wurden bereits in Kapitel 3, »Apache 2 im Überblick«, erläutert.

Wenn auf Ihrem Webserver sehr viele Anfragen eintreffen, müssen Sie sich möglicherweise trotzdem Gedanken über Maßnahmen zur Performance-Steigerung machen. Diese Maßnahmen betreffen vier verschiedene Gesichtspunkte; bis auf die letztgenannte gehören sie nicht zum Themenbereich dieses Buches:

- **Hardware**
  Die beste Softwareoptimierung nützt nichts, wenn eine Hardwarekomponente einen »Flaschenhals« (einen Leistungsengpass) bildet. Die allerwichtigste Voraussetzung für den reibungslosen Betrieb eines hoch belasteten Webservers ist ein ausreichender Arbeitsspeicher.
- **Netzwerkanbindung**
  Wenn Ihre Verbindung zum Netzwerk beziehungsweise Internet nichts taugt, nützt Ihnen selbst eine Aufrüstung der Hardware nichts mehr.
- **Betriebssystem**
  Neuere Versionen Ihres Betriebssystems enthalten möglicherweise verbesserte Fassungen von Systemaufrufen und Bibliotheksfunktionen; durch ein einfaches Upgrade lässt sich auf diese Weise oft die Leistung steigern. Denken Sie zum Beispiel an den Linux-Kernel, der seit Version 2.6 einen sogenannten *O(1)-Scheduler* enthält.[1]
- **Apache-Konfiguration**
  Im folgenden Abschnitt finden Sie einige kurze Konfigurationstipps zur Performance-Verbesserung. Weiter unten werden zwei Core-Direktiven und ein

1 Die O-Notation bezeichnet die Komplexität und damit das Geschwindigkeitsverhalten von Algorithmen. O(1) steht für eine absolut konstante Komplexität. Das Scheduling – die Entscheidung, welcher Prozess ausgeführt werden soll – funktioniert bei der neuesten Linux-Generation also ungeachtet der Prozessanzahl immer gleich schnell.

Modul vorgestellt, die ebenfalls zur Verbesserung der Performance beitragen können.

- **Vorschalt-Cache**
  Wie Sie im nächsten Kapitel ausführlich erfahren werden, kann Apache als Webcache eingesetzt werden, um fertig zusammengestellte, dynamisch generierte Seiten zu cachen oder auch als Proxy-Cache für weitere Server zu dienen. Es gibt allerdings auch Drittanbieter-Software, die hochgradig auf diese eine Aufgabe optimiert ist. Infrage kommen beispielsweise der Klassiker *squid* (*http://www.squid-cache.org*) oder der neuere *varnish* (*https://www.varnish-cache.org/*), der beispielsweise von Haus aus auf mehrere Prozessorkerne skalieren kann (bei squid wird dies erst allmählich nachgerüstet).

### 12.2.1 Allgemeines

Die folgenden Punkte stellen lediglich eine stichwortartige Zusammenfassung von Performance-Tipps dar, die an anderen Stellen dieses Buches im jeweiligen Zusammenhang erläutert werden:

- **Vermeiden Sie DNS-Lookups.**
  Jede Direktive, die direkt oder indirekt zur Befragung eines Nameservers führt, kostet Zeit. Sie sollten also in aller Regel nicht die Direktive `HostNameLookups` aktivieren, sondern die Hostnamen zu den IP-Adressen in Log-Dateien nachträglich ermitteln (siehe Kapitel 11, »Logging«). Darüber hinaus sollten in der Konfiguration, wann immer möglich, IP-Adressen anstelle von Hostnamen verwendet werden, beispielsweise zur Definition virtueller Hosts (siehe weiter oben in diesem Kapitel) oder für die Zugriffskontrolle mit `Oder`, `Allow` und `Deny` (siehe Kapitel 6, »Grundkonfiguration«).
- **Setzen Sie mod_rewrite mit Bedacht ein.**
  Wie Sie in Kapitel 8, »Weiterleitungen und Indizes«, erfahren haben, ist `mod_rewrite` ein überaus mächtiges Werkzeug. Allerdings geht diese Leistungsfähigkeit auch mit einem gewissen Ressourcenhunger einher. Sie sollten sich deshalb genau überlegen, ob Sie für eine bestimmte Konfigurationseinstellung wirklich `Rewrite*`-Direktiven benötigen oder ob sich dasselbe Ergebnis beispielsweise auch mit `mod_alias`-Direktiven erreichen lässt. Wenn Sie auf `mod_rewrite` angewiesen sind, sollten Sie wiederum exzessives Rewrite-Logging vermeiden; dies dient eigentlich nur der gezielten Fehlersuche.
- **Vermeiden Sie .htaccess-Dateien.**
  Bereits in Kapitel 6, »Grundkonfiguration«, wurde ausgeführt, dass der Einsatz von *.htaccess*-Dateien nur in wenigen Fällen sinnvoll ist. In aller Regel sollten Sie überall `AllowOverride None` setzen, damit Apache keine Zeit mit der Suche nach *.htaccess*-Dateien vergeudet. Dieser Tipp wird in der Apache-Originaldo-

kumentation genannt, aber ein Test, den ich vor Kurzem in einer Produktivumgebung durchgeführt habe, zeigt, dass der Verzicht auf *.htaccess* nur eine sehr geringfügige Performance-Steigerung mit sich bringt.

- **Verwenden Sie Type-Maps anstelle von MultiViews.**
  Wenn Sie Content-Negotiation einsetzen, sollten Sie Ihre Type-Map-Dateien nach Möglichkeit selbst schreiben, anstatt `MultiViews` zu verwenden. Letzteres ist relativ langwierig, weil Apache den Inhalt der Verzeichnisse durchsuchen muss, um die Type-Maps automatisch zu konstruieren. Die genaue Vorgehensweise wird in Kapitel 7, »Header und MIME-Einstellungen«, beschrieben.
- **Aktivieren Sie nur benötigte Module.**
  Wenn Sie Module als DSOs laden, sollten Sie nur die `LoadModule`-Anweisungen für diejenigen Module stehenlassen, die Sie tatsächlich benötigen; bei Konfigurationen mit statischen Modulen sollten Sie überflüssige Module sogar gar nicht erst einkompilieren.
- **Optimieren Sie Ihre MPM-Einstellungen.**
  In Kapitel 6, »Grundkonfiguration«, finden Sie zahlreiche Direktiven zur Manipulation verschiedener Aspekte der Multiprocessing-Module. Besonders die Einstellungen zur Anzahl der Child-Prozesse beziehungsweise Threads sollten Sie sich näher anschauen, um die optimalen Werte für Ihre Situation herauszufinden.
- **Überwachen Sie den Server-Status.**
  Das Modul `mod_status` stellt den Handler `server-status` bereit, der Informationen über die aktuelle Belastung des Servers veröffentlichen kann. Näheres dazu finden Sie in Kapitel 8, »Weiterleitungen und Indizes«.

### 12.2.2 Benchmarks mit ab

Apache 2 wird mit einem Kommandozeilen-Hilfsprogramm namens `ab` (Apache Benchmark) ausgeliefert. Es ermöglicht Performance-Tests beliebiger URLs, nicht nur Ihres eigenen Servers. Das Utility befindet sich im Verzeichnis `bin` Ihrer Apache-Installation. Der folgende Beispielaufruf sendet in zehn nebenläufigen Prozessen 10.000 Anfragen an *www.mynet.de/* (der abschließende Slash ist hier wichtig, andernfalls funktioniert es aufgrund der Weiterleitung nicht):

```
# ab -n 10000 -c 10 http://www.mynet.de/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright (c) 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking www.mynet.de (be patient)
```

```
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Finished 10000 requests


Server Software:        Apache/2.2.9
Server Hostname:        www.mynet.de
Server Port:            80

Document Path:          /
Document Length:        159 bytes

Concurrency Level:      10
Time taken for tests:   163.374922 seconds
Complete requests:      10000
Failed requests:        0
Write errors:           0
Total transferred:      5110000 bytes
HTML transferred:       1590000 bytes
Requests per second:    61.21 [#/sec] (mean)
Time per request:       163.375 [ms] (mean)
Time per request:       16.337 [ms] (mean, across all concurrent
requests)
Transfer rate:          30.54 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    6  14.2     10     480
Processing:    60  155  52.0    140    1001
Waiting:        0  119  70.1    110     550
Total:         60  162  53.2    140    1011

Percentage of the requests served within a certain time (ms)
  50%    140
  66%    180
  75%    190
  80%    190
  90%    200
  95%    210
```

```
  98%    230
  99%    250
 100%   1011 (longest request)
```

Tabelle 12.1 zeigt sämtliche Kommandozeilenoptionen des Programms im Überblick.

| ab-Option | Erläuterung |
|---|---|
| `-n Anzahl` | Anzahl durchzuführender Anfragen |
| `-c Anzahl` | Anzahl nebenläufiger Anfragen (Standard: 1) |
| `-t Sekunden` | Maximaldauer der Anfragen; impliziert -n 50.000 |
| `-i` | `HEAD`- anstelle von `GET`-Anfragen verwenden |
| `-p Dateiname` | Datei mit Body für `POST`-Anfragen |
| `-T MIME-Type` | `Content-Type`-Header für `POST`-Body |
| `-k` | persistente Verbindung (Keepalive) |
| `-s` | SSL verwenden |
| `-A Benutzername:Passwort` | Anmeldedaten für Basic-Authentifizierung |
| `-C Name=Wert` | Cookie setzen; Option kann mehrfach verwendet werden |
| `-H String` | Anfrage-Header hinzufügen; kann mehrfach verwendet werden |
| `-X Adresse[:Port]` | Proxy verwenden |
| `-P Benutzername:Passwort` | Anmeldedaten zur Basic-Authentifizierung am Proxy |
| `-e Dateiname` | Datei mit kommaseparierten Werten schreiben, die z. B. in Tabellenkalkulationen importiert werden kann |
| `-g Dateiname` | Datei mit tabseparierten Werten (GNUPlot-Format) ausgeben |
| `-w` | Ergebnis als HTML-Tabelle ausgeben |
| `-x String` | Attribute für `<table>`-Tag |
| `-y String` | Attribute für `<tr>`-Tag |
| `-z String` | Attribute für `<td>`-Tag |
| `-v Zahl` | Ausführlichkeit; 4: mit Headern, 3: mit HTTP-Statuscode, 2: mit Warnungen |
| `-S` | Mittelwert und Standardabweichung weglassen |
| `-d` | Meldung `percentage served within XX [ms] table` unterdrücken |
| `-q` | Fortschrittsanzeige (`Completed 1000 Requests` usw.) deaktivieren |
| `-h` | Hilfe ausgeben und beenden |
| `-V` | Version anzeigen und beenden |

**Tabelle 12.1** Sämtliche Kommandozeilenoptionen des Apache-Benchmark-Tools

### 12.2.3 Performance-bezogene Core-Direktiven

In diesem Abschnitt werden Ihnen zwei wichtige Core-Konfigurationsdirektiven vorgestellt, die die Performance Ihres Apache-Webservers unter Umständen verbessern können. Es handelt sich um Optionen für Memory-Mapping – das Kopieren von Dateien in den Arbeitsspeicher – sowie `sendfile()`, einen Systemaufruf, der Dateien direkt über das Netzwerk sendet.

#### EnableMMAP

Die Verwendung von Memory-Mapping steuern

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | EnableMMAP On\|Off |
| Standardwert | On |

Normalerweise ist der Einsatz von Memory-Mapping grundsätzlich aktiviert. Das bedeutet, dass geöffnete Dateien im Arbeitsspeicher abgebildet werden, was grundsätzlich die Performance verbessert. Sie sollten lediglich in zwei Fällen über eine Deaktivierung von `EnableMMAP` nachdenken:

- Wenn Sie Verzeichnisse ausliefern, die über NFS von einem anderen Rechner im Netzwerk eingebunden wurden, kann es in diesen Verzeichnissen zu Speicherfehlern (segmentation faults) kommen. In einem solchen Fall sollten Sie die Direktive für das entsprechende Verzeichnis gezielt auf **`Off`** setzen:

  ```
  <Location /nfs-dateien>
  EnableMMAP Off
  </Location>
  ```

- Bei einigen Mehrprozessorsystemen kann Memory-Mapping die Performance vermindern, anstatt sie zu verbessern. Sie sollten dies in einer ruhigen Minute mithilfe von `ab` überprüfen und `EnableMMAP` gegebenenfalls auf `Off` setzen.

#### EnableSendfile

Die Verwendung des direkten Dateiversands durch den Kernel steuern

| Seit Version | 2.0.44 |
|---|---|
| Modul | core |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |

| Syntax | `EnableSendfile On\|Off` |
|---|---|
| Standardwert | `On`, seit 2.3.9 `off` |

Der Kernel mancher Betriebssysteme enthält den Systemaufruf `sendfile()`. Er ermöglicht den direkten Versand einer Datei über das Netzwerk, was erheblich schneller geht als das Öffnen, Lesen und Ausgeben. Genau wie Memory-Mapping ist auch die `sendfile()`-Unterstützung standardmäßig aktiviert. Auch `EnableSendFile` muss unter Umständen deaktiviert werden, wenn ein über das Netzwerk gemountetes Verzeichnis betroffen ist. Darüber hinaus gibt es einige Betriebssysteme mit fehlerbehafteten `sendfile()`-Implementierungen; wenn Sie ein solches System verwenden, müssen Sie `EnableSendfile` ganz abschalten.

### 12.2.4 mod_file_cache: häufig genutzte Dateien vorausladen

`mod_file_cache` ist nicht etwa eine Speicherlösung für `mod_cache` wie `mod_cache_disk` (siehe Kapitel 13, »Proxy- und Cache-Funktionen«), sondern ein unabhängiges Modul, das eine etwas andere Aufgabe erfüllt: Es bietet die Möglichkeit, häufig genutzte Dateien beim Start von Apache in den Arbeitsspeicher zu laden und ihre Dateideskriptoren geöffnet zu halten, um den Zugriff darauf zu beschleunigen.

Beachten Sie, dass der Server neu gestartet werden muss, wenn eine der betroffenen Dateien geändert wird. Sie sollten hier also in aller Regel keine Dateien aufnehmen, die durch dynamische Prozesse modifiziert werden, sondern möglichst nur große, statische Dokumente.

#### MMapFile

Liste von Dateien, die beim Start in den Arbeitsspeicher geladen werden sollen

| Modul | `mod_file_cache` |
|---|---|
| Kontext | Server |
| Syntax | `MMapFile Dateipfad [Dateipfad ...]` |
| Standardwert | nicht gesetzt |

Mit dieser Direktive werden Dateien mithilfe des Systemaufrufs `mmap()` im Arbeitsspeicher abgebildet. Apache braucht sie also nicht mehr von der Festplatte zu lesen, wenn sie angefordert werden. Hier ein Beispiel:

```
MMapFile /usr/local/apache2/htdocs/images/foto.jpg
```

Beachten Sie, dass es mit Bordmitteln von `mod_file_cache` keine Möglichkeit gibt, alle Inhalte oder gar Unterverzeichnisse eines Verzeichnisses in den Cache

zu kopieren; Platzhalter sind in den Dateinamen ebenfalls nicht zulässig. Wenn bei Ihnen `mod_perl` installiert ist, können Sie sich leicht mit folgenden Anweisungen behelfen, um alle Inhalte des Verzeichnisses */usr/local/apache2/htdocs/images* in den Speicher zu kopieren:

```
<Perl>
  opendir (DIR, "/usr/local/apache2/htdocs/images");
  while ($file = readdir (DIR)) {
    if ($file !~ /^\./) {
      push @MMapFile, $file;
    }
  }
  closedir (DIR);
</Perl>
```

Ansonsten bleibt Ihnen (zumindest unter UNIX) noch folgende Lösung, die sinngemäß in der Originaldokumentation zu `mod_file_cache` vorgeschlagen wird:

```
# find /usr/local/apache2/htdocs/media -type f -print \
| sed -e 's/.*/MMapFile &/' \
> /usr/local/apache2/conf/mmap.conf
```

Die neu erzeugte Datei *mmap.conf*, die alle Dateien aus dem URL-Pfad */media* enthält, können Sie mithilfe einer Include-Anweisung (siehe Kapitel 6, »Grundkonfiguration«) in Ihre *httpd.conf* einbinden.

#### CacheFile

Eine Liste von Dateihandles beim Apache-Start in den Cache kopieren

| | |
|---|---|
| **Modul** | `mod_file_cache` |
| **Kontext** | Server |
| **Syntax** | `CacheFile Dateipfad [Dateipfad ...]` |
| **Standardwert** | nicht gesetzt |

Diese Direktive öffnet beim Start von Apache alle angegebenen Dateien und hält ihre Dateideskriptoren im Cache bereit. Da also nicht der gesamte Inhalt der Dateien in den Arbeitsspeicher kopiert wird, bringt diese Direktive nicht ganz so viel Performance-Steigerung wie `MMapFile`, benötigt dafür aber weniger Speicher.

Hier sehen Sie ein einfaches Beispiel:

```
CacheFile /usr/local/apache2/htdocs/index.html \
/usr/local/apache2/htdocs/images/logo.gif
```

## 12.3 Load-Balancing

Manche Websites sind derart vielen Zugriffen ausgesetzt, dass einfaches Performance-Tuning nicht mehr weiterhilft. In einer solchen Situation wird es Zeit, sich Gedanken über *Load-Balancing* (Last- oder Ladeverteilung) zu machen. Bei diesem Verfahren werden die eingehenden Anfragen möglichst gerecht auf mehrere physikalische Server verteilt.

Load-Balancing-Verfahren bieten auf einen Blick die folgenden Vorteile:

- Die Belastung der einzelnen Server-Hosts wird vermindert.
- Die Verfügbarkeit der Website wird gesteigert – wenn einer der Server-Rechner ausfällt, können die restlichen den Betrieb der Site aufrechterhalten.
- Unter Umständen sind die Kosten für mehrere verhältnismäßig einfache Rechner geringer als diejenigen für eine einzelne Hochleistungsmaschine.

Bevor Sie über Load-Balancing nachdenken, sollten Sie sich aber auch über die Nachteile und Probleme im Klaren sein:

- Es genügt nicht, einfach ein paar Rechner aufzustellen, auf denen sich dieselbe Website befindet. Da Websites sich recht häufig ändern, müssen Sie auch sicherstellen, dass ihre Inhalte identisch bleiben, Sie brauchen also mit anderen Worten eine effiziente Lösung zur Replikation.
- Der Einsatz Session-basierter Server-Anwendungen wird bei der Verwendung von Load-Balancing zum Problem: Es muss sichergestellt werden, dass die Anfragen derselben Session nur von einem der Server-Hosts beantwortet werden.
- Unter Umständen liegt der Engpass Ihrer Site gar nicht an einer Überlastung des Server-Rechners, sondern an der überforderten Netzwerkverbindung. Bevor Sie also Zeit und Geld in eine Load-Balancing-Lösung verwenden, sollten Sie dies prüfen – wenn mehr physikalische Rechner am Netz hängen, könnte sich die Netzwerkbelastung sogar noch vergrößern.

Abgesehen von diesen Schwierigkeiten, besteht eine große Herausforderung beim Load-Balancing darin, eine wirklich gerechte Verteilung der Anfragen zu erzielen. Der einfachste Ansatz ist, jede eingehende Anfrage per Zufallsgenerator an einen anderen Server weiterzuleiten. Allerdings gewährleistet das nicht unbedingt eine wirklich faire Ressourcenbelastung. Deshalb verwenden professionelle Load-Balancing-Systeme ausgeklügeltere Verfahren, bei denen zur Auswahl der Server-Rechner permanent die Anzahl der übertragenen Bytes und die Geschwindigkeit der Netzwerkverbindungen überwacht werden.

In vielen Fällen besteht eine praktischere und einfachere Lösung darin, kein Load-Balancing-System einzurichten, sondern Ihre Site in unterschiedliche funktionale Bereiche zu unterteilen, die Sie auf unterschiedlichen physikalischen Servern unterbringen können. Sie brauchen sich bei dieser Methode keine Gedanken über die Replikation zu machen und benötigen auch keinen zusätzlichen Rechner, der als Load-Balancer arbeitet.

Die »Extremlösung« in puncto Load-Balancing ist die Nutzung der diversen *Cloud-Computing*-Angebote wie beispielsweise Amazon EC2. Hier können Sie stündlich flexibel Server hinzuschalten, wenn der Traffic steigt, und sie wieder entfernen, wenn er wieder sinkt. Die Angebote beschränken sich dabei nicht auf Webserver, sondern es können auch Datenbanken oder gar fertige Webanwendungen als Cloud-Dienste gemietet werden.

Hier wird zunächst eine »selbst gestrickte« Load-Balancing-Lösung auf der Basis von `mod_rewrite` gezeigt; anschließend erhalten Sie einige Hinweise zu Drittanbieter-Lösungen. Im nächsten Kapitel lernen Sie noch das Modul `mod_proxy_balancer` für Proxy-Load-Balancing kennen.

Die einfachste Lösung zur Verteilung von Anfragen auf mehrere Hosts ist Round-Robin-DNS. Die Einrichtung dieses Verfahrens mit dem Nameserver BIND wurde bereits in Kapitel 1, »TCP/IP-Netzwerke, Internet und WWW«, angesprochen. Allerdings sollten Sie wissen, dass diese Art der Lastverteilung durch den Nameserver nicht sonderlich gerecht ist, weil bei Weitem nicht jede Ermittlung der Adresse eines Webservers Teil einer HTTP-Anfrage ist.

### 12.3.1 Load-Balancing mit mod_rewrite

Die genaue Funktionsweise des Moduls `mod_rewrite` und seiner Direktiven wurde bereits in Kapitel 8, »Weiterleitungen und Indizes«, besprochen. Hier werden einige der URL-Manipulationsfähigkeiten des Moduls verwendet, um folgende Load-Balancing-Konfiguration einzurichten:

- Der Server, bei dem die Client-Anfragen zunächst eintreffen (*www.mynet.de*), fungiert ausschließlich als Load-Balancer: Er entscheidet mithilfe der zufälligen Auswahl eines Weiterleitungsziels, welcher der anderen Server-Hosts die Anfrage verarbeiten soll. Der Hostname des gewählten Rechners wird mithilfe einer temporären Weiterleitung an den Client versendet.
- Die eigentlichen Webserver (*www1.mynet.de* bis *www4.mynet.de*) enthalten alle dieselbe Website. Interne Hyperlinks in den HTML-Dokumenten enthalten ausschließlich relative URLs, sodass es keine Rolle spielt, von welchem dieser Hosts der Client bedient wird. Damit gewährleistet dieses Verfahren außer-

dem, dass Session-basierte Anwendungen kein Problem sind, weil die gesamte Kommunikation mit einem bestimmten Client komplett über einen der Server abgewickelt wird.

Auf dem Load-Balancer wird eine `RewriteMap` vom Typ `rnd` verwendet. Wie in Kapitel 8, »Weiterleitungen und Indizes«, ausführlich beschrieben wurde, liefert diese Art der Map bei jedem Zugriff ein zufällig gewähltes Element aus der Liste. Die Datei für dieses Beispiel befindet sich unter */usr/local/apache2/lb* und heißt *balance_map.txt*. Sie besitzt folgenden Inhalt:

```
server www1|www2|www3|www4
```

Nun können Sie folgende Rewrite-Anweisungen festlegen, die jeweils einen anderen Host auswählen:

```
RewriteEngine On
RewriteMap hosts rnd:/usr/local/apache2/lb/balance_map.txt
RewriteRule ^/(.*) http://${hosts:server}.mynet.de/$1 [R]
```

Angenommen, ein Client fordert die Adresse *www.mynet.de/bilder/test.gif* an. Der URL-Pfad ist somit */bilder/test.gif*. Die `RewriteRule`-Anweisung liest zunächst einen zufälligen Namen aus der Map – beispielsweise `www2`. Aus diesen Informationen wird also die neue URL *www2.mynet.de/bilder/test.gif* zusammengesetzt. Diese wird per Redirect – mit dem Flag `[R]` – an den Client übermittelt.

Möglicherweise möchten Sie nicht, dass der Client bei jedem Zugriff einen anderen Hostnamen zu Gesicht bekommt. In diesem Fall können Sie das Verfahren dahingehend erweitern, dass der Load-Balancer zugleich als Reverse-Proxy fungiert, das heißt die URLs der anderen Hosts in seinen eigenen URL-Namensraum übernimmt. Diese Erweiterung ist unvorstellbar einfach – da `mod_rewrite` die Reverse-Proxy-Unterstützung bereits mitbringt, müssen Sie die Weiterleitung lediglich durch die Flags `[P]` (Proxy) und `[L]` (»Last«, letzte `RewriteRule`) ersetzen:

```
RewriteEngine On
RewriteMap hosts rnd:/usr/local/apache2/lb/balance_map.txt
RewriteRule ^/(.*) http://${hosts:server}.mynet.de/$1 [P,L]
```

Natürlich muss der Hauptserver nun immer noch jede Anfrage selbst verarbeiten. Deshalb ist diese Lösung besonders dann geeignet, wenn auf den eigentlichen Webserver-Hosts komplexe Webanwendungen ausgeführt werden. Die einfache Weiterleitung von Proxy-Antworten verursacht nämlich erheblich weniger Arbeit.

Beachten Sie, dass dieses Basismodell erweitert werden muss, wenn Sie Unterstützung für Authentifizierung, SSL oder sonstige Zusatzfunktionen benötigen.

Außerdem müssen Sie einen Client dazu zwingen, dauerhaft einen bestimmten Server zu verwenden, wenn Session-basierte Webanwendungen zum Einsatz kommen. Die einfachste Lösung für dieses Problem besteht darin, dass ein Server beim Initialisieren der Session ein Cookie mit seiner Identität setzt. In der Client-Anfrage wird dieses Cookie dann wieder mitgeschickt und kann vor den anderen Rewrite-Rules ausgewertet werden, um den entsprechenden Client jeweils auf »seinen« Server zu leiten. Hier die entsprechend ergänzte Konfiguration, wobei das Cookie den Namen »XServer« trägt und sicherheitshalber geprüft wird, ob es tatsächlich einen gültigen Servernamen enthält:

```
RewriteEngine On
RewriteCond %{HTTP_COOKIE} XServer=^(www[1-4])$
RewriteRule ^/(.*) http://%1.mynet.de/$1 [P,L]
RewriteMap hosts rnd:/usr/local/apache2/lb/balance_map.txt
RewriteRule ^/(.*) http://${hosts:server}.mynet.de/$1 [P,L]
```

### 12.3.2 Open-Source-Lösungen für Load-Balancing

Neben Eigenbau-Load-Balancern gibt es übrigens auch zahlreiche Komplettlösungen – sowohl kommerzielle als auch Open-Source-Produkte. Da dies ein Buch über freie Software ist, werden hier lediglich Letztere berücksichtigt. Ihre nähere Beschreibung würde den Rahmen dieses Buches sprengen; deshalb werden sie hier nur mit den zugehörigen Webverweisen aufgelistet:

- **Pound**
  Dieses GPL-Programm hat zunächst einmal nichts mit Apache selbst zu tun – es arbeitet als Reverse-Proxy und/oder Load-Balancer für unterschiedliche Webserver, ist aber selbst kein HTTP-Server. Informationen sowie Download-Quellen finden Sie unter *www.apsis.ch/pound*.
- **Eddie**
  Auch dies ist ein freies Standalone-Programm, das mit Apache und anderen Webservern zusammenarbeiten kann. Die Website des Projekts ist *eddie.sourceforge.net*.
- **lbnamed**
  Dies ist ein freier DNS-basierter Load-Balancer. Im Grunde erledigt das Programm also dieselbe Aufgabe wie Round-Robin-DNS, sorgt aber durch Unterscheidung der Anfragetypen für eine gerechtere Verteilung. Näheres erfahren Sie unter *www.stanford.edu/~schemers/docs/lbnamed/lbnamed.html*.

## 12.4 Zusammenfassung

Der Betrieb besonders stark belasteter Sites sowie der Einsatz von Apache 2 für Webhosting-Dienstleistungen bringen völlig neue Herausforderungen mit sich. Dieses Kapitel kann und will nur einen Einstieg in die Server-Skalierung bieten. Vieles, was dieses Thema betrifft, hat nämlich nichts mehr mit der Apache-Administration zu tun, sondern betrifft Fragen der Hardwareauswahl oder der allgemeinen Systemoptimierung.

Für umfangreiche Websites ist der Einsatz virtueller Hosts interessant: Sie können einen einzelnen Webserver nach außen so erscheinen lassen, als seien es unterschiedliche Rechner. Apache 2 ermöglicht Ihnen die Einrichtung virtueller Hosts nach Hostname, IP-Adresse oder Portnummer; Letzteres ist vor allem auch für SSL beziehungsweise den Proxy-Betrieb geeignet. Speziell für Webhoster wurde schließlich das Modul `mod_vhost_alias` konzipiert, das die dynamische Einrichtung virtueller Hosts auf der Basis von Verzeichnisstrukturen ermöglicht.

Um die Performance des Servers zu speichern, gibt es verschiedene Möglichkeiten. Natürlich müssen Sie gegebenenfalls Ihre Hardware aufrüsten; vor allem RAM-Speicher kann niemals genug vorhanden sein. Auch Ihre Netzwerkanbindung muss möglicherweise verbessert werden. Daneben sollten Sie die Tipps zur Apache-Konfiguration – beispielsweise die Vermeidung von DNS-Lookups oder *.htaccess*-Dateien – beherzigen.

Wenn auch Performance-Tuning nicht mehr weiterhilft, sollten Sie sich gegebenenfalls über Load-Balancing Gedanken machen. In diesem Kapitel wurde eine einfache Lösung auf der Basis von `mod_rewrite` skizziert; darüber hinaus wurden einige Quellen für den freien Bezug externer Load-Balancing-Software empfohlen.

*»Wenn man ein Kenner ist,*
*darf man keine Sammlung anlegen.«*
*– Jean-Jacques Rousseau*

# 13 Proxy- und Cache-Funktionen

In diesem Kapitel wird erläutert, wie Sie Apache als Proxy-Server einrichten können. Damit können Sie ihn verwenden, um Netzwerkanfragen für Clients durchzuführen (Forward-Proxy) oder auch um Anfragen an andere Server unter lokalen URLs einzubinden (Reverse-Proxy). Anschließend werden die Apache-Module für das Caching, also für jede Form der Zwischenspeicherung von Daten zur Zugriffsbeschleunigung, vorgestellt. Web-Caches werden nämlich sehr häufig (wenn auch nicht ausschließlich) im Zusammenhang mit Proxys eingesetzt.

Beachten Sie im Zusammenhang mit Caching und Proxys auch die Ausführungen zu HTTP-Headern in Kapitel 2, »Funktionsweise von Webservern«, und Kapitel 7, »Header und MIME-Einstellungen«. Besonders wichtig ist hier der `Expires`-Header; das passende Apache-Modul `mod_expires` wurde ebenfalls in Kapitel 7, »Header und MIME-Einstellungen«, vorgestellt.

## 13.1 Apache als Proxy-Server

Ein Proxy-Server ist ein Stellvertreter, der im Namen von Clients mit anderen Servern interagiert. Der Vorgang ist völlig transparent: Die Clients können die gewünschten Operationen durchführen, als wären sie tatsächlich mit dem Ziel-Server verbunden; der Server wiederum kommuniziert nur mit dem Proxy und weiß nichts über den Client, für den der Proxy die Anfrage weiterleitet. Es gibt zwei grundlegende Arten von Proxys:

- Ein *Forward-Proxy* wird vom Client aktiv kontaktiert, um für diesen Anfragen an einen anderen Server zu senden. Der Proxy nimmt anschließend die Antwort des Servers entgegen und leitet sie an den Client weiter. Gegenüber dem Server agiert der Proxy dabei ebenfalls als Client, erzeugt also eigene Anfragen.

- Ein *Reverse-Proxy* ist im Gegensatz dazu ein Server, der hinter den Kulissen Anfragen an andere Server stellt und sie an den Client weiterreicht, ohne dass dieser von deren Existenz erfährt. Auf diese Weise können Sie die Inhalte anderer Websites unter URLs Ihres eigenen Servers anbieten. Der Zweck dieser Übung ist allerdings in der Regel kein böswilliger Content-Diebstahl. Vielmehr bildet dieses Konzept beispielsweise die Grundlage der Einbindung anderer Protokolle und einiger Load-Balancing-Lösungen.

Proxys sind übrigens keineswegs auf HTTP beschränkt. Es gibt Proxy-Server für fast jedes wichtige TCP/IP-Anwendungsprotokoll wie FTP, E-Mail (SMTP, POP3 und IMAP), NNTP oder IRC. Daneben stellen die sogenannten *SOCKS-Proxys* eine Weiterleitung beliebiger TCP-Verbindungen zur Verfügung. Apache kann zurzeit als Proxy für HTTP, FTP, AJP (Tomcat) und sichere Verbindungen (über `CONNECT`-Anfragen) eingesetzt werden; zudem steht Proxy-Load-Balancing zur Verfügung.

Proxy-Server haben im Wesentlichen drei Hauptaufgaben:

- Sie ermöglichen den (indirekten) Zugriff auf Netzwerkdienste, die der Client eigentlich nicht erreichen kann – beispielsweise, weil er sich hinter einer Firewall befindet oder weil der Internet-Provider formal nur Verbindungen für einen einzelnen Rechner zulässt. Zu diesem Zweck werden Proxys sehr häufig in Firmen eingesetzt.
- Da keine direkte Verbindung zwischen Client und Server besteht, kann der Proxy bei Bedarf beliebige Filter- und Kontrollfunktionen ausführen. Diese Nutzung von Proxys ist oft zweifelhaft; sie wird von einigen autoritären Regierungen als Mittel der Zensur und Kontrolle eingesetzt.[1] Auch manche Internet-Provider stehen in dem Ruf, durch Zwangs-Proxys den Zugriff auf das Internet einzuschränken.
- Ein Proxy-Server kann mit einem Cache kombiniert werden, um einmal abgerufene Inhalte für andere Clients vorzuhalten und auf diese Weise den Netzwerk-Traffic zu reduzieren.

Bei Apache 2 stellt das Modul `mod_proxy` die grundlegende Proxy-Funktionalität bereit; für die tatsächliche Vermittlung bestimmter Protokolle verwendet es jedoch Hilfsmodule. Mit Apache werden die folgenden Module geliefert:

1 Dazu muss man nicht unbedingt nach China oder in den Iran blicken – hierzulande hat beispielsweise der Regierungspräsident von Düsseldorf schon viele Jahre vor Ursula von der Leyen die Provider aufgefordert, gewisse Websites für ihre Kunden zu sperren. Zwar ging es dabei vor allem um rechtsradikale Sites, um die es gewiss nicht schade ist, aber dennoch bleibt im Hinblick auf Artikel 5 GG die Frage, ob ein nicht einmal demokratisch legitimierter Verwaltungsbeamter Zensur üben darf wie einst die »Fürsten von Gottes Gnaden«.

- `mod_proxy_http` – HTTP-Proxy
- `mod_proxy_ftp` – FTP-Proxy
- `mod_proxy_connect` – HTTPS-Tunnel-Proxy
- `mod_proxy_ajp` – Proxy für das Apache JServ Protocol 1.3; dient insbesondere der Tomcat-Anbindung
- `mod_proxy_balancer` – Proxy-Load-Balancing

In der neuen Version 2.3/2.4 kommen einige neue Proxy-Module hinzu:

- `mod_proxy_express` – dynamische Masseneinstellungen für Reverse-Proxys, analog zu `mod_vhost_alias`
- `mod_proxy_fcgi` – FastCGI-Proxy, der insbesondere mit dem Hilfsprogramm *fcgistarter* zusammenarbeitet (siehe auch Kapitel 14, »CGI«)
- `mod_proxy_fdpass` – Weitergabe des Client-Sockets über ein lokales UNIX-Domain-Socket (wird hier nicht näher behandelt, da es zurzeit nur für Entwickler interessant ist)
- `mod_proxy_html` – Rewriting von HTML-Hyperlinks, um sie in einer Proxy-Umgebung weiterverwenden zu können
- `mod_proxy_scgi` – SCGI-Gateway/Proxy

Das Konzept macht es leicht, Proxy-Lösungen für andere Protokolle zu entwickeln; in künftigen Apache-Versionen werden möglicherweise noch mehr von ihnen verfügbar sein.

Wenn Sie Proxy-Funktionalität für ein größeres Unternehmensnetzwerk benötigen, sollten Sie übrigens besser ein auf diese Aufgabe spezialisiertes Produkt wie `squid` verwenden; die Apache-Proxy-Module funktionieren zwar recht gut, sind aber nicht ganz so leistungsfähig.

### 13.1.1 Proxy-Grundkonfiguration

Bevor im Folgenden die einzelnen Proxy-Direktiven vorgestellt werden, finden Sie hier als ersten Anhaltspunkt zwei einfache, aber vollständige Proxy-Konfigurationen – die erste für einen Forward-Proxy, die zweite für einen Reverse-Proxy.

Beachten Sie, dass es sehr wichtig ist, einen Proxy-Server vor unberechtigten Zugriffen zu schützen. Schließlich könnte ansonsten jeder Ihren Proxy verwenden, um auf beliebige Internetinhalte zuzugreifen! In aller Regel werden Proxy-Zugriffe deshalb auf das lokale Netzwerk beschränkt; Authentifizierung ist ebenfalls möglich und kann in den meisten Browsern, die den entsprechenden Proxy ansprechen, automatisiert werden.

Zunächst einmal müssen Sie die Module für die konkrete Proxy-Funktionalität aktivieren: `mod_proxy` muss für jede Art von Proxy-Betrieb aktiv sein; daneben können Sie eines oder mehrere der soeben aufgeführten `mod_proxy_*`-Module aktivieren.

Beachten Sie, dass keines dieser Hilfsmodule eigene Direktiven definiert; die Konfiguration ist allein Sache von `mod_proxy`. In dieser Grundanleitung wird zunächst einmal nur die Einrichtung eines HTTP-Proxys vorgestellt; Einzelheiten für die anderen Protokolle entnehmen Sie bitte der anschließenden Direktivenreferenz.

Mit folgenden Anweisungen machen Sie Apache zum Forward-Proxy für HTTP-Anfragen, der aus Sicherheitsgründen auf das lokale Netzwerk beschränkt ist (die IP-Adressen müssen Sie gegebenenfalls an Ihr eigenes Netzwerk anpassen):

```
# Module laden - nur bei DSO-Betrieb:
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy.so

# Proxy einschalten
ProxyRequests On
ProxyVia On
# Konfiguration für Proxy-Anfragen
<Proxy *>
  Order deny,allow
  Deny from all
  Allow from 192.168.0 127.0.0.1
</Proxy>
```

Mit dieser Einstellung werden Proxy-Anfragen über den Standard-TCP-Port abgewickelt, an dem Apache lauscht. Damit wird allerdings jede Anfrage als Proxy-Anfrage betrachtet, sodass Sie den entsprechenden Apache-Server nicht mehr als gewöhnlichen Webserver benutzen können. Falls das auf einem bestimmten Host Ihr Wunsch war, ist es in Ordnung. In allen anderen Fällen sollten Sie einen zusätzlichen Port konfigurieren und einen virtuellen Host als HTTP-Proxy einrichten. Die folgende Variante verarbeitet Proxy-Anfragen nur auf Port 3128, der auch von reinen Proxy-Cache-Produkten wie `squid` verwendet wird:

```
# Module laden - nur bei DSO-Betrieb:
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy.so

# Zusätzlicher Port für Proxy-Anfragen
Listen 3128
```

```
# Virtueller Host für Proxy-Port 3128
<VirtualHost _default_:3128>
  # Proxy einschalten
  ProxyRequests On
  ProxyVia On

  # Konfiguration für Proxy-Anfragen
  <Proxy *>
    Order deny,allow
    Deny from all
    Allow from 192.168.0 127.0.0.1
  </Proxy>

</VirtualHost>
```

In den Browsern Ihres lokalen Netzwerks können Sie nun den neuen Proxy aktivieren; dies funktioniert je nach Browser unterschiedlich, und eine Beschreibung würde den Rahmen dieser Anleitung sprengen.

Das nächste kleine Beispiel zeigt die Reverse-Proxy-Funktionalität; es speist Anfragen für den URL-Pfad */sales* aus der URL *http://sales.mynet.de*. Die zweite Direktive korrigiert die URL in HTTP-Redirect-Headern – Umleitungsadressen würden sonst nicht in den eingebundenen Adressraum, sondern auf den Reverse-Proxy selbst verweisen.

```
ProxyPass /sales http://sales.mynet.de
ProxyPassReverse /sales http://sales.mynet.de
```

Für Reverse-Proxy-Funktionen brauchen Sie `ProxyRequests` nicht zu aktivieren, und Sie benötigen auch keinen virtuellen Host und keinen `<Proxy>`-Container. Schließlich soll das Einbinden der externen Ressource in der Regel für alle Anfragen gelten. Wenn Sie eine feinere Kontrolle über Reverse-Proxy-Operationen benötigen, können Sie die entsprechenden Funktionen von `mod_rewrite` verwenden (siehe Kapitel 8, »Weiterleitungen und Indizes«).

Das letzte kurze Beispiel demonstriert die Funktionsweise des (seit Apache 2.1-beta verfügbaren) Proxy-Load-Balancings; dazu muss zusätzlich das Modul `mod_proxy_balancer` eingebunden werden. Hier wird der jeweils identische Inhalt des Verzeichnisses */media* der drei Server *media1.mynet.de* bis *media3.mynet.de* unter dem URL-Pfad */media* eingebunden:

```
<Proxy balancer://mediahosts>
   BalancerMember http://media1.mynet.de
   BalancerMember http://media2.mynet.de
   BalancerMember http://media3.mynet.de
</Proxy>
ProxyPass /media balancer://mediahosts/media
```

Auf die AJP-Proxy-Funktionalität zur Einbindung der Servlet- und JSP-Engine Tomcat wird in Kapitel 15, »Technologien zur Webprogrammierung«, eingegangen – sowohl mit als auch ohne Load-Balancer.

### 13.1.2 Referenz der Proxy-Direktiven

Im Folgenden werden sämtliche Direktiven des Moduls `mod_proxy` vorgestellt. Bitte beachten Sie, dass einige neuere Funktionen, insbesondere im Zusammenhang mit dem Proxy-Load-Balancing, offiziell undokumentiert sind und zudem noch weiterentwickelt werden. Über gravierende Neuerungen werden Sie jeweils auf der Website zum Buch informiert.

#### <Proxy> ... </Proxy>

Einstellungen für Proxy-Ressourcen

| | |
|---|---|
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `<Proxy URL-Muster> ... </Proxy>` |
| **Standardwert** | nicht gesetzt |

Direktiven, die in diesem Container stehen, gelten ausschließlich für Proxy-Anfragen, die die angegebene URL anfordern. Sie können Shell-Wildcards verwenden, um mehrere URLs durch ein Muster zu beschreiben; für echte reguläre Ausdrücke ist dagegen die Variante `<ProxyMatch>` zuständig. In vielen Fällen wird einfach `*` verwendet, um Regeln für sämtliche Proxy-Anfragen zu bilden.

Das folgende Beispiel wurde bereits im Zusammenhang gezeigt; es erlaubt nur Rechnern aus dem lokalen Netzwerk Proxy-Anfragen:

```
<Proxy *>
  Order deny,allow
  Deny from all
  Allow from 192.168.0
</Proxy>
```

Das nächste Beispiel verhindert, dass irgendjemand über den Forward-Proxy auf die Site *http://www.trojan-horse.com* zugreift:

```
<Proxy http://www.trojan-horse.com/*>
  Order deny,allow
  Deny from all
</Proxy>
```

Ein spezieller Verwendungszweck dieses Containers ergibt sich im Zusammenhang mit dem Proxy-Load-Balancing. Hier wird als URL-Muster ein Balancer-Aliasname mit dem Pseudoprotokoll `balancer://` angegeben; innerhalb des Containers stehen `BalancerMember`-Direktiven zur Definition der verschiedenen Einzel-Server. Beispiel:

```
<Proxy balancer://music>
   BalancerMember music1.mynet.de
   BalancerMember music2.mynet.de
</Proxy>
```

Der Alias `balancer://music` kann anschließend verwendet werden, um Pfade der verschiedenen Server per Reverse-Proxy-Funktionalität einzubinden. Beispiel:

```
ProxyPass /mp3 balancer://music/mp3
```

#### <ProxyMatch> ... </ProxyMatch>

Einstellungen für Gruppen von Proxy-Ressourcen, auf die ein RegExp zutrifft

| | |
|---|---|
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `<ProxyMatch RegExp> ... </ProxyMatch>` |
| **Standardwert** | nicht gesetzt |

Dieser Container funktioniert genau wie `<Proxy>`, ermöglicht aber die Angabe eines Perl-kompatiblen regulären Ausdrucks als URL-Muster. Das folgende Beispiel sorgt für die Deflate-Komprimierung aller GIF-, JPEG- und PNG-Bilder, die der Apache-Proxy ausliefert:

```
<ProxyMatch \.(gif)|(jpg)|(png)$>
  SetOutputFilter DEFLATE
</ProxyMatch>
```

#### ProxyRequests

Proxy-Anfragen zulassen/verbieten

| | |
|---|---|
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyRequests On\|Off` |
| **Standardwert** | `Off` |

Wenn Sie diese Direktive auf `On` setzen, werden Proxy-Anfragen zugelassen. Weiter oben wurde bereits gezeigt, wie Sie dafür einen virtuellen Host mit einer

bestimmten Portnummer einrichten können. Es ist sehr wichtig, den Server gegen Zugriffe durch fremde Benutzer abzusichern, bevor Sie `ProxyRequests` einschalten.

### ProxyRemote

Bestimmte Anfragen an einen anderen Proxy weiterreichen

| | |
|---|---|
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyRemote Muster Remote-Server` |
| **Standardwert** | nicht gesetzt |

Diese Direktive leitet Proxy-Anfragen, die der lokale Proxy empfangen hat, an einen anderen Proxy-Server weiter. Der erste Parameter ist das URL-Muster der Proxy-Anfragen, der zweite die vollständige URL des Remote-Servers. Das folgende Beispiel leitet alle Proxy-Anfragen an *otherproxy.mynet.de* weiter:

```
ProxyRemote * http://otherproxy.mynet.de
```

### ProxyRemoteMatch

Gruppen von Anfragen weiterreichen, die einem RegExp entsprechen

| | |
|---|---|
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyRemoteMatch RegExp Remote-Server` |
| **Standardwert** | nicht gesetzt |

Die Direktive `ProxyRemoteMatch` funktioniert genau wie `ProxyRemote` mit dem Unterschied, dass das URL-Muster für die Proxy-Anfragen ein regulärer Ausdruck ist. Hier sehen Sie ein Beispiel, das alle Proxy-Anfragen für Sounddateien (*mp3*, *wav* und *aif*) an *soundproxy.mynet.de* weiterreicht:

```
ProxyRemoteMatch \.(mp3|wav|aif)$ \
http://soundproxy.mynet.de
```

### NoProxy

Adressen, für die kein Remote-Proxy verwendet wird

| | |
|---|---|
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `NoProxy Host [Host ...]` |
| **Standardwert** | nicht gesetzt |

Diese Direktive nimmt die angegebenen Hosts oder Adressblöcke von der Vermittlung der Anfrage an einen weiteren Proxy durch `ProxyRemote` aus. Für die Liste der Hostnamen beziehungsweise IP-Adressen sind folgende Schreibweisen erlaubt:

- **Hostname**
  ein vollständiger DNS-Hostname – Beispiel: `www.mynet.de`
- **Domain-Name**
  ein beliebig kurzes Teilstück, mit dem DNS-Namen enden können; stets mit führendem Punkt, um sie von Hostnamen zu unterscheiden – Beispiele: `.mynet.de` oder `.de`
- **IP-Adresse**
  Die IP-Adresse eines einzelnen Hosts – Beispiel: `196.23.17.52`
- **Teilnetz**
  ein Adressfragment, das ein bestimmtes IP-Adress-Teilnetz kennzeichnet; optional mit CIDR-Netzwerkbits oder Teilnetzmaske – Beispiele: `196.23`, `196.23.17.0/24` oder `196.23.17.0/255.255.255.0`

Nähere Erläuterungen zu diesen verschiedenen Adressangaben finden Sie in Kapitel 6, »Grundkonfiguration«, bei der Beschreibung der Direktive `Allow`.

Hier ein Komplettbeispiel, das Anfragen für einen bestimmten Host und eine Domain aus der `ProxyRemote`-Weiterleitung ausschließt:

```
# Alle Proxy-Anfragen an otherproxy.mynet.de weiterleiten
ProxyRemote * otherproxy.mynet.de
# Einzelne Adressen ausschließen
NoProxy .somenet.de www.othernet.de
```

#### ProxyBadHeader

Bestimmt den Umgang mit ungültigen Antwort-Headern.

| | |
|---|---|
| **Seit Version** | 2.0.44 |
| **Modul** | `mod_proxy` |
| **Kontext** | `Server, <VirtualHost>` |
| **Syntax** | `ProxyBadHeader IsError \| Ignore \| StartBody` |
| **Standardwert** | `IsError` |

Mithilfe dieser Direktive wird festgelegt, was geschehen soll, wenn die Antwort eines entfernten Servers einen Fehler enthält. Folgende Verhaltensweisen sind möglich:

- `IsError`: Apache beendet die Verarbeitung der Antwort und sendet dem Client eine Meldung mit dem Status `502 Bad Gateway`. Dies ist die Voreinstellung.
- `Ignore`: Der fehlerhafte Header wird ignoriert und einfach weitergeleitet.
- `StartBody`: Die Verarbeitung von Headern wird abgebrochen; der Rest der Antwort wird als Body betrachtet.

#### ProxyBlock

Ressourcen, für die Apache nicht als Proxy fungieren soll

| | |
|---|---|
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyBlock * \| Wort \| Host \| Domain [Wort\|Host\|Domain ...]` |
| **Standardwert** | nicht gesetzt |

`ProxyBlock` ermöglicht die Angabe beliebiger Adressbestandteile – wenn diese gefunden werden, wird die entsprechende Proxy-Anfrage abgelehnt. Die möglichen Elemente bedeuten Folgendes:

- **Wort**: ein einzelner Partikel – z.B. `mynet`
- **Host**: ein vollständiger Hostname – beispielsweise *www.mynet.de*
- **Domain**: ein beliebiges Teilstück eines Domain-Namens – etwa *mynet.de*

Das folgende Beispiel deaktiviert den Proxy-Zugriff auf alle Adressen, die den Bestandteil »pause« enthalten:

```
ProxyBlock pause
```

#### ProxyDomain

Domain-Name, der bei Proxy-Anfragen im Intranet automatisch angehängt wird

| | |
|---|---|
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyDomain Domain` |
| **Standardwert** | nicht gesetzt |

Diese Direktive ermöglicht die Definition eines Domain-Suffixes, das der Proxy automatisch an Hostnamen ohne Domain anhängt. Beispiel:

```
ProxyDomain .mynet.de
```

Falls der Proxy eine Anfrage mit der Adresse *www* erhält, wird diese automatisch zu *www.mynet.de* ergänzt. Dies ist nützlich, wenn die Rechner in Ihrem Intranet ohne Domain-Suffix angesprochen werden.

### ProxyErrorOverride

Bestimmt, ob der Proxy Fehlerseiten überschreiben soll.

| Seit Version | 2.0 |
|---|---|
| Modul | mod_proxy |
| Kontext | Server, <VirtualHost> |
| Syntax | ProxyErrorOverride On\|Off |
| Standardwert | Off |

Wenn Sie diese Direktive auf On setzen, ersetzt Ihr Proxy die Fehlermeldungsseiten entfernter Server durch seine eigenen. Da Sie Fehlermeldungen mithilfe von ErrorDocument anpassen können (siehe Kapitel 8, »Weiterleitungen und Indizes«), können Sie auf diese Weise auch im Proxy-Betrieb Ihre eigenen Fehlerseiten verwenden. Dies ist besonders bei einem Reverse-Proxy nützlich, weil damit neben den angezeigten URLs auch das Aussehen der Fehlermeldungen modifiziert werden kann.

### ProxyAddHeaders

Bestimmt, ob der Proxy seine Informationen zu X-Forwarded*-Headern hinzufügen soll.

| Seit Version | 2.3.10 |
|---|---|
| Modul | mod_proxy |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files> |
| Syntax | ProxyAddHeaders On\|Off |
| Standardwert | On |

Normalerweise fügt Apache in seiner Funktion als Proxy Informationen zu den HTTP-Erweiterungs-Headern X-Forwarded-For, X-Forwarded-Host und X-Forwarded-Server hinzu. Wenn Sie dies explizit verhindern möchten, können Sie es wie folgt deaktivieren:

```
ProxyAddHeaders off
```

### ProxySourceAddress

Lokale IP-Adresse für ausgehende Proxy-Verbindungen

| Seit Version | 2.3.9 |
|---|---|
| Modul | mod_proxy |
| Kontext | Server, <VirtualHost> |

| Syntax | ProxySourceAddress IP-Adresse |
|---|---|
| Standardwert | nicht gesetzt |

Mit dieser Direktive können Sie festlegen, dass ausgehende Proxy-Verbindungen eine bestimmte lokale IP-Adresse als Quelle verwenden sollen. Wenn Sie dies nicht tun, wird für gewöhnlich diejenige IP-Adresse verwendet, über die die ursprüngliche Client-Anfrage empfangen wurde. Beispiel:

```
ProxySourceAddress 192.168.0.8
```

#### ProxyFtpDirCharset

Legt den Zeichensatz für FTP-Verzeichnislistings fest.

| Seit Version | 2.2.7 |
|---|---|
| Modul | mod_proxy_ftp; vor 2.3.5 mod_proxy |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files> |
| Syntax | ProxyFtpDirCharset Zeichensatz |
| Standardwert | ISO-8859-1 |

Diese Direktive bestimmt den Zeichensatz, der verwendet wird, wenn mod_proxy_ftp ein FTP-Verzeichnislisting erzeugt. Der Standardwert ist ISO-8859-1; das folgende Beispiel legt UTF-8 fest:

```
ProxyFtpDirCharset UTF-8
```

#### ProxyFtpEscapeWildcards

Escaping von Platzhalterzeichen beim Senden an den FTP-Server aktivieren/deaktivieren

| Seit Version | 2.3-beta |
|---|---|
| Modul | mod_proxy_ftp |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files> |
| Syntax | ProxyFtpEscapeWildcards on\|off |
| Standardwert | on |

Standardmäßig werden die Platzhalterzeichen *, ?, [, { und ~ mit einem Backslash escaped, bevor sie an den FTP-Server gesendet werden. Manche FTP-Server können damit nicht umgehen; in diesem Fall können Sie das Escaping wie folgt ausschalten:

```
ProxyFtpEscapeWildcards off
```

### ProxyFtpListOnWildcard

Bei Platzhalterzeichen im Dateinamen eine Dateiliste oder eine Datei anfordern

| Seit Version | 2.3-beta |
|---|---|
| Modul | mod_proxy_ftp |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files> |
| Syntax | ProxyFtpListOnWildcard on\|off |
| Standardwert | on |

Wenn die angeforderte Datei eines der Platzhalterzeichen *, ?, [, { oder ~ enthält, fordert Apache normalerweise eine Dateiliste an, auf die das Dateimuster passt. Wenn Sie stattdessen Dateien anfordern möchten, die diese Platzhalter im Dateinamen enthalten, müssen Sie diese Direktive explizit auf off setzen:

```
ProxyFtpListOnWildcard off
```

### ProxyIOBufferSize

Größe des Proxy-Datenpuffers

| Modul | mod_proxy |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | ProxyIOBufferSize Bytes |
| Standardwert | 8192 |

Diese Direktive bestimmt die Größe des internen Puffers, den mod_proxy für gelesene, aber noch nicht wieder ausgegebene Daten verwendet. Die Voreinstellung 8.192 ist der Höchstwert; in aller Regel gibt es keinen Grund, einen kleineren Wert anzugeben.

### ProxyMaxForwards

Höchstzahl von Proxys, durch die eine Anfrage weitergeleitet wird

| Seit Version | 2.0 |
|---|---|
| Modul | mod_proxy |
| Kontext | Server, <VirtualHost> |
| Syntax | ProxyMaxForwards Anzahl |
| Standardwert | 10 |

Mithilfe von ProxyMaxForwards können Sie festlegen, wie viele aufeinanderfolgende Proxy-Server eine Anfrage passieren darf. Die Einstellung gilt nur für

Anfragen, die keinen `Max-Forwards`-Header enthalten, und kann unfreiwillige Proxy-Schleifen verhindern oder DoS-Attacken erschweren. Beispiel:

```
ProxyMaxForwards 8
```

### ProxyPass

Bildet entfernte URLs in den lokalen Verzeichnisbaum ab.

| | |
|---|---|
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | 2.0: `ProxyPass [Pfad] !\|URL`<br>ab 2.1: `ProxyPass [Pfad] !\|URL [Schlüssel=Wert [Schlüssel=Wert ...]] [nocanon] [interpolate]` |
| **Standardwert** | nicht gesetzt |

Dies ist die Hauptdirektive für die Einrichtung von Reverse-Proxy-Optionen. Sie funktioniert formal ähnlich wie `Alias` (siehe Kapitel 8, »Weiterleitungen und Indizes«), nur wird unter dem angegebenen URL-Pfad kein lokales Verzeichnis, sondern eine externe URL eingebunden. Hier sehen Sie ein Beispiel, das unter dem URL-Pfad */test* die Inhalte der Site *www.mytest.com* veröffentlicht:

```
ProxyPass /test http://www.mytest.com
```

Anstelle einer URL können Sie auch das spezielle Element `!` angeben – es bedeutet, dass unter dem angegebenen URL-Pfad keine Proxy-Operation stattfinden soll. Eine solche Einschränkung muss vor der eigentlichen `ProxyPass`-Einstellung erfolgen.

Das folgende Beispiel bindet unter dem URL-Pfad `/othernet` die Inhalte von *www.othernet.de* ein – außer in dem Verzeichnis `/othernet/local`, das vom lokalen Server verarbeitet wird:

```
ProxyPass /othernet/local !
ProxyPass /othernet http://www.othernet.de
```

Seit Apache 2.1-beta verfügt `mod_proxy` über Connection-Pooling-Funktionen. Diese und andere Optionen werden in dieser Direktive mithilfe von `Schlüssel=Wert`-Paaren eingestellt. Sie alle können auch für die im weiteren Verlauf beschriebene Direktive `BalancerMember` verwendet werden. Im Einzelnen sind folgende Schlüssel verfügbar:

- `min` – Mindestanzahl von Verbindungen zum Backend-Server. Der Standardwert ist 0.

- `max` – Absolute Höchstanzahl (hard limit) der Verbindungen; kein Standardwert.
- `smax` – Höchstzahl der Verbindungen, die geöffnet bleiben und keiner maximalen Lebensdauer unterliegen. Der Standardwert entspricht dem Wert von `max`.
- `ttl` – Lebensdauer von Verbindungen, die über die Anzahl `smax` hinausgehen, in Sekunden. Standardmäßig nicht gesetzt.
- `timeout` – Maximale Wartezeit auf das Senden von Daten. Standardmäßig wird der Wert der Direktive `ProxyTimeout` verwendet.
- `connectiontimeout` – Maximale Wartezeit beim Verbindungsaufbau. Der Standardwert entspricht der Einstellung für `timeout`.
- `acquire` – Maximale Wartezeit auf eine freie Verbindung im Connection-Pool; standardmäßig nicht gesetzt.
- `disablereuse` – Wenn Sie diesem Parameter den Wert `on` zuweisen, wird das Connection-Pooling verhindert, und jede Verbindung wird sofort nach Gebrauch wieder geschlossen. Das kann notwendig sein, wenn Firewalls oder Round-Robin-DNS des Ursprungsservers verhindern, dass persistente Reverse-Proxy-Verbindungen funktionieren. Der Standardwert `off` erlaubt dagegen das Connection-Pooling.
- `flushpackets` – Diese Option ist zurzeit nur für AJP implementiert. Sie bestimmt, ob Ausgabedaten des Servers gepuffert oder sofort gesendet werden. Der Standardwert `off` besagt, dass die Daten standardmäßig gepuffert und nur im Bedarfsfall gesendet werden. `on` sendet sie stets sofort. Der dritte mögliche Wert, `auto`, sendet die Daten, wenn für eine bestimmte Anzahl von Millisekunden (durch `flushwait` festgelegt) keine Eingabe empfangen wurde.
- `flushwait` – Die Anzahl von Millisekunden, die `flushpackets` im `auto`-Modus warten soll, bis die gepufferten Daten gesendet werden. Der Standardwert ist 10.
- `keepalive` – Wenn Sie den Wert `on` setzen, sendet Apache regelmäßig `KEEP_ALIVE`-Pakete. Wichtig bei Firewalls, die inaktive Verbindungen zu schnell unterbrechen. Der Standardwert ist `off`.
- `retry` – Wartezeit bis zu einem neuen Verbindungsversuch, falls eine Verbindung im Pool einen Fehler gemeldet hat. Der Standardwert beträgt 60 Sekunden.
- `lbset` – Mit dieser Option können Sie einem `BalancerMember`-Worker eine Ordnungszahl zuweisen. Der Load-Balancer probiert zunächst die Worker mit der niedrigsten Ordnungszahl durch, bevor er solche mit höheren Ordnungszahlen verwendet. Auf diese Weise können Sie Fallback-Load-Balancing-Cluster errichten, die aktiv werden, wenn das Hauptcluster ausfällt oder überlastet ist. Der Standardwert ist 0.

- `ping` – Nur für AJP. Das AJP13-Protokoll unterstützt eine spezielle Anfrage namens `CPING`, auf die eine `CPONG`-Antwort erwartet wird, vergleichbar mit dem gewöhnlichen ICMP-`ping`. Der Standardwert 0 deaktiviert dieses Feature; bei einem höheren Wert sendet der Proxy `CPING`-Anfragen und wartet die angegebene Anzahl von Sekunden auf die `CPONG`-Antwort; er verwendet den betreffenden Server nur dann für die eigentliche Anfrage, wenn das `CPONG` rechtzeitig eintrifft. Dieses Verfahren erhöht leicht den Netzwerk-Traffic, verhindert dafür aber das fruchtlose Versenden vollständiger Anfragen. Benötigt als Server eine Tomcat-Version ab 3.3.2, 4.1.28, 5.0.13, 5.5.0 oder 6.0.
- `loadfactor` – Bestimmt auf einer Prozentskala von 1 bis 100, wie stark der einzelne Rechner als Teil eines Load-Balancing-Clusters belastet wird. Standardwert: 1.
- `route` – Route eines Einzelrechners beim Load-Balancing, normalerweise nicht gesetzt. Dieser Wert wird an eine eventuelle Session-ID angehängt.
- `redirect` – Umleitungsroute. Anfragen ohne Session-ID werden an den Rechner im Load-Balancing-Cluster weitergeleitet, dessen `route`-Wert dieser Angabe entspricht.
- `status` – Status, den ein Worker beim Start besitzen soll. Standardmäßig besitzt ein Worker zunächst keinen besonderen Status. Die möglichen Werte sind die folgenden einzelnen Buchstaben:
  - `D` – deaktiviert (disabled)
  - `S` – angehalten (stopped)
  - `I` – Fehler ignorieren (ignore errors)
  - `H` – Bereitschaft (hot-standby)
  - `E` – Fehlerzustand (error)

  Sie können mehrere Statusbuchstaben verwenden und ihnen optional ein `+` voranstellen, um sie hinzuzufügen (Standard), oder ein `-`, um sie zu entfernen.

Bei der Weiterleitung an eine virtuelle `balancer://`-URL können Sie noch die folgenden speziellen Parameter angeben:

- `lbmethod` – Legt die Load-Balancing-Methode fest: `byrequests` (Standard) verteilt die Last nach der Anzahl der Anfragen, während `bytraffic` die tatsächlich übertragenen Bytes in Betracht zieht.

  In Apache 2.3 wurden diese Funktionalitäten ausgelagert, die keine eigenen Direktiven bereitstellen, aber geladen sein müssen, um die jeweilige `lbmethod`-Einstellung verwenden zu können. Gleichzeitig wurden neue Methoden hinzugefügt:

  - `mod_lbmethod_bybusiness` (Methodenname `bybusiness`) merkt sich, wie viele wartende Anfragen jedem Worker gerade zugewiesen sind, und weist eine neue Anfrage demjenigen mit der geringsten Anzahl zu. Dies ist besonders sinnvoll, wenn die Worker eigene Warteschlangen für Anfragen besitzen.
  - `mod_lbmethod_byrequests` stellt die bereits genannte Methode `bybusiness` zur Verfügung.
  - `mod_lbmethod_bytraffic` stellt die ebenfalls schon genannte Methode `bytraffic` bereit.
  - `mod_lbmethod_heartbeat` (Methodenname `heartbeat`) orientiert sich an »Lebenszeichen«-Anfragen (Heartbeats) der Worker, die innerhalb von Apache beispielsweise durch das ebenfalls neue Modul `mod_heartbeat` erzeugt werden können.
- `stickysession` – Hier wird ein Wert wie `JSESSIONID` oder `PHPSESSIONID` eingetragen, damit eine bestimmte Session durchgehend von ein und demselben Einzelrechner des Clusters verarbeitet wird.
- `nofailover` – Wenn Sie diese Option auf `on` setzen, wird eine Session abgebrochen, sobald der entsprechende Einzel-Server nicht mehr verfügbar ist oder einen Fehler meldet. Auch dies ist wichtig, wenn Ihre Anwendungen keinen Wechsel einer Session auf einen anderen Server vertragen.
- `timeout` – Wartezeit, bis ein Rechner des Balancing-Clusters verfügbar ist. Der Standardwert ist 0, das heißt, es wird normalerweise nicht gewartet.
- `maxattempts` – Die maximale Anzahl möglicher Fehlversuche, bevor der Verbindungsversuch aufgegeben wird. Der Standardwert ist 1.
- `scolonpathdelim` – Wird diese Option auf `on` gesetzt, dann gilt das Semikolon als Trennzeichen in Session-IDs. Dies emuliert das Verhalten von `mod_jk` (siehe Kapitel 15, »Technologien zur Webprogrammierung«) im Umgang mit zusammengesetzten Session-Parametern wie `JSESSIONID=3a4c9fa0d;foo=bar`.

Hinter den `Schlüssel=Wert`-Paaren können Sie optional noch die folgenden beiden Schlüsselwörter ohne Werte setzen:

- `nocanon` – Normalerweise wird die URL der Anfrage kanonisiert, bevor die Proxy-Anfrage weitergeleitet wird; konkret wird der kanonische Server-Name verwendet (siehe `UseCanonicalName` in Kapitel 6, »Grundkonfiguration«) und `Path-Info` (Pfadangaben hinter der eigentlichen URL) entfernt. Diese Option sorgt dafür, dass die ungeparste URL übergeben wird; in einigen Fällen kann dies notwendig sein.
- `interpolate` – Sorgt in Zusammenarbeit mit der weiter unten beschriebenen Direktive `ProxyPassInterpolateEnv` dafür, dass Umgebungsvariablen in der Form `${Variablenname}` in der URL für die Proxy-Anfrage interpoliert werden.

Beachten Sie, dass in diesem Kontext nicht alle CGI-Umgebungsvariablen (siehe Kapitel 14, »CGI«) zur Verfügung stehen, sodass Sie einige Ersetzungen mittels `mod_rewrite` vornehmen müssen. Diese Option ist seit Apache 2.2.9 verfügbar.

### ProxySet

Optionen für Balancer und `BalancerMember` setzen

| | |
|---|---|
| **Seit Version** | 2.2 |
| **Modul** | `mod_proxy` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `ProxySet URL Schlüssel=Wert`<br>`[Schlüssel=Wert ...]` |
| **Standardwert** | nicht gesetzt |

`ProxySet` ist eine alternative Methode, sämtliche unter `ProxyPass` vorgestellten `Schlüssel=Wert`-Optionen für Load-Balancer und `BalancerMember` zu setzen. Wenn Sie die Direktive innerhalb eines `<Proxy>`-Containers einsetzen, können Sie den ersten Parameter (URL) weglassen, da die URL bereits für den Container gesetzt wird. Beispiele:

```
ProxySet balancer://cluster lbmethod=bytraffic

<Proxy balancer://cluster>
  BalancerMember www1.mynet.de
  BalancerMember www2.mynet.de
  BalancerMember www3.mynet.de
  ProxySet keepalive=On
</Proxy>
ProxyPass /balanced balancer://cluster
```

### ProxyPassMatch

Bildet entfernte URLs mithilfe regulärer Ausdrücke in den lokalen Verzeichnisbaum ab.

| | |
|---|---|
| **Seit Version** | 2.2.5 |
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `ProxyPassMatch [Regexp] !\|URL`<br>`[Schlüssel=Wert [Schlüssel=Wert ...]]` |
| **Standardwert** | nicht gesetzt |

Diese Direktive funktioniert genau wie `ProxyPass`, mit der Ausnahme, dass anstelle des zu ersetzenden Pfads ein regulärer Ausdruck angegeben werden kann, was die Konfiguration flexibler macht. In der Proxy-Anfrage-URL können geklammerte Teilausdrücke aus dem regulären Ausdruck dann wie üblich als `$1`, `$2` usw. eingesetzt werden.

Das folgende Beispiel bindet gängige Mediendateien unter dem URL-Pfad */media* per Reverse-Proxy-Anfrage an *media.mynet.de* ein:

```
ProxyPassMatch ^/media/(.*)\.(gif|jpg|png|swf|avi) \
http://media.mynet.de/$1.$2
```

#### ProxyPassInterpolateEnv

Interpoliert Umgebungsvariablen in Proxy-Anfrage-URLs.

| | |
|---|---|
| **Seit Version** | 2.2.9 |
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `ProxyPassInterpolateEnv On\|Off` |
| **Standardwert** | `Off` |

Wenn Sie dieser Direktive den Wert `On` zuweisen, wird die Interpolation von Umgebungsvariablen in `ProxyPass`, `ProxyPassReverse`, `ProxyPassReverseCookieDomain` und `ProxyPassReverseCookiePath` grundsätzlich aktiviert. Zusätzlich muss in der entsprechenden Direktive selbst noch das Schlüsselwort `interpolate` gesetzt werden (siehe Beschreibung der jeweiligen Direktive). Danach können Sie einige Umgebungsvariablen in der Form `${Variablenname}` in den Proxy-Anfrage-URLs verwenden. Da dieses Feature zulasten der Server-Performance geht, sollten Sie diese Direktive nur dann aktivieren, wenn es wirklich notwendig ist.

#### ProxyPassReverse

URL im HTTP-Antwort-Header eines Reverse-Proxys anpassen

| | |
|---|---|
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `ProxyPassReverse [Pfad] URL [interpolate]` |
| **Standardwert** | nicht gesetzt |

Diese Direktive arbeitet mit `ProxyPass` zusammen; häufig werden sogar für beide Direktiven dieselben Argumente angegeben. `ProxyPassReverse` sorgt dafür, dass die URL in bestimmten HTTP-Antwort-Headern von Weiterleitungen (3xx-Status-

codes) angepasst wird. Namentlich geht es um die Header `Location`, `Content-Location` und `URI`. Betrachten Sie dazu folgendes Beispiel:

```
ProxyPass /othernet http://www.othernet.de
ProxyPassReverse /othernet http://www.othernet.de
```

Angenommen, ein Client fordert die URL *www.mynet.de/othernet/test* an. Dies entspricht aufgrund der `ProxyPass`-Direktive in Wirklichkeit der Adresse *www.othernet.de/test*. Wenn */test* ein Verzeichnis auf dem entsprechenden Server ist, können Sie davon ausgehen, dass die Antwort eine Weiterleitung auf *www.othernet.de/test/* ist (das Problem mit dem abschließenden Slash wird in Kapitel 6, »Grundkonfiguration«, genau erläutert). Die relevanten Elemente der HTTP-Antwort des Servers *www.othernet.de* sehen mit anderen Worten folgendermaßen aus:

```
HTTP/1.1 301 Moved Permanently
Location: http://www.othernet.de/test/
```

An dieser Stelle sorgt `ProxyPassReverse` dafür, dass die URL wieder in das URL-Schema des Proxys umgewandelt wird. Der Location-Header, den der Client zu Gesicht bekommt, sieht also folgendermaßen aus:

```
Location: http://www.mynet.de/othernet/test/
```

Der Hostname in den geänderten Headern wird übrigens nicht unbedingt aus der ursprünglichen Anfrage übernommen, sondern gemäß der `UseCanonicalName`-Einstellung gesetzt.

Auch für diese Direktive können Sie die bei `ProxyPass` erläuterte Option `interpolate` setzen, um Umgebungsvariablen mittels `${Variablenname}` zu interpolieren; dazu muss wiederum die Direktive `ProxyPassInterpolateEnv` eingeschaltet sein.

#### ProxyPassReverseCookieDomain

Domain in Cookies anpassen, die über den Reverse-Proxy übertragen werden

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyPassReverseCookieDomain`<br>`interne-Domain externe-Domain [interpolate]` |
| **Standardwert** | nicht gesetzt |

Auf ähnliche Weise und aus demselben Grund wie `ProxyPassReverse` ändert diese Direktive den Domain-Bestandteil in `SetCookie`-Headern. Beispiel:

```
ProxyPassReverseCookieDomain mynet.de othernet.de
```

Näheres über Cookies und ihre Bestandteile erfahren Sie im PHP-Abschnitt von Kapitel 15, »Technologien zur Webprogrammierung«.

#### ProxyPassReverseCookiePath

Pfad in Cookies anpassen, die über den Reverse-Proxy übertragen werden

| Seit Version | 2.1-beta |
|---|---|
| Modul | mod_proxy |
| Kontext | Server, <VirtualHost> |
| Syntax | ProxyPassReverseCookiePath<br>interner-Pfad externer-Pfad [interpolate] |
| Standardwert | nicht gesetzt |

Diese Direktive ändert entsprechend den Pfadbestandteil eines SetCookie-Headers.

#### BalancerMember

Definition eines Einzel-Servers in einem Load-Balancing-Cluster

| Seit Version | 2.1-beta |
|---|---|
| Modul | mod_proxy |
| Kontext | Server, <VirtualHost> |
| Syntax | BalancerMember Host<br>[Schlüssel=Wert [Schlüssel=Wert ...]] |
| Standardwert | nicht gesetzt |

Mithilfe dieser Direktive legen Sie die einzelnen Hosts eines Load-Balancing-Clusters fest. Dies muss innerhalb eines <Proxy>-Containers mit einer balancer://-Alias-URL geschehen. Diese URL können Sie anschließend für ProxyPass-Direktiven verwenden, um die Last auf die verschiedenen BalancerMember zu verteilen.

Das folgende Beispiel definiert drei HTTP-Server als Load-Balancing-Cluster und bindet dieses Cluster anschließend unter dem URL-Pfad */balanced* ein:

```
<Proxy balancer://cluster>
  BalancerMember www1.mynet.de
  BalancerMember www2.mynet.de
  BalancerMember www3.mynet.de
</Proxy>
ProxyPass /balanced balancer://cluster
```

Optional können Sie für jeden `BalancerMember` eine Reihe von Optionen im Format `Schlüssel=Wert` angeben. Die möglichen Schlüssel wurden bereits bei der `Direktive ProxyPass` erläutert.

#### BalancerGrowth

Nachträgliches Wachstumspotenzial für die Anzahl der BalancerMember

| | |
|---|---|
| **Seit Version** | 2.3.13 |
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `BalancerGrowth Anzahl` |
| **Standardwert** | `5` |

Diese Direktive legt fest, wie viele BalancerMember für einen virtuellen Host über die vorkonfigurierte Anzahl hinaus hinzugefügt werden können.

#### ProxyPreserveHost

`Host`-Header der Anfrage an Reverse-Proxy durchreichen

| | |
|---|---|
| **Seit Version** | 2.0.31 |
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyPreserveHost On\|Off` |
| **Standardwert** | `Off` |

Wenn Sie diese Direktive aktivieren, wird bei Reverse-Proxy-Anfragen der `Host`-Header aus der Client-Anfrage an den entfernten Server durchgereicht. In den meisten Fällen ist dies nicht sinnvoll; nur im Zusammenhang mit speziellen Konfigurationen für namensbasierte virtuelle Hosts kann es manchmal erforderlich sein, den ursprünglichen `Host`-Header weiterzuleiten.

#### ProxyReceiveBufferSize

Netzwerkpuffergröße für Proxy-Verbindungen

| | |
|---|---|
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyReceiveBufferSize Bytes` |
| **Standardwert** | `0` |

Diese Direktive ermöglicht die Festlegung des TCP-Puffers für Proxy-Verbindungen. Normalerweise bestimmt der Vorgabewert 0, dass die Vorgabe des jeweiligen Betriebssystems verwendet werden soll. Wenn Sie den Puffer manuell vergrößern möchten, um den Datendurchsatz zu steigern, muss der Wert größer als 512 sein. Beispiel:

```
ProxyReceiveBufferSize 1024
```

#### ProxyTimeout

Timeout für Proxy-Anfragen

| | |
|---|---|
| **Seit Version** | 2.0.31 |
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyTimeout Sekunden` |
| **Standardwert** | `300` |

Mit dieser Direktive können Sie festlegen, wie lange der Proxy auf die Antwort eines entfernten Servers warten soll, bevor er die Anfrage als gescheitert betrachtet. Die Voreinstellung von 300 Sekunden (5 Minuten) ist bei Weitem ausreichend.

#### ProxyVia

Konfiguration des `Via`-Headers für Proxy-Antworten

| | |
|---|---|
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyVia On\|Off\|Full\|Block` |
| **Standardwert** | `Off` |

Diese Direktive bestimmt, wie der Proxy mit dem `Via`-Header in HTTP-Anfragen und -Antworten umgehen soll. Dieser Header wird in Kapitel 2, »Funktionsweise von Webservern«, näher erläutert. Er sollte eine Liste sämtlicher Proxys enthalten, über die die Anfrage weitergeleitet wurde. Sie können einen der vier folgenden Werte angeben:

- `On`. Der Proxy fügt den eigenen Hostnamen zum `Via`-Header jeder Anfrage und Antwort hinzu.
- `Off`. Eventuell vorhandene `Via`-Header werden unbehelligt weitergeleitet; Apache fügt keinen eigenen Eintrag hinzu.

- `Full`. Apache erzeugt `Via`-Header wie bei der Einstellung `On` und fügt zusätzlich seine Versionsinformationen als `Via`-Kommentar hinzu.
- `Block`. Der Proxy fügt keinen `Via`-Eintrag hinzu und entfernt darüber hinaus alle bereits vorhandenen `Via`-Header.

#### AllowCONNECT

Legt fest, über welche TCP-Ports `CONNECT`-Anfragen erlaubt sind.

| | |
|---|---|
| **Modul** | `mod_proxy_connect`; vor 2.3.5 `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `AllowCONNECT Port [Port ...]`<br>seit 2.3.7:<br>`AllowCONNECT Port[-Port] [Port[-Port] ...]` |
| **Standardwert** | `443 563` |

Diese Direktive bestimmt das Verhalten der SSL-Tunnelung über `mod_proxy_connect`: Sie bestimmt, für welche Server-TCP-Ports `CONNECT`-Anfragen weitergeleitet werden. Standardmäßig sind nur die Ports 443 (HTTPS) und 563 (`snews`) aktiviert. Wenn Sie in Ausnahmefällen noch andere Ports benötigen – etwa weil Sie Ihren SSL-Server nicht für Port 443 konfiguriert haben –, können Sie sie durch Leerzeichen getrennt angeben. Beispiel:

```
AllowCONNECT 443 563 4433
```

Seit Version 2.3.7 können anstelle einzelner Ports auch Port-Bereiche in der Schreibweise `Startportnummer-Endportnummer` angegeben werden. Beispiel:

```
AllowCONNECT 443 563 4433-4439
```

#### ProxyStatus

Bestimmt, ob `mod_status` Informationen über den Load-Balancer-Status veröffentlichen soll.

| | |
|---|---|
| **Seit Version** | 2.2 |
| **Modul** | `mod_proxy` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyStatus On\|Off\|Full` |
| **Standardwert** | `Off` |

`ProxyStatus` bestimmt, ob das in Kapitel 8, »Weiterleitungen und Indizes«, vorgestellte Modul `mod_status` Informationen über den Status des Proxy-Load-Balan-

cings veröffentlicht. Wenn Sie die Direktive auf `On` oder `Full` (Synonym) setzen, werden diese Informationen ausgegeben, bei der Standardeinstellung `Off` nicht.

### 13.1.3 mod_proxy_html

Das Modul `mod_proxy_html` wurde längere Zeit als Drittanbieter-Modul gepflegt und erst im »Endspurt« der 2.4-Entwicklung in die offizielle Distribution aufgenommen. Es dient als Ausgabefilter (siehe auch Kapitel 16, »SSI und Filter«) und ermöglicht das Rewriting von HTML-Hyperlinks für eine Proxy-Umgebung.

Genauer bietet das Modul die Möglichkeit, die Links in HTML-Dokumenten innerhalb einer durch eine Reverse-Proxy-Konfiguration angebundenen Website durch ihre externen Gegenstücke zu ersetzen, genau wie es `ProxyPassReverse` mit den aufgerufenen URLs macht.

Angenommen, ein Reverse-Proxy ist wie folgt eingebunden:

```
ProxyPass /external http://internal.mynet.de
```

Eine Webseite innerhalb der Website unter *http://internal.mynet.de* enthält beispielsweise folgenden Link:

```
<a href="http://internal.mynet.de/info.html">Info</a>
```

Mit `mod_proxy_html` können Sie dafür sorgen, dass der Link automatisch wie folgt geändert wird, um über den Proxy zugänglich zu sein:

```
<a href="http://proxy.mynet.de/external/info.html">Info</a>
```

#### ProxyHTMLEnable

Aktiviert die Funktionalität von `mod_proxy_html`.

| | |
|---|---|
| **Seit Version** | 2.4, für ältere 2.x-Versionen als externes Zusatzmodul |
| **Modul** | `mod_proxy_html` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `ProxyHTMLEnable on\|off` |
| **Standardwert** | `off` |

Mit dieser Direktive können Sie die Funktionalität von `mod_proxy_html` für den jeweiligen Kontext aktivieren:

```
ProxyHTMLEnable on
```

Das Rewriting erfolgt nur für HTML-Dokumente (MIME-Types `text/html` oder `application/xhtml+xml`). Wenn Sie auch andere Dateitypen entsprechend bear-

beiten möchten, müssen Sie die Umgebungsvariable `PROXY_HTML_FORCE` setzen. Beachten Sie jedoch, dass dies manche Dateien – insbesondere Binärformate, die an manchen Stellen zufällig HTML enthalten – unbrauchbar machen kann.

Wenn das Modul `mod_xml2enc` aktiv ist (siehe Kapitel 16, »SSI und Filter«), dann wird automatisch die durch dieses Modul gebotene Internationalisierung aktiviert.

#### ProxyHTMLURLMap

Regeln für das HTML-URL-Rewriting definieren

| | |
|---|---|
| **Seit Version** | 2.4, für ältere 2.x-Versionen als externes Zusatzmodul |
| **Modul** | `mod_proxy_html` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `ProxyHTMLURLMap Suchmuster Ersatzmuster [Flags] [Bedingung]` |
| **Standardwert** | nicht gesetzt |

Nachdem Sie die Funktionalität mit `ProxyHTMLEnable` grundsätzlich aktiviert haben, definiert `ProxyHTMLURLMap` die eigentlichen Ersetzungsregeln. Das folgende einfache Beispiel führt die in der Einführung beschriebene Ersetzung von Links auf *http://internal.mynet.de* nach */external* auf dem aktuellen Host (*proxy.mynet.de*) durch:

```
ProxyHTMLEnable on
ProxyHTMLURLMap http://internal.mynet.de /external
```

Die ersten beiden Parameter, Suchmuster und Ersatzmuster, bilden im Prinzip die Umkehrung der Parameter von Direktiven wie `ProxyPass`.

Als dritten Parameter können Sie beliebig viele der folgenden Flags angeben (dabei wird zwischen Groß- und Kleinschreibung unterschieden):

- `h` – HTML-Links ignorieren (unverändert durchreichen)
- `e` – Skript-Events ignorieren (unverändert durchreichen)
- `c` – eingebettete Skript- und Style-Abschnitte unverändert durchreichen
- `L` – letzte Regel; wenn die aktuelle Ersetzungsregel zutrifft, werden keine weiteren mehr ausgeführt (bei HTML-Links geschieht dies automatisch)
- `l` – das Gegenteil von `L`: auch auf HTML-Links gegebenenfalls weitere Regeln anwenden
- `R` – Als Suchmuster wird ein regulärer Ausdruck verwendet, und im Ersatzmuster können die Treffervariablen `$1` bis `$9` für `()`-geklammerte Teilausdrücke aus dem Suchmuster verwendet werden.

- `x` – POSIX-erweiterte reguläre Ausdrücke verwenden (nur zusammen mit `R` verwendbar)
- `i` – beim Suchen keine Groß- und Kleinschreibung beachten (nur zusammen mit `R` verwendbar)
- `n` – den Speicher für reguläre Ausdrücke zur Beschleunigung deaktivieren (nur zusammen mit `R` verwendbar)
- `s` – zeilenbasierte Suche nach regulären Ausdrücken (nur zusammen mit `R` verwendbar)
- `^` – Treffer nur am String-Anfang (nur für Strings, nicht für reguläre Ausdrücke; spielt bei HTML-Links keine Rolle)
- `$` – Treffer nur am String-Ende (nur für Strings, nicht für reguläre Ausdrücke; spielt bei HTML-Links keine Rolle)
- `V` – Umgebungsvariablen im Ersatzmuster interpretieren (dazu muss ProxyHTMLInterp aktiv sein). Das Format im Ersatzmuster ist `${Variablenname|Standardwert}`. Wenn die entsprechende Variable definiert ist, wird an der jeweiligen Stelle ihr Wert eingefügt, andernfalls der angegebene Standardwert (Letzterer ist jedoch optional).
- `v` – Umgebungsvariablen im Suchmuster interpretieren (Details siehe bei `V`)

Als letztes Argument kann schließlich optional ein `ap_expr`-Ausdruck angegeben werden. Die Regel wird dann nur beachtet, wenn dieser Ausdruck zutrifft. Alle Einzelheiten über Ausdrücke stehen in der Beschreibung der Direktive `<If>` in Kapitel 6, »Grundkonfiguration«.

#### ProxyHTMLBufSize

Anzahl Bytes, um die der interne Puffer für den HTML-Proxy bei Bedarf erhöht wird

| | |
|---|---|
| **Seit Version** | 2.4, für ältere 2.x-Versionen als externes Zusatzmodul |
| **Modul** | `mod_proxy_html` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `ProxyHTMLBufSize Bytes` |
| **Standardwert** | `8192` |

Während des Link-Rewritings muss der Parser Nicht-HTML-Inhalte wie CSS und JavaScript in einem Puffer zwischenspeichern. Wenn dieser Puffer zu klein wird, dann wird er jeweils in Blöcken der hier angegebenen Byte-Anzahl vergrößert. Wenn Sie wissen, dass viele der zu verarbeitenden Dokumente Stylesheet- oder Skript-Blöcke enthalten, die größer als 8 KByte sind, dann sollten Sie den Wert

erhöhen, weil die dynamische Vergrößerung des Puffers während der Verarbeitung zu Lasten der Performance geht.

Das folgende Beispiel stellt die doppelte Größe ein:

```
ProxyHTMLBufSize 16384
```

#### ProxyHTMLCharsetOut

Zeichensatz für die Ausgabe von `mod_proxy_html` festlegen

| Seit Version | 2.4, für ältere 2.x-Versionen als externes Zusatzmodul |
|---|---|
| Modul | `mod_proxy_html` |
| Kontext | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| Syntax | `ProxyHTMLCharset Zeichensatz\|*` |
| Standardwert | `UTF-8` |

Mit dieser Direktive können Sie einen Ausgabezeichensatz für das Ergebnis der HTML-Proxy-Verarbeitung angeben. Der spezielle Wert * verwendet den Zeichensatz des Ursprungsdokuments. Wenn es möglich ist, sollten Sie aus Performance-Gründen bei UTF-8 bleiben, weil libxml2 auch intern diesen Zeichensatz verwendet und alles andere zu Lasten der Performance geht.

Hier dennoch ein Beispiel mit iso-latin-1:

```
ProxyHTMLCharsetOut ISO-8859-1
```

#### ProxyHTMLDocType

Eine HTML- oder XHTML-Doctype-Deklaration hinzufügen

| Seit Version | 2.4, für ältere 2.x-Versionen als externes Zusatzmodul |
|---|---|
| Modul | `mod_proxy_html` |
| Kontext | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| Syntax | `ProxyHTMLDocType HTML\|XHTML [Legacy]`<br>oder<br>`ProxyHTMLDocType fpi [SGML\|XML]` |
| Standardwert | nicht festgelegt |

Diese Direktive setzt den `<!DOCTYPE>`-Header für die ausgegebenen HTML-Dokumente. In der ersten Form haben Sie die Wahl zwischen `HTML` (ergibt HTML 4.01) und `XHTML` (XHTML 1.0). Wenn Sie das Schlüsselwort `Legacy` hinzufügen, wird der Kompatibilitätsmodus Transitional aktiviert, der weniger genau ist als der Standardmodus Strict. Wenn die Direktive in diesem Format verwendet wird,

versucht mod_proxy_html zusätzlich, das HTML des verarbeiteten Dokuments im Sinne des gewählten Standards zu säubern; wo dies nicht gelingt, werden entsprechende Fehler mit dem LogLevel debug ins ErrorLog geschrieben.

Die andere Möglichkeit ist die Angabe eines eigenen Formal Public Identifiers (FPI) wie beispielsweise "-//W3C//DTD HTML 4.01//EN". In diesem Fall gibt der zusätzliche Parameter an, ob SGML/HTML- oder XML/XHTML-Syntax verwendet werden soll.

#### ProxyHTMLEvents

Liste von Attributen, die als Skript-Events interpretiert werden

| | |
|---|---|
| **Seit Version** | 2.4, für ältere 2.x-Versionen als externes Zusatzmodul |
| **Modul** | mod_proxy_html |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files> |
| **Syntax** | ProxyHTMLEvents Attribut [Attribut ...] |
| **Standardwert** | nicht gesetzt |

Eine Reihe von HTML-Attributen sind JavaScript-Event-Handler, beispielweise onclick oder onfocus. Diese können JavaScript-Anweisungen enthalten, in denen sich wiederum URLs befinden können, die umgeschrieben werden sollten. mod_proxy_html sucht in allen hier angegebenen Attributen danach und ersetzt sie. In der Standardkonfiguration werden alle HTML-4.0.1- und XHTML-1.0-Event-Attribute angegeben:

```
ProxyHTMLEvents onclick ondblclick onmousedown onmouseup \
onmouseover onmousemove onmouseout onkeypress \
onkeydown onkeyup onfocus onblur onload \
onunload onsubmit onreset onselect onchange
```

Es ist eine gute Idee, auch das Attribut style hinzuzufügen. Dies löst zwar kein JavaScript aus, sondern enthält Inline-CSS, aber dies kann URL-Angaben wie background-image:url(...) enthalten, die gegebenenfalls geändert werden müssen.

#### ProxyHTMLExtended

Bestimmt, ob die Ersetzungsregeln auch auf Inline-Skripte, -Stylesheets und Skript-Ereignisse angewendet werden.

| | |
|---|---|
| **Seit Version** | 2.4, für ältere 2.x-Versionen als externes Zusatzmodul |
| **Modul** | mod_proxy_html |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files> |

| Syntax | ProxyHTMLExtended on\|off |
|---|---|
| Standardwert | off |

Wenn diese Direktive den Wert off hat, werden die Ersetzungsregeln nur auf HTML-Links angewendet, bei on auch auf Inline-Skripte, -Stylesheets und Skript-Ereignisse (sofern die ProxyHTMLURLMap-Flags entsprechend gesetzt sind). Wenn Sie diese Funktionalität einschalten, hat der Parser mehr zu tun, und die Performance kann leiden. Sie sollten on also nur verwenden, wenn es wirklich notwendig ist.

#### ProxyHTMLFixups

Beseitigung einfacher HTML-Fehler

| Seit Version | 2.4, für ältere 2.x-Versionen als externes Zusatzmodul |
|---|---|
| Modul | mod_proxy_html |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files> |
| Syntax | ProxyHTMLFixups [lowercase] [dospath] [reset] |
| Standardwert | nicht gesetzt |

Wenn diese Direktive gesetzt wird, kann mod_proxy_html einfache, verbreitete HTML-Link-Fehler beseitigen:

- lowercase – URLs werden in Kleinschreibung umgewandelt.
- dospath – Backslashes (\) werden in Forward-Slashes (/) umgewandelt.
- reset – Einstellungen der nächsthöheren Konfigurationsebene werden zurückgesetzt.

Sie können beliebig viele dieser Werte angeben.

Bitte beachten Sie, dass die vermeintlichen Fehler in manchen HTML-Dokumenten Absicht sind; verwenden Sie diese Direktive daher nur, wenn Sie genau wissen, dass der Backend-Server die entsprechenden Fehler produziert.

#### ProxyHTMLInterp

Ersetzungsregeln für jede Anfrage neu interpretieren

| Seit Version | 2.4, für ältere 2.x-Versionen als externes Zusatzmodul |
|---|---|
| Modul | mod_proxy_html |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files> |
| Syntax | ProxyHTMLInterp on\|off |
| Standardwert | off |

Standardmäßig werden die Ersetzungsregeln beim Start von Apache vorkompiliert und dann einfach angewendet. Wenn Sie diese Direktive auf `on` setzen, werden sie dagegen bei jeder einzelnen Anfrage neu interpretiert. Dies beeinträchtigt die Performance, schafft aber beispielsweise die Möglichkeit, Umgebungsvariablen zu verwenden.

#### ProxyHTMLLinks

Legt die HTML-Elemente und -Attribute fest, die als Links gelten.

| | |
|---|---|
| **Seit Version** | 2.4, für ältere 2.x-Versionen als externes Zusatzmodul |
| **Modul** | `mod_proxy_html` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `ProxyHTMLLinks Element Attribut [Attribut ...]` |
| **Standardwert** | nicht gesetzt |

Diese Direktive legt die HTML-Elemente (Tags) und -Attribute fest, die als zu verarbeitende Links betrachtet werden. Sie benötigen für jedes Element eine Instanz dieser Direktive, können aber beliebig viele Attribute angeben. Die Standardkonfiguration definiert alle HTML-4.0.1- und XHTML-1.0-Links:

```
ProxyHTMLLinks a  href
ProxyHTMLLinks area  href
ProxyHTMLLinks link  href
ProxyHTMLLinks img  src longdesc usemap
ProxyHTMLLinks object  classid codebase data usemap
ProxyHTMLLinks q  cite
ProxyHTMLLinks blockquote cite
ProxyHTMLLinks ins  cite
ProxyHTMLLinks del  cite
ProxyHTMLLinks form  action
ProxyHTMLLinks input  src usemap
ProxyHTMLLinks head  profile
ProxyHTMLLinks base  href
ProxyHTMLLinks script  src for
```

#### ProxyHTMLStripComments

HTML-Kommentare entfernen

| | |
|---|---|
| **Seit Version** | 2.4, für ältere 2.x-Versionen als externes Zusatzmodul |
| **Modul** | `mod_proxy_html` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `ProxyHTMLStripComments on\|off` |
| **Standardwert** | `off` |

Wenn Sie diese Direktive auf `on` setzen, werden HTML-Kommentare aus dem verarbeiteten Dokument entfernt. Beachten Sie, dass dabei auch Skripte und Stylesheets verschwinden, die in Kommentare eingebettet sind. Außerdem würden auch die Anweisungen für kommentarbasierte Präprozessoren wie SSI und ESI gelöscht; daher müssen diese in der Filterkette vor `mod_proxy_html` verarbeitet werden (siehe Kapitel 16, »SSI und Filter«).

### 13.1.4 Massen-Reverse-Proxys mit mod_proxy_express

Bis einschließlich Apache 2.2 bestand die einfachste Möglichkeit zur dynamischen Konfiguration einer größeren Gruppe von Reverse-Proxys darin, Rewriting mit dem Flag `[P]` zu verwenden (siehe den `mod_rewrite`-Teil in Kapitel 8, »Weiterleitungen und Indizes«). Für Apache 2.4 wurde zu diesem Zweck das Modul `mod_proxy_express` hinzugefügt, erstmals in Version 2.3.13. Das Modul ist in seiner Bedeutung mit derjenigen von `mod_vhost_alias` für Massen-VHosts zu vergleichen.

#### ProxyExpressEnable

ProxyExpress-Funktionalität aktivieren

| | |
|---|---|
| **Seit Version** | 2.3.13 |
| **Modul** | `mod_proxy_express` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyExpressEnable On\|Off` |
| **Standardwert** | `Off` |

Wenn Sie die Funktionalität von ProxyExpress benötigen, müssen Sie sie mithilfe dieser Direktive explizit einschalten:

```
ProxyExpressEnable on
```

#### ProxyExpressDBMFile

DBM-Datei mit der konkreten ProxyExpress-Konfiguration

| | |
|---|---|
| **Seit Version** | 2.3.13 |
| **Modul** | `mod_proxy_express` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyExpressDBMFile Dateipfad` |
| **Standardwert** | nicht gesetzt |

Die konkrete ProxyExpress-Konfiguration befindet sich in einer DBM-Datei (siehe unter anderem `RewriteMap` in Kapitel 8, »Weiterleitungen und Indizes«,

sowie die DBM-basierte Autorisierung in Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«). Am einfachsten ist es, eine einfache Textdatei zu erstellen und diese dann mit dem ebenfalls in Kapitel 8, »Weiterleitungen und Indizes«, beschriebenen Hilfsprogramm `httxt2dbm` in das DBM-Format umzuwandeln.

Hier ein Beispiel für eine Textdatei vor der Umwandlung:

```
www1.mynet.de http://192.168.0.11:8000
www2.mynet.de http://192.168.0.12:8000
www3.mynet.de http://192.168.0.13:8000
```

Geben Sie folgenden Befehl ein, um die Textdatei *express_map.txt* in die DBM-Datei *expressmap* umzuwandeln:

```
# httxt2dbm -i express_map.txt -o expressmap
```

Anschließend können Sie die erstellte DBM-Datei mit folgender Konfiguration zum Einsatz bringen:

```
ProxyExpressEnable on
ProxyExpressDBMFile Pfad/zu/expressmap
```

#### ProxyExpressDBMType

Typ der verwendeten DBM-Datei

| | |
|---|---|
| **Seit Version** | 2.3.13 |
| **Modul** | `mod_proxy_express` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ProxyExpressDBMType Default \| SDBM \| GDBM \| NDBM \| DB` |
| **Standardwert** | `Default` |

Mit dieser Direktive können Sie gegebenenfalls den konkreten Typ der verwendeten DBM-Datei angeben, falls dies notwendig ist. Die möglichen Werte werden in Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«, bei der Direktive `AuthDBMType` näher erläutert.

## 13.2 Cache-Funktionen

Ein *Cache* ist ein Zwischenspeicher, der Zugriffe beschleunigt. Das Caching-Prinzip wird heutzutage für unzählige Aspekte von Hard- und Software genutzt: Der mehrstufige CPU-Cache umgeht den Arbeitsspeicher, der Dateisystem-Cache ver-

mindert Festplattenzugriffe usw. Apache 2 enthält einige Module, die dieses Verfahren auf verschiedene Weise verwirklichen. Besonders praktisch ist das Caching von Inhalten im Zusammenhang mit einem Proxy – einmal angeforderte Seiten bleiben für weitere Anfragen gespeichert und reduzieren auf diese Weise den Netzwerk-Traffic. Aber auch auf die allgemeine Performance von Apache kann sich Caching positiv auswirken.

### 13.2.1 Cache-Grundkonfiguration

Im Einzelnen existieren folgende Cache-Module:

- `mod_cache`. Dieses Modul stellt die grundlegende Caching-Funktionalität bereit. Es benötigt stets die Anwesenheit eines zusätzlichen Moduls, das sich um die eigentliche Speicherung der Daten kümmert. Apache wird mit zwei passenden Speicherlösungen geliefert:
  - `mod_cache_disk` (bis Version 2.2 `mod_disk_cache`) realisiert den Cache auf der Festplatte. Dadurch kann er bei Bedarf ziemlich groß werden, ist aber langsamer als ein Cache im Arbeitsspeicher.
  - `mod_mem_cache` (in Apache 2.3 abgeschafft) richtet den Cache im Arbeitsspeicher ein. Diese Lösung sorgt für einen schnelleren, aber tendenziell natürlich kleineren Cache.
- `mod_file_cache`. Trotz seines Namens hat dieses Modul nichts mit den anderen Cache-Modulen zu tun. Es ermöglicht Ihnen, häufig angeforderte Dokumente beim Start von Apache in den Arbeitsspeicher zu laden. Aus diesem Grund wurde es bereits in Kapitel 12, »Skalierung und Performance-Tuning«, behandelt.

Hier sehen Sie zunächst ein komplettes Beispiel, das einen Speicher-Cache für sämtliche Dokumente unter dem URL-Pfad */media* einrichtet:

```
# Module laden, falls DSO-basiert
LoadModule cache_module modules/mod_cache.so
LoadModule mem_cache_module modules/mod_mem_cache.so

# Cache-Konfiguration
CacheEnable mem /media
MCacheSize 8192
MCacheMinObjectSize 1024
MCacheMaxObjectCount 200
```

Die Größe des Arbeitsspeicherbereichs für den Cache beträgt 8 Megabyte (der Wert von `MCacheSize` wird in Kilobyte angegeben). Der Cache speichert nur

Objekte ab einer Größe von 1 Kilobyte (bei `MCacheMinObjectSize` sind es Bytes). Es werden maximal 200 Dateien zwischengespeichert.

In Apache 2.3/2.4 funktioniert dieses Beispiel nicht mehr, da die speicherbasierte Cache-Engine abgeschafft wurde. Durch Prefetching, Memory-Mapping und andere moderne Techniken ist der plattenbasierte Cache ohnehin schnell genug und belastet zudem nicht permanent den Arbeitsspeicher. Wenn Sie wirklich einen speicherbasierten Cache benötigen, können Sie Drittanbieter-Lösungen wie memcached (*http://memcached.org*) verwenden und mit Apache als Reverse-Proxy darauf zugreifen.

Das folgende Beispiel zeigt, wie sich die Konfiguration des weiter oben vorgestellten Forward-Proxys um einen Festplatten-Cache ergänzen lässt:

```
# Module laden, falls DSO-basiert
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule cache_module modules/mod_cache.so
LoadModule cache_disk_module modules/mod_cache_disk.so
# Die entsprechende Zeile für Apache bis 2.2:
#LoadModule disk_cache_module modules/mod_disk_cache.so

# Zusätzlicher Port für Proxy-Anfragen
Listen 3128

# Virtueller Host für Proxy-Port 3128 mit Cache
<VirtualHost _default_:3128>
  # Proxy einschalten
  ProxyRequests On
  ProxyVia On
  # Cache (128 MB auf der Festplatte)
  CacheEnable disk/
  CacheRoot /usr/local/apache2/cache
  CacheSize 131072

  # Konfiguration für Proxy-Anfragen
  <Proxy *>
    Order deny,allow
    Deny from all
    Allow from 192.168.0 127.0.0.1
  </Proxy>

</VirtualHost>
```

### 13.2.2 Cache-Direktiven

Im Folgenden werden die Direktiven der Module `mod_cache`, `mod_disk_cache` und `mod_mem_cache` beschrieben. Für Apache 2.4 wurden einige neue Direktiven hinzugefügt, die noch nicht ausreichend dokumentiert sind und fortlaufenden Änderungen unterliegen. Bitte beachten Sie entsprechende Hinweise auf der Website zum Buch.

#### CacheEnable

Caching bestimmter URLs über eine bestimmte Speicherlösung abwickeln

| | |
|---|---|
| **Modul** | `mod_cache` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `CacheEnable Cache-Typ URL-String` |
| **Standardwert** | nicht gesetzt |

Mit dieser Direktive wird das Caching lokaler URLs unter dem angegebenen URL-Pfad aktiviert. Als Cache-Typ können Sie einen der folgenden drei Werte angeben:

- `mem` – Cache im Arbeitsspeicher (benötigt zusätzlich `mod_mem_cache`; nur bis Apache 2.2)
- `fd` – Cache im Arbeitsspeicher nach Dateideskriptoren (erfordert ebenfalls `mod_mem_cache`; nur bis 2.2)
- `disk` – Cache auf der Festplatte (`mod_cache_disk` beziehungsweise `mod_disk_cache` muss aktiviert sein)

Beispiele:

```
CacheEnable mem /pages
CacheEnable disk /media/images
```

Es können beispielsweise auch alle Dateien eines bestimmten Protokolls gecached werden, die per Reverse-Proxy eingebunden werden. Das folgende Beispiel aktiviert den Cache für FTP-Daten:

```
CacheEnable disk ftp://
```

#### CacheDisable

Caching einzelner URLs deaktivieren

| | |
|---|---|
| **Modul** | `mod_cache` |
| **Kontext** | Server, `<VirtualHost>` |

| Syntax | CacheDisable URL-String |
|---|---|
| Standardwert | nicht gesetzt |

Mit dieser Direktive können Sie URL-Pfade vom Caching ausschließen. Das ist wichtig, wenn Sie das Caching eines bestimmten URL-Pfads mittels CacheEnable eingeschaltet haben, die Inhalte eines Unterverzeichnisses davon aber nicht im Cache haben möchten. Hier ein Komplettbeispiel:

```
CacheEnable mem /media
CacheDisable mem /media/movies
```

#### CacheDefaultExpire

Dauer, wie lange ein Dokument ohne explizite Gültigkeitsdauer standardmäßig im Cache bleibt

| Modul | mod_cache |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | CacheDefaultExpire Sekunden |
| Standardwert | 3600 |

Diese Direktive legt fest, wie lange ein Dokument im Cache gültig bleiben soll, das keinen Expires-Header enthielt. Der Wert wird in Sekunden angegeben, der Standardwert beträgt also eine Stunde. Das folgende Beispiel legt zwölf Stunden fest:

```
CacheDefaultExpire 43200
```

#### CacheHeader

Fügt den Header X-Cache zur Antwort hinzu.

| Seit Version | 2.3.9 |
|---|---|
| Modul | mod_cache |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (All) |
| Syntax | CacheHeader on\|off |
| Standardwert | off |

Wenn Sie diese Direktive auf on setzen, wird der Header X-Cache zu den Server-Antworten im entsprechenden Kontext hinzugefügt. Dieser enthält Informationen darüber, ob die Antwort aus dem Cache bedient wurde (HIT), nicht dort vorhanden war (MISS) oder veraltet ist und daher neu erstellt werden muss (REVALIDATE). Clients können diese Angaben beispielsweise nutzen, um die Effizienz des Cachings zu messen.

### CacheDetailHeader

Fügt den Header `X-Cache-Detail` zur Antwort hinzu.

| Seit Version | 2.3.9 |
|---|---|
| Modul | mod_cache |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (All) |
| Syntax | CacheDetailHeader on\|off |
| Standardwert | off |

Der Header `X-Cache-Detail` – den Sie zur Antwort hinzufügen können, indem Sie diese Direktive auf `on` setzen –, enthält detailliertere Informationen als `X-Cache`, die eine noch genauere Analyse ermöglichen. Insbesondere der Grund für eine Caching-Entscheidung wird angegeben – Beispiel:

```
X-Cache-Detail: "conditional cache hit: entity refreshed" from
localhost
```

### CacheIgnoreCacheControl

Ignoriert `CacheControl: no-cache` in Client-Anfragen.

| Modul | mod_cache |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | CacheIgnoreCacheControl On\|Off |
| Standardwert | Off |

Normalerweise richtet sich Apache beim Caching nach den Bestimmungen des `CacheControl`-Headers (siehe Kapitel 2, »Funktionsweise von Webservern«). Vor allem werden Dokumente für gewöhnlich nicht im Cache abgelegt, wenn `Cache-Control: no-cache` gesetzt ist. Wenn Sie diese Direktive einschalten, ignoriert Apache diesen Header und legt das entsprechende Dokument trotzdem im Cache ab.

### CacheIgnoreNoLastMod

Ignoriert, dass eine Anfrage keinen `LastModified`-Header besitzt.

| Modul | mod_cache |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | CacheIgnoreNoLastMod On\|Off |
| Standardwert | Off |

Apache speichert in der Standardkonfiguration nur solche Dokumente im Cache, die einen `Last-Modified`-Header besitzen. Dieser Wert kann nämlich zur Berech-

nung des Verfallsdatums herangezogen werden, wenn kein `Expires`-Header vorhanden ist. Wenn Sie möchten, dass Dokumente ohne diesen Header in den Cache aufgenommen werden, können Sie die Direktive `CacheIgnoreNoLastMod` auf `On` setzen. Für das Verfallsdatum wird in diesem Fall der Wert von `CacheDefaultExpire` zur aktuellen Zeit addiert.

#### CacheIgnoreQueryString

Query-Strings von URLs beim Caching ignorieren

| | |
|---|---|
| **Seit Version** | 2.2.6 |
| **Modul** | `mod_cache` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `CacheIgnoreQueryString On\|Off` |
| **Standardwert** | `Off` |

URLs mit Query-String werden gemäß RFC 2616 für jeden einzelnen Query-String-Wert separat im Cache abgelegt, und dies geschieht auch nur dann, wenn die Antworten einen `Expires`-Header aufweisen. Wenn Sie diese Direktive auf `On` setzen, werden URLs mit Query-String dagegen unabhängig vom Vorhandensein des Verfallsdatums gespeichert, und der Query-String selbst wird ignoriert. Beachten Sie, dass auf diese Weise falsche Inhalte angezeigt werden können, wenn der Inhalt dynamisch generierter Websites vom Wert des Query-Strings abhängt.

#### CacheKeyBaseURL

Basis-URL für die Erstellung des Cache-Schlüssels für Reverse-Proxys

| | |
|---|---|
| **Seit Version** | 2.3.9 |
| **Modul** | `mod_cache` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `CacheKeyBaseURL` |
| **Standardwert** | `http://example.com` |

Wenn diese Direktive nicht gesetzt ist, werden Schema, Hostname und Port des aktuellen virtuellen Hosts verwendet, um den Schlüssel zu berechnen, unter dem Daten für eine Reverse-Proxy-Konfiguration im Cache abgelegt werden. Wenn Sie ein Cluster mehrerer Reverse-Proxys verwenden und deren Daten unter demselben Schlüssel cachen möchten, können Sie diese Direktive setzen, um dies zu bewerkstelligen. Beispiel:

```
CacheKeyBaseURL http://mynet.de
```

### CacheLastModifiedFactor

Faktor zur Errechnung eines Verfallsdatums anhand von `LastModified`

| Modul | mod_cache |
|---|---|
| Kontext | Server, `<VirtualHost>` |
| Syntax | `CacheLastModifiedFactor Float` |
| Standardwert | `0.1` |

Diese Direktive legt einen Faktor fest, der zur Berechnung des Verfallsdatums aus dem `LastModified`-Header verwendet wird, wenn kein `Expires`-Header vorliegt. Dazu wird die Zeit, die seit dem Zeitpunkt der letzten Änderung vergangen ist, mit diesem Faktor multipliziert. Dieser Wert wird dann zur aktuellen Zeit addiert, um das Verfallsdatum zu erzeugen. Tabelle 13.1 zeigt ein Berechnungsbeispiel.

| Aktuelle Zeit | 19.09.2011, 19:00 Uhr |
|---|---|
| Letzte Änderung | 18.09.2011, 23:00 Uhr |
| CacheLastModifiedFactor | 0.1 |
| Zeit seit der letzten Änderung | 20 Stunden |
| Berechnete Gültigkeitsdauer | 20 Stunden * 0.1 = 2 Stunden |
| Berechnetes Verfallsdatum | 19.08.2011, 21:00 Uhr |

**Tabelle 13.1** Berechnung des Verfallsdatums mithilfe von »CacheLastModifiedFactor«

### CacheMaxExpire

Bestimmt, wie lange ein Dokument ohne Prüfung des Originals höchstens im Cache verbleibt.

| Modul | mod_cache |
|---|---|
| Kontext | Server, `<VirtualHost>` |
| Syntax | `CacheMaxExpire Sekunden` |
| Standardwert | `86400` |

Mit dieser Direktive können Sie festlegen, wie lange ein Dokument aus dem Cache ausgeliefert werden soll, ohne die Original-URL zu überprüfen. Nach der angegebenen Sekundenzahl wird der Inhalt der URL auch dann überprüft, wenn sein Verfallsdatum noch nicht erreicht ist. Der Standardwert ist ein Tag; das folgende Beispiel legt zwei Stunden fest:

```
CacheMaxExpire: 7200
```

### CacheMinExpire

Bestimmt, wie lange ein Dokument mindestens im Cache verbleibt.

| Seit Version | 2.3 |
|---|---|
| Modul | mod_cache |
| Kontext | Server, <VirtualHost> |
| Syntax | CacheMaxExpire Sekunden |
| Standardwert | 0 |

Diese Direktive gibt an, wie lange ein Dokument mindestens im Cache verbleiben soll, ohne die Original-URL zu überprüfen. Dies gilt nur für Dokumente, die keinen gültigen Expires-Header besitzen.

### CacheQuickHandler

Dokumente in der Quick-Handler-Phase aus dem Cache laden

| Seit Version | 2.3 |
|---|---|
| Modul | mod_cache |
| Kontext | Server, <VirtualHost> |
| Syntax | CacheQuickHandler on\|off |
| Standardwert | on |

Standardmäßig werden Antworten während der Quick-Handler-Phase aus dem Cache bedient, also ziemlich früh während der Anfrage-Verarbeitung. Dies macht den Cache besonders effizient.

Wenn Sie CacheQuickHandler auf off setzen, können Sie dieses Verhalten auch ausschalten und den Cache so als normalen Handler ausführen. Dies erlaubt die Weiterverarbeitung der Antworten aus dem Cache durch Autorisierung, Ausgabefilter und so weiter. Der Cache steht dann auch als Filter-Provider mit dem Namen CACHE zur Verfügung, und er kann an einer selbst gewählten Stelle in die Filter-Kette eingefügt werden (siehe Kapitel 16, »SSI und Filter«).

### CacheStaleOnError

Dokumente aus dem Cache liefern, obwohl die Originale nicht mehr verfügbar sind

| Seit Version | 2.3.9 |
|---|---|
| Modul | mod_cache |

| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (All) |
|---|---|
| Syntax | CacheStaleOnError on\|off |
| Standardwert | on |

Wenn der Cache ein Dokument enthält, dessen Original einen Server-Fehler (Status 5xx) produziert, liefert Apache dieses Dokument normalerweise dennoch aus dem Cache aus, anstatt den Fehler an den Client weiterzugeben. Wenn Sie diese Direktive auf `off` setzen, wird dagegen der Fehler an den Client durchgereicht, aber nicht im Cache gespeichert.

#### CacheStoreExpired

Caching abgelaufener Dokumente

| Seit Version | 2.3 |
|---|---|
| Modul | mod_cache |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (All) |
| Syntax | CacheStoreExpired on\|off |
| Standardwert | off |

Seit Version 2.2.4 werden Dokumente, die der Backend-Server als abgelaufen meldet (`Expires`-Wert in der Vergangenheit), nicht mehr im Cache gespeichert. Wenn Sie diese Direktive auf `on` setzen, können Sie dieses Verhalten ändern.

#### CacheStoreNoStore

Inhalte trotz `Cache-Control: no-store` im Cache ablegen

| Seit Version | 2.1-beta |
|---|---|
| Modul | mod_cache |
| Kontext | Server, <VirtualHost> |
| Syntax | CacheStoreNoStore On\|Off |
| Standardwert | Off |

Normalerweise werden Ressourcen nicht im Cache gespeichert, wenn der Header `Cache-Control` den Wert `no-store` aufweist (siehe Kapitel 2, »Funktionsweise von Webservern«). Wenn Sie diese Direktive auf `On` setzen, wird dieses Verhalten überschrieben. Da `no-store` allerdings gemäß HTTP-Spezifikation (RFC 2616) dazu dient, vertrauliche Informationen vor der Speicherung und der versehentlichen Weitergabe zu schützen, sollten Sie dies nur in begründeten Ausnahmefällen in Erwägung ziehen.

### CacheStorePrivate

Inhalte trotz `Cache-Control: private` im Cache ablegen

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_cache` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `CacheStorePrivate On\|Off` |
| **Standardwert** | `Off` |

`CacheStorePrivate` erledigt dieselbe Aufgabe wie `CacheStoreNoStore`, allerdings für Ressourcen, bei denen der `Cache-Control`-Header den Wert `private` aufweist. Auch hier gilt die Warnung, dass diese Inhalte eigentlich ausdrücklich vom Caching ausgenommen wurden, sodass Sie den Wert dieser Direktive in aller Regel nicht auf `On` setzen sollten.

### CacheRoot

Stammverzeichnis für die Cache-Dateien

| | |
|---|---|
| **Modul** | `mod_cache_disk` (bis 2.2 `mod_disk_cache`) |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `CacheRoot Verzeichnis` |
| **Standardwert** | nicht gesetzt |

Dies ist die wichtigste Direktive von `mod_cache_disk`: Sie legt das Verzeichnis fest, unter dem der Festplatten-Cache angelegt wird. Wenn Sie sie weglassen, steht der Cache nicht zur Verfügung. Beispiele (für UNIX und Windows):

```
CacheRoot /usr/local/apache2/cache
CacheRoot "C:/Apache2/cache"
```

### CacheMaxFileSize

Maximale Dateigröße eines Dokuments im Cache

| | |
|---|---|
| **Modul** | `mod_cache_disk` (bis 2.2 `mod_disk_cache`) |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `CacheMaxFileSize Bytes` |
| **Standardwert** | `1000000` |

Mit dieser Direktive können Sie die maximale Größe für Dateien festlegen, die in den Festplatten-Cache aufgenommen werden. Der Vorgabewert beträgt eine Mil-

lion Byte (etwa 977 Kilobyte). Das folgende Beispiel legt genau zwei Megabyte fest:

```
CacheMaxFileSize 2097152
```

### CacheMinFileSize

Minimale Dateigröße eines Dokuments im Cache

| | |
|---|---|
| **Modul** | mod_disk_cache |
| **Kontext** | Server, <VirtualHost> |
| **Syntax** | CacheMinFileSize Bytes |
| **Standardwert** | 1 |

Diese Direktive legt die Minimalgröße von Dateien fest, die im Festplatten-Cache gespeichert werden. Das folgende Beispiel stellt ein Kilobyte als Mindestgröße ein:

```
CacheMinFileSize 1024
```

### CacheDirLength

Anzahl der Zeichen in Cache-Verzeichnisnamen

| | |
|---|---|
| **Modul** | mod_cache_disk (bis 2.2 mod_disk_cache) |
| **Kontext** | Server, <VirtualHost> |
| **Syntax** | CacheDirLength Anzahl |
| **Standardwert** | 2 |

Diese Direktive betrifft die interne Struktur des Festplatten-Caches: Sie legt fest, wie viele Zeichen die Namen der Unterverzeichnisse im Cache-Verzeichnis innehaben sollen. Beachten Sie, dass das Produkt CacheDirLength * CacheDirLevels nicht größer als 20 sein darf. Beispiel:

```
CacheDirLength 3
```

### CacheDirLevels

Anzahl der Unterverzeichnisebenen im Cache

| | |
|---|---|
| **Modul** | mod_cache_disk (bis 2.2 mod_disk_cache) |
| **Kontext** | Server, <VirtualHost> |
| **Syntax** | CacheDirLevels Anzahl |
| **Standardwert** | 3 |

Mit `CacheDirLevels` wird die Anzahl der Unterverzeichnisebenen im Cache-Verzeichnis festgelegt. Auch bei dieser Direktive müssen Sie den Höchstwert 20 für `CacheDirLength` * `CacheDirLevels` beachten. Beispiel:

```
CacheDirLevels 6
```

### MCacheSize

Maximale Speichergröße für den Cache

| | |
|---|---|
| **Version** | bis 2.2 |
| **Modul** | `mod_mem_cache` |
| **Kontext** | Server |
| **Syntax** | `MCacheSize KBytes` |
| **Standardwert** | `100` |

Diese Direktive legt die maximale Größe des Arbeitsspeicher-Caches fest. Syntax und Funktion entsprechen ansonsten `CacheSize` – wobei Sie natürlich einen erheblich kleineren Wert wählen sollten als für den Festplatten-Cache; die Voreinstellung beträgt 100 Kilobyte.

### MCacheMaxObjectCount

Maximale Anzahl der Elemente im Cache

| | |
|---|---|
| **Version** | bis 2.2 |
| **Modul** | `mod_mem_cache` |
| **Kontext** | Server |
| **Syntax** | `MCacheMaxObjectCount Wert` |
| **Standardwert** | `1009` |

Diese Direktive bestimmt, wie viele Objekte sich maximal im Arbeitsspeicher-Cache befinden dürfen. Sobald die Grenze erreicht ist, wird nach den Regeln der Direktive `MCacheRemovalAlgorithm` eine Datei entfernt, bevor eine neue hinzugefügt wird.

### MCacheMaxObjectSize

Maximale Größe eines Dokuments im Cache

| | |
|---|---|
| **Version** | bis 2.2 |
| **Modul** | `mod_mem_cache` |
| **Kontext** | Server |

| Syntax | MCacheMaxObjectSize Bytes |
|---|---|
| Standardwert | 10000 |

Diese Direktive setzt die maximale Größe eines Dokuments, das im Arbeitsspeicher-Cache abgelegt werden kann. Syntax und Funktionsweise entsprechen CacheMaxFileSize.

### MCacheMinObjectSize

Mindestgröße eines Dokuments, das im Cache abgelegt wird

| Version | bis 2.2 |
|---|---|
| Modul | mod_mem_cache |
| Kontext | Server |
| Syntax | MCacheMinObjectSize Bytes |
| Standardwert | 0 |

Mit dieser Direktive wird die Mindestgröße für ein Objekt im Speicher-Cache festgelegt. Sie ähnelt der Direktive CacheMinFileSize des Festplatten-Caches.

### MCacheMaxStreamingBuffer

Maximale Puffergröße, bevor das Caching eines Dokuments abgebrochen wird

| Version | bis 2.2 |
|---|---|
| Modul | mod_mem_cache |
| Kontext | Server |
| Syntax | MCacheMaxStreamingBuffer Bytes |
| Standardwert | entweder 100000 oder MCacheMaxObjectSize<br>(immer der jeweils kleinere Wert) |

Wenn Server-Antworten über Streaming-Protokolle geliefert werden (beispielsweise durch HTTP mit Chunks realisiert), steht die Gesamtgröße des Dokuments nicht von vornherein fest. Die Prüfung, ob die Datei kleiner als MCacheMaxObjectSize ist, kann deshalb nicht stattfinden. Deshalb bestimmt der Wert dieser Direktive, wie viele Bytes der Antwort gepuffert werden, bevor entschieden wird, dass die Datei zu groß für den Cache ist. Wenn Sie die Direktive weglassen, wird der Wert von MCacheObjectSize verwendet, höchstens aber 100.000 Byte (etwa 98 Kilobyte). Das folgende Beispiel legt 300 Kilobyte fest:

```
MCacheMaxStreamingBuffer 307200
```

**MCacheRemovalAlgorithm**

Algorithmus zur Auswahl der Dokumente, die aus dem Cache entfernt werden

| | |
|---|---|
| **Version** | bis 2.2 |
| **Modul** | `mod_mem_cache` |
| **Kontext** | Server |
| **Syntax** | `MCacheRemovalAlgorithm LRU\|GDSF` |
| **Standardwert** | `GDSF` |

Mit dieser Direktive können Sie festlegen, nach welchem Verfahren bei Bedarf die Objekte ausgewählt werden, die aus dem Arbeitsspeicher-Cache entfernt werden. Sie können einen der beiden folgenden Werte angeben:

- `LRU` (Last Recently Used): Es werden die Dateien entfernt, auf die am längsten nicht mehr zugegriffen wurde.
- `GDSF` (GreadyDual-Size): Dieses Verfahren berechnet für jedes Objekt im Cache eine Priorität, die sich aus dem nötigen Aufwand für eine erneute Anfrage und aus der Dateigröße zusammensetzt. Die Dateien mit der niedrigsten Priorität werden entfernt.

### 13.2.3 htcacheclean

Das Apache-Hilfsprogramm `htcacheclean` wurde in Apache 2.1-beta neu eingeführt. Es dient dazu, den Festplatten-Cache aufzuräumen, und ersetzt einige für Apache 2.0 geplante, aber niemals umgesetzte Garbage-Collection-Direktiven. Das Programm besitzt folgende Kommandozeilenoptionen:

- `-d<Minuten>`
  Als Daemon ausführen und jeweils nach der angegebenen Anzahl von Minuten einen Säuberungsdurchlauf starten. Diese Option kann nicht zusammen mit `-D`, `-v` und `-r` verwendet werden.
- `-D`
  Einen Testdurchlauf (»dry run«) durchführen, aber nichts löschen
- `-v`
  Ausführliche Meldungen und Statistiken ausgeben (verbose mode)
- `-r`
  Gründlich säubern (im Prinzip alles löschen). Apache darf nicht laufen, während Sie diese Option verwenden.
- `-n`
  Reduziert die Priorität, mit der dieses Programm ausgeführt wird (nice). Diese Option ist besonders für den Daemon-Betrieb geeignet.

- `-t`
  Nicht nur Dateien, sondern auch leere Verzeichnisse löschen. Standardmäßig werden nur die gespeicherten Dateien, aber nicht die zur Verwaltung verwendeten Verzeichnisstrukturen gelöscht. Falls Sie den Cache mithilfe der Direktiven `CacheDirLength` und `CacheDirLevels` so konfiguriert haben, dass besonders viele Verzeichnisse erzeugt werden, sollten Sie diese Option von Zeit zu Zeit nutzen.
- `-p<Pfad>`
  Gibt das Stammverzeichnis des Festplatten-Caches an. Der Wert sollte der bereits beschriebenen Direktive `CacheRoot` entsprechen.
- `-l<Limit>`
  Angabe der maximalen Cache-Größe. Die Einheit ist Byte, es sei denn, Sie schreiben `K` für Kilobyte oder `M` für Megabyte hinter den Wert.
- `-i`
  Nur ausführen, falls es Änderungen im Cache gab (intelligent mode). Diese Option kann nur zusammen mit `-d` verwendet werden.
- `-a` (seit Version 2.3)
  Eine Liste der im Cache gespeicherten URLs anzeigen
- `-A` (seit Version 2.3)
  Eine ausführliche Liste der im Cache gespeicherten URLs mit folgenden Detailinformationen anzeigen: URL, Header-Größe, Body-Größe, Status, Entity-Version, Datum, Verfallsdatum, Anfragezeit, Antwortzeit, Body vorhanden, Header vorhanden

Ebenfalls seit Version 2.3 können Sie hinter den Optionen explizit eine Liste der zu löschenden URLs angeben.

Das folgende Beispiel startet das Programm als Daemon, der den Cache alle 30 Minuten säubert, und zwar nur bei Bedarf:

```
$ htcacheclean -d30 -i -p/usr/local/apache2/cache
```

Hier ein weiteres Beispiel, das einen Testdurchlauf mit ausführlicher Ausgabe durchführt:

```
$ htcacheclean -D -v -p/usr/local/apache2/cache
```

## 13.3 Zusammenfassung

Apache kann sowohl als Forward-Proxy konfiguriert werden, der den Stellvertreterzugriff auf andere Sites und Protokolle vornimmt, als auch als Reverse-Proxy, der fremde Inhalte in den lokalen URL-Namensraum einbindet. Apache 2 verwendet ein intelligentes Modell für den Proxy-Betrieb: Das Basismodul `mod_proxy` stellt die allgemeine Proxy-Funktionalität bereit, während Helfermodule die Vermittlung der einzelnen Protokolle übernehmen. Zurzeit werden daher HTTP, FTP, AJP und SSL unterstützt; mit weiteren ist zu rechnen. In der neuesten Version wurde zudem eine eingebaute Lösung für Proxy-Load-Balancing eingeführt.

Nicht nur im Zusammenhang mit Proxys ist das Thema Caching interessant: Die Fähigkeit, eigene und fremde Inhalte für den schnelleren Zugriff zwischenzuspeichern, kann die Performance verbessern und die Netzwerkbandbreite schonen. Auch der Cache wird bei Apache durch ein zweistufiges Modell realisiert: `mod_cache` stellt die Grundfunktion bereit; verschiedene Speicherlösungen realisieren den Cache z. B. im Arbeitsspeicher oder auf der Festplatte.

*»Nichts ist so gefährlich wie das Allzumodernsein.*
*Man gerät in Gefahr, plötzlich aus der Mode zu kommen.«*
*– Oscar Wilde*

# 14 CGI

Das *Common Gateway Interface* (CGI) ist die klassische Schnittstelle für serverseitige Webanwendungen. Es handelt sich nicht um eine bestimmte Programmiersprache oder Technologie, sondern um eine standardisierte Möglichkeit für den Webserver, mit externen Programmen zu kommunizieren. Diese Programme erhalten Formulardaten als Eingabe; ihre Standardausgabe wird vom Webserver an den Client gesendet. Auf diese Weise lassen sich einfache interaktive Anwendungen realisieren.

Im vorliegenden Kapitel wird diese Technologie vorgestellt, und Sie erfahren vor allem, wie Sie Apache konfigurieren müssen, um sie zu verwenden. Selbstverständlich gibt es heutzutage zahlreiche modernere, raffiniertere und performantere Technologien für Webserver-Anwendungen – einige von ihnen werden im nächsten Kapitel vorgestellt. Trotzdem ist CGI noch heute weit verbreitet – vor allem, weil es unzählige fertige Lösungen und gute Anleitungen für jede erdenkliche Gelegenheit bereithält. Im Übrigen lässt sich das Grundprinzip webbasierter Anwendungen am einfachsten anhand dieser Basistechnik erläutern. Wenn Sie mit ihr zurechtkommen, werden Sie auch fortgeschrittenere Verfahren leichter verstehen.

Da die CGI-Beispiele in diesem Kapitel in Perl geschrieben sind, lassen sie sich überdies fast eins zu eins auf das im nächsten Kapitel vorgestellte Apache-Modul `mod_perl` übertragen.

## 14.1 Die CGI-Schnittstelle

Schon die ursprünglichen Webserver des CERN und der NCSA waren mit CGI-Fähigkeiten ausgestattet. Die Grundlagen des bis heute gültigen CGI/1.1-Standards können Sie unter *hoohoo.ncsa.uiuc.edu/cgi* nachlesen; das Dokument stammt von 1995 und bezieht sich auf den damals noch aktuellen NCSA-HTTPd.

Die Definition der CGI-Schnittstelle selbst hat sich seitdem tatsächlich nicht mehr weiterentwickelt. Allerdings gibt es etwa für Perl – die Sprache, in der die meisten CGI-Skripte geschrieben werden – einige Erweiterungen, die insbesondere die Performance steigern. Beispiele dafür sind FastCGI sowie das im nächsten Kapitel behandelte Apache-Modul `mod_perl`.

Noch ein Hinweis zur Terminologie: Normalerweise ist in Dokumentationen von CGI-Skripten die Rede; manchmal werden sie aber auch als CGI-Programme bezeichnet. Formal besteht ein kleiner Unterschied zwischen diesen Begriffen: Ein CGI-Programm ist ein Binary Executable, das beispielsweise in C geschrieben wurde, während ein Skript in einer interpretierten Sprache wie Perl, Python oder gar einer Shell-Skriptsprache verfasst wird. Da es inzwischen virtuelle Maschinen für Binärprogramme und Just-in-time-Compiler für Skriptsprachen gibt, wird dieser Unterschied immer unerheblicher. Deshalb werden die beiden Begriffe hier synonym verwendet.

Bevor Sie CGI-Einstellungen in Ihrem Apache-Webserver vornehmen und CGI-Skripte schreiben, soll Ihnen eine kurze Beschreibung der Schnittstelle an dieser Stelle ein wenig Hintergrundwissen bieten. CGI-Programmierung ist einfacher umzusetzen, wenn Sie nachvollziehen können, auf welche Weise der Server mit einem externen Programm Daten austauscht. Ein grundlegendes Schema der CGI-Kommunikation finden Sie in Abbildung 14.1.

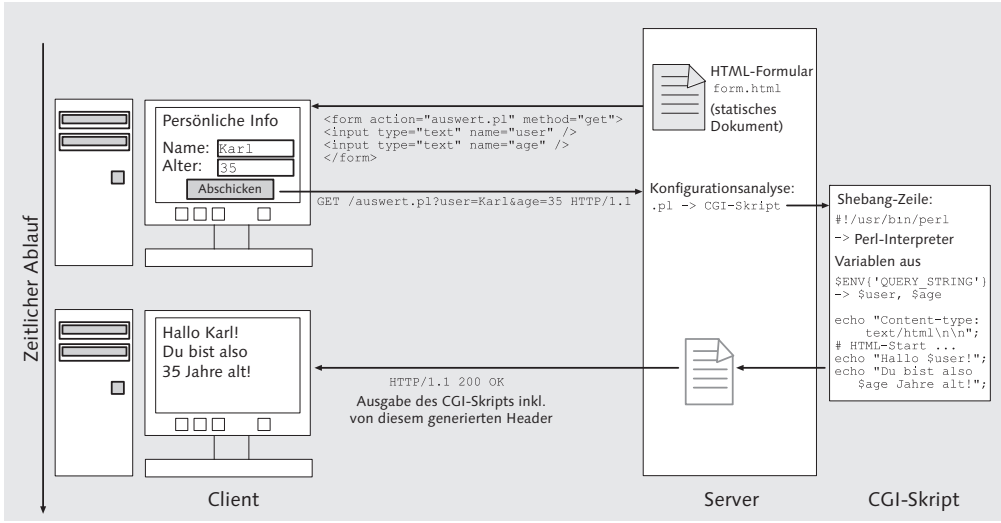



**Abbildung 14.1** Schematischer Ablauf der Abarbeitung eines CGI-Skripts

Prinzipiell werden bei der Abarbeitung eines CGI-Skripts immer dieselben Schritte ausgeführt:

1. Wenn der Client eine bestimmte URL anfordert, überprüft Apache anhand des Verzeichnisses beziehungsweise der Dateiendung, ob es sich um die URL eines CGI-Programms handelt. Im in Abbildung 14.1 gezeigten Beispiel bemerkt der Server es an der Endung `.pl`.
2. Falls tatsächlich ein CGI-Skript angefordert wurde, startet der Webserver zunächst einen neuen Prozess (besonders dieser Schritt ist für das Performance-Problem von CGI-Skripten auf stark beanspruchten Webservern verantwortlich).
3. Der neu erzeugte Prozess wird dazu verwendet, das CGI-Programm auszuführen. Falls es sich um ein binäres Programm handelt, wird es einfach gestartet. Ist es dagegen ein Skript, das in einer Interpretersprache geschrieben wurde (im in Abbildung 14.1 gezeigten Beispiel handelt es sich um Perl), dann muss auch noch der entsprechende Interpreter geladen werden – ein weiterer Performance-Engpass ist die Folge.
4. Das gestartete CGI-Programm kann Formulardaten entgegennehmen, die mit der HTTP-Anfrage versendet wurden (siehe Kapitel 2, »Funktionsweise von Webservern«): bei einer `GET`-Anfrage aus dem Query-String, der als URL-Anhang hinter dem `?` folgt; bei `POST` aus dem Anfrage-Body, der dem Programm über die Standardeingabe (`stdin`) bereitgestellt wird. Im Beispiel ist es eine `GET`-Anfrage mit den an die URL angehängten CGI-Parametern `user=Karl` und `age=35`.
5. Daneben kann das CGI-Skript zahlreiche Informationen über den Client, den Server und die HTTP-Anfrage aus Umgebungsvariablen auslesen, die der Webserver exportiert.
6. Das Skript kann nun beliebige Arbeitsschritte durchführen. In der Regel wird aus den Eingabedaten, aus unveränderlichen Vorlagetexten und gegebenenfalls aus weiteren Quellen wie etwa Datenbanken dynamisch ein HTML-Dokument erstellt. Es gibt allerdings auch CGI-Skripte, die Bilder oder – mithilfe geeigneter Zusatzmodule – sogar PDF-Dokumente und Flash-Filme erzeugen. Im Beispiel ist es ein HTML-Dokument, von dem nur die Zeile mit dem dynamischen Inhalt gezeigt wird.
7. Alles, was das Skript auf die Standardausgabe (`stdout`) schreibt, wird an den Webserver zurückgegeben, der es unverändert als HTTP-Antwort an den Client schickt. Sie müssen deshalb darauf achten, die nötigen HTTP-Header (mindestens `Content-Type`) selbst zu generieren; bei CGI-Skripten erledigt Apache das nicht automatisch für Sie.

### 14.1.1 FastCGI

Aufgrund der Problematik, dass CGI für jede Anfrage einen neuen Prozess starten muss, wurde die Variante *FastCGI* entwickelt. Bei dieser gibt es einen einzelnen Prozess, der alle Anfragen für CGI-Skripte ausführt; dies verbessert die CGI-Performance erheblich.

Bis einschließlich Version 2.2 gehört die FastCGI-Unterstützung nicht zum Lieferumfang von Apache; Sie müssen ein Drittanbieter-Modul herunterladen und installieren. Die bekannteste und empfehlenswerteste Implementierung für Apache 2 ist `mod_fcgid`. Informationen und Downloads finden Sie unter *http://fastcgi.coremail.cn*. Eine Binärversion für Windows gibt es bei ApacheLounge (*http://www.apachelounge.com*), einer Website, die sich auf den Apache-Betrieb unter Windows spezialisiert hat.

Bei Apache 2.3-beta wird das Hilfsprogramm *fcgistarter* mitgeliefert. Es ermöglicht den separaten Start eines FastCGI-Programms oder -Skripts. Die Syntax ist sehr einfach:

```
fcgistarter -c Befehl -p Port [-i Interface] -N Instanzenzahl
```

Die Parameter bedeuten Folgendes:

- `-c Befehl` gibt den Pfad des auszuführenden Skripts oder Programms an.
- `-p Port` gibt den TCP-Port an, auf dem *fcgistarter* lauschen soll.
- `-i Interface` ist optional und gibt die IP-Adresse einer bestimmten Netzwerkschnittstelle an, auf der *fcgistarter* Verbindungen entgegennehmen soll. Wenn Sie den Parameter weglassen, wird auf allen verfügbaren Schnittstellen gelauscht.
- `-N Instanzenanzahl` gibt an, wie viele Instanzen des Skripts oder Programms ausgeführt werden sollen. Sie können sich dabei an der maximalen Anzahl von Worker-Prozessen beziehungsweise -Threads orientieren.

Das folgende Beispiel startet 20 Instanzen des Perl-Skripts *test.pl* als FastCGI-Skript auf Port 4000 und wechselt durch das angehängte `&` in den Hintergrund:

```
# fcgistarter -c /usr/local/scripts/test.pl -p 4000 -N 20 &
```

Die Verbindung zu Apache wird über das Proxy-Modul `mod_proxy_fcgi` (siehe auch Kapitel 13, »Proxy- und Cache-Funktionen«) hergestellt. Das folgende Beispiel stellt das soeben gestartete Skript über den Pfad */test-cgi* bereit:

```
ProxyPass /test-cgi/ fcgi://localhost:4000/
```

Dazu müssen die beiden Module `mod_proxy` und `mod_proxy_fcgi` aktiviert sein.

## 14.2 Apache für CGI-Skripte konfigurieren

Zunächst einmal müssen Sie dafür sorgen, dass das Modul `mod_cgi` aktiv ist. Bei einem normal installierten Apache ist dies bereits der Fall. Wurde das Modul als DSO kompiliert, dann müssen Sie es natürlich mit folgender Direktive aktivieren:

```
LoadModule cgi_module modules/mod_cgi.so
```

Im Übrigen benötigen Sie für die normale CGI-Konfiguration noch `mod_alias` (siehe Kapitel 8, »Weiterleitungen und Indizes«); dieses Modul stellt die Direktiven `ScriptAlias` und `ScriptAliasMatch` bereit.

Wenn Sie ein Multithreading-MPM unter UNIX verwenden, sollten Sie stattdessen `mod_cgid` einsetzen; dies wird in entsprechenden Apache-Installationen automatisch vorkonfiguriert. Diese Variante verbessert durch ein weiter unten kurz beschriebenes Verfahren die Performance. Die möglichen Konfigurationsdirektiven für beide Module sind identisch; für `mod_cgid` gibt es lediglich die zusätzliche Direktive `ScriptSock`.

### 14.2.1 CGI-Verzeichnisse

Standardmäßig ist die Ausführung von CGI-Skripten in der `DocumentRoot` einer Website nicht aktiviert. Stattdessen gibt es in der Normalkonfiguration ein separates Verzeichnis, das sich aus Sicherheitsgründen außerhalb des Website-Verzeichnisses befindet. Dieses Verzeichnis wird durch die Direktive `ScriptAlias` aus dem Modul `mod_alias` bereitgestellt.

#### ScriptAlias

Bildet ein Verzeichnis unter den Verzeichnisbaum der Website ab und aktiviert darin CGI-Skripte.

| | |
|---|---|
| **Modul** | `mod_alias` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ScriptAlias URL-Pfad Dateisystem-Pfad` |
| **Standardwert** | nicht gesetzt |

Der traditionelle Name des CGI-Verzeichnisses ist *cgi-bin*. Bei einer normalen Apache-Installation wird dieses Verzeichnis unter der `ServerRoot` angelegt und mit einer Anweisung wie dieser in das *htdocs*-Verzeichnis abgebildet:

```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/
```

Ein Zugriff auf die URL *http://www.mynet.de/cgi-bin/skript.pl* spricht dann in Wirklichkeit das Skript */usr/local/apache2/cgi-bin/skript.pl* an.

Im Grunde ist diese Direktive nur eine Abkürzung für folgende Schreibweise:

```
Alias /cgi-bin/ /usr/local/apache2/cgi-bin/
<Directory "/usr/local/apache2/cgi-bin/">
  SetHandler cgi-script
</Directory>
```

Bei einem Verzeichnis, das mithilfe der Direktive `ScriptAlias` eingebunden wird, gelten die enthaltenen Dateien daher automatisch als CGI-Skripte und werden vom Server auch so behandelt. Aus diesem Grund benötigen sie keine spezielle Dateiendung. CGI-Skripte müssen aber stets einige andere spezielle Voraussetzungen erfüllen:

- Auf einem UNIX-System müssen sie als ausführbar für alle Benutzer gekennzeichnet sein. Sie müssen also nach dem Erstellen eines solchen Skripts das folgende Dateirecht setzen:

  ```
  # chmod a+x skript.pl
  ```

  Unter Windows entfällt dieser Schritt; diese Art von Dateirechten existiert hier nicht.

- Falls es sich um ein interpretiertes Skript handelt, muss dem Webserver mitgeteilt werden, welcher Interpreter für seine Ausführung zuständig ist. Die traditionelle UNIX-Methode dafür ist die Shebang in der ersten Zeile des Skripts. Sie gibt den vollständigen Pfadnamen des Programms an, z.B.:

  ```
  #!/usr/bin/perl
  ```

  Windows kennt diesen Mechanismus eigentlich nicht. Hier entscheidet normalerweise allein die Dateiendung darüber, welches Programm ein Dokument oder Skript öffnen soll. Apache benötigt allerdings auch hier die Shebang-Zeile, um die Zuordnung vorzunehmen (es sei denn, Sie ändern die weiter unten beschriebene spezielle Win32-Konfigurationsdirektive `ScriptInterpreterSource`). Deshalb ist dies eine der wichtigsten Anpassungen, die Sie an einem Perl-CGI-Skript vornehmen müssen, das ursprünglich für UNIX geschrieben wurde. Beispiel:

  ```
  #!C:/Perl/bin/perl.exe
  ```

  Ein kompiliertes, binäres Programm braucht sich natürlich nicht an diese Regel zu halten, da es für sich selbst ausgeführt werden kann. Solche Programme werden aber in der Praxis eher selten als CGIs eingesetzt, zum einen

aus Sicherheitsgründen und zum anderen, weil andere Sprachen praktischere Hilfsmittel für die CGI-Programmierung enthalten.

- Wie bereits erwähnt, muss sich das CGI-Programm nicht nur um die Erzeugung eines vollständigen (meist HTML-)Dokuments auf der Standardausgabe kümmern, sondern auch die nötigen HTTP-Antwort-Header selbst generieren. Die Mindestanforderung ist ein `Content-Type`-Header mit einem zur Ausgabe passenden MIME-Type. Hinter den Headern muss bekanntermaßen eine Leerzeile folgen.

Wenn Sie gleich eine ganze Gruppe von Verzeichnissen nach diesem Schema CGI-fähig machen möchten, können Sie dazu die Direktive `ScriptAliasMatch` verwenden.

#### ScriptAliasMatch

Bildet eine durch einen regulären Ausdruck beschriebene Gruppe von Verzeichnissen in die Website ab und aktiviert in ihnen CGI-Skripte.

| | |
|---|---|
| **Modul** | `mod_alias` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ScriptAliasMatch URL-Pfadmuster Dateisystem-Pfadmuster` |
| **Standardwert** | nicht gesetzt |

Um das Verhalten der Standardkonfiguration der Direktive `ScriptAlias` mithilfe von `ScriptAliasMatch` nachzuahmen, können Sie die folgende Formulierung verwenden:

```
ScriptAliasMatch ^/cgi-bin(.*) /usr/local/apache2/cgi-bin$1
```

Diese Verwendung von `ScriptAliasMatch` bringt zwar keinen praktischen Nutzen, demonstriert aber das Prinzip: Muster, die dem URL-Pfad (erster Parameter) entsprechen, werden auf den im zweiten Parameter angegebenen Pfad umgesetzt. Geklammerte Ausdrücke aus dem URL-Muster können dabei wie gewohnt als `$1`, `$2` usw. im Ersetzungsteil verwendet werden. Einen kurzen Überblick über reguläre Ausdrücke finden Sie in Anhang G.

Hier noch drei praxistauglichere Beispiele:

- Angenommen, Sie haben CGI-Skripte für verschiedene Aufgaben und wollen diese in unterschiedlichen Verzeichnissen ablegen. Die Verzeichnisse heißen `cgi-bin0` bis (maximal) `cgi-bin9` und befinden sich unterhalb der `ServerRoot`. Natürlich ist es kein Problem, sie mit einzelnen `ScriptAlias`-Direktiven wie den folgenden einzubinden:

```
ScriptAlias /cgi-bin0/ /usr/local/apache2/cgi-bin0/
ScriptAlias /cgi-bin1/ /usr/local/apache2/cgi-bin1/
[...]
ScriptAlias /cgi-bin9/ /usr/local/apache2/cgi-bin9/
```

Sehr effizient ist diese Lösung allerdings nicht; außerdem könnte es auch passieren, dass Sie nicht 10, sondern 100 Verzeichnisse auf diese Weise behandeln müssen. Mit einer `ScriptAliasMatch`-Direktive lässt sich dies in einem Schritt erledigen:

```
ScriptAliasMatch ^/cgi-bin([0-9])/(.*)  \
     /usr/local/apache2/cgi-bin$1/$2
```

- Wenn Sie CGI-Skripte haben, die in mehreren Sprachen geschrieben sind (in diesem Beispiel Perl, Shell-Skripte und binäre C-Programme), möchten Sie diese möglicherweise in unterschiedlichen Verzeichnissen wie `cgi-perl`, `cgi-sh` und `cgi-bin` (eben Binaries!) speichern. Mit `ScriptAliasMatch` lässt sich dies folgendermaßen erledigen:

```
ScriptAliasMatch ^/cgi-((perl)|(sh)|(bin))/(.*)   \
     /usr/local/apache2/cgi-$1/$5
```

- Angenommen, Ihre Website besitzt drei interaktive Bereiche: einen datenbankgestützten Katalog, einen Shop und ein Diskussionsforum. Diese Bereiche befinden sich unter *www.mynet.de/katalog*, *www.mynet.de/shop* beziehungsweise *www.mynet.de/forum*. Jeder dieser Bereiche soll sein eigenes CGI-Verzeichnis erhalten; die URLs dieser Verzeichnisse sollen von außen als *www.mynet.de/katalog/cgi-bin* usw. angesprochen werden, in Wirklichkeit aber unter der `ServerRoot` in Verzeichnissen wie *cgi-bin/katalog* liegen. Das Konzept soll von vornherein verallgemeinert werden – alle URLs nach dem Schema *www.mynet.de/[a-z]+/cgi-bin* sollen in entsprechende CGI-Verzeichnisse umgeleitet werden:

```
ScriptAliasMatch ^/([a-z]+)/cgi-bin/(.*)  \
     /usr/local/apache2/cgi-bin/$1/$2
```

### 14.2.2 CGI in normalen Verzeichnissen aktivieren

Die Verwendung spezieller CGI-Verzeichnisse ist nur eine von zwei Möglichkeiten der CGI-Konfiguration. Die andere Methode besteht darin, Dateien mit bestimmten Endungen allgemein zu CGI-Skripten zu erklären. Beachten Sie, dass dies aus sicherheitstechnischen Erwägungen nicht die Ideallösung ist. Es handelt sich meistens um interpretierte Skripte, mindestens aber um ausführbare Programme, deren Funktionalität hinter den Kulissen niemanden etwas angeht.

Schlimmer: Manche CGI-Skripte enthalten äußerst kritische Informationen wie beispielsweise Zugangsdaten für Datenbanken oder geschützte Verzeichnisse. Wenn Sie diese nicht ausreichend schützen, erleichtern Sie Crackern auf diese Weise ihr gefährliches Handwerk.

Allerdings kann es einen wichtigen Grund für diese Konfiguration geben: Wenn Sie Hoster sind und die Websites zahlreicher Kunden als virtuelle Hosts auf Ihren Server-Rechnern ablegen, gibt es kaum eine andere Möglichkeit, Ihren Kunden den Einsatz von CGI-Skripten zu gestatten. Schließlich greifen die Kunden von außen auf Ihren Webspace zu, und Sie können es sich aus Sicherheitsgründen kaum leisten, ihnen den Zugriff auf ein separates CGI-Verzeichnis zu gestatten. Sind Sie umgekehrt Kunde eines solchen Hosters, dann müssen Sie die Apache-Installation, auf der Sie Ihre Website im eigenen Netzwerk testen, an diese Besonderheit anpassen.

Um Dateien in einem bestimmten Kontext CGI-fähig zu machen, müssen Sie die Kerndirektive `Options` anpassen und die von `mod_mime` definierte Konfigurationsanweisung `AddHandler` hinzufügen. Die grundlegende Verwendung dieser beiden Optionen wurde bereits in Kapitel 6, »Grundkonfiguration«, beziehungsweise Kapitel 7, »Header und MIME-Einstellungen«, erläutert.

`Options` benötigt für das betreffende Verzeichnis (oder den gesamten Server beziehungsweise Virtual Host) die Eigenschaft `ExecCGI`, die auch in der allgemeinen Optionenliste `All` enthalten ist. Sie brauchen also bezüglich `Options` nichts mehr zu tun, wenn im gewünschten (oder übergeordneten) Kontext bereits die folgende Konfigurationsanweisung gesetzt ist:

```
Options All
```

Andernfalls müssen Sie `ExecCGI` zu den bestehenden Optionen hinzufügen. Angenommen, Sie finden im betreffenden Kontext folgende Anweisung:

```
Options Indexes FollowSymLinks
```

Dann müssen Sie die Direktive folgendermaßen ergänzen:

```
Options Indexes FollowSymLinks ExecCGI
```

Um nur in einem untergeordneten Kontext, beispielsweise einem speziellen Unterverzeichnis, die CGI-Ausführung zuzulassen, können Sie auch die folgende Direktive für den entsprechenden Kontext verwenden:

```
Options +ExecCGI
```

Diese Schreibweise lässt alle anderen bereits im übergeordneten Kontext gesetzten Optionen unverändert.

Die AddHandler-Direktive muss die Endungen aller Dateitypen auflisten, die als CGI-Skripte gelten sollen. Dazu wird für die CGI-Konfiguration folgendes Schema verwendet:

```
AddHandler cgi-script .Endung1 [.Endung2 ...]
```

Wenn Sie beispielsweise Dateien mit den Endungen .cgi und .pl als ausführbare CGI-Skripte konfigurieren möchten, müssen Sie folgende Anweisung verwenden:

```
AddHandler cgi-script .cgi .pl
```

In der Regel wird diese Direktive einmal für die gesamte Server-Konfiguration gesetzt. Natürlich steht es Ihnen frei, sie alternativ nur für einzelne Verzeichnisse oder andere Unterkontexte einzustellen.

Es kann ein erhebliches Sicherheitsproblem sein, diesen Schritt zu vergessen: Wenn Ihr Server außerhalb der speziellen cgi-bin-Verzeichnisse CGI-Skripte enthält, deren Endungen nicht mittels AddHandler als ausführbar markiert sind, werden diese vom Server im Quelltext ausgeliefert. Warum das so gefährlich ist, wurde bereits erläutert.

Es gibt noch eine (eigentlich veraltete) Alternative zur Verwendung von AddHandler: Sie können die gewünschten Dateiendungen dem MIME-Type application/x-httpd-cgi zuordnen. Dies erreichen Sie durch eine AddType-Direktive im gewünschten Kontext, z. B.:

```
AddType application/x-httpd-cgi .cgi .pl
```

Sie könnten diese Zuordnung stattdessen auch in Ihrer MIME-Type-Datei (durch TypesConfig definiert, standardmäßig conf/mime.types) eintragen. Dieser Eintrag sieht beispielsweise so aus:

```
application/x-httpd-cgi  cgi pl
```

Wie Sie sehen, werden die Dateiendungen hier ohne führenden Punkt eingetragen. Zu empfehlen ist dieses Verfahren allerdings nicht, weil sich Änderungen in der *TypesConfig*-Datei nur schwer wiederfinden lassen. Nähere Informationen zur allgemeinen MIME-Type-Konfiguration finden Sie in Kapitel 7, »Header und MIME-Einstellungen«.

### 14.2.3 Konfigurationsanweisungen für mod_cgi und mod_cgid

Wie Sie in Abschnitt 14.2.2, »CGI in normalen Verzeichnissen aktivieren«, erfahren haben, können Sie die Funktionalität von mod_cgi beziehungsweise mod_cgid einsetzen, ohne eine einzige Konfigurationsdirektive dieser Module selbst zu set-

zen. Für die eigentliche Ausführung von CGI-Skripten sind diese Direktiven auch nicht notwendig; sie konfigurieren vor allem das Logging der CGI-Skripte. In diesem kurzen Abschnitt werden alle Direktiven von `mod_cgi` und `mod_cgid` beschrieben.

### ScriptLog

Pfad der Log-Datei für CGI-Skripte

| | |
|---|---|
| **Modul** | `mod_cgi`, `mod_cgid` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `ScriptLog Pfad` |
| **Standardwert** | nicht gesetzt |

Falls der Pfad relativ angegeben wird, bezieht er sich auf die `ServerRoot`. Das folgende Beispiel legt eine *ScriptLog*-Datei mit konventionellem UNIX-Namen im normalen *logs*-Verzeichnis an:

```
ScriptLog logs/script_log
```

Unter Windows würde die Datei dagegen standardmäßig *script.log* heißen.

Wenn Sie die Direktive `ScriptLog` setzen, wird eine zusätzliche Log-Datei für CGI-Skriptfehler angelegt. Während im `ErrorLog` nur eine lapidare einzeilige Fehlermeldung erscheint, wenn ein CGI-Skript einen Fehler enthält, finden Sie hier eine ausführliche Analyse. Das `ScriptLog` sollten Sie nur zu Debugging-Zwecken während der Entwicklung von CGI-Skripten aktivieren – im laufenden Betrieb kostet seine Verwendung zu viel Performance.

Ein wenig aufpassen müssen Sie mit den Zugriffsrechten für die *ScriptLog*-Datei: Die Einträge werden von der User-ID vorgenommen, unter der das Skript ausgeführt wird. Der betreffende User benötigt also Schreibrechte an der Log-Datei. Falls sie sich in einem separaten Verzeichnis befindet, können Sie dem CGI-Benutzer einfach Schreibrechte für dieses Verzeichnis erteilen. Befindet sie sich dagegen im allgemeinen *logs*-Verzeichnis, ist dies ein zu großes Sicherheitsrisiko. In diesem Fall sollten Sie die Datei manuell erzeugen und dem CGI-Benutzer nur das Schreiben dieser einen Datei erlauben.

### ScriptLogBuffer

Maximale Byte-Anzahl des HTTP-Anfrage-Bodys im `ScriptLog`

| | |
|---|---|
| **Modul** | `mod_cgi`, `mod_cgid` |
| **Kontext** | Server, `<VirtualHost>` |

| Syntax | ScriptLogBuffer Bytes |
|---|---|
| Standardwert | 1024 |

Bei einem CGI-Skriptfehler, der auf eine HTTP-Anfrage vom Typ `POST` oder `PUT` folgt, werden nicht nur die HTTP-Header in das `ScriptLog` geschrieben, sondern auch der Beginn des Entity-Bodys. Damit dies nicht zu Speicherengpässen führt oder gar für DoS-Attacken missbraucht wird, wird die maximale Anzahl von Bytes des HTTP-Anfrage-Bodys, die in die Log-Datei aufgenommen werden, mit dieser Direktive beschränkt. Beispiel:

```
ScriptLogBuffer 2048
```

#### ScriptLogSize

Maximale Größe der *ScriptLog*-Datei

| Modul | mod_cgi, mod_cgid |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | ScriptLogSize Bytes |
| Standardwert | 10385760 |

Ähnlich wie die Größe des einzelnen Eintrags (siehe `ScriptLogBuffer`) kann auch die Gesamtgröße der *ScriptLog*-Datei beschränkt werden. Diese Maßnahme schützt ebenfalls vor Festplattenüberlauf sowie vor Angriffen. Beispiel:

```
ScriptLogSize 20971520    # 20 MByte
```

#### ScriptSock

Pfadname des UNIX-Domain-Sockets für `mod_cgid`

| Modul | mod_cgid |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | ScriptSock Pfad |
| Standardwert | logs/cgisock |

Beim Start des Webservers erzeugt `mod_cgid` automatisch einen zusätzlichen Prozess, in dem die einzelnen CGI-Skripte als Threads ausgeführt werden. Apache kommuniziert mit diesem Prozess über ein UNIX-Domain-Socket. Dies ist eine Pipe-ähnliche, bidirektionale Einrichtung im Dateisystem, die der Interprozess-Kommunikation dient. In jeder anderen Hinsicht funktioniert sie genau wie ein Netzwerk-Socket.

Diese Direktive legt den Pfadnamen des verwendeten UNIX-Domain-Sockets fest. Die Ressource wird mit den Rechten des Benutzers geöffnet, der Apache gestartet hat – in der Regel also `root`. Achten Sie darauf, dass kein anderer Benutzer Schreibrechte in diesem Verzeichnis besitzt.

Hier ein Beispiel für die Verwendung der Direktive:

```
ScriptSock /var/run/apache2/cgid.sock
```

### 14.2.4 Plattformspezifische Einstellungen

Die bisher aufgeführten CGI-Konfigurationsdirektiven gelten für alle Plattformen. Daneben gibt es noch zwei plattformspezifische Konfigurationsanweisungen: `ScriptInterpreterSource` für Windows und `CGIMapExtension` für NetWare.

#### ScriptInterpreterSource

Bestimmt, wie der Interpreter für CGI-Skripte ermittelt wird.

| | |
|---|---|
| **Seit Version** | 1.x; `Registry-Strict` seit 2.0 (nur Windows) |
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `ScriptInterpreterSource Registry \| Registry-Strict \| Script` |
| **Standardwert** | `Script` |

Die Direktive `ScriptInterpreterSource` ist nur für Win32 verfügbar. Sie legt fest, ob Windows den Interpreter für ein CGI-Skript aus der Shebang-Zeile (Standard; siehe oben) oder aus den Einstellungen der Windows-Registry für die Dateiendung des Skripts ermitteln soll. Die Direktive kann die drei folgenden festgelegten Werte annehmen:

- `Registry`
  Apache ermittelt das Programm zur Ausführung des CGI-Skripts aus der Windows-Registry. Dazu schlägt er zunächst unter *HKEY\CLASSES\ROOT* (*HKCR*) die entsprechende Dateiendung nach. Der dort vorgefundene Wert ist der Name eines Dateityps, der ebenfalls einen Schlüssel unter *HKCR* bildet. Apache liest, falls vorhanden, den Unterschlüssel *Shell\ExecCGI\Command* dieses Dateityps und verwendet den dort eingetragenen Befehl zur Ausführung des Skripts. Falls *Shell\ExecCGI\Command* nicht vorhanden ist (dies ist der Normalfall), verwendet der Server hilfsweise den Wert des Unterschlüssels *Shell\Open\Command*, der normalerweise bei einem Doppelklick auf die entsprechende Datei zum Einsatz kommt.

Angenommen, es wird das Programm für ein CGI-Skript namens `test.pl` gesucht. Apache ermittelt zunächst die Zeichenfolge unter *HKCR\.pl*; bei einer normalen ActivePerl-Installation lautet sie *»Perl«*. Da *HKCR\Perl\Shell\ExecCGI\Command* nicht existiert, falls Sie diesen Unterschlüssel nicht manuell erstellt haben, verwendet der Webserver den Eintrag unter *HKCR\Perl\Shell\Open\Command*; dieser lautet etwa *»C:\Perl\bin\ perl.exe %1«*. Das bedeutet, dass Apache das Programm *C:\Perl\bin\ perl.exe* (den Perl-Interpreter) mit dem Argument `test.pl` ausführen soll.

Bitte beachten Sie, dass dieses Verfahren in `ScriptAlias`-Verzeichnissen sehr gefährlich ist: Da Apache 2 in einem solchen Verzeichnis jede Datei als CGI-Skript behandelt, öffnet er auch jede Datei mit dem in der Registry eingestellten Programm. Falls sich also beispielsweise ein gewöhnliches HTML-Dokument mit der Endung `.htm` oder `.html` im Verzeichnis befindet, öffnet Apache auf dem Server-Rechner bei jeder Anfrage eine Instanz des Standardbrowsers; die Belastung des Rechners kann sehr schnell Dimensionen erreichen, die eine weitere Bearbeitung von Anfragen unmöglich machen – gewissermaßen ein DoS-Eigentor.

- `Registry-Strict`
  Der Wert `Registry-Strict`, der erst mit Apache 2 eingeführt wurde, verhindert genau das bei `Registry` geschilderte Sicherheitsproblem: Es wird nur der Wert des Registry-Schlüssels *HKCR\Dateityp\Shell\ExecCGI\Command* verwendet. Ist er nicht vorhanden, wird nicht versucht, das Skript mittels *Shell\Open\Command* auszuführen. Bei dieser Variante werden also definitiv nur diejenigen Dateitypen ausgeführt, für die Sie selbst ein Programm festgelegt haben. Wie man Schlüssel in der Windows-Registry erzeugt, wird in Kapitel 5, »Apache in Betrieb nehmen« (Abschnitt 5.1.2, »Apache unter Windows steuern«), erläutert – zwar in einem anderen Zusammenhang, aber das Prinzip ist dennoch immer dasselbe. Der Wert für den Schlüssel muss *»Pfad\des\Interpreters %1«* (ohne Eingabe der Anführungszeichen) lauten, wobei *%1* für das erste übergebene Argument steht – in diesem Fall für den Pfad des CGI-Skripts.

- `Script`
  Dies ist die Voreinstellung – Apache liest den Namen des Interpreters wie auf UNIX-Systemen aus der Shebang, der ersten Zeile des Skripts selbst. Diese muss dazu folgendes Format besitzen:

  ```
  #!Pfad/des/Interpreters
  ```

  Wie in der Apache-Konfigurationsdatei und bei den meisten anderen Anwendungen, die ursprünglich aus der UNIX-Welt stammen, müssen Sie also den UNIX-Slash (/) als Pfadtrenner verwenden. Beispiel:

  ```
  #!E:/Perl/bin/perl.exe
  ```

Falls sich das *Perl/bin*-Verzeichnis in Ihrem `PATH` befindet, können Sie die Verzeichnisangabe auch weglassen:

```
#!perl.exe
```

In der Regel gibt es kaum einen Grund, die Voreinstellung `Script` zu ändern. Da Sie CGI-Skripte aus Sicherheitsgründen ohnehin *niemals* ungeprüft aus dem Web herunterladen und auf Ihrem Server installieren sollten, können Sie bei der Gelegenheit auch gleich die Shebang-Zeile solcher meist für UNIX konzipierten Fertig-CGIs anpassen – der Wegfall dieser Änderung ist im Wesentlichen der einzige Vorteil von `Registry-Strict`; von `Registry` ist aus den angeführten Gründen ohnehin ganz abzuraten.

#### CGIMapExtension

Pfad des CGI-Interpreters

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `CGIMapExtension Pfad Dateiendung` |
| **Standardwert** | nicht gesetzt |

Diese Direktive gibt auf NetWare-Systemen den Pfad zu einem CGI-Interpreter-Programm an, das für Dateien mit der genannten Endung aufgerufen werden soll. Das folgende Beispiel führt Skripte mit der Dateiendung `.pl` mithilfe von `sys:\perl.lnm` aus:

```
CGIMapExtension sys:\perl.lnm .pl
```

### 14.2.5 Das Modul mod_actions

Ein weiteres CGI-relevantes Modul, das zur Grundausstattung von Apache 2 gehört, ist `mod_actions`. Es stellt zwei Konfigurationsdirektiven zur Verfügung, die die einfache Definition von CGI-Skripten für bestimmte Handler beziehungsweise HTTP-Methoden ermöglichen.

#### Action

CGI-Skript für Handler oder MIME-Type aktivieren

| | |
|---|---|
| **Modul** | `mod_actions` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `Action Action-Type CGI-Skript` |
| **Standardwert** | nicht gesetzt |

`Action` ermöglicht es, für einen bestimmten MIME-Type oder einen Handler ein CGI-Skript anzugeben, das anstelle der aufgerufenen URL ausgeführt wird. Falls sich dieses Skript in einem Verzeichnis befindet, in dem `AcceptPathInfo` aktiviert wurde, erhält es den relativen Pfad der angeforderten Datei über die Umgebungsvariable `PATH_INFO` und den absoluten (Dateisystem-)Pfad über `PATH_TRANSLATED`. Dies lässt sich nutzen, um die eigentlich angeforderte Datei zunächst in irgendeiner Form durch das CGI-Skript zu manipulieren und dann auszugeben. Beachten Sie jedoch, dass sich die meisten derartigen Aufgaben erheblich komfortabler und mit besserer Performance durch Ausgabefilter (siehe Kapitel 16, »SSI und Filter«) umsetzen lassen.

Das folgende Beispiel leitet sämtliche HTML-Dateien im Unterverzeichnis `/actiontest` der `DocumentRoot` an das Skript `/cgi-bin/test.pl` weiter:

```
<Location /actiontest>
  Action text/html /cgi-bin/test.pl
</Location>
```

Damit das Skript irgendetwas Sinnvolles mit der angeforderten Datei anfangen kann, muss das *cgi-bin*-Verzeichnis folgende Konfigurationsanweisung enthalten (oder erben):

```
AcceptPathInfo On
```

Alternativ können Sie das Skript einem Handler statt einem MIME-Type zuweisen; Handler können mittels `AddHandler` für eine bestimmte Dateiendung oder durch `SetHandler` für alle Dateien in einem Kontext festgelegt werden. Das folgende Beispiel aktiviert für alle Dateien im aktuellen Kontext, die die Endung `.htm` (aber nicht `.html`) tragen, das Skript `/cgi-bin/modify.pl`:

```
AddHandler htmodify .htm
Action htmodify /cgi-bin/modify.pl
```

Beispiele für solche Skripte sehen Sie dort, wo die eigentliche CGI-Programmierung besprochen wird, im Verlauf dieses Kapitels.

#### Script

CGI-Skript für eine bestimmte HTTP-Methode aktivieren

| | |
|---|---|
| **Modul** | `mod_actions` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `Script HTTP-Methode CGI-Skript` |
| **Standardwert** | nicht gesetzt |

Mithilfe dieser Direktive können Sie für einzelne HTTP-Anfrage-Methoden generell ein bestimmtes CGI-Skript aktivieren. Sinnvoll ist dies beispielsweise für `PUT`-Anfragen: Das Skript kann erst einmal genau überprüfen, ob der übermittelte Inhalt unbedenklich ist, bevor es ihn auf der Festplatte des Server-Rechners speichert.

Dieses Beispiel leitet `PUT`-Anfragen im aktuellen Kontext an `/cgi-bin/putter.pl` weiter:

```
Script PUT /cgi-bin/putter.pl
```

Beachten Sie, dass dieses Verfahren bei `GET`-Anfragen nur funktioniert, wenn sich hinter der angeforderten URL ein Query-String, das heißt ein `?` mit nachfolgender Parameterliste, befindet. Andernfalls liefert Apache einfach die angeforderte Datei aus.

## 14.3 Umgebungsvariablen

Für die CGI-Programmierung ist es sehr angenehm, dass Apache eine Reihe von Informationen über sich selbst, über den anfragenden Client und die Details der Anfrage selbst als Umgebungsvariablen veröffentlicht. Darüber hinaus existieren mit `mod_env` und `mod_setenvif` zwei Standardmodule, die die Manipulation dieser Variablen ermöglichen. In diesem Abschnitt finden Sie zunächst eine Übersicht über die verfügbaren Umgebungsvariablen und lernen anschließend die Konfigurationsdirektiven kennen, mit denen sich eigene Variablen setzen lassen.

Der Zugriff auf Umgebungsvariablen aus CGI-Programmen heraus funktioniert je nach Programmiersprache unterschiedlich:

- In Perl liegen alle Variablen in dem Hash[1] `%ENV` vor, in dem die Variablennamen die Schlüssel und ihre Werte die Elemente sind. `$ENV{'HTTP_USER_AGENT'}` liefert also beispielsweise den Namen und die Version des anfragenden Client-Programms.
- In PHP (Näheres in Kapitel 15, »Technologien zur Webprogrammierung«) erfüllt das Array `$_SERVER` dieselbe Aufgabe. Auch hier bilden die Variablennamen die Schlüssel, da es keinen Unterschied zwischen Arrays und Hashes

1 Ein Hash entspricht dem Konstrukt, das in anderen Sprachen als *assoziatives Array* bezeichnet wird: Die Indizes sind keine Nummern, sondern Strings. In Perl wird ein ganzer Hash durch das Präfix `%` gekennzeichnet (z.B. `%hash`). Der Zugriff auf ein einzelnes Element erfolgt durch die Angabe des Schlüssels in geschweiften Klammern. Beispiel: `$hash{'info'}`. Um einen neuen Hash mit Anfangswerten zu erzeugen, können Sie folgendes Schema verwenden: `my %hash = (key1 => 'Wert1', key2 => 'Wert2');`.

gibt. `$_SERVER ['REQUEST_METHOD']` z.B. liefert die HTTP-Anfrage-Methode zurück.

- In Java (Näheres dazu ebenfalls im nächsten Kapitel) können Umgebungsvariablen von Haus aus nicht gelesen werden, weil Java-Programme in einer virtuellen Maschine laufen, die nichts über die tatsächliche Systemumgebung weiß. Die Klasse `HttpServletRequest` der Servlet-API bietet dafür allerdings eine Reihe von Methoden nach dem Schema `getVariablenname()`. Anstelle der Unterstriche werden nach Java-Konvention große Anfangsbuchstaben für das jeweils nächste Wort verwendet. `getServerName()` liefert beispielsweise den Wert von `SERVER_NAME`, also den DNS-Namen des Webservers.
- Im Web kursieren einige alte CGI-Beispiele, die als UNIX-Shell-Skripte geschrieben sind (was aus Sicherheits- und Performance-Gründen nicht zu empfehlen ist). Hier können die Werte der Umgebungsvariablen einfach als `$VARIABLEN_NAME` angesprochen werden, z.B. `$SCRIPT_NAME` für den Pfad des aktuellen CGI-Skripts. Unter Windows können analog dazu die (noch weniger empfehlenswerten) MS-DOS-Batch-Dateien mit der Dateiendung `.bat` verwendet werden; hier lautet das Schema `%VARIABLEN_NAME%`, also z.B. `%QUERY_STRING%` für an die URL angehängte Formulardaten.

### 14.3.1 Die Umgebungsvariablen im Überblick

Tabelle 14.1 zeigt die CGI-Umgebungsvariablen im Überblick. Die Liste ist im Prinzip vollständig; es wurden allerdings nicht alle automatisch aus den Anfrage-Headern erzeugten Variablen aufgelistet. Der Aufbau dieser speziellen Variablen ist immer derselbe: Dem (hier vollständig in Großbuchstaben geschriebenen) Namen des Headers (zu finden in Kapitel 2, »Funktionsweise von Webservern«) wird das Präfix `HTTP_` vorangestellt. Beispiele: `HTTP_HOST` (übermittelter Hostname der Anfrage) oder `HTTP_COOKIE` (an den Server übermittelte Cookies).

| Variable | Beschreibung | Beispielwert |
|---|---|---|
| `SERVER_SOFTWARE` | Bezeichnung und Version des Webservers; Apache-Direktive: `ServerTokens` (siehe Kapitel 6, »Grundkonfiguration«) | `Apache/2.3.15 (UNIX)` |
| `SERVER_NAME` | Hostname oder IP-Adresse des Servers `ServerName` (siehe Kapitel 6, »Grundkonfiguration«) | *www.mynet.de* |
| `GATEWAY_INTERFACE` | verwendete CGI-Version | `CGI/1.1` |

**Tabelle 14.1** Übersicht über die Standard-CGI-Umgebungsvariablen

| Variable | Beschreibung | Beispielwert |
|---|---|---|
| SERVER_PROTOCOL | Name und Version des Anwendungsprotokolls, das die Anfrage übermittelt (normalerweise HTTP) | HTTP/1.1 |
| SERVER_PORT | TCP-Port des Servers: Listen (siehe Kapitel 6, »Grundkonfiguration«) | 80 |
| REQUEST_METHOD | HTTP-Anfrage-Methode | GET |
| PATH_INFO | Dies ist die zusätzliche Pfadangabe hinter dem eigentlichen Namen des Skripts. Existiert nur, wenn AcceptPathInfo gesetzt ist. | /more/path/info |
| PATH_TRANSLATED | vom Server umgewandelte Variante von PATH_INFO | /translated/path/info |
| SCRIPT_NAME | Pfad des aktuellen Skripts selbst | /cgi-bin/script.pl |
| QUERY_STRING | Inhalt des URL-Anhangs hinter dem ? | artikelnr=102 |
| REMOTE_HOST | Hostname des anfragenden Clients, falls verfügbar (sonst identisch mit REMOTE_ADDR) | xdsl-213-196-226-224.netcologne.de |
| REMOTE_ADDR | IP-Adresse des anfragenden Clients | 213.196.226.224 |
| AUTH_TYPE | verwendete Art der Benutzerauthentifizierung (siehe Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«) | Basic |
| REMOTE_USER | Name des entfernten Benutzers, falls Authentifizierung verwendet wurde | Gast |
| REMOTE_IDENT | RFC-1431-Identität des entfernten Benutzers, falls verfügbar | i. d. R. nicht vorhanden (spezielles identd-Format) |
| CONTENT_TYPE | MIME-Type des Bodys einer POST- oder PUT-Anfrage (durch Content-Type-Header angegeben) | application/x-www-form-urlencoded |

**Tabelle 14.1** Übersicht über die Standard-CGI-Umgebungsvariablen (Forts.)

| Variable | Beschreibung | Beispielwert |
|---|---|---|
| CONTENT_LENGTH | Anzahl der Bytes im Body einer POST- oder PUT-Anfrage | 32768 |
| HTTP_* (z. B. HTTP_USER_AGENT oder HTTP_ACCEPT) | Alle HTTP-Header der Client-Anfrage (siehe Kapitel 2, »Funktionsweise von Webservern«) werden mit dem Präfix HTTP_ versehen und an das CGI-Skript durchgereicht. | Mozilla/5.0 (Windows; U; Windows NT 6.0; de; rv:1.9.0.3) Gecko/2008092417 Firefox/3.0.3 |

**Tabelle 14.1** Übersicht über die Standard-CGI-Umgebungsvariablen (Forts.)

### 14.3.2 Umgebungsvariablen in der Apache-Konfiguration setzen

Umgebungsvariablen dienen nicht ausschließlich der CGI-Programmierung. Da sie jedoch für CGI eine besonders wichtige Bedeutung haben, ist dieses Kapitel trotzdem ein passender Ort für die Apache-Konfigurationsdirektiven, die dem Setzen von Umgebungsvariablen dienen. Für diese Aufgabe gibt es zwei Standardmodule: mod_env ist das grundlegende Modul zum Setzen der Variablen, während mod_setenvif verschiedene Bedingungen auswerten kann, bevor Variablen gesetzt werden. Darüber hinaus bestimmt die core-Direktive AcceptPathInfo, ob die Umgebungsvariablen PATH_INFO und PATH_TRANS-LATED generiert werden.

Wie Sie bereits in Kapitel 8, »Weiterleitungen und Indizes«, erfahren haben, lassen sich ähnliche Aufgaben auch mithilfe von mod_rewrite umsetzen.

#### SetEnv

Umgebungsvariablen setzen

| | |
|---|---|
| **Modul** | mod_env |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| **Syntax** | SetEnv Umgebungsvariable Wert |
| **Standardwert** | nicht gesetzt |

Diese Direktive setzt die angegebene Variable auf den gewünschten Wert. Sie kann dann auf die oben beschriebene Art und Weise in CGI-Skripten ausgelesen werden. Es ist leider nicht möglich, die Werte der aus der Anfrage übernommenen oder vom Server selbst generierten Variablen zu überschreiben. Der Sinn der Konfigurationsanweisung besteht vielmehr darin, beliebige zusätzliche Variablen zu definieren. Beispiel:

```
SetEnv DATA_PATH /etc/misc_data
```

### PassEnv

Shell-Umgebungsvariablen an CGI-Skripte weitergeben

| Modul | mod_env |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | PassEnv Umgebungsvariable [Umgebungsvariable ...] |
| Standardwert | nicht gesetzt |

Neben den CGI-Umgebungsvariablen, die Apache automatisch an CGI-Skripte weitergibt, existieren auch die Umgebungsvariablen der Shell. Mithilfe von `PassEnv` können Sie diese ebenfalls an alle CGI-Programme durchreichen. Das sollten Sie allerdings mit Umsicht tun, weil manche Variablen aus Sicherheitsgründen nicht in die relativ ungeschützte CGI-Umgebung gehören. Das folgende Beispiel leitet die Umgebungsvariable `SHELL` (den Pfad der aktuellen Login-Shell) an CGI-Skripte weiter:

```
PassEnv SHELL
```

### UnsetEnv

Umgebungsvariable entfernen

| Modul | mod_env |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | UnsetEnv Umgebungsvariable [Umgebungsvariable ...] |
| Standardwert | nicht gesetzt |

Diese Direktive ermöglicht es, eine zuvor gesetzte Variable in einem untergeordneten Kontext wieder zu entfernen. Beispiel:

```
UnsetEnv DATA_PATH
```

### SetEnvIf

Umgebungsvariablen je nach Bedingungen in der Anfrage setzen

| Modul | mod_setenvif |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | SetEnvIf Attribut RegExp [!]Umgebungsvariable[=Wert] [[!] Umgebungsvariable[=Wert] ...] |
| Standardwert | nicht gesetzt |

Die Direktive `SetEnvIf` setzt eine Variable, wenn eine bestimmte Umgebung zutrifft beziehungsweise nicht zutrifft. Das zu überprüfende Attribut kann im Grunde jedes Header-Feld der Client-Anfrage sein (siehe Kapitel 2, »Funktionsweise von Webservern«). Sie können auch einen regulären Ausdruck verwenden, um eine Gruppe von Headern anzugeben. Zusätzlich sind folgende Attribute definiert:

- `Remote_Host` – Hostname des entfernten Client-Rechners
- `Remote_Addr` – IP-Adresse des Client-Rechners
- `Server_Addr` – IP-Adresse des Server-Hosts
- `Remote_User` – User-ID des entfernten Benutzers
- `Request_Method` – HTTP-Anfrage-Methode (`GET`, `POST` usw.)
- `Request_Protocol` – Protokoll der Client-Anfrage (normalerweise HTTP)
- `Request_URI` – die angeforderte Ressource
- Die zu prüfende Bedingung ist ein beliebiger regulärer Ausdruck.
- Für die zu setzende Variable gibt es drei verschiedene Optionen:
- `variable` (ohne Wertangabe): Die Variable erhält den Wert 1 (»gesetzt«).
- `!variable` entfernt die bereits definierte Variable.
- `variable=Wert` setzt die Variable auf den angegebenen Wert.

Hier einige Beispiele:

- `SetEnvIf Request_URI "\.(gif)|(jpg)|(png)$" image_request`
  Setzt die Variable `image_request`, wenn die angeforderte URL auf `.gif`, `.jpg` oder `.png` endet.
- `SetEnvIf Referer "www\.google\.([a-z]+)" \`
  `is_google_search=$1`
  Wenn das Header-Feld `Referer` den Bestandteil `www.google.` `<beliebige TLD>` enthält, wird `is_google_search` auf die gefundene Top-Level-Domain gesetzt. Damit wissen Sie, dass die angeforderte URL durch eine Google-Suche gefunden wurde. Außerdem erhalten Sie durch die TLD die Information, welche »Landesfiliale« von Google befragt wurde.
- `SetEnvIf ^X\- .+ extra_headers`
  Wenn die Anfrage Erweiterungs-Header enthält, die mit `X-` beginnen (dargestellt durch einen Wert, der mindestens ein Zeichen besitzt), wird die Variable `extra_headers` gesetzt.

#### SetEnvIfNoCase

Umgebungsvariablen je nach Bedingungen (ohne Unterschied bei Groß-/Kleinschreibung) setzen

| Seit Version | 1.3 |
|---|---|
| Modul | mod_setenvif |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | SetEnvIfNoCase Attribut RegExp [!]Umgebungsvariable[=Wert] [[!]Umgebungsvariable[=Wert] ...] |
| Standardwert | nicht gesetzt |

SetEnvIfNoCase funktioniert genau wie SetEnvIf – mit der einzigen Ausnahme, dass bei den Suchmustern nicht zwischen Groß- und Kleinschreibung unterschieden wird.

#### BrowserMatch

Browser-Bedingung zum bedingten Setzen von Umgebungsvariablen

| Modul | mod_setenvif |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | BrowserMatch RegExp [!]Umgebungsvariable[=Wert] [[!]Umgebungs-variable[=Wert] ...] |
| Standardwert | nicht gesetzt |

BrowserMatch vergleicht den Header User-Agent (Eigenname, den das anfragende Client-Programm übermittelt) mit einem regulären Ausdruck und setzt auf dieser Grundlage eine Variable. Im Grunde genommen ist

```
BrowserMatch RegExp Variable
```

also lediglich ein Synonym für folgende SetEnvIf-Formulierung:

```
SetEnvIf User-Agent RegExp Variable
```

Beispielsweise setzt die folgende Anweisung die Variable is_windows, wenn die User-Agent-Angabe win enthält:

```
BrowserMatch win is_windows
```

Speziell für den Einsatz mit BrowserMatch gibt es einige besondere (natürlich auch mit SetEnv oder SetEnvIf verwendbare) Variablen, die das Verhalten von Apache bezüglich der Antwort unmittelbar beeinflussen:

- downgrade-1.0 – Apache behandelt die Anfrage als HTTP/1.0.
- force-no-vary – Vor der Auslieferung werden alle Vary-Header aus der Antwort entfernt; einige Clients kommen nicht mit ihnen zurecht.

- `force-response-1.0` – Apache liefert die Antwort im HTTP/1.0-Format.
- `gzip-only-text/html` – Wenn diese Variable den Wert 1 hat, werden nur HTML-Dokumente durch den `deflate`-Filter komprimiert.
- `no-gzip` – Die `deflate`-Komprimierung wird völlig abgeschaltet.
- `nokeepalive` – Deaktiviert persistente Verbindungen.
- `prefer-language` – Setzt die bevorzugte Sprache (ein ISO-Sprachkürzel, siehe Anhang E) für Content-Negotiation.
- `redirect-carefully` – Sorgt für besondere Sorgfalt beim Setzen von Weiterleitungs-Headern.
- `suppress-error-charset` (seit Apache 2.0.40) – Unterdrückt das Senden einer Zeichensatzinformation mit servergenerierten Fehlerseiten.

Die bei der Installation automatisch generierte Konfigurationsdatei enthält eine Reihe von `BrowserMatch`-Anweisungen mit diesen besonderen Variablen, um bekannten Inkompatibilitäten mit bestimmten Clients zu begegnen:

```
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 \
     force-response-1.0
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0
```

#### BrowserMatchNoCase

Browser-Bedingung ohne Beachtung von Groß-/Kleinschreibung

| | |
|---|---|
| **Seit Version** | 1.2 (früher in `mod_browser`; seit 1.3 in `mod_setenvif`) |
| **Modul** | `mod_setenvif` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `BrowserMatchNoCase RegExp [!]Umgebungsvariable[=Wert] [[!]Umgebungsvariable[=Wert] ...]` |
| **Standardwert** | nicht gesetzt |

Diese Direktive ist identisch mit `BrowserMatch`, außer dass sie nicht zwischen Groß- und Kleinschreibung unterscheidet.

#### AcceptPathInfo

Zulässigkeit von Pfadangaben hinter der eigentlichen URL

| Seit Version | 2.0.30 |
|---|---|
| Modul | core |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | AcceptPathInfo On\|Off\|Default |
| Standardwert | Default |

AcceptPathInfo bestimmt, ob Apache Pfadangaben akzeptiert, die sich hinter der angeforderten URL befinden. Die drei möglichen Werte bedeuten Folgendes:

- Off
  Apache weigert sich, Anfragen anzunehmen, bei denen der Pfad hinter dem angeforderten Dateinamen weitergeht. Wenn */info/test.html* beispielsweise eine existierende Datei ist, und ein Client übergibt eine Anfrage mit dem Pfad */info/test.html/extra*, dann gibt der Server den Fehler 404 Not Found zurück.
- On
  Wenn AcceptPathInfo eingeschaltet ist, werden zusätzliche Pfadangaben akzeptiert: Eine Anfrage mit der URL */info/test.html/extra* liefert die vorhandene Datei */info/test.html* aus und speichert den Rest der Pfadangabe in der Umgebungsvariablen PATH_INFO. Darüber hinaus erstellt Apache den zugehörigen absoluten Pfad mithilfe der ServerRoot (im vorliegenden Fall ergibt sich beispielsweise */usr/local/httpd/extra*) und stellt ihn als PATH_TRANSLATED zur Verfügung.
- Default
  Dies ist die Voreinstellung: Apache überlässt dem jeweiligen Handler, der die Anfrage bearbeitet, die Entscheidung, ob dieser zusätzliche Pfadangaben akzeptiert oder nicht. Beispielsweise lässt der Handler cgi-script diese Pfadangaben zu, während der core-Handler sie ablehnt.

#### AllowEncodedSlashes

URLs mit codierten Slashes zulassen

| Seit Version | 2.0.46 |
|---|---|
| Modul | core |
| Kontext | Server, <VirtualHost> |
| Syntax | AllowEncodedSlashes On\|Off |
| Standardwert | Off |

Diese Direktive bestimmt, ob eine URL an irgendeiner Stelle codierte Pfadtrennzeichen (/, unter Windows auch \) enthalten darf – gemeint sind die URL-codierten Zeichenfolgen %2F beziehungsweise %5F. Wenn AllowEncodedSlashes nicht

ausdrücklich auf `On` gesetzt wird, werden Anfragen nach solchen URLs mit dem Statuscode `404 Not Found` beantwortet, andernfalls können CGI-Skripte möglicherweise etwas Sinnvolles damit anfangen – genau wie mit `PathInfo`.

## 14.4 Session-Verwaltung mit mod_session

HTTP ist ein zustandsloses (stateless) Protokoll – jedes Anfrage/Antwort-Paar steht für sich allein und wird weder von einer vorherigen Anfrage beeinflusst, noch beeinflusst es seinerseits eine nachfolgende. Für komplexere Webanwendungen ist dies ein Problem, weil solche Anwendungen sich bestimmte Zustände und Einstellungen über beliebig viele Anfragen hinweg merken müssen – denken Sie beispielsweise an das Warenkorbsystem eines Online-Shops oder an benutzerdefinierte Einstellungen für Aussehen und Inhalte bestimmter Websites.

Um dieses Problem zu lösen, wurden im Prinzip zwei Wege gefunden: Cookies und Sessions. Cookies werden in Form des HTTP-Headers `Set-Cookie` mit einer Server-Antwort mitgeschickt und vom Client gespeichert. Bei der nächsten Anfrage an denselben Server wird das Cookie im Header `Cookie` wieder zurückgeschickt. Der Vorteil von Cookies besteht darin, dass diese auch über eine Browser-Sitzung hinaus gültig bleiben können. Ein Nachteil ist, dass Benutzer das Speichern von Cookies in ihren Browsern ausschalten oder Cookies gezielt löschen können – und da Cookies auch zum Usertracking durch Werbetreibende verwendet werden, gibt es viele User, die dies tun.

Sessions sind dagegen serverseitig: Die Webanwendung erzeugt einen eindeutigen Schlüssel, die Session-ID, und speichert mithilfe dieser ID die Daten für eine bestimmte Benutzersitzung. Wenn der Benutzer Cookies zulässt, werden diese oft als Speichermechanismus für Session-Daten verwendet; andernfalls muss die Session-ID Teil der URL oder des Query-Strings jeder Anfrage an den Server sein.

Bisher waren Sessions ausschließlich Sache der jeweiligen Webanwendung; im nächsten Kapitel wird dies am Beispiel von PHP gezeigt. Apache 2.3/2.4 bietet erstmals Module für die Verwaltung von Sessions auf Webserver-Ebene: `mod_session` und einige Hilfsmodule. Das in Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«, besprochene Modul `mod_auth_form` benötigt `mod_session`; außerdem können Sie die Sessions für eigene CGI-Skripte oder Webanwendungen verwenden (deshalb wird das Modul auch in diesem Kapitel beschrieben).

### 14.4.1 Ein einfaches Konfigurationsbeispiel

Um Sessions zu verwenden, muss die Funktionalität zunächst aktiviert werden – dafür ist die Direktive `Session` zuständig. Um in die Session zu schreiben, muss ein HTTP-Erweiterungs-Header gesetzt werden, der mit `SessionHeader` festgelegt wird. Schließlich wird der Speicherort bestimmt – am einfachsten zu konfigurieren ist ein Cookie, dessen Name mithilfe der Direktive `SessionCookieName` aus dem Modul `mod_session_cookie` festgelegt wird. Das folgende Beispiel verwendet den Session-Header-Namen `X-Replace-Session` und den Cookie-Namen `session`:

```
Session On
SessionHeader X-Replace-Session
SessionCookieName session path=/
```

Nun können Sie in einem CGI-Skript beispielsweise folgendermaßen in die Session schreiben:

```
#!/usr/bin/perl -w
use strict;

print "Content-type: text/html\n";
print "X-Replace-Session: key1=val1&key2=val2\n";
print "\n";
# Beginn des HTTP-Bodys ...
```

Wie Sie an der Zeile `print "X-Replace-Session ..."` erkennen können, werden Schlüssel-Wert-Paare in die Session geschrieben; das Format entspricht einem Query-String.

Die einfachste Möglichkeit, Daten aus der Session lesbar zu machen, besteht darin, sie in Form der Umgebungsvariablen `HTTP_SESSION` bereitzustellen. Dazu wird die Direktive `SessionEnv` aktiviert:

```
SessionEnv On
```

In der Einleitung des vorangegangenen Abschnitts wurde bereits beschrieben, wie in verschiedenen Programmiersprachen auf Umgebungsvariablen zugegriffen wird – in Perl würden Sie beispielsweise `$ENV{'HTTP_SESSION'}` schreiben, in PHP `$_SERVER['HTTP_SESSION']`. Wie sich ein Query-String zerlegen und verarbeiten lässt, erfahren Sie im Abschnitt »Formulardaten einlesen«.

Sessions können statt in Cookies auch mittels DBD in einer Datenbank gespeichert werden; dafür ist das Modul `mod_session_dbd` zuständig. Außerdem können Sie die Session-Daten mithilfe von `mod_session_crypto` verschlüsseln. Näheres erfahren Sie in der Beschreibung der einzelnen Direktiven.

### 14.4.2 Session-Direktiven

Im Folgenden werden die einzelnen Direktiven der Module `mod_session`, `mod_session_cookie`, `mod_session_dbd` und `mod_session_crypto` beschrieben.

#### Session

Die Session-Funktionalität aktivieren

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_session` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `Session On\|Off` |
| **Standardwert** | `Off` |

Setzen Sie diese Direktive auf `On`, um die Verwendung einer Session für den gewünschten Kontext grundsätzlich zu aktivieren. Beachten Sie, dass andere Direktiven wie `SessionHeader` oder `SessionCookieName` benötigt werden, um tatsächlich einen praktischen Nutzen davon zu haben.

#### SessionEnv

Den Inhalt der Session in der Umgebungsvariablen `HTTP_SESSION` bereitstellen

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_session` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `SessionEnv On\|Off` |
| **Standardwert** | `Off` |

Wenn Sie diese Direktive auf `On` setzen, wird der Inhalt der Session in der Umgebungsvariablen `HTTP_SESSION` zur Verfügung gestellt. Dies ist die einfachste Art, auf die Session-Daten zuzugreifen.

#### SessionExclude

URL-Pfade aus der Session-Verwaltung ausschließen

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_session` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |

| Syntax | SessionExclude Pfad |
|---|---|
| Standardwert | none |

Mit dieser Direktive können Sie einen URL-Pfad angeben, für den die Session-Konfiguration ignoriert wird. Dies ermöglicht es Ihnen, statische Bestandteile dynamischer Webanwendungen ohne Session zu betreiben, sodass diese beispielsweise problemlos gecached werden können.

Das Verfahren ähnelt der Angabe eines Pfads für Cookies. Dieser wird jedoch nicht automatisch gesetzt, sondern muss über die Option `path` der Direktive `SessionCookieName` explizit angegeben werden.

Das folgende Beispiel richtet eine Session für den URL-Pfad */myapp* ein, schließt aber das Unterverzeichnis *static* aus:

```
<Location /myapp>
  Session On
  SessionEnv On
  SessionHeader X-Replace-Session
  SessionCookieName session path=/myapp
  SessionExclude /myapp/static
</Location>
```

`SessionExclude` hat Vorrang vor der ebenfalls verfügbaren Direktive `SessionInclude`.

#### SessionHeader

Name des HTTP-Headers für die Session

| Seit Version | 2.3 |
|---|---|
| Modul | mod_session |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SessionHeader Name |
| Standardwert | none |

Diese Direktive gibt den Namen des HTTP-Headers an, in den die Session-Daten geschrieben werden. Sie sollten einen Erweiterungs-Header mit dem Präfix `X-` verwenden. Beispiel:

```
SessionHeader X-Replace-Session
```

### SessionInclude

URL-Pfad, für den die Session gültig ist

| Seit Version | 2.3 |
|---|---|
| Modul | mod_session |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SessionInclude Pfad |
| Standardwert | alle URLs |

Diese Direktive ist die Umkehrung von `SessionExclude`: Wenn Sie sie setzen, wird die Session nur für den angegebenen URL-Pfad innerhalb des Konfigurationskontextes gesetzt. Das folgende Beispiel konfiguriert die Session innerhalb des URL-Pfads */mysite*, verwendet sie aber nur innerhalb von */mysite/scripts*:

```
<Location /mysite>
  Session On
  SessionEnv On
  SessionHeader X-Replace-Session
  SessionCookieName session path=/mysite/scripts
  SessionInclude /mysite/scripts
</Location>
```

### SessionMaxAge

Gültigkeitsdauer von Sessions

| Seit Version | 2.3 |
|---|---|
| Modul | mod_session |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SessionMaxAge Sekunden |
| Standardwert | 0 |

Die Direktive `SessionMaxAge` definiert eine Anzahl von Sekunden seit der letzten Änderung der Session, nach der diese ungültig wird. Dies ist beispielsweise nützlich, wenn die Session im Zusammenhang mit formularbasierter Authentifizierung verwendet wird (siehe Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«) – ein Benutzer, der über einen gewissen Zeitraum nicht aktiv ist, kann so automatisch ausgeloggt werden. Das folgende Beispiel stellt eine Stunde ein:

```
SessionMaxAge 3600
```

Mit dem Standardwert `0` sind Sessions beliebig lange gültig.

### SessionCookieName

Name des RFC-2109-Cookies, in dem die Session-Daten gespeichert werden

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_session_cookie` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `SessionCookieName Name Attribute` |
| **Standardwert** | `none` |

Wenn Sie diese Direktive setzen, die durch das Modul `mod_session_cookie` bereitgestellt wird, dann werden die Session-Daten clientseitig in einem RFC-2109-Cookie mit dem angegebenen Namen zwischengespeichert. Neben dem Namen können Sie einige optionale, durch Semikolon getrennte Attribute angeben, die in Kapitel 2, »Funktionsweise von Webservern«, in der Beschreibung des HTTP-Headers `Set-Cookie` erläutert wurden.

Das folgende Beispiel setzt ein Cookie mit dem Namen `session`, dem Pfad */myapp* und der Domain *app.mynet.de*:

```
SessionCookieName session path=/myapp;domain=app.mynet.de
```

### SessionCookieName2

Name des RFC-2965-Cookies, in dem die Session-Daten gespeichert werden

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | `mod_session_cookie` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `SessionCookieName2 Name Attribute` |
| **Standardwert** | `none` |

Entspricht der Beschreibung von `SessionCookieName`, verwendet jedoch eine modernere Cookie-Implementierung gemäß RFC 2965.

### SessionCookieRemove

Session-Cookies aus den HTTP-Headern eingehender Anfragen entfernen

| Seit Version | 2.3 |
|---|---|
| Modul | mod_session_cookie |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SessionCookieRemove On\|Off |
| Standardwert | Off |

Wenn Sie Apache als Reverse-Proxy verwenden, sollten Sie die Session-Cookies unter Umständen nicht an den Backend-Server weitergeben. Dazu können Sie diese Direktive explizit auf `On` setzen.

### SessionDBDCookieName

Name des RFC-2109-Cookies, in dem die Session-ID gespeichert wird

| Seit Version | 2.3 |
|---|---|
| Modul | mod_session_dbd |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SessionDBDCookieName Name Attribute |
| Standardwert | none |

Wenn Sie die Session-Daten mittels DBD in einer SQL-Datenbank speichern, kann diese Direktive verwendet werden, um die Session-ID in einem Cookie zu speichern. Syntax und Funktionsweise entsprechen ansonsten `SessionCookieName`.

### SessionDBDCookieName2

Name des RFC-2965-Cookies, in dem die Session-ID gespeichert wird

| Seit Version | 2.3 |
|---|---|
| Modul | mod_session_dbd |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SessionDBDCookieName2 Name Attribute |
| Standardwert | none |

Wie `SessionDBDCookieName`, aber mit moderneren RFC-2965-konformen Cookies.

### SessionDBDCookieRemove

Session-ID-Cookies aus den Headern eingehender Anfragen entfernen

| Seit Version | 2.3 |
|---|---|
| Modul | mod_session_dbd |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SessionDBDCookieRemove On\|Off |
| Standardwert | Off |

Entspricht `SessionCookieRemove`, gilt jedoch für die durch `mod_session_dbd` gesetzten Session-ID-Cookies.

### SessionDBDDeleteLabel

SQL-Abfrage zum Entfernen von Sessions aus der Datenbank

| Seit Version | 2.3 |
|---|---|
| Modul | mod_session_dbd |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SessionDBDDeleteLabel Label |
| Standardwert | deletesession |

Die `SessionDBD*Label`-Direktiven geben die Label-Namen von SQL-Abfragen an, die für die verschiedenen Session-Datenbankoperationen verwendet werden. Die Labels beziehen sich auf Abfragen, die zuvor mittels `DBDPrepareSQL` vorbereitet wurden (siehe Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«).

Das folgende Beispiel definiert MySQL-basierte Abfragen für alle Session-Aufgaben. Die `SessionDBD*Label`-Direktiven sind eigentlich nicht notwendig, weil jeweils die Standard-Label-Namen verwendet werden; sie wurden nur der Vollständigkeit halber hinzugefügt:

```
DBDriver mysql
DBDParams "dbname=sessions user=sessionuser
  password=PASSWORT host=localhost"
DBDPrepareSQL "DELETE FROM session WHERE key = '%s'" \
  deletesession
DBDPrepareSQL "UPDATE SESSION SET value = '%s', expiry = %d WHERE
key = '%s'" \
  updatesession
```

```
DBDPrepareSQL "INSERT INTO session (value, expiry, key) values
('%s', %d, '%s')" \
  insertsession
DBDPrepareSQL "SELECT value FROM session WHERE key = '%s' AND
(expiry = 0 OR expiry > %d)" \
  selectsession
SessionDBDDeleteLabel deletesession
SessionDBDUpdateLabel updatesession
SessionDBDInsertLabel insertsession
SessionDBDSelectLabel selectsession
```

Ein Problem besteht zurzeit noch darin, dass die Datenbank im Lauf der Zeit mit abgelaufenen Sessions angefüllt wird. Sie sollten einen externen Cronjob einrichten, der diese von Zeit zu Zeit löscht. Analog zu den obigen Beispielen könnte eine entsprechende SQL-Abfrage so aussehen:

```
DELETE FROM session WHERE expiry != 0 AND expiry < %d
```

#### SessionDBDInsertLabel

SQL-Abfrage zum Einfügen von Sessions in die Datenbank

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | mod_session_dbd |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| **Syntax** | SessionDBDInsertLabel Label |
| **Standardwert** | insertsession |

Mit der durch diese Direktive angegebenen SQL-Query werden neue Sessions in die Datenbank geschrieben. Nähere Infos und ein Beispiel finden Sie in der Beschreibung von SessionDBDDeleteLabel.

#### SessionDBDDeleteLabel

SQL-Abfrage zum Entfernen von Sessions aus der Datenbank

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | mod_session_dbd |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| **Syntax** | SessionDBDDeleteLabel Label |
| **Standardwert** | deletesession |

### SessionDBDPerUser

Benutzernamenbasierte Sessions

| Seit Version | 2.3 |
|---|---|
| Modul | mod_session_dbd |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SessionDBDPerUser On\|Off |
| Standardwert | Off |

Wenn Sie diese Direktive auf `On` setzen und ein Benutzer eingeloggt wird, dann wird der Session-Key auf der Basis des jeweiligen Benutzernamens erzeugt. Andernfalls wird die Direktive ignoriert.

### SessionDBDSelectLabel

SQL-Abfrage zum Lesen von Sessions aus der Datenbank

| Seit Version | 2.3 |
|---|---|
| Modul | mod_session_dbd |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SessionDBDSelectLabel Label |
| Standardwert | selectsession |

Mit `SessionDBDSelectLabel` geben Sie die SQL-Abfrage zum Lesen (Auswählen) von Session-Daten aus der Datenbank an – für nähere Informationen und ein Beispiel siehe `SessionDBDDeleteLabel`.

### SessionDBDUpdateLabel

SQL-Abfrage zum Ändern von Sessions in der Datenbank

| Seit Version | 2.3 |
|---|---|
| Modul | mod_session_dbd |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SessionDBDDeleteLabel Label |
| Standardwert | deletesession |

Die durch diese Direktive angegebene SQL-Abfrage wird zum Ändern existierender Sessions verwendet. Siehe `SessionDBDDeleteLabel` für Details.

### SessionCryptoPassphrase

Passphrase zur Verschlüsselung der Session-Daten

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | mod_session_crypto |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| **Syntax** | SessionCryptoPassphrase Passphrase [Passphrase ...] |
| **Standardwert** | none |

Wenn Sie Session-Daten in clientseitigen Cookies speichern, können Sie diese optional verschlüsseln. Dazu wird das Modul mod_session_crypto verwendet. Mit der Direktive SessionCryptoPassphrase wird die Passphrase zur symmetrischen Verschlüsselung angegeben. Wenn Sie mehrere Passphrasen angeben, wird die erste zur Verschlüsselung verwendet und alle in der Liste für den Versuch der Entschlüsselung. Sie können eine Schlüsselrotation durchführen, indem Sie von Zeit zu Zeit einen weiteren Schlüssel am Ende der Liste anfügen und den vordersten entfernen.

Die Passphrasen sollten möglichst lang sein und aus zufälligen Zeichenfolgen bestehen. Hier ein komplettes Konfigurationsbeispiel:

```
Session On
SessionEnv On
SessionCookieName session path=/
SessionHeader X-Replace-Session
SessionCryptoPassphrase \
  "}X9FJ`rrS\~rIP6[ihT;@SL9K~^[<nOAv:T8mL" \
  "kkDQ-En}Y:nC$qjUC==<\zCeH}$spFEtx`8) Y" \
  ">qMgBb8=y&={eUK05hs.cypSm<Ze|Bc/c8r'Ik"
```

### SessionCryptoDriver

Treiber, der zur Verschlüsselung der Sessions verwendet wird

| | |
|---|---|
| **Seit Version** | 2.3 |
| **Modul** | mod_session_crypto |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| **Syntax** | SessionCryptoDriver Name [Parameter[=Wert] ...] |
| **Standardwert** | none |

Wenn Sie Apache mit verschiedenen Verschlüsselungstreibern kompiliert haben, können Sie mithilfe dieser Direktive auswählen, welcher verwendet wird. Das

folgende Beispiel verwendet den Treiber NSS (Network Security Services, siehe *http://www.mozilla.org/projects/security/pki/nss/*):

```
SessionCryptoDriver nss
```

#### SessionCryptoCipher

Verschlüsselungsverfahren für die Session-Daten

| Seit Version | 2.3 |
|---|---|
| Modul | mod_session_crypto |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (AuthConfig) |
| Syntax | SessionCryptoCipher Name |
| Standardwert | aes256 |

Diese Direktive wählt das konkrete Verschlüsselungsverfahren für die Session-Daten aus. Die möglichen Werte sind:

- 3des192
- aes128
- aes192
- aes256

Welche von diesen Werten tatsächlich zur Verfügung stehen, hängt gegebenenfalls vom gewählten Treiber ab (SessionCryptoDriver).

## 14.5 Grundlagen der CGI-Programmierung

Nachdem Ihr Webserver nun in der Lage ist, CGI-Skripte auszuführen, soll hier ein kurzer Überblick über ihre Programmierung folgen. Da die meisten CGI-Programme in der Praxis in Perl geschrieben werden, wird die Sprache auch für diese Beschreibung verwendet; die grundlegenden Vorgehensweisen sind aber unabhängig von der gewählten Programmiersprache immer dieselben. Die Installation der Programmiersprache Perl selbst wird im nächsten Kapitel in Abschnitt 15.2, »mod_perl«, beschrieben.

Für das Durcharbeiten dieses Abschnitts sind grundlegende Perl-Kenntnisse erforderlich. Grundlegende Konstrukte der Sprache wie Variablen, Kontrollstrukturen oder Ein- und Ausgabe werden nicht erläutert. Es ist sehr wahrscheinlich, dass die UNIX-Administratoren unter den Lesern dieses Buches über diese Kenntnisse verfügen. Aber auch unter Windows kann Perl sehr nützliche Dienste leis-

ten, nicht nur zur CGI-Programmierung. Falls Sie diese faszinierende Sprache noch nicht kennen sollten, empfehle ich Ihnen das Buch [SCHWA2005].

### 14.5.1 Das erste Beispiel

Ein erstes Beispiel für ein Perl-CGI-Skript sehen Sie in Listing 14.1. Es erzeugt lediglich eine statische Ausgabe, die Datum und Uhrzeit enthält:

```
#!/usr/bin/perl -w
$now = scalar localtime;
print << "ENDHTML";
Content-Type: text/html

<html>
<head>
<title>CGI-Test</title>
</head>
<body>
<h1>Willkommen!</h1>
<p>Heute ist $now.</p>
</body>
</html>
ENDHTML
```

**Listing 14.1** Ein erstes kleines CGI-Beispiel

Wenn Sie das Skript in einem Browser betrachten[2] und die CGI-Konfiguration funktioniert hat, sollten Sie eine Ausgabe nach folgendem Schema erhalten:

```
Willkommen!
Heute ist Tue Sep 13 13:58:06 2011.
```

In dem Skript wird ein HIER-Dokument verwendet, um eine lange Reihe von `print`-Befehlen zu vermeiden. Beachten Sie dabei unbedingt, dass die Schlussmarkierung (in diesem Fall `ENDHTML`) allein in einer Zeile stehen muss. Sie muss also in der ersten Spalte beginnen und mit einem Zeilenumbruch enden – selbst dann, wenn es sich um die letzte Zeile des Skripts handelt. Da der Name der Markierung bei der `print`-Anweisung in doppelten Anführungszeichen steht, ist innerhalb des Textes die Variablensubstitution eingeschaltet. Im Beispiel wird so die Variable `$now` ausgewertet, die Datum und Uhrzeit enthält.

2 Natürlich müssen Sie sich das Skript auch dann über eine HTTP-URL von Apache liefern lassen, wenn sich Client und Server auf demselben Host befinden. Nur dies sorgt für die Ausführung des Skripts. Dieser Hinweis ist wichtig für den Fall, dass Sie es gewohnt sind, lokale HTML-Dokumente direkt als Datei (file:-URLs) per Doppelklick in den Browser zu laden.

Falls Ihr Perl-Interpreter einen anderen Pfad hat als `/usr/bin/perl`, müssen Sie die Shebang-Zeile anpassen. Wenn Sie dieses Skript beispielsweise auf einem Windows-Host verwenden, könnte sie folgendermaßen lauten:

```
#!C:/Perl/bin/perl.exe -w
```

Die Perl-Kommandozeilenoption `-w` gibt zusätzliche Warnungen über einen unsauberen Programmierstil aus. Diese werden auf `stderr` ausgegeben; wenn Ihr `LogLevel` auf `warn` oder niedriger gestellt ist, landen sie deshalb im `ErrorLog`.

### 14.5.2 Formulardaten einlesen

Eine der wichtigsten Aufgaben von CGI-Skripten besteht in der Verarbeitung von Daten aus HTML-Formularen. Diese Daten werden je nach HTTP-Anfrage-Methode auf unterschiedliche Weise an das Skript übergeben:

- Wenn `GET` verwendet wurde, bilden die Daten einen Teil der URL, der durch ein Fragezeichen von der Pfadangabe getrennt ist. Dieser URL-Anhang wird durch die Umgebungsvariable `QUERY_STRING` abgebildet. Umgebungsvariablen stehen in Perl-Skripten über den Hash `%ENV` zur Verfügung.
- Bei `POST` befinden sich die Daten im Body der HTTP-Anfrage und können über die Standardeingabe eingelesen werden; die Umgebungsvariable `CONTENT_LENGTH` enthält die Anzahl der Datenbytes.

Die verwendete HTTP-Methode kann aus der Umgebungsvariablen `REQUEST_METHOD` gelesen werden. Der folgende Codeabschnitt liest sämtliche Formulardaten in beiden Fällen in die Variable `$formdata` ein:

```
my $formdata;
if $ENV{'REQUEST_METHOD'} eq 'POST' {
   read (STDIN, $formdata, $ENV{'CONTENT_LENGTH'});
} else {
   $formdata = $ENV{'QUERY_STRING'};
}
```

Als Nächstes müssen die URL-codierten Daten in einzelne `Name=Wert`-Paare zerlegt und decodiert werden. Dies funktioniert beispielsweise so:[3]

```
my @formfields = split (/&/, $formdata);
my %formvars;
```

3 In diesem Beispiel wird von dem klassischen Ampersand (`&`) als Feldtrennzeichen ausgegangen. Einige CGI-Implementierungen (insbesondere das im weiteren Verlauf erläuterte Perl-Modul `CGI.pm`) verwenden dagegen ein Semikolon (`;`) – Sie müssten das Skript also entsprechend anpassen.

```
foreach $formfield (@formfields) {
   my ($var, $val) = split (/=/, $formfield);
   $val =~ tr/+/ /;
   $val =~ s/%([\da-f]{2})/pack ("C", hex ($1))/egi;
   $formvars{$var} = $val;
}
```

Zunächst wird `$formdata` in die einzelnen `Name=Wert`-Gruppen aufgeteilt. Aus `kunde=G%FCnther+Schmitz&land=de` würden so die beiden Teile `kunde=G%FCnther+Schmitz` und `land=de`.

Nun wird nacheinander jedes der entstandenen Elemente bearbeitet. Zunächst werden sie noch einmal am =-Zeichen zerlegt. Der Teil hinter diesem Zeichen, der den Wert eines Feldes bildet, muss anschließend noch durch zwei Operationen in normalen Text umgewandelt werden:

- +-Zeichen werden mithilfe des einfachen Textersetzungsoperators `tr//` (der keine regulären Ausdrücke akzeptiert) durch Leerzeichen ersetzt.
- Die zweite, schwierigere Operation ersetzt Sonderzeichen, die durch ein Prozentzeichen und den hexadezimalen Zeichencode gekennzeichnet sind, durch das entsprechende Zeichen selbst. Dazu sucht der `s///`-Operator nach dem passenden Muster und verwendet im Ersetzungsteil die Funktion `pack()`, um das gewünschte Zeichen zu generieren. Der Modifikator `e` sorgt dafür, dass überhaupt ein Ausdruck anstelle einer einfachen Zeichenkette als Ersatz verwendet werden kann, während `g` für globale Ersetzung (und nicht nur eine) und `i` für die Nichtbeachtung von Groß- und Kleinschreibung im Suchmuster steht.

Zu guter Letzt wird der fertig bearbeitete Wert in den Hash `%formvars` gepackt, in dem die Feldnamen die Schlüssel und die Formulardaten die Werte bilden.

Diese Vorgehensweise zum Auslesen der Formulardaten ist nicht zu empfehlen und wird hier nur erwähnt, um die zugrunde liegende Technik zu erläutern. Sie ist ziemlich umständlich und obendrein fehlerträchtig. Nicht nur dies ist ein überzeugender Grund dafür, das im weiteren Verlauf erläuterte Perl-Modul `CGI.pm` zu verwenden.

## 14.6 Das Perl-Modul CGI.pm

Das Modul `CGI.pm` von Lincoln D. Stein, das schon seit langer Zeit zur Standarddistribution von Perl gehört, vereinfacht das Erstellen von CGI-Skripten erheblich. Zum einen gibt es eine standardisierte Methode zum Lesen von Formulardaten, zum anderen werden zahlreiche Funktionen geliefert, die das automatische Erstellen von HTTP-Headern und HTML-Elementen ermöglichen.

### 14.6.1 CGI.pm im Überblick

Zunächst einmal müssen Sie das Modul in Ihr Skript einbinden. Wie viele Perl-Module besitzt `CGI.pm` zwei verschiedene Schnittstellen, eine objektorientierte und eine prozedurale. Da die objektorientierte Verwendung heute üblicher ist, wird sie automatisch aktiviert, wenn Sie das Modul folgendermaßen importieren:

```
use CGI;
```

Um die prozedurale Schnittstelle freizuschalten, müssen Sie die `use`-Anweisung mit dem Tag `:standard` kombinieren:

```
use CGI qw/:standard/;
```

Wenn Sie diese Option wählen, können Sie die von `CGI.pm` exportierten Grundfunktionen prozedural (aber natürlich auch weiterhin objektorientiert) aufrufen – z.B. `param()` zum Auslesen von Formulardaten. Wenn Sie nicht den vollen Funktionsumfang von `CGI.pm` benötigen, können Sie anstelle von `:standard` einige spezielle Funktionslisten oder sogar einzelne Funktionen importieren. Umgekehrt können Sie über andere Import-Tags auch noch Funktionalität hinzufügen. Die entsprechenden Optionen werden im Verlauf dieses Kapitels noch beschrieben.

Während der Entwicklung eines CGI-Skripts sollten Sie auch noch folgende Zeile einfügen:

```
use CGI::Carp qw/fatalsToBrowser/;
```

Das Modul `CGI::Carp` ermöglicht ein ähnlich elegantes Abfangen und Verfolgen von Fehlern wie `Carp` für Konsolen-Perl-Skripte. Das Pragma `'fatalsToBrowser'` sendet Fehlermeldungen direkt über den Webserver an den Browser, mit dem das Skript getestet wird. Auf diese Weise erhalten Sie sofort Feedback, wenn etwas nicht richtig funktioniert, und brauchen nicht zuerst das `ErrorLog` zu öffnen oder gar das `ScriptLog` einzuschalten. Wenn Sie das Skript gründlich genug getestet haben, sollten Sie die Zeile allerdings wieder entfernen (oder einfach auskommentieren). Denn erstens verwirrt die Skriptfehlerausgabe gewöhnliche User Ihrer Website, und zweitens können Fehlermeldungen potenziellen Crackern wertvolle Hinweise über Schwachstellen des Skripts liefern.

Wenn Sie sich für die objektorientierte Variante entschieden haben, dann müssen Sie als Nächstes ein `CGI`-Objekt erzeugen. Beispiel:

```
my $cgi = CGI->new();
```

Anschließend werden alle `CGI.pm`-Funktionen als Methoden dieser Instanz aufgerufen, beispielsweise `$cgi->param()`. Das ist zwar einerseits etwas mehr Schreib-

arbeit, ermöglicht aber andererseits eine saubere Abgrenzung der CGI-spezifischen Anweisungen vom allgemeinen Perl-Code.

Im restlichen Teil dieses Abschnitts gilt der Name `$cgi` ohne weiteren Hinweis als Referenz auf ein so erzeugtes CGI-Objekt.

**CGI::Pretty**

Wenn Sie `CGI.pm` zur Erzeugung von HTML-Code mit den im Folgenden beschriebenen Methoden einsetzen, werden Sie feststellen, dass der HTML-Code einfach ohne Zeilenumbrüche erzeugt wird. Einem Browser ist das völlig egal, aber es stört manchmal beim Debugging des ausgegebenen Codes. Wenn Sie umbrochenes und eingerücktes HTML benötigen, können Sie das Modul `CGI::Pretty.pm` von Brian Paulsen verwenden. Es gehört wie `CGI.pm` zum Lieferumfang von Perl und besitzt identische Funktionen. Nur der Import und die eventuelle Instanziierung sehen natürlich anders aus:

```
use CGI::Pretty;
my $cgi = CGI::Pretty->new();
```

Leider wird die Pretty-Formatierung nur auf Standard-HTML-Elemente angewendet, nicht auf die speziellen Formularfunktionen.

`param()` ist die wichtigste Methode (beziehungsweise Funktion) des gesamten Moduls. Sie liest Formulardaten aus und ermittelt dabei automatisch, ob diese mit `GET` oder mit `POST` versandt wurden; Sie brauchen sich nicht mehr selbst um dieses Detail zu kümmern. Selbstverständlich übernimmt sie auch gleich das Unescaping (Umwandlung aus der URL-Codierung in Klartext). Mithilfe dieser Methode können Sie die Werte einzelner Formularfelder anhand ihres Namens lesen. Darüber hinaus liefert ein Aufruf von `param()` ohne Argumente ein Array mit den Namen aller übergebenen Formularfelder zurück. Angenommen, das aktuelle Skript wurde durch folgendes HTML-Formular aufgerufen:

```
<form action="test.pl" method="get">
  Name: <input type="text" name="user" /><br />
  Alter: <input type="text" name="age" /><br />
</form>
```

Mit `param ('user')` können Sie den eingegebenen Benutzernamen prozedural ermitteln, mit `$cgi->param ('user')` objektorientiert:

```
my $user = param('user');              # prozedural
my $user = $cgi->param('user');        # objektorientiert
```

Wenn Sie den nachfolgenden Code eingeben, wird die Liste der Feldnamen im Array `@cgivars` gespeichert:

```
my @cgivars = param();                 # prozedural
my @cgivars = $cgi->param();           # objektorientiert
```

Da der Unterschied zwischen prozeduraler und objektorientierter Syntax immer diesem Schema folgt, wird im Folgenden nur noch letztere Syntax verwendet.

Die Liste der Feldnamen können Sie beispielsweise benutzen, um sämtliche Formularfelder in einen Hash einzulesen, in dem die Feldnamen die Schlüssel und ihre Einträge die Werte darstellen. Das folgende Beispiel liest sie in einen Hash namens `%formfields` ein:

```
my @formnames = $cgi->param();          # Namen auslesen
my %formfields;                         # Hash deklarieren
foreach $name (@formnames) {
   $formfields{$name} = $cgi->param ($name);
}
```

Beachten Sie, dass eine bestimmte Sorte von Formularfeldern auf diese Weise nicht vollständig gelesen wird: Einige Feldsorten können mehrere Werte enthalten; diese stehen in einem Perl-Skript als Array zur Verfügung. Es handelt sich konkret um Checkbox-Gruppen und Auswahlmenüs mit Mehrfachauswahloption. Betrachten Sie als Beispiel das folgende Formular:

```
<form action="test.pl" method="get">
   Welche Programmiersprachen setzen Sie ein?
   <select name="plang" multiple="true">
     <option value="perl">Perl</option>
     <option value="java">Java</option>
     <option value="csharp">C#</option>
     <option value="cpp">C++</option>
     <option value="vb">Visual Basic</option>
  </select>
  <br />
  Was brauchen Sie zum Programmieren?
  <input type="checkbox" name="pneed" value="cof" />Kaffee
  <input type="checkbox" name="pneed" value="col" />Cola
  <input type="checkbox" name="pneed" value="piz" />Pizza
  <input type="checkbox" name="pneed" value="ice" />Eis
</form>
```

Bei beiden Fragen können mehrere Antworten zutreffen. Angenommen, jemand wählt aus dem Menü (mit gedrückter (Strg)-Taste) die beiden Sprachen Perl und Java aus und kreuzt die beiden Checkboxen Cola und Pizza an. Dann ist der Rückgabewert von `$cgi->param('plang')` und `$cgi->param('pneed')` kein Skalar, sondern jeweils ein Array. Sollten Sie die Werte, wie bereits gezeigt, dennoch als Skalare auslesen, steht Ihnen nur jeweils der erste zur Verfügung. Um beispielsweise die Programmiersprachen korrekt auszulesen, müssen Sie also Folgendes schreiben:

```
my @languages = $cgi->param('plang');
```

Ein weiterer praktischer Nutzen von `$cgi->param()` ohne Argumente ist die Überprüfung, ob überhaupt Formulardaten eingegeben wurden. Das ist besonders bei Formularen nützlich, die ihre Daten nicht an eine andere URL versenden, sondern an dasselbe Skript: Wenn noch keine Formulardaten übergeben wurden, handelt es sich offenbar um den ersten Aufruf des Skripts – eine Behandlung der Benutzereingaben kann nach einem Schema wie diesem entfallen:

```
# Wurden Parameter übergeben?
if ($cgi->param()) {
   # ... Behandlung der Eingaben
}
```

Die andere wichtige Aufgabe von `CGI.pm` besteht in der Bereitstellung zahlreicher Methoden, die den HTTP-Header und gängige HTML-Elemente ausgeben. Das folgende Beispiel definiert einen Standard-Header mit dem `Content-Type text/html`:

```
print $cgi->header;
```

Um die Werte von Header-Feldern anzupassen oder eigene zu setzen, wird das folgende Standardverfahren verwendet: Die einzelnen Felder werden als Hash-Referenz nach dem Muster `-header_name => "Wert"` angegeben – mit führendem Minuszeichen und Unterstrichen anstelle von Bindestrichen innerhalb der Feldnamen. Dieses Beispiel stellt anstelle von ISO-Latin-1 den für Türkisch verwendeten Zeichensatz `iso-8859-9` ein und setzt den zusätzlichen Header `Content-Language` auf den Wert `tr` (Türkisch):

```
print $cgi->header
     (-content_type="text/html; charset=iso-8859-9",
      -content_language="tr");
```

Sogar Cookies lassen sich auf diese Weise setzen. Die CGI-Methode `cookie()` dient dabei sowohl zum Lesen als auch zum Schreiben des Cookies. Das folgende Beispiel versucht zunächst, ein zuvor gesetztes Cookie mit dem Namen `lastvisit` aus der Client-Anfrage zu lesen, bevor es dieses Cookie mit Datum und Uhrzeit neu setzt:

```
# Cookie lastvisit lesen
my $readcookie = $cgi->cookie (-name => 'lastvisit');
# Datum und Uhrzeit
my $now = scalar localtime;
# Cookie lastvisit neu erzeugen
my $setcookie = $cgi->cookie (-name => 'lastvisit',
                              -value => $now,
                              -expires => '+7d',
                              -path => '/');
```

```
# Header erzeugen und das neu erstellte Cookie setzen
print $cgi->header (-cookie => $setcookie);
```

Den Standardrahmen eines HTML-Dokuments können Sie mithilfe der beiden Methoden `start_html` und `end_html` ausgeben. `start_html` nimmt optional einen Titel als Argument entgegen. Beispiel:

```
print $cgi->start_html ('Testseite');
```

Dies erzeugt den folgenden Teil des Ausgabedokuments:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US"><head>
<title>Testseite</title>
</head><body>
```

Wenn Sie anstelle von XHTML lieber klassisches HTML 4.0 verwenden möchten, müssen Sie das Modul wie folgt mit dem Pragma `-no_xhtml` importieren:

```
use CGI qw/-no_xhtml/;            # objektorientiert
```

beziehungsweise

```
use CGI qw/:standard -no_xhtml/;  # prozedural
```

`end_html` schließt den Rahmen des Dokuments ab und wird folgendermaßen aufgerufen:

```
print $cgi->end_html;
```

Dies erzeugt folgende Ausgabe:

```
</body></html>
```

Zur Erzeugung von HTML-Tags gibt es zwei unterschiedliche Ansätze, je nachdem, ob diese nur einfachen Text (beziehungsweise gar nichts) oder zahlreiche weitere Elemente umschließen.

Zunächst einmal existiert für jedes HTML-Tag eine Methode (beziehungsweise Funktion), deren Name dem Tag entspricht. Beispielsweise wird eine `<h1>`-Überschrift folgendermaßen definiert:

```
print $cgi->h1 ('Haupt&uuml;berschrift');
```

Noch einfacher funktioniert natürlich ein völlig leeres Tag wie der Zeilenumbruch `<br />`:

```
print $cgi->br();
```

Auch bei HTML-Tags werden Attribute nach dem oben gezeigten Schema verwendet, wenn auch mit einem kleinen Unterschied: Um Attribute von Inhalten eines Tags unterscheiden zu können, stehen die Attribute in einem anonymen Hash, der von geschweiften Klammern `{ ... }` umschlossen wird. Das folgende Beispiel erzeugt ein `<img>`-Tag zum Einbetten des Bildes `test.gif` mit der Breite 200 Pixel, der Höhe 100 Pixel und dem Alternativtext »Testbild«:

```
print $cgi->img ({-src => 'test.gif', -width => 200,
                  -height => 200, -alt => 'Testbild'});
```

Enthält ein Tag Textinhalt, wird dieser als zweiter Parameter nach der Attributliste angegeben. Beispielsweise erzeugt diese Anweisung eine `<h1>`-Überschrift, die (auf veraltete Weise) zentriert dargestellt wird:

```
print $cgi->h1 ({-align => 'center'}, 'Hallo Welt!');
```

Eine etwas größere Herausforderung stellen ineinander verschachtelte HTML-Tags dar. Das folgende Beispiel erstellt eine Tabelle mit zwei Zeilen, die je zwei Zellen enthalten:

```
print ($cgi->table({-border => '2', -cellpadding => '4'},
   $cgi->Tr(
      $cgi->td('Aktuelles 2.2-Release:'),
      $cgi->td('2.2.10'),
   ), $cgi->Tr(
      $cgi->td('Aktuelles 2.0-Release:'),
      $cgi->td('2.0.63')));
```

Durch eine vernünftige Zeilenumbruch- und Einrückungspraxis, wie sie hier gezeigt wird, können Sie dennoch einigermaßen den Überblick behalten. Beachten Sie übrigens, dass `Tr()` großgeschrieben werden *muss*; der Name würde sonst mit dem Transliterationsoperator `tr//` in Konflikt geraten. Dasselbe gilt übrigens für `Link()` und `Select()`.

Wie bereits erwähnt, gibt es für solche verschachtelten Tags auch noch eine andere Lösung: Sie können sie in eine `start_*`- und eine `end_*`-Methode aufteilen und erhalten auf diese Weise Folgendes:

```
print $cgi->start_table({-border => '2',
                         -cellpadding => '4'});
   print $cgi->start_Tr;
      print $cgi->td('Aktuelles 2.2-Release:');
      print $cgi->td('2.2.10');
   print $cgi->end_Tr;
   print $cgi->start_Tr;
```

```
      print $cgi->td('Aktuelles 2.0-Release:');
      print $cgi->td('2.0.63');
   print $cgi->end_Tr;
print $cgi->end_table;
```

Wie Sie sehen, wird `Tr()` in `start_Tr()` und `end_Tr()` aus Konsistenzgründen weiterhin großgeschrieben, obwohl hier natürlich keine Verwechslungsgefahr mehr besteht.

Auf ähnlich einfache Weise lassen sich mittels `CGI.pm` HTML-Formulare erzeugen. Neben den normalen HTML-Tags für Formulare und ihre Elemente, die durch die soeben gezeigten üblichen Methoden wie `start_form()`, `end_form()` oder `input()` erzeugt werden, gibt es etliche praktische Kurzfassungen: Beispielsweise erzeugt `textfield()` ein Texteingabefeld, für das Sie normalerweise `input({-type => 'text' ...})` schreiben müssten; ein anderes Beispiel ist die Methode `radio_group()`, die gleich eine ganze Gruppe von Radio-Buttons bereitstellt. Genaueres erfahren Sie in den Beispielen und der Referenz im Folgenden.

### 14.6.2 Beispiel: Ein kleiner Taschenrechner

Das kleine Beispiel in Listing 14.2 stellt ein Formular zur Eingabe zweier Zahlen und vier Radio-Buttons zur Auswahl einer der vier Grundrechenarten zur Verfügung. Ein Klick auf den des BERECHNEN-Button sendet die Formulardaten an das Skript selbst, das nun zusätzlich zur erneuten Anzeige des Formulars auch das Rechenergebnis (oder eine Fehlermeldung bei ungültigen Eingaben oder einer Division durch 0) anzeigt.

```
#!/usr/bin/perl -w
use strict;
use CGI;
use CGI::Carp qw/fatalsToBrowser/;

# CGI-Objekt erzeugen
my $cgi = CGI->new();

print $cgi->header;
print $cgi->start_html ("CGI-Rechner");
print $cgi->h1 ("Kleiner CGI-Rechner");

# Wurden Parameter übergeben?
if ($cgi->param()) {
   my $zahl1 = $cgi->param ('z1');
   my $zahl2 = $cgi->param ('z2');
   my $op = $cgi->param ('op');
```

```
   if (!test_num ($zahl1) || !test_num ($zahl2)) {
      print "Fehler: Ung&uuml;tiger Operand.";
   } elsif ($op eq '/' && $zahl2 == 0) {
      print "Fehler: Division durch 0.";
   } else {
      print "Ergebnis: $zahl1 $op $zahl2 = "
            .eval ("$zahl1 $op $zahl2");
   }
}
print $cgi->start_form (-method => "get");
print "1. Zahl: ";
print $cgi->textfield (-name => "z1");
print $cgi->br();
print "2. Zahl: ";
print $cgi->textfield (-name => "z2");
print $cgi->br();
print $cgi->br();
print "Operation: ";
print $cgi->radio_group (-name => 'op',
                         -values => ['+', '-', '*', '/'],
                         -default => '+');
print $cgi->br();
print $cgi->submit (-value => "Berechnen");
print $cgi->end_form;
print $cgi->end_html;

sub test_num
{
   my $num = shift;
   if ($num =~ /^\-?\d*\.?\d*$/) {
      return 1;
   }
   return 0;
}
```

**Listing 14.2** Ein kleiner CGI-Taschenrechner

Wenn Sie das CGI-Skript in ein CGI-fähiges Verzeichnis Ihrer Apache-Installation kopieren, die entsprechende URL (z.B. *www.mynet.de/cgi-bin/calc.pl*) in Ihren Browser eingeben und anschließend eine Berechnung ausführen, sollte das Ganze etwa so aussehen wie in Abbildung 14.2. Wie Sie sehen, ist ein durch `CGI.pm` erzeugtes Formular automatisch »sticky«, das heißt, die vor dem Absenden eingegebenen Formularwerte werden automatisch wieder in die entsprechenden Felder eingetragen.

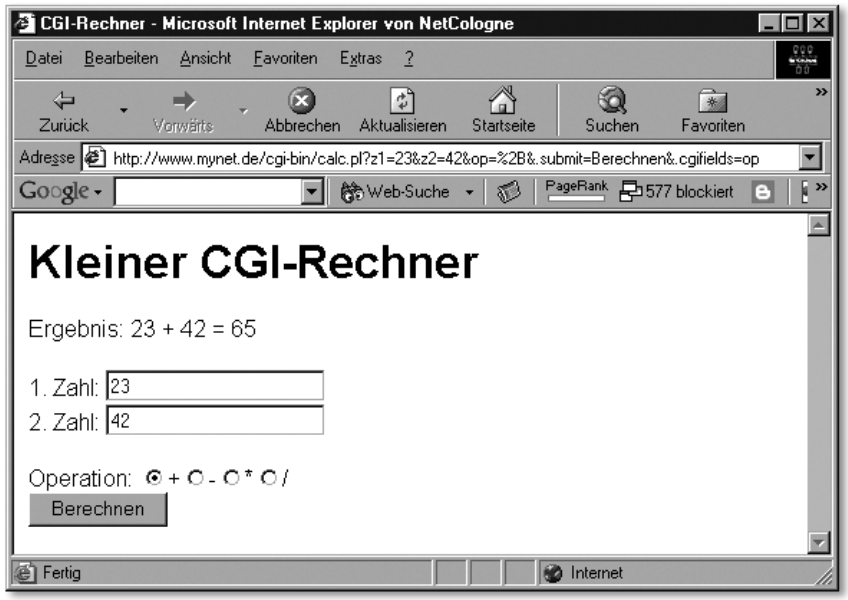



**Abbildung 14.2** Beispielberechnung des »CGI.pm«-Taschenrechners

Die meisten Informationen über die Funktionsweise des Beispiels haben Sie bereits weiter oben in der Einführung erhalten. Zusätzlich sind noch folgende Punkte anzumerken:

- Das Skript ruft den Perl-Interpreter mit der Option `-w` (Warnings) auf. Dies erzeugt zusätzliche Warnungen, die Auskunft über einen unsauberen Programmierstil oder mögliche Fehlerquellen geben. Bei gleichzeitiger Verwendung von `CGI::Carp` werden auch diese Warnungen im Browser angezeigt. Die Verwendung der Option `-w` ist grundsätzlich zu empfehlen.
- Mithilfe von `use strict` wird die strenge Typüberprüfung eingeschaltet. Diese Option verlangt, dass sämtliche Variablen vor der ersten Verwendung mit Gültigkeitsbereichen wie `my`, `our` oder `local` deklariert werden. In längeren Skripten verhindert dies z.B. das Auftreten logischer Ausführungsfehler durch falsch geschriebene Variablennamen – eine Fehlersorte, die sich sonst nur in sehr mühseligen Verfahren finden lässt.
- Zur Berechnung des Ergebnisses der Rechenoperation wird die Funktion `eval()` verwendet. Sie betrachtet den übergebenen String als Ausdruck und liefert dessen Ergebnis zurück. Dies erspart langwieriges Hantieren mit `if-else`-Konstrukten. Voraussetzung ist natürlich, dass das Formular als Werte für die Rechenoperationen (die Radio-Button-Gruppe `op`) die eigentlichen Operatoren übergibt.

- Wie Sie wahrscheinlich wissen, benötigen HTML-Formulare normalerweise das Attribut `action="URL"`, das die URL angibt, an die die Formulardaten verschickt werden sollen. Die Methode `start_form()` in diesem Beispiel besitzt dieses Attribut allerdings nicht. Das liegt daran, dass `CGI.pm` aus Bequemlichkeitsgründen automatisch die URL des aktuellen Skripts selbst einträgt, wenn Sie den Parameter weglassen.
- Im Formular wird die Methode `radio_group()` verwendet, um die Radio-Buttons zur Auswahl der Rechenoperation zu erzeugen:

```
print $cgi->radio_group (-name => 'op',
                         -values => ['+', '-', '*', '/'],
                         -default => '+');
```

Der Parameter `-name` ist der gemeinsame Name aller vier Buttons. Das anonyme Array, das den Wert von `-values` bildet, enthält die Werte, die übermittelt werden, wenn der jeweilige Button ausgewählt wird – und legt damit auch gleich die Anzahl der Radio-Buttons in der Gruppe fest. Das optionale Argument `-default` gibt den Wert des Buttons an, der zu Beginn automatisch ausgewählt ist. Der erzeugte HTML-Code sieht mit diesen Optionen so aus:

```
<input type="radio" name="op" value="+" checked="checked" />+
<input type="radio" name="op" value="-" />-<input type="radio"
name="op" value="*" />*
<input type="radio" name="op" value="/" />/<br />
<input type="submit" name=".submit" value="Berechnen" />
<input type="hidden" name=".cgifields" value="op" />
```

Wenn die Radio-Buttons anstelle der normalerweise verwendeten `-value`-Felder eine andere Beschriftung erhalten sollen, können Sie den Parameter `-labels` hinzufügen. Es handelt sich um einen anonymen Hash (oder eine Hash-Referenz), wobei die Feldwerte die Schlüssel und die gewünschten Beschriftungen die Elemente bilden. Mit einem anonymen Hash sieht die Anweisung für das Rechenoperationsbeispiel so aus:

```
print $cgi->radio_group (-name => 'op',
                         -values => ['+', '-', '*', '/'],
                         -labels => {'+' => 'Addieren',
                                     '-' => 'Subtrahieren',
                                     '*' => 'Multiplizieren',
                                     '/' => 'Dividieren'},
                         -default => '+');
```

Eigentlich besteht der Sinn des speziellen Trennsymbols => in einem Hash (neben der Verdeutlichung der Beziehung zwischen Schlüsseln und Werten) darin, dass die Anführungszeichen um die Schlüssel weggelassen werden können. Allerdings funktioniert das nicht, wenn die Schlüssel – wie hier – Symbole mit einer programmiertechnischen Bedeutung sind. Alternativ können Sie zuerst eine Hash-Variable erzeugen und dann als Wert von `-labels` eine Referenz darauf angeben:

```
my %labels = ('+' => 'Addieren',
              '-' => 'Subtrahieren',
              '*' => 'Multiplizieren',
              '/' => 'Dividieren');
print $cgi->radio_group (-name => 'op',
                         -values => ['+', '-', '*', '/'],
                         -labels => \%labels,
                         -default => '+');
```

- Die Subroutine `test_num` verwendet den folgenden regulären Ausdruck, um zu ermitteln, ob das übergebene Argument eine Zahl ist:

  ```
  /^\-?\d*\.?\d*$/
  ```

  Dieses Muster lässt sich folgendermaßen zerlegen, um es zu erläutern:

  - `^` Textanfang
  - `\-?` ein Minus (–) oder auch keins
  - `\d*` beliebig viele Ziffern
  - `\.?` ein Punkt (.) oder auch keiner
  - `\d*` beliebig viele Ziffern
  - `$` Textende

  Eine kurze Übersicht über reguläre Ausdrücke finden Sie in Anhang G.

### 14.6.3 CGI.pm-Kurzreferenz

In diesem Abschnitt finden Sie eine Referenz der wichtigsten Bestandteile von `CGI.pm`: Import-Tags und -Pragmata sowie CGI- und Formularmethoden. Eine Übersicht über sämtliche HTML-Erzeugungsfunktionen ist dagegen nicht dabei, weil sie alle nach demselben, bereits besprochenen Schema funktionieren und den bekannten HTML-Tags entsprechen. Diese können Sie bequem online im Perl-Teil von SelfHTML (*http://de.selfhtml.org/perl/module/cgi.htm*) nachschlagen.

### Importreferenz

Wie bereits erwähnt, gibt es eine Reihe spezieller Tags und Pragmata für den Import des CGI-Moduls. Diese werden hier kurz vorgestellt. Zunächst finden Sie in Tabelle 14.2 die verschiedenen Import-Tags für unterschiedliche Funktionslisten. Daneben können Sie auch immer einzelne Funktionen angeben, indem Sie einfach ihre Namen in die Importliste schreiben.

| Tag | Beschreibung |
|---|---|
| `:cgi` | Importiert CGI-relevante Methoden; vor allem `param()`. |
| `:form` | Importiert die speziellen Formularmethoden wie `radio_group()` oder `textfield()`. |
| `:html2` | Importiert Methoden für HTML-2.0-Tags (grundlegender Satz von HTML-Tags). |
| `:html3` | Importiert zusätzliche Methoden für HTML-3.2-Tags. |
| `:html4` | Importiert zusätzliche Methoden für HTML-4.0-Tags. |
| `:netscape` | Importiert zusätzliche Methoden für Netscape-HTML-Erweiterungen. |
| `:html` | Importiert alle HTML-Methoden, also `:html2`, `:html3`, `:html4` und `:netscape`. |
| `:standard` | Importiert Standardfunktionen sowie alle HTML-, CGI- und Formularmethoden. |
| `:all` | Importiert alle Methoden. |

**Tabelle 14.2** Import-Tags für die prozedurale »CGI.pm«-Programmierung

Beachten Sie, dass diese Funktionslisten nur für die prozedurale `CGI.pm`-Programmierung wichtig sind. Wenn Sie mit der objektorientierten Syntax auf das Modul zugreifen, stehen Ihnen über Ihre CGI-Instanz auf jeden Fall alle Methoden zur Verfügung.

In Tabelle 14.3 sehen Sie die diversen Pragmata, die jeweils ein spezielles Verhalten der `CGI.pm`-Methoden zur Folge haben. Im Gegensatz zu den Funktionslisten sind sie für beide Ansätze der CGI-Programmierung relevant.

| Pragma | Beschreibung |
|---|---|
| `-any` | Akzeptiert beliebige Formulierungen, die dem `CGI.pm`-Syntaxschema folgen, als auszugebende HTML-Tags (praktisch, falls browserspezifische Erweiterungen verwendet werden sollen, die normalerweise nicht integriert sind). |
| `-compile` | Kompiliert die importierten Methoden. Nützlich, wenn CGI-Skripte unter FastCGI oder `mod_perl` mehrfach ausgeführt werden. |

**Tabelle 14.3** Import-Pragmata, die das Verhalten von »CGI.pm« beeinflussen

| Pragma | Beschreibung |
|---|---|
| -nosticky | Formulare sollen nicht »sticky« sein: Daten werden nach dem Versand nicht mehr automatisch in die Felder eingetragen. |
| -no_undef_params | Verhindert HTML-Attribute ohne Wert wie readonly bei Formularfeldern. |
| -no_xhtml | Generiert klassisches HTML anstelle des normalerweise erzeugten XHTML. |
| -nph | Erzeugt »No Parsed Header«-CGIs (NPH), bei denen der Header nicht durch den Server manipuliert wird – Apache sendet noch nicht einmal die Statuszeile. |
| -newstyle_urls | Verwendet Semikola als Feldertrennzeichen in URL-codierten Formulardaten (altes Pragma; heute ohnehin Standard). |
| -oldstyle_urls | Verwendet Ampersands (&) anstelle von Semikola als Feldertrennzeichen. |
| -autoload | Gibt unbekannte Funktionen aus Skripten automatisch an CGI.pm weiter, sodass kein Funktionsimport nötig ist (verbraucht weniger Speicher). |
| -no_debug | Deaktiviert das alternative Einlesen von Parametern über Kommandozeilenparameter oder die Standardeingabe. |
| -debug | Aktiviert das alternative Einlesen von Parametern über Kommandozeilenparameter oder die Standardeingabe, um das Skript offline zu testen. |
| -private_tempfiles | Erschwert den sicherheitsgefährdenden Zugriff auf Upload-Dateien über das lokale Dateisystem. |

**Tabelle 14.3** Import-Pragmata, die das Verhalten von »CGI.pm« beeinflussen (Forts.)

### Kurzreferenz der CGI-Methoden

Es folgt eine Übersicht über die wichtigsten CGI-Methoden des Moduls. Einige von ihnen haben Sie bereits weiter oben in ihrer praktischen Anwendung kennengelernt; hier finden Sie systematisch alle wichtigen Parameter dazu.

- header(): Diese Methode sendet einen gültigen HTTP-Header. Standardmäßig wird nur das Feld Content-Type gesetzt, normalerweise mit dem MIME-Type text/html. Darüber hinaus können Sie beliebige weitere Header setzen. Dazu müssen Sie den Namen des Headers klein und mit einem führenden Minuszeichen (-) schreiben; Trennstriche innerhalb des Feldnamens werden durch Unterstriche ersetzt. Aus Content-Type wird also beispielsweise -content_type. Übrigens fügt Apache zu ausgegebenen Skripten standardmäßig noch die Header Date und Server hinzu.

  Neben den in Kapitel 2, »Funktionsweise von Webservern«, besprochenen HTTP-Antwort-Headern können Sie zusätzlich den speziellen Header-Schlüs-

sel `-status` verwenden, um den (numerischen) Status der Antwort zu setzen. Bei einem korrekt ausgeführten CGI-Skript wird natürlich standardmäßig `200 OK` gesendet, was in der Regel auch erwünscht ist. Einige Skripte erzeugen jedoch gar keine eigene Ausgabe, sondern führen nach getaner Arbeit eine Weiterleitung durch. Zu diesem Zweck könnten Sie den Status `303 See Other` und einen `-location`-Header mit dem neuen Status senden:

```
print $cgi->header (-status => 303,
                    -location => 'other.pl');
```

- `redirect()`: Einfacher lässt sich die Weiterleitung mit der speziellen Methode `redirect()` realisieren. Sie benötigt nur ein einziges Argument, nämlich die (absolute oder relative) URL des Weiterleitungsziels. Beispiel:

  ```
  print $cgi->redirect ('other.pl');
  ```

- `cookie()`: Die Methode `cookie()` dient sowohl zum Lesen als auch zum Erzeugen von Cookies. Das Lesen funktioniert, indem Sie den Namen des gewünschten Cookies angeben, und muss vor dem Senden des Headers erledigt werden:

  ```
  my $readcookie = $cgi->cookie (-name => 'Cookie-Name');
  ```

  Um ein neues Cookie zu erzeugen, wird die Methode ebenfalls nach diesem Schema aufgerufen. Sie müssen mindestens ein `-value`-Feld hinzufügen, damit das Cookie einen sinnvollen Zweck erfüllt:

  ```
  my $setcookie = $cgi->cookie (-name  => 'Cookie-Name',
                                -value => 'Cookie-Wert');
  ```

  Alle Attribute, die ein Cookie erhalten kann, finden Sie in Tabelle 14.4.

| Cookie-Attribut | Beschreibung |
|---|---|
| `-name` | Name des Cookies (beliebig) |
| `-value` | Wert des Cookies (beliebige Zeichenkette) |
| `-path` | Pfad, für den das Cookie gelten soll. Standardwert: `"/"` – das bereits gesetzte Cookie wird an alle CGI-Skripte auf dem Server versendet. |
| `-domain` | Domain, für die das Cookie gelten soll. Besonders wichtig bei namensbasierten virtuellen Hosts, damit das Cookie nicht beim falschen Empfänger unter derselben IP-Adresse landet. |
| `-expires` | Verfallsdatum. Meist relativ im Format `'+nEinheit'`, wobei folgende Einheiten möglich sind: Sekunden (`s`), Minuten (`m`), Stunden (`h`), Tage (`d`), Monate (`M`), Jahre (`y`). `'+3d'` bedeutet z. B. drei Tage Gültigkeit. Fehlt der Wert, gilt das Cookie nur für die aktuelle Browser-Session. |

**Tabelle 14.4** Attribute der »CGI.pm«-Methode »cookie( )«

| Cookie-Attribut | Beschreibung |
|---|---|
| -secure | Falls true, wird das Cookie nur gesetzt, wenn eine SSL-Verbindung besteht. |

**Tabelle 14.4** Attribute der »CGI.pm«-Methode »cookie( )« (Forts.)

Denken Sie daran, dass Sie das Cookie nach der Erzeugung über den Header ausgeben müssen, damit es tatsächlich gesetzt wird:

```
print $cgi->header (-cookie => $setcookie);
```

- start_html(): Die bereits erwähnte Methode start_html() erzeugt den Beginn des HTML-Dokuments bis zum <body>-Tag inklusive. Wenn Sie lediglich ein einziges Argument angeben, wird es als Titel des Dokuments (<title> ... </title>) interpretiert. Alternativ können Sie nach dem bewährten Schema zahlreiche benannte Argumente angeben; die wichtigsten sehen Sie in Tabelle 14.5.[4]

| Argument | Beschreibung | Beispiele |
|---|---|---|
| -title | die benannte Variante des Dokumenttitels | -title => 'Testseite' |
| -author | Setzt ein link-Tag mit der E-Mail-Adresse des Autors. | -author => 'editor@mynet.de' |
| -base | Setzt das Tag <base href="aktuelle URL"> – der Wert des Parameters ist egal. | -base => 'true' |
| -target | Vorgabeziel für Hyperlinks (praktisch bei Sites mit Frames) | -target => 'rightframe' |
| -style | Stylesheets;<br>String: <style>-Block<br>Hash: externe CSS-Datei | -style =><br>'h1 {color: #0000FF}'<br><br>-style =><br>{src => 'normal.css'} |
| -meta | Setzt Meta-Tags mit den in einem Hash angegebenen name/value-Paaren. | -meta =><br>{description<br>=> 'CGI-Test',<br>keywords<br>=> 'Apache',<br>'CGI, Test'} |

**Tabelle 14.5** Argumente der »CGI.pm«-Methode »start_html( )«

4 Beachten Sie, dass externe CSS- und JavaScript-Dateien nicht in einem *cgi-bin*-Verzeichnis liegen dürfen, weil Apache versuchen würde, sie vor der Auslieferung auszuführen.

| Argument | Beschreibung | Beispiele |
|---|---|---|
| -script | JavaScript;<br>String: Code, der zwischen <script> und </script> im Head stehen soll<br>Hash: JavaScript aus externer .js-Datei | -script =><br>'alert (\'Hallo,<br>Welt!\');'<br><br>-script =><br>{-language =><br>'JavaScript',<br>-src =><br>'myscript.js'} |
| -onLoad | JavaScript-Event-Handler im <body>-Tag | -onLoad =><br>'alert<br>(\'Startskript\');' |
| -BGCOLOR<br>-TEXT<br>usw. | veraltete Grundfarbangaben für das <body>-Tag (besser -style verwenden) | -BGCOLOR => '#FF0000' |

**Tabelle 14.5** Argumente der »CGI.pm«-Methode »start_html( )« (Forts.)

- end_html(): Diese Methode hat keine Parameter. Sie erzeugt den festgelegten Text </body></html> für das Dokumentende.
- url(): Mithilfe dieser Methode kann das Skript seine eigene URL ermitteln; einige Parameter (jeweils mit dem Wert 1 oder einem anderen »true«-Wert) ermöglichen genauere Einstellungen. Hier die wichtigsten: -absolute gibt die absolute URL zurück, -relative dagegen die relative. Mit -query erhalten Sie zusätzlich die angehängten Parameter.
- self_url(): Diese parameterlose Methode erzeugt die URL des aktuellen Skripts mit allen zurzeit gesetzten Formularfeldern. Dies ist vor allem für seiteninterne Hyperlinks nützlich. Beispiel:

```
my $self = $cgi->self_url;
print $cgi->a({-href => "$self#unten"}, 'Siehe unten'});
```

### Kurzreferenz der Formularmethoden

Im Folgenden finden Sie eine Übersicht über die wichtigsten Methoden zur Erzeugung von HTML-Formularen und ihren Elementen. Bitte beachten Sie, dass hier nur diejenigen Funktionen berücksichtigt werden, die nicht der normalen HTML-Syntax entsprechen. Die Entsprechungen der HTML-Tags stehen natürlich ebenfalls zur Verfügung, und nichts (außer der Tatsache, dass es unnötige Arbeit ist) hält Sie davon ab, eine Gruppe von Radio-Buttons folgendermaßen zu erzeugen:

```
print $cgi->h3 ('Was sind Sie?');
print $cgi->input ({-type => 'radio', -name => 'prog',
                    -value => 'yes'});
print $cgi->b ('Programmierer');
print ' oder ';
print $cgi->input ({-type => 'radio', -name => 'prog',
                    -value => 'no'});
print $cgi->b ('Nichtprogrammierer');
```

- `start_form()`: Durch diese Methode wird ein HTML-Formular eingeleitet. Wenn Sie keine Parameter angeben, wird als Action-URL das aktuelle Skript selbst, als Methode `POST` und als Encoding-Type `application/x-www-form-urlencoded` eingetragen. Bei Bedarf können Sie diese Angaben durch die benannten Parameter `-action`, `-method` und `-enctype` ändern. Für die JavaScript-Programmierung ist es obendrein nützlich, das Formular zu benennen; dies erledigt der zusätzliche Parameter `-name`. Beispiel:

  ```
  print $cgi->start_form (-action  => 'auswert.pl',
                          -method  => 'GET');
  ```

- `end_form()`: Diese Methode gibt einfach `</form>` zurück.
- `textfield()`: Erzeugt ein Texteingabefeld in einem Formular. Die möglichen Parameter sind `-name` (der Feldname), `-default` (Vorgabewert), `-size` (sichtbare Zeichenzahl) und `-maxlength` (maximale Zeichenanzahl). Hier ein Beispiel:

  ```
  print "Ihre E-Mail-Adresse: ";
  print $cgi->textfield (-name =>      'mail',
                         -size =>       40,
                         -maxlength => 80);
  ```

- `password_field()`: Diese Methode besitzt dieselben Parameter wie `textfield()`. Der Unterschied besteht darin, dass die Eingabe als `***` dargestellt wird. Beispiel:

  ```
  print "Ihr Kundenkennwort: ";
  print $cgi->password_field (-name =>       'pass',
                              -size =>       20,
                              -maxlength => 8);
  ```

- `filefield()`: Mithilfe dieser Methode wird ein Datei-Upload-Feld erzeugt; die Parameter sind wiederum identisch mit `textfield()`. Es ermöglicht Benutzern, zusammen mit dem Formular eine lokale Datei zu versenden. Beachten Sie, dass der Encoding-Type des Formulars dafür auf `multipart/form-data`

gesetzt werden muss. Der eigentliche Upload funktioniert immer nach folgendem Schema:[5]

```
# Das Dateihandle aus dem Formularfeld lesen
my $file = $cgi->param ('dateifeld');
# Die Datei einlesen
while (my $line = <$file>) {
   chomp $line;
   # Die eingelesene Zeile beliebig verarbeiten
   # ...
}
```

- `popup_menu()`: Diese Methode erzeugt ein Auswahlmenü. Die notwendigen Parameter geben den Feldnamen (`-name`) und ein Array mit den Werten (`-values`) an. Zusätzlich können Sie noch einen Standardwert (`-default`) sowie einen Hash mit von den Werten abweichenden Beschriftungen (`-labels`) angeben. Beispiel:

```
print $cgi->h3 ("Welchen Webserver benutzen Sie?");
print $cgi->popup_menu (-name => 'webserver',
                        -value => ['ap', 'ms', 'tc', xx'],
                        -default => 'ap',
                        -labels => {ap => 'Apache',
                                    ms => 'Microsoft IIS',
                                    tc => 'Tomcat',
                                    xx => 'Sonstige'});
```

- `radio_group()` wurde weiter oben bereits im Beispiel verwendet. Die Methode erzeugt eine Gruppe von Radio-Buttons. Es können alle Parameter von `popup_menu()` übernommen werden; die Bedeutung ist identisch. Zusätzlich können Sie noch das Argument `-linebreak => 'true'` verwenden, wenn Ihre Radio-Buttons untereinander und nicht nebeneinander stehen sollen. Mit `-rows => $zeilenzahl` und `-cols => $spaltenzahl` lassen sich die Buttons sogar automatisch in einer Tabelle anordnen. Das folgende Beispiel verwendet drei Zeilen mit je zwei Spalten:

```
print $cgi->h3 ("Unter welchem OS l&auml;uft Ihr Webserver?");
print $cgi->radio_group (-name => 'os',
                         -value => ['lx', 'bsd', 'mox',
                                    'ux', 'w32', 'xx'],
                         -default => 'lx',
```

5 Binärdateien sollten Sie gegebenenfalls mit `read()` und nicht mit dem Zeilenoperator `<>` einlesen, um Ärger mit der Zeilenumbruchlogik zu vermeiden. Ein Beispiel für `read()` zeigt der kleine Perl-Webserver in Kapitel 2, »Funktionsweise von Webservern«.

```
                -labels => {lx =>  'Linux',
                            bsd => '*BSD',
                            mox => 'Mac OS X',
                            ux =>  'Anderes UNIX',
                            w32 => 'Windows',
                            xx =>  'Sonstiges'},
                -rows => 3,
                -cols => 2);
```

- `checkbox_group()` erzeugt eine Gruppe von Checkboxen. Die Argumente sind mit `radio_group()` identisch – bis auf eine kleine, aber wichtige Ausnahme: Da Benutzer mehrere Checkboxen einer Gruppe auswählen können, ist es auch möglich, mehrere vorab anzukreuzen. Aus diesem Grund ist der Wert von `-default` hier kein String, sondern ein Array.
- `submit()` erstellt das wichtigste Formularelement: den Absendebutton. Die Argumente sind `-name` (nur wichtig, wenn Sie mehrere SUBMIT-Buttons haben und unterscheiden möchten, *welcher* angeklickt wurde) sowie `-value`, die Beschriftung. Beispiel:

  ```
  print $cgi->submit (-name =>  'standardversand',
                      -value => 'Abschicken');
  ```

- `reset()` erzeugt zu guter Letzt einen RESET-Button, der das Formular auf den Zustand beim Laden der Seite zurücksetzt. Da die Beschriftung das einzige Argument ist, können Sie es einfach ohne Parameternamen angeben. Beispiel:

  ```
  print $cgi->reset ('Zuruecksetzen');
  ```

## 14.7 Zusammenfassung

CGI ist zwar eine etwas in die Jahre gekommene Technologie, bildet aber noch immer die Grundlage vieler dynamischer Webserver-Anwendungen. Die Schnittstellendefinition selbst ist recht einfach: Als Eingabe steht einem CGI-Skript der URL-Anhang einer `GET`-Anfrage beziehungsweise der Body-Inhalt einer `POST`-Anfrage zur Verfügung. Alles, was das Programm über die Standardausgabe ausgibt, leitet Apache als Antwort an den Client weiter.

Apache kann auf vielfältige Weise für die Ausführung von CGI-Skripten konfiguriert werden. Standardmäßig wird ein *cgi-bin*-Verzeichnis definiert, dessen Inhalte über eine `ScriptAlias`-Direktive als ausführbar gekennzeichnet werden. Dies ist eine relativ sichere Methode, weil sich das CGI-Verzeichnis nicht unterhalb der `DocumentRoot` befindet. Alternativ können Sie aber auch Verzeichnisse innerhalb der Website CGI-fähig machen, indem Sie die Option `ExecCGI` setzen.

Wenn diese Option aktiv ist, lassen sich bestimmte Dateien mittels `AddHandler` oder `AddType` zu CGI-Skripten erklären.

Ein praktisches Hilfsmittel für die CGI-Programmierung sind Umgebungsvariablen: Sie liefern einem CGI-Skript zahlreiche Informationen über den Client, den Server und die HTTP-Anfrage. Apache enthält standardmäßig die beiden Module `mod_env` und `mod_setenvif` zum Setzen und Manipulieren zusätzlicher Variablen.

So gut wie jede Programmiersprache kann verwendet werden, um CGI-Programme zu erstellen; im nächsten Kapitel werden Sie beispielsweise sehen, dass sich auch PHP nicht nur als Apache-Modul, sondern auch per CGI einbinden lässt. Die wichtigste Programmiersprache für CGI-Skripte ist jedoch traditionell Perl. Das liegt nicht nur daran, dass es für diese Sprache die meisten Beispiele und Dokumentationen gibt, sondern vor allem auch an der Flexibilität und den überragenden Fähigkeiten zur Textmanipulation, die Perl zu bieten hat. Sie können die CGI-Funktionalität entweder manuell erzeugen, oder – was wesentlich empfehlenswerter ist – das hervorragende Perl-Modul `CGI.pm` verwenden: Es bietet zahlreiche praktische Methoden zur Parameterverarbeitung sowie zur HTML- und Formularerzeugung.

*»Kennst du viele Sprachen – hast du viele Schlüssel für ein Schloss.«*
*– Voltaire*

# 15 Technologien zur Webprogrammierung

Dass es mehr Möglichkeiten als CGI gibt, wenn es um Webprogrammierung geht, dürfte sich inzwischen herumgesprochen haben. In diesem Kapitel erfahren Sie genau, wie Sie einige der wichtigsten Programmiersprachen und Technologien installieren und Apache für die Zusammenarbeit mit ihnen konfigurieren können. Im Einzelnen geht es um folgende Sprachen und Produkte:

- **PHP**
  Diese auf Webanwendungen spezialisierte Skriptsprache ist laut NetCraft das beliebteste aller optionalen Apache-Module. Der besondere Vorteil besteht darin, dass Sie PHP-Anweisungen einfach an den passenden Stellen in den statischen HTML-Code einbetten können. Darüber hinaus ist PHP besonders beliebt, weil es hervorragend mit der freien relationalen Datenbank MySQL zusammenarbeitet. Da keine vernünftige Webanwendung mehr ohne Datenbankunterstützung auskommt, wird hier auch die Installation von MySQL erläutert; zudem erfahren Sie in den anderen Abschnitten dieses Kapitels, wie Sie aus weiteren Programmiersprachen auf MySQL-Datenbanken zugreifen können.
- **mod_perl**
  Das Projekt `mod_perl` ist eine seit Jahren bewährte und ebenfalls beliebte Technologie für Webserver-Anwendungen. Mit `mod_perl` können Sie Ihre vorhandenen Perl-CGIs nach kleinen Anpassungen mit bis zu fünfzigfacher Geschwindigkeit und erheblich geringerem Ressourcenverbrauch ausführen. Das Modul kann aber noch viel mehr: Da es einen Perl-Interpreter in den laufenden Apache-Webserver einbettet, können Sie Perl-Code in die Apache-Konfigurationsdateien schreiben und sogar vollwertige Apache-Module in Perl verfassen.
- **Tomcat**
  Eigentlich ist Tomcat selbst ein recht gut ausgestatteter Webserver. Auf diesen Aspekt der Software wird an dieser Stelle aber nur kurz eingegangen, denn die Hauptaufgabe von Tomcat liegt in der Bereitstellung einer leistungsfähigen

Engine für Java-Webanwendungen, und zwar sowohl für Java Servlets (vollständige Java-Programme mit Webanwendungen) als auch für JavaServer Pages (JSP, eine PHP-ähnliche Möglichkeit zur schnellen Einbettung von Programmcode in HTML). Über das spezielle Modul `mod_jk` oder per Apache-Reverse-Proxy können Sie Tomcat mit Ihrem Apache 2 verbinden.

- **mod_lua**
  Für die neue Apache-Version 2.3/2.4 wurde das bisher als Drittanbieter-Modul verfügbare `mod_lua` in die offizielle Distribution aufgenommen. Es bettet die einfache, leichtgewichtige Skriptsprache Lua in Apache ein. Das Modul gilt noch als experimentell, sodass die Features sich im Lauf der 2.4-Unterversionen möglicherweise noch ändern.

In Abschnitt 15.5, »Weitere Programmierschnittstellen«, werden noch weitere Technologien angesprochen, aber nicht weiter vertieft. Beispielsweise erfahren Sie dort etwas über das nur für Windows verfügbare Modul `mod_isapi`, über `mod_python` sowie über `mod_ruby`.

## 15.1 PHP

Die Skriptsprache PHP – ein rekursives Akronym für *PHP:Hypertext Preprocessor* – wurde 1995 von Rasmus Lerdorf entwickelt. Ursprünglich ging es vor allem um eine Art Makrosammlung für Websites; anfangs stand der Name PHP sogar für »Personal Homepage Tools«. Das Projekt entwickelte sich aber schnell zu einer voll ausgestatteten Webserver-Programmiersprache. PHP arbeitet neben Apache noch mit einigen anderen Webservern zusammen; außerdem können PHP-Skripte seit Version 4.2 auch wie Perl oder Python auf der Kommandozeile ausgeführt werden. Darauf wird hier natürlich nicht weiter eingegangen.

Die aktuelle stabile Version ist 5.3; das Release ist zurzeit 5.3.8. Die neue Version 5.4 ist fast fertig; zurzeit ist der erste Release Candidate verfügbar. Auf der DVD zum Buch finden Sie beide Versionen für UNIX (Quellcode) und Windows (Binaries).

Die wichtigsten Neuerungen in PHP 5 sind folgende:

- **Zend Engine II**
  Schon seit Längerem bildet die Zend Engine der Firma Zend Technologies die stabile und leistungsfähige Basis von PHP. Die neue Version wurde noch einmal erheblich erweitert und verbessert.
- **MySQLi**
  Dies ist eine verbesserte, objektorientierte Schnittstelle zu MySQL-Datenbanken.

- **XML-Unterstützung**
  PHP 5 bietet gleichzeitig einfachere und erweiterte Fähigkeiten zur Verarbeitung von XML-Dateien.

Bitte beachten Sie, dass Sie beim Einsatz von PHP als Apache-Modul auf UNIX-Systemen mit Thread-basierten MPMs vorsichtig sein sollten – vereinzelt gibt es noch PHP-Erweiterungsbibliotheken, die nicht Thread-sicher sind.

### 15.1.1 MySQL installieren

MySQL ist der beliebteste Open-Source-Datenbank-Server der Welt. Er ist schnell, stabil und für viele verschiedene Plattformen verfügbar. Zudem bildet er zusammen mit Apache und PHP (oder Perl oder manchmal auch Python) ein LAMP- oder WAMP-System, die am meisten verbreitete Umgebung für Webanwendungen unter Linux beziehungsweise Windows. Hier erfahren Sie nur das Wichtigste in Kürze; wenn MySQL Sie näher interessiert, empfehle ich Ihnen mein Buch *Praktischer Einstieg in MySQL mit PHP* (O'Reilly Verlag, 2. Auflage, Köln 2007). Unter *http://www.oreilly.de/german/freebooks/einmysql2ger/* können Sie es als kostenloses Openbook im PDF-Format herunterladen.

Damit es mit der MySQL-Unterstützung in PHP keine Probleme gibt, empfiehlt es sich, zuerst die Datenbank und erst danach PHP zu installieren. MySQL finden Sie für verschiedene Betriebssysteme auf der DVD zum Buch, ansonsten auch unter *www.mysql.com*.

Die derzeit stabile MySQL-Version ist 5.5; ihre Installation und Konfiguration werden hier beschrieben. Ein weiteres hier angesprochenes Tool, das sich zur MySQL-Administration anbietet, ist *phpMyAdmin*. Sie finden es auf der DVD zum Buch oder im Web unter *www.phpmyadmin.net*. Es wird einfach unter einer PHP-fähigen URL Ihrer Website zugänglich gemacht. Der Vorteil ist natürlich, dass Sie Ihre MySQL-Datenbanken damit auch über das Web pflegen können.

#### Installation unter UNIX

Auf der Website *www.mysql.com* finden Sie für zahlreiche UNIX-Varianten Binärpakete. Gängige Versionen für x86-32-, x86-64- und Mac-OS-X-Systeme sind auch auf der beiliegenden DVD enthalten. Die Linux-Versionen brauchen Sie in der Regel nur mit `gunzip` und `tar` zu entpacken und per Symlink in ein Verzeichnis wie */usr/local/mysql* abzubilden; die Mac-Version lässt sich per Doppelklick mounten und enthält einen komfortablen grafischen Installer. Anschließend geht es mit `mysql_install_db` (siehe im Folgenden) und den folgenden Schritten weiter.

Für so gut wie alle UNIX-Versionen ist natürlich die Quellcode-Distribution geeignet. Diese müssen Sie wie üblich als Erstes entpacken; anschließend können Sie in das neu erstellte Verzeichnis wechseln:

```
# tar -xzvf mysql-5.5.17.tar.gz
# cd mysql-5.5.17
```

Aus Sicherheitsgründen sollten Sie einen neuen Benutzer und eine neue Gruppe erstellen; der MySQL-Server wird dann unter dieser User- und Group-ID ausgeführt. Dazu können Sie unter Linux die folgenden Befehle verwenden:

```
# groupadd mysql
# useradd -g mysql mysql
```

Für andere UNIX-Varianten gelten ähnliche Kommandos; schlagen Sie im Zweifelsfall in der Dokumentation Ihres Systems nach.

Da dieser »Benutzer« sich niemals persönlich anmelden wird, können Sie sich als Passwort einen vollkommen unmöglichen Zeichensalat ausdenken, den Sie sich nicht einmal zu merken brauchen.

Nun wird die bekannte Befehlsfolge zur Konfiguration und Kompilierung eingegeben:

```
# ./configure --prefix=/usr/local/mysql \
--with-mysqld-user=mysql
# make
# make install
```

Natürlich können Sie `configure` vorher auch wieder mit der Option `--help` aufrufen, um Informationen über weitere Einstellungen zu erhalten. Die hier verwendeten Optionen legen */usr/local/mysql* als Stammverzeichnis fest und sorgen dafür, dass der Server unter der soeben angelegten User-ID ausgeführt wird.

Nach der eigentlichen Installation (sowohl einer Source- als auch einer Binary-Version) müssen Sie die sogenannten *MySQL Grant Tables* erstellen, die die Authentifizierungs- und Berechtigungsdaten enthalten. Dazu können Sie folgendes Skript im *bin*-Verzeichnis Ihrer MySQL-Installation (variiert je nach `PREFIX`) ausführen:

```
# ./mysql_install_db
```

Als Nächstes sollten Sie einige Besitzrechte setzen: Die ausführbaren MySQL-Programme sollten auf `root` übertragen werden, das Datenverzeichnis auf den neu angelegten User `mysql`. Wenn Sie MySQL mit dem `PREFIX /usr/local/mysql` installiert haben und sich darin befinden, können Sie dies folgendermaßen tun:

```
# chown -R root.
# chown -R mysql data
# chgrp -R mysql.
```

Nun können Sie den `mysql`-Server starten:

```
# /usr/local/mysql/bin/mysqld_safe --user=mysql &
```

Wie Ihnen bekannt sein dürfte, führt das angehängte `&` dazu, dass der Befehl im Hintergrund ausgeführt wird, sodass Sie das entsprechende Terminal für andere Aufgaben weiterverwenden können.

Wenn Sie MySQL beim Hochfahren des Systems automatisch starten möchten, können Sie sich sinngemäß an die ausführliche Anleitung halten, die in Kapitel 5, »Apache in Betrieb nehmen«, für den Apache-Webserver gegeben wurde. Das Skript *support-files/mysql.server* im MySQL-Verzeichnis kann dabei als Startskript dienen.

Aus Sicherheitsgründen sollten Sie als Nächstes den Kommandozeilen-Client `mysql` (im *bin*-Verzeichnis der Installation) starten, ein Passwort für `root@localhost` festlegen und die Benutzertabelle `user` in der Verwaltungsdatenbank `mysql` anschließend von allen Datensätzen befreien, die ohne Benutzernamen oder ohne Passwort sind:

```
$ mysql
mysql> SET PASSWORD FOR root@localhost = PASSWORD('geheim');
mysql> DELETE FROM mysql.user WHERE user="" OR password="";
mysql> exit
```

SQL-Abfragen müssen in diesem Client-Programm stets mit einem Semikolon abgeschlossen werden; bis zu diesem Zeichen dürfen sie sich über beliebig viele Zeilen erstrecken. Selbstverständlich müssen Sie sich ein besseres Passwort einfallen lassen als `'geheim'`. Falls Sie ältere Clients oder Programmierschnittstellen einsetzen, muss übrigens die Funktion `OLD_PASSWORD()` anstelle von `PASSWORD()` zur Verschlüsselung eingesetzt werden.[1] In Zukunft muss der Client mit Benutzernamen und Passwort gestartet werden:

```
$ mysql -u root -p
[root-Passwort eingeben]
```

#### Installation unter Windows

Für Windows wird ein komfortabler MySQL-Installer angeboten. Sie finden die aktuelle Version auf der Buch-DVD. Starten Sie nach dem Entpacken des ZIP-

1 Der MySQL-User root sollte aber sowieso nicht für datenbankbasierte Anwendungen eingesetzt werden.

Archivs zunächst den Installer *Setup.exe*. Die Installation selbst ist nicht der Rede wert; Sie brauchen nur Installationsort und Ausführlichkeit der Installation zu wählen. Wenn Sie sich danach entscheiden, den Server sofort zu konfigurieren, wird der *MySQL Server Instance Configuration Wizard* gestartet, der Ihnen eine komfortable Grundkonfiguration Ihres MySQL-Servers ermöglicht. Hier werden Sie nacheinander durch folgende Bildschirme geführt, die Sie nach der Optionsauswahl jeweils mit NEXT abschließen müssen:

1. **Konfigurationstyp**
   Wählen Sie DETAILED CONFIGURATION, um den folgenden Anweisungen zu folgen.
2. **Server-Typ**
   Für eine Arbeitsstation, an der Sie Webanwendungen entwickeln, ist DEVELOPER MACHINE die richtige Wahl. SERVER MACHINE ist für einen Server-Rechner gedacht, auf dem mehrere Server-Dienste ausgeführt werden, z.B. MySQL und Apache 2. Die Auswahl DEDICATED MYSQL SERVER MACHINE optimiert MySQL dagegen für einen Server, auf dem ausschließlich MySQL ausgeführt wird – solche Maschinen laufen allerdings selten unter Windows.
3. **Datenbanktypen**
   MySQL verwendet zur Speicherung von Datenbanktabellen verschiedene Storage Engines mit unterschiedlichen Stärken und Schwächen. Die praktischste Wahl ist daher MULTIFUNCTIONAL DATABASE – die Engine kann für jede Tabelle frei gewählt werden. TRANSACTIONAL DATABASE ONLY optimiert den MySQL-Server für die transaktionsfähige Engine InnoDB, kann aber notfalls auch mit den klassischen MyISAM-Tabellen umgehen. NON-TRANSACTIONAL DATABASE ONLY schließlich erlaubt nur die transaktionslose, aber stark performanceoptimierte MyISAM-Engine.
4. **Tablespace**
   Wählen Sie hier, wo Ihre Datenbanken gespeichert werden sollen. Die Vorgabe ist das MySQL-Installationsverzeichnis; wenn Sie wechseln möchten, sollten Sie die größte freie NTFS-Partition wählen.
5. **Anzahl gleichzeitiger Verbindungen**
   Für einen Entwickler-Rechner genügt DECISION SUPPORT, optimiert für bis zu 20 gleichzeitige Client-Verbindungen. Für einen hoch belasteten Produktionsserver sollte ONLINE TRANSACTION PROCESSING gewählt werden. Schließlich besteht auch noch die Möglichkeit, die Anzahl der Verbindungen von Hand einzugeben.
6. **Netzwerkoptionen**
   Hier können Sie die TCP/IP-Netzwerkunterstützung einstellen (ausschalten sollten Sie sie in aller Regel nicht); auch die Standardportnummer 3306 soll-

ten Sie beibehalten, da manche Clients sie voraussetzen. Enable Strict Mode aktiviert ein strengeres Verhalten von SQL-Abfragen, wie es zu einem modernen, transaktionsorientierten SQL-Server passt. Beachten Sie aber, dass einzelne Client-Anwendungen nicht damit zurechtkommen.

7. **Zeichensatz**
   Sofern Sie nichts Besonderes vorhaben, ist Standard Character Set (ISO-8859-1) zu empfehlen; Sie können trotzdem den Zeichensatz und die darauf aufbauende Sortierreihenfolge jeder Tabelle und jeder einzelnen Spalte wählen.
8. **Windows-Optionen**
   Die empfehlenswerte Option Install As Windows Service erzeugt einen MySQL-Dienst (Windows-Dienste werden am Beispiel von Apache in Kapitel 5, »Apache in Betrieb nehmen«, erläutert); Launch the MySQL Server automatically startet diesen automatisch beim Booten. Sie sollten auch Include Bin Directory in Windows PATH aktivieren – das MySQL-*bin*-Verzeichnis wird zur Umgebungsvariablen `PATH` hinzugefügt, sodass Sie die MySQL-Dienstprogramme nach dem nächsten Neustart von einem beliebigen Arbeitsverzeichnis aus aufrufen können.
9. **Sicherheit**
   Wählen Sie Modify Security Settings, und geben Sie ein starkes Passwort für den MySQL-User `root` ein. Die Option Enable root access from remote machines sollten Sie nur in begründeten Ausnahmefällen aktivieren – wenn Sie den MySQL-Server unbedingt mit voller Kontrolle über das Netzwerk oder Internet administrieren müssen, sollten Sie dafür einen separaten MySQL-User erstellen. Auch Create An Anonymous Account (anonymer Benutzer ohne Passwort) ist meist nicht zu empfehlen.
10. **Abschluss**
    Klicken Sie auf Execute, um die Einstellungen anzuwenden. Falls dies nicht funktioniert, liegt es meist daran, dass noch ein alter MySQL-Dienst existiert. Starten Sie den Wizard in diesem Fall neu; er sollte Ihnen die Option anbieten, einen bestehenden MySQL-Dienst zu entfernen. Andernfalls ist Ihre Firewall schuld; hier müssen Sie gegebenenfalls den Port 3306 öffnen.

#### Eine Testdatenbank

Die einfachen MySQL-Beispiele in diesem Kapitel beziehen sich alle auf eine kleine Datenbank namens `buecher`, in der eine einzige gleichnamige Tabelle enthalten ist. Die Felder dieser Datenbanktabelle sehen Sie in Tabelle 15.1.

Damit die Beispiele nicht allzu umfangreich werden, ist die Tabelle sehr einfach gehalten: Das relationale Datenbankmodell würde eigentlich nicht erlauben, dass

der Autor (dessen Name aus Vor- und Nachnamen besteht) in einem einzelnen Feld steht – eigentlich darf er nicht einmal in derselben Tabelle stehen, weil ein Buch mehrere Autoren besitzen kann und weil ein und derselbe Autor mehrere Bücher geschrieben haben kann. Letzteres gilt auch für den Verlag. In den Beispielinhalten erscheint deshalb jeweils nur der Nachname des Autors, und es werden nur Bücher von Einzelautoren aufgenommen.

| Feldbezeichnung | Datentyp |
|---|---|
| isbn | char(13), Primärschlüssel |
| autor | varchar(30) |
| titel | varchar(40) |
| verlag | varchar(30) |

**Tabelle 15.1** Inhalte der Beispieldatenbanktabelle »buecher«

Sie können die Datenbank und die Tabelle erstellen, indem Sie im Kommandozeilen-Client `mysql` nacheinander folgende SQL-Befehle eintippen (die Antworten des MySQL-Clients wurden hier weggelassen):

```
mysql> CREATE DATABASE buecher;
mysql> use buecher
mysql> CREATE TABLE buecher (
    ->    isbn CHAR(10),
    ->    autor varchar(30),
    ->    titel varchar(40),
    ->    verlag varchar(30),
    ->    PRIMARY KEY (isbn)
    -> );
```

`CREATE DATABASE` und `CREATE TABLE` sind SQL-Standardbefehle zum Erzeugen einer neuen Datenbank beziehungsweise Tabelle; `use <Datenbank>` ist dagegen ein interner Befehl des MySQL-Clients, der zur Auswahl der gewünschten Datenbank verwendet wird. Daher endet Letzterer auch nicht mit einem Semikolon.

Als Nächstes benötigt die Datenbank einige Beispielinhalte. Sie können entweder die Werte aus Tabelle 15.2 übernehmen oder Ihre eigenen benutzen.

| ISBN | Autor | Titel | Verlag |
|---|---|---|---|
| 9783898426947 | Bongers | XSLT 2.0 und XPath 2.0 | Galileo Press |
| 9783897217201 | Friedl | Reguläre Ausdrücke | O'Reilly |
| 9783836212335 | Erlhofer | Suchmaschinen-Optimierung | Galileo Press |
| 9783897214057 | Schroder | Linux-Kochbuch | O'Reilly |

**Tabelle 15.2** Beispieldatensätze für die Datenbanktabelle »buecher«

Die Werte können Sie zur Not ebenfalls mithilfe des Clients `mysql` eintippen:

```
mysql> INSERT INTO buecher VALUES
      ('9783898426947', 'Bongers', 'XSLT 2.0 und XPath 2.0', 'Galileo
Press'),
      ('9783897217201', 'Friedl', 'Reguläre Ausdrücke', 'O\'Reilly'),
       ...
       ('9783897214057', 'Schroder', 'Linux-Kochbuch', 'O\'Reilly');
```

Beachten Sie bitte das Semikolon hinter dem letzten Datensatz (im Gegensatz zu den Kommata bei allen anderen) sowie den Backslash vor dem Apostroph bei `'O\'Reilly'` – alternativ können Sie die Felder durch doppelte Anführungszeichen umschließen.

Alle diese Schritte können Sie sich sparen, wenn Sie die Datei *mkbuecher.sql* aus dem Verzeichnis *listings* der DVD zum Buch ausführen. Dies funktioniert im MySQL-Kommandozeilen-Client folgendermaßen:

```
mysql> source mkbuecher.sql
```

Für die im Folgenden genannten PHP-Beispiele sollten Sie nun noch einen User erstellen, der auf diese (und nur auf diese) Datenbank zugreifen darf:

```
mysql> CREATE USER buchuser@localhost;
mysql> GRANT ALL PRIVILEGES ON buecher.* TO buchuser@localhost
    -> IDENTIFIED BY 'passwort';
```

Auch hier müssen Sie sich bemühen, ein besseres Passwort zu finden. Wenn Sie vorhaben, mit alten Clients (z.B. PHP-Versionen vor 4.4 oder 5.0.3) zusammenzuarbeiten, müssen Sie den Bereich `IDENTIFIED BY 'passwort'` weglassen und die folgende zusätzliche Zeile eingeben:

```
mysql> SET PASSWORD FOR buchuser@localhost =
    -> OLD_PASSWORD('passwort');
```

### 15.1.2 PHP installieren

PHP bietet zwei verschiedene Betriebsmodi für die Zusammenarbeit mit dem Apache-Webserver: als externer CGI-Interpreter oder als Modul (Letzteres wird im PHP-Jargon als *Server-API-* oder kurz *SAPI-Modul* bezeichnet). Beide Versionen haben ihre Vor- und Nachteile:

- Die CGI-Variante wird seit der ersten PHP-Version unterstützt, ist also seit fast 13 Jahren unzähligen Praxistests ausgesetzt. Sie kann daher mit Recht als stabil und ausgereift bezeichnet werden. Dafür leidet sie unter den in Kapitel 14, »CGI«, erwähnten Nachteilen von CGI: Das Forking eines neuen Child-Prozesses (oder das Erzeugen eines neuen Threads) für jeden Aufruf eines PHP-CGI-

Skripts erzeugt bei stark belasteten Websites schnell Speicher- und Performance-Engpässe. FastCGI kann eine interessante Alternative sein.

- Die SAPI-Unterstützung wurde erst mit der PHP-Version 4 eingeführt und galt lange als experimentell. Inzwischen ist sie allerdings praxistauglich; immer mehr Sites benutzen PHP als Server-Modul, vor allem natürlich mit Apache. Durch das direkte Einfügen des PHP-Interpreters in den Webserver kann dieser PHP-Skripte ohne zusätzliche Belastung unmittelbar ausführen. Selbstverständlich benötigt Ihre Apache-Installation DSO-Unterstützung, damit diese Variante eingesetzt werden kann.

In diesem Abschnitt werden die Installation und Konfiguration beider PHP-Einsatzarten erklärt.

#### Installation unter UNIX

Auf UNIX-Systemen ist – wie bei freier Software üblich – die Kompilierung des Quellcodes zu bevorzugen. Das Konfigurationsskript, das die Makefiles für PHP erzeugt, wurde genau wie bei Apache 2 mit GNU Autoconf und Automake erstellt. Die Installation funktioniert deshalb im Prinzip genau wie der in Kapitel 4, »Apache kompilieren und installieren«, erläuterte Apache-Build-Prozess.

Zunächst einmal müssen Sie sich das Quellcode-Paket der gewünschten PHP-Version besorgen. Sie finden PHP 5.3 im aktuellen Release auf der DVD zum Buch; auf der Site *www.php.net* beziehungsweise auf den dort angegebenen Mirrors können Sie sich aber auch jeweils die neueste Version herunterladen. Zunächst einmal müssen Sie das Archiv entpacken:

```
# tar -xzvf php-5.3.8.tar.gz
```

Wechseln Sie anschließend in das neu erstellte Verzeichnis, in dem sich der entpackte Quellcode-Baum befindet:

```
# cd php-5.3.8
```

Nun wird das `configure`-Skript aufgerufen. Die allgemeinen Optionen sind Autoconf-Standard und wurden in Kapitel 4, »Apache kompilieren und installieren«, bereits für Apache erläutert, andere Kommandozeilenargumente, insbesondere die zahlreichen Schalter im Format `--enable-feature` oder `--with-option`, sind PHP-spezifisch. Geben Sie folgenden Befehl ein, wenn Sie alle Optionen in Ruhe lesen möchten:

```
# ./configure --help |less
```

Sie sollten auf jeden Fall die Option `--with-mysql=MySQL-Verzeichnis` verwenden, um die wichtige MySQL-Unterstützung zu aktivieren. Wenn Sie eine

MySQL-Version ab 4.1.2 verwenden, sollten Sie (zusätzlich oder ausschließlich) die `mysqli`-Extension aktivieren. Dazu müssen Sie `--with-mysqli=/Pfad/zu/mysql_config` (befindet sich im Verzeichnis *bin* Ihrer MySQL-Installation) angeben. Diese Extension bietet eine moderne, objektorientierte Schnittstelle zu MySQL-Datenbanken. Im weiteren Verlauf des Kapitels wird die Programmierung mit beiden Varianten erläutert.

Eine weitere Entscheidung, die Sie schon an dieser Stelle treffen müssen, ist die Frage, ob Sie PHP als CGI- oder als SAPI-Variante verwenden möchten. Für letztere Variante ist nämlich die Option `--with-apxs2[=/Pfad/zu/apxs]` erforderlich, wenn Sie die notwendigen Anpassungen nicht manuell vornehmen möchten. In aller Regel empfiehlt sich die Installation als Apache-Modul.

Hier sehen Sie einen `configure`-Aufruf, der die Makefiles zur Installation als SAPI-Modul mit `mysql-` und `mysqli`-Unterstützung in das Basisverzeichnis `/usr/local/php5` erstellt, wenn sich `apxs` unter `/usr/local/apache2/bin` und MySQL unter `/usr/local/mysql` befindet:

```
# ./configure --prefix=/usr/local/php5 \
--with-mysql=/usr/local/mysql \
--with-mysqli=/usr/local/mysql/bin/mysql_config \
--with-apxs2=/usr/local/apache2/bin/apxs
```

Wünschen Sie dagegen traditionelle CGI-Unterstützung, können Sie folgende Variante des Konfigurationsbefehls (hier nur mit der klassischen `mysql`-Extension) einsetzen:

```
# ./configure --prefix=/usr/local/php5 \
--with-mysql=/usr/local/mysql
```

In beiden Fällen wird die bekannte `Checking...`-Orgie in Gang gesetzt, die je nach Systemgeschwindigkeit und gewählten Optionen ziemlich lange dauern kann. Wenn `configure` beendet wurde, erfolgen der eigentliche Build-Vorgang und die anschließende Installation über die beiden folgenden bekannten Befehle:

```
# make
# make install
```

Wenn Sie PHP mit der Option `--with-apxs2` konfiguriert haben, sollte Ihre *httpd.conf*-Datei automatisch folgende Zusatzzeile erhalten haben:

```
LoadModule php5_module modules/libphp5.so
```

In beiden Fällen müssen Sie selbst dafür sorgen, dass Apache 2 Dateien mit der Endung `.php` (oder, wenn Sie möchten, auch mit den veralteten Endungen `.php3`,

`.php4` oder `.phtml`) als PHP-Skripte betrachtet. Dazu wird im gewünschten Kontext eine der folgenden Zeilen eingesetzt:

```
AddHandler php5-script .php
# Alternative zur Verknüpfung weiterer Endungen:
AddHandler php5-script .php .php3 .php4 .phtml
```

Alternativ können Sie die – eigentlich veraltete – `AddType`-Variante verwenden:

```
AddType application/x-httpd-php5 .php
# Alternative zur Verknüpfung weiterer Endungen:
AddType application/x-httpd-php5 .php .php3 .php4 .phtml
```

Nach einem Neustart Ihres Apache-Servers können Sie bereits die PHP-Funktionalität testen. Speichern Sie dazu folgende kleine Datei unter dem Namen *info.php* in einem Kontext Ihrer Server-Verzeichnisse, in dem die soeben eingestellte Typzuordnung gilt:

```
<?php

   phpinfo();

?>
```

Wenn Sie diese Datei anschließend über einen Browser öffnen, sollten Sie ein Ergebnis wie in Abbildung 15.1 zu Gesicht bekommen. Die Funktion `phpinfo()` erzeugt nämlich ein umfangreiches HTML-Dokument mit zahlreichen Informationen zur PHP- und Apache-Konfiguration. Falls dieses Ergebnis bei Ihnen aus irgendeinem Grund nicht eintritt, überprüfen Sie noch einmal Schritt für Schritt die gesamte Beschreibung.

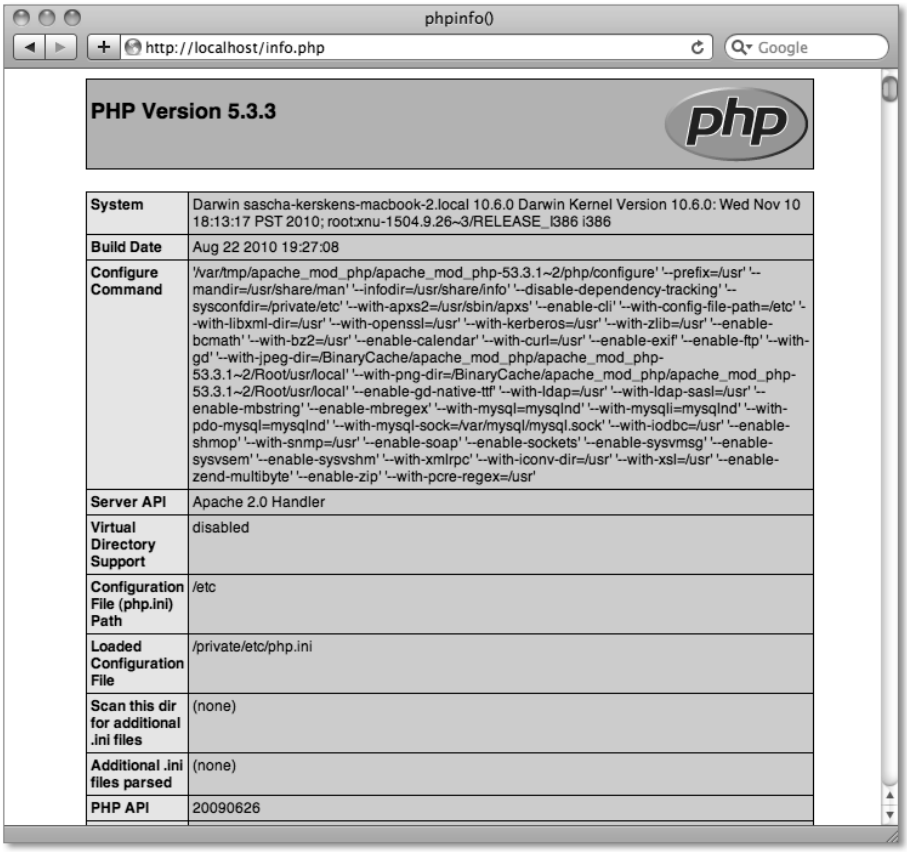
Wenn Sie PHP als CGI-Anwendung verwenden möchten, müssen Sie *httpd.conf* um folgende Zeilen erweitern:

```
Alias /php/ /usr/local/php5/bin/
AddHandler php5-script .php
# Alternative zu AddHandler:
#AddType application/x-httpd-php5 .php
Action application/x-httpd-php5 "/php/php"
```

Suchen Sie sich am besten für jede der drei Zeilen eine Stelle, an der dieselbe Direktive verwendet wird, oder kommentieren Sie die ganze Direktivengruppe als »PHP-CGI-Konfiguration«.

Die PHP-Entwickler stellen für UNIX keine Binaries zur Verfügung. Aber vielleicht gibt es für Ihre konkrete UNIX-Sorte anstelle des Quellcode-Pakets auch ein

vorgefertigtes Installer-Paket Ihres Distributors. Schauen Sie einfach auf den Installations-CDs oder auf der Website Ihres Systems nach. Für Apache 2 müssen Sie sicherstellen, dass es sich mindestens um PHP 4.3 handelt; ältere Versionen arbeiten nicht einwandfrei (oder möglicherweise gar nicht) mit der aktuellen Apache-Version zusammen.[2] Näheres entnehmen Sie bitte der Dokumentation Ihres Betriebssystems.




| | |
|---|---|
| System | Darwin sascha-kerskens-macbook-2.local 10.6.0 Darwin Kernel Version 10.6.0: Wed Nov 10 18:13:17 PST 2010; root:xnu-1504.9.26~3/RELEASE_I386 i386 |
| Build Date | Aug 22 2010 19:27:08 |
| Configure Command | '/var/tmp/apache_mod_php/apache_mod_php-53.3.1~2/php/configure' '--prefix=/usr' '--mandir=/usr/share/man' '--infodir=/usr/share/info' '--disable-dependency-tracking' '--sysconfdir=/private/etc' '--with-apxs2=/usr/sbin/apxs' '--enable-cli' '--with-config-file-path=/etc' '--with-libxml-dir=/usr' '--with-openssl=/usr' '--with-kerberos=/usr' '--with-zlib=/usr' '--enable-bcmath' '--with-bz2=/usr' '--enable-calendar' '--with-curl=/usr' '--enable-exif' '--enable-ftp' '--with-gd' '--with-jpeg-dir=/BinaryCache/apache_mod_php/apache_mod_php-53.3.1~2/Root/usr/local' '--with-png-dir=/BinaryCache/apache_mod_php/apache_mod_php-53.3.1~2/Root/usr/local' '--enable-gd-native-ttf' '--with-ldap=/usr' '--with-ldap-sasl=/usr' '--enable-mbstring' '--enable-mbregex' '--with-mysql=mysqlnd' '--with-mysqli=mysqlnd' '--with-pdo-mysql=mysqlnd' '--with-mysql-sock=/var/mysql/mysql.sock' '--with-iodbc=/usr' '--enable-shmop' '--with-snmp=/usr' '--enable-soap' '--enable-sockets' '--enable-sysvmsg' '--enable-sysvsem' '--enable-sysvshm' '--with-xmlrpc' '--with-iconv-dir=/usr' '--with-xsl=/usr' '--enable-zend-multibyte' '--enable-zip' '--with-pcre-regex=/usr' |
| Server API | Apache 2.0 Handler |
| Virtual Directory Support | disabled |
| Configuration File (php.ini) Path | /etc |
| Loaded Configuration File | /private/etc/php.ini |
| Scan this dir for additional .ini files | (none) |
| Additional .ini files parsed | (none) |
| PHP API | 20090626 |

**Abbildung 15.1** Beispielausgabe der PHP-Funktion »phpinfo( )«, hier für PHP 5.3.3 unter Mac OS X

### Installation unter Windows

Für Windows wird PHP bevorzugt in Binärform angeboten; die eigenständige Kompilierung ist etwas komplexer als bei Apache und soll hier nicht vertieft wer-

2 Der Support für den kompletten PHP-4-Entwicklungszweig wurde aber ohnehin bereits vor längerer Zeit eingestellt, sodass Sie von Versionen vor PHP 5.x die Finger lassen sollten.

den. PHP-Release-Versionen (zurzeit 5.3.8) werden für Windows in zwei verschiedenen Varianten angeboten:

1. **Installer**
   Dieses Paket ist eine EXE-Datei, die Sie per Doppelklick starten können. Es unterstützt ausschließlich die CGI-Version von PHP. Genau wie bei der binären Apache-Installation (siehe Kapitel 4, »Apache kompilieren und installieren«) müssen Sie einige Fragen beantworten – unter anderem die nach dem Installationsverzeichnis. Für einige Webserver wie Microsoft IIS oder OmniHTTPd kann der Installer auch gleich automatisch die PHP-Konfiguration vornehmen. Auch Apache wird laut Liste inzwischen unterstützt, aber mein letzter Test verlief erfolglos (möglicherweise funktioniert das Paket unter Windows 7 und Vista nicht).
2. **ZIP-Archiv**
   Dieses Paket enthält den vollständigen Leistungsumfang von PHP: Neben der CGI-Funktionalität werden die SAPI-Module für diverse Webserver mitgeliefert, unter anderem auch für Apache (gemeint ist 1.3) und Apache 2.

Der Binär-Installer stellt Ihnen eine Reihe von Fragen, die im Prinzip einigen der im Folgenden vorgestellten `php.ini`-Einträge entsprechen. Nach der Installation müssen Sie die PHP-CGI-Einstellungen in Ihrer *httpd.conf* vornehmen; das folgende Beispiel geht davon aus, dass Sie PHP, wie vom Installer vorgeschlagen, im Verzeichnis *C:\PHP* installiert haben:

```
ScriptAlias /php/ "c:/php/"
AddHandler php5-script .php
# Veraltete Alternative zu AddHandler:
#AddType application/x-httpd-php .php
Action application/x-httpd-php "/php/php.exe"
```

Die Einrichtung der ZIP-Datei erfordert ein wenig mehr Handarbeit, ist aber auch nicht allzu schwierig. Der Vorteil ist, dass Sie PHP als Apache-Modul installieren können. Im Wesentlichen müssen Sie die folgenden Schritte durchführen:

1. Entpacken Sie das ZIP-Archiv, das zurzeit (Stand: November 2011) beispielsweise *php-5.3.8-Win32.zip* heißt, mit WinZip oder einem anderen ZIP-fähigen Programm. Als Verzeichnis empfiehlt sich wiederum *C:\PHP*. Die Ausführungen in den folgenden Punkten gehen von diesem Vorschlag aus; wenn Sie ein anderes Verzeichnis wählen, müssen Sie einige Angaben entsprechend anpassen.
2. Das Verzeichnis, in das Sie PHP entpackt haben, enthält eine Datei namens *php.ini-dist*. Es handelt sich um die vorgefertigte Version der Konfigurations-

datei *php.ini*. Am einfachsten ist es, diese Datei nach `%SystemRoot%` (z.B. *C:\Windows*) zu kopieren und dort in *php.ini* umzubenennen.

3. Nehmen Sie an Ihrer neuen *php.ini*-Datei folgende Änderungen vor (die entsprechenden Zeilen sind bereits enthalten; sie besitzen aber noch keine Pfadangabe):

```
doc_root = C:\Apache2\htdocs    ; Ihre DocumentRoot
extension_dir = C:\PHP\ext      ; Ihr PHP-Verzeichnis
```

   Wie Sie sehen, wird in der Windows-Version dieser Datei der systemübliche Backslash (\) als Pfadtrennzeichen verwendet. Das Semikolon leitet einen Kommentar ein, der bis zum Zeilenende reicht. Wie die Kommentare schon andeuten, müssen Sie die Angaben an Ihr Apache- beziehungsweise PHP-Verzeichnis anpassen.

4. Fügen Sie das Verzeichnis *C:\PHP* zur Umgebungsvariablen `PATH` hinzu. Dies geschieht unter START • SYSTEMSTEUERUNG • SYSTEM • ERWEITERT • UMGEBUNGSVARIABLEN; anschließend müssen Sie den Rechner leider komplett neu starten.

5. Kopieren Sie die Datei `php5ts.dll` aus Ihrem PHP-Verzeichnis nach `%SystemRoot%\System32`.

6. Nehmen Sie zu guter Letzt folgende Ergänzungen an Ihrer *httpd.conf*-Datei vor:

```
LoadModule php5_module "C:/PHP/php5apache2.dll"
AddHandler php5-script .php
# Veraltete Alternative zu AddHandler:
#AddType application/x-httpd-php .php
```

   Natürlich muss der PHP-Pfad wiederum angepasst werden; zum Thema Dateiendungen wurde an entsprechender Stelle bereits alles gesagt.

7. Starten Sie nun Ihren Apache-Webserver neu. Im Apache-Monitor und in automatisch generierten Dokumenten (bei entsprechender `ServerTokens`-Einstellung) müsste er sich nun mit einer Versionsangabe wie `Apache/2.3.15 (Win32) PHP/5.3.8` melden.

### 15.1.3 Die PHP-Konfigurationsdatei php.ini

Wie bereits im Zusammenhang mit der Installation erwähnt wurde, besitzt PHP seine eigene Konfigurationsdatei, nämlich *php.ini*. Wie der Dateiname vermuten lässt, wurde ihre Syntax (auch unter UNIX) den klassischen Windows-INI-Dateien nachempfunden:

- Sie enthält verschiedene Abschnitte, die durch Schlüsselwörter in eckigen Klammern gekennzeichnet sind. Beispiel:

  ```
  [PHP]
  ```

- Jede Einstellung steht in einer eigenen Zeile und hat das Format `Name=Wert`. Beispiel:

  ```
  doc_root = /usr/local/apache2/htdocs
  ```

  Unter Windows wird in Pfaden übrigens, wie bereits erwähnt, der plattformspezifische Backslash (\) verwendet.

- Ein Semikolon leitet einen Kommentar ein, der bis zum Ende der jeweiligen Zeile reicht. Beispiel:

  ```
  ; doc_root: Die DocumentRoot der Website
  doc_root = "C:\Programme\Apache Group\Apache2\htdocs" ; Win32
  ```

In Tabelle 15.3 finden Sie die wichtigsten Optionen für die Datei. Beachten Sie, dass die Syntax in der Regel toleranter ist, als es in der Tabelle den Anschein erweckt. Die Boolean-Alternativen `On|Off`, `"On"|"Off"` und `"1"|"0"` sind beispielsweise in der Regel synonym – die Angaben stammen aber aus der Originaldokumentation, sodass Sie auf der sicheren Seite sind, wenn Sie sich daran halten.

| Einstellung | Standardwert | Erläuterung |
|---|---|---|
| `short_open_tag = On\|Off` | `On` | Bestimmt, ob `<? ... ?>` anstelle von `<?php ... ?>` erlaubt ist. |
| `asp_tags = On\|Off` | `Off` | Bestimmt, ob die ASP-Syntax `<% ... %>` anstelle von `<?php ... ?>` zulässig ist. |
| `max_execution_time` | `30` | maximale Ausführungsdauer von PHP-Skripten, bevor sie abgebrochen werden |
| `precision = "INT"` | `"14"` | Genauigkeit von Fließkommazahlen als Ziffernanzahl |
| `y2k_compliance = On\|Off` | `Off` | Jahr-2000-Kompatibilität |
| `expose_php = On\|Off` | `On` | Bestimmt, ob die PHP-Existenz z. B. in den `ServerTokens` veröffentlicht wird. |
| `memory_limit = "...M\|K\|B"` | `"8M"` (8 Megabyte) | Maximale Speichermenge, die ein Skript verbrauchen darf. Existiert nur, wenn PHP mit `--enable-memory-limit` kompiliert wurde. |

**Tabelle 15.3** Die wichtigsten Optionen der Datei »php.ini«

| Einstellung | Standardwert | Erläuterung |
|---|---|---|
| track-vars = "On"\|"Off" | "On" | Anfrage- und Server-Daten werden in $_ENV, $_GET, $_POST, $_COOKIE und $_SERVER zur Verfügung gestellt. |
| arg_separator.output = "STRING" | "&" | Trennzeichen in Query-Strings, die von PHP erzeugt wurden |
| arg_separator.input = "STRING" | "&" | Trennzeichen für die Auswertung von Query-Strings (jedes Zeichen des Werts wird einzeln geprüft, z.B. "&;") |
| variables_order = "STRING" | "EGPCS" | Reihenfolge, in der Variablenarten für register_globals ausgewertet werden:<br>E = Umgebungsvariablen<br>G = GET-Felder<br>P = POST-Felder<br>C = Cookies<br>S = Server-Variablen |
| register_globals = "On"\|"Off" | "Off" | Stellt Formularfelder, Umgebungsvariablen, Server-Variablen und Cookies automatisch als globale Variablen mit ihrem Feldnamen bereit. Praktisch, aber sicherheitstechnisch riskant; daher seit 4.2 standardmäßig "Off". In PHP 5.4 abgeschafft. |
| register_argc_argv = "On"\|"Off" | "On" | Stellt GET-Variablen und Kommandozeilenargumente als C-ähnliche argc/argv bereit. |
| register_long_arrays = "On"\|"Off" | "On" | Definiert $HTTP_POST_VARS und $HTTP_GET_VARS zusätzlich zu $_POST und $_GET (seit 5.0 konfigurierbar; davor immer eingeschaltet). |
| post_max_size = "...M\|K\|B" | "8M" | maximale Größe von POST-Daten |
| auto_prepend_file = "DATEIPFAD" | "" | Include-Datei, die automatisch am Kopf jeder Datei importiert wird |
| auto_append_file = "DATEIPFAD" | "" | Include-Datei, die automatisch am Fuß jeder Datei importiert wird |
| default_mimetype = "TYPE/SUBTYPE" | "text/html" | Standardtyp für den Header Content-Type |

**Tabelle 15.3** Die wichtigsten Optionen der Datei »php.ini« (Forts.)

| Einstellung | Standardwert | Erläuterung |
|---|---|---|
| `default_charset = "ZEICHENSATZ"` | `"iso-8859-1"` | Standardzeichensatz für den Header `Content-Type` |
| `always_populate_raw_post_data = "0"\|"1"` | `"0"` | Bestimmt, ob `$HTTP_RAW_POST_DATA` (`POST`-Daten ohne Variablenaufteilung) immer erzeugt wird. |
| `allow_webdav_methods = "0"\|"1"` | `"0"` | Bestimmt, ob WebDAV-Methoden zulässig sind. |
| `include_path = "PFAD1 [:PFAD2:...]"` | Umgebungsvariable `PHP_INCLUDE_PATH` | Verzeichnisse, in denen nach `Include`-Dateien gesucht wird |
| `doc_root = "PFAD1[:PFAD2:...]"` | `PHP_INCLUDE_PATH` | übergeordnetes Verzeichnis für PHP-Skripte (z.B. Apache-`DocumentRoot`) |
| `user_dir = "VERZEICHNIS"` | `NULL` | Name des Benutzerverzeichnisses mit PHP-Dateien (entspricht dem `mod_userdir`-Konzept, siehe Kapitel 8, »Weiterleitungen und Indizes«) |
| `extension_dir = "PFAD"` | Umgebungsvariable `PHP_EXTENSION_DIR` | Verzeichnis mit Extension-Dateien, die beim PHP-Start geladen werden |
| `cgi.rfc2616_headers = "0"\|"1"` | `"0"` | ausgeschaltet `Status:` Header; eingeschaltet RFC-2616-Header |
| `file_uploads = "0"\|"1"` | `"1"` | Datei-Uploads zulässig |
| `upload_tmp_dir = "PFAD"` | `NULL` (Systemvorgabe) | Temp-Verzeichnis für Uploads |
| `upload_max_filesize = "...M\|K\|B"` | `"2M"` | maximale Größe für Upload-Datei |
| `sql.safe_mode = "0"\|"1"` | `"0"` | sicheres SQL; Schutz vor SQL-Injection (aber normale PHP-MySQL-Beispiele funktionieren nicht mehr) |

**Tabelle 15.3** Die wichtigsten Optionen der Datei »php.ini« (Forts.)

### 15.1.4 phpMyAdmin einrichten

phpMyAdmin ist ein in PHP geschriebenes, webbasiertes Administrations- und Datenbankbearbeitungs-Tool. Die Installation beschränkt sich auf das Entpacken einer Archivdatei und das Ausführen eines Konfigurationsskripts. Auf der beiliegenden DVD finden Sie eine ZIP-Datei für Windows und eine *.tar.gz*-Datei für UNIX.

Es empfiehlt sich, `phpMyAdmin` aus Sicherheitsgründen in ein Verzeichnis außerhalb Ihrer `DocumentRoot` zu entpacken und mittels `Alias` einzubinden. Im folgen-

den Konfigurationsbeispiel wurde die Archivdatei nach */usr/local/apache2* (in die `ServerRoot`) entpackt, sodass die `Alias`-Direktive wie folgt lautet:

```
Alias /phpmyadmin /usr/local/apache2/phpMyAdmin-3.4.7.1-all-
languages
```

Gegebenenfalls müssen Sie noch einen `<Directory>`- oder `<Location>`-Container einrichten, um Berechtigungen und Optionen für das Verzeichnis zu setzen.

Erzeugen Sie im Verzeichnis *phpMyAdmin-3.4.7.1-all-languages* ein Unterverzeichnis mit dem Namen *config*; es benötigt Schreibrechte für jeden. Unter UNIX funktioniert dies so:

```
# cd phpMyAdmin-3.4.7.1-all-languages
# mkdir config
# chmod o+rw config
```

Auf Windows-Systemen funktionieren die betreffenden Schritte so:

```
> cd phpMyAdmin-3.4.7.1-all-languages
> md config
> cacls config /e /t /g SYSTEM:F
```

`SYSTEM` müssen Sie gegebenenfalls durch den User ersetzen, unter dem Apache tatsächlich läuft; dies lässt sich unter Start • Systemsteuerung • Verwaltung • Dienste ermitteln oder ändern (siehe Kapitel 5, »Apache in Betrieb nehmen«).

Als Nächstes wird ein spezielles MySQL-Benutzerkonto benötigt, das die Verwaltungsdatenbank `mysql` lesen darf, um die `phpMyAdmin`-Authentifizierung zu überprüfen.

Starten Sie dazu den MySQL-Konsolen-Client `mysql` als `root`, und geben Sie folgende Anweisungen ein (bitte mit eigenem Passwort):

```
mysql> CREATE USER phpcontroller@localhost
    -> IDENTIFIED BY 'phpCtrl';
mysql> GRANT SELECT ON mysql.* TO
    -> phpcontroller@localhost;
```

Starten Sie Apache danach neu, und geben Sie die Adresse *http://localhost/phpmyadmin/scripts/setup.php* in die Adressleiste Ihres Browsers ein, um die interaktive Konfigurationshilfe zu starten. In diesem Dialog können Sie zahlreiche Einstellungen für `phpMyAdmin` vornehmen. Die wichtigste von ihnen ist die Einrichtung der MySQL-Benutzeranmeldung; genau diese wird hier beschrieben.

Klicken Sie in der Kategorie Servers auf Add, um den ersten MySQL-Server zu konfigurieren. (Eine `phpMyAdmin`-Installation kann mit MySQL-Servern auf mehreren Hosts zusammenarbeiten.)

Geben Sie unter Server hostname den Hostnamen des Datenbank-Servers ein; bei einer lokalen Installation können Sie die Voreinstellung localhost beibehalten. Server-Port und Server-Socket können in der Regel auch ihre Standardwerte behalten. Falls der Punkt PHP Extension to use den Eintrag mysqli enthält, ist diese Schnittstelle wahrscheinlich verfügbar und sollte bevorzugt verwendet werden.

Unter Authentication type sollten Sie die Option http wählen; config ist zu unsicher und unflexibel, da die Benutzerdaten dabei dauerhaft in der Konfigurationsdatei gespeichert werden. Entfernen Sie den Wert root aus dem Feld User for config auth, denn bei der HTTP-Authentifizierung kann sich jeder zulässige MySQL-Benutzer anmelden. In die Felder phpMyAdmin Control user und phpMyAdmin Control user password werden nun die Daten des soeben angelegten Benutzers `phpcontroller` eingetragen. Klicken Sie auf Add, um den neuen Server hinzuzufügen.

Bei Bedarf können Sie sich nun weitere Konfigurationsmöglichkeiten anschauen. Dabei geht es um `phpMyAdmin`-Feineinstellungen wie Zeichensätze, Sicherheitseinstellungen oder Layout. Zum Schluss müssen Sie im Abschnitt Configuration die Option Save anklicken.

Verschieben Sie nun die neu erstellte Konfigurationsdatei in das `phpMy-Admin-` Stammverzeichnis. Wenn Sie sich in diesem befinden, lautet die entsprechende Eingabe unter UNIX:

```
# mv config/config.inc.php.
```

Unter Windows:

```
> move config\config.inc.php.
```

Danach können Sie das leer gewordene *config*-Verzeichnis löschen.

Aktualisieren Sie die Ansicht des Browser-Fensters, in dem `phpMyAdmin` angezeigt wird. Geben Sie im HTTP-Anmelde-Dialog die Daten eines MySQL-Benutzers ein. Sie gelangen auf die `phpMyAdmin`-Startseite. Hier müssen Sie zuerst im linken Frame eine Datenbank auswählen oder in der Mitte eine neue erstellen. Danach geht die Arbeit mit dem Tool recht intuitiv vonstatten.

### 15.1.5 PHP-Programmierung

Dieser Abschnitt ist weder eine umfassende Einführung noch eine ausführliche Referenz zur Programmiersprache PHP. Beides würde den Rahmen dieses Buches sprengen. Bei Galileo Computing ist aber eine ganze Reihe von PHP-5-Büchern mit unterschiedlichen Schwerpunkten erschienen. Im Übrigen ist auch die

Online-Dokumentation von PHP unter *http://www.php.net/* sehr ausführlich und recht gut strukturiert.

Wenn Sie PHP programmieren, brauchen Sie nicht – wie bei der CGI-Programmierung in Perl und anderen Sprachen – den HTML-Code durch Ausgabebefehle zu erzeugen. Stattdessen können Sie an einer beliebigen Stelle eines HTML-Dokuments einen PHP-Block einfügen, in dem der PHP-Interpreter nach PHP-Anweisungen sucht und diese ausführt.[3] Das impliziert, dass jedes HTML-Dokument – abgesehen von der Dateiendung – ein gültiges PHP-Skript ist, auch wenn der PHP-Interpreter darin noch nichts zu tun hat. Ein PHP-Block sieht folgendermaßen aus:

```
<?php

   ANWEISUNG1;
   [ANWEISUNG2;
    ...]

?>
```

Hier ein konkretes Beispiel, das die berühmten Worte »Hallo Welt!« auf die Website ausgibt:

```
<?php

   echo ("Hallo Welt!");

?>
```

Der PHP-Block kann übrigens an jeder beliebigen Stelle des HTML-Dokuments stehen – wenn Sie möchten, sogar innerhalb der Anführungszeichen des Werts eines HTML-Attributs. Angenommen, Sie haben irgendwo zu Beginn Ihres Skripts eine Variable namens `$farbe` definiert (z.B. aus einem Cookie ausgelesen), können Sie sie folgendermaßen als Seitenhintergrundfarbe setzen:

```
<body bgcolor="<?php echo ($farbe); ?>">
```

Wie Sie sehen, sind die Zeilenumbrüche im PHP-Block optional – der Browser erhält ohnehin nur die Ausgabe des `echo()`-Befehls, also beispielsweise Folgendes:

```
<body bgcolor="#00FFFF">
```

3 Dieser vermeintliche Vorteil wird allerdings zum Nachteil, wenn Sie codelastige Webanwendungen schreiben – sie werden schnell sehr unübersichtlich. Hier kann die Auslagerung in `include`-Dateien (siehe an entsprechender Stelle) helfen.

Sprachformalitäten von PHP – die Definition von Variablen, Funktionen, Klassen, Methoden und Objekten – können hier nicht explizit behandelt werden. Sie werden in den Beispielen einfach ohne weitere Erläuterung verwendet. Wenn Sie noch nie in PHP programmiert haben, sollten Sie eine der erwähnten Informationsquellen konsultieren.

### Formulardaten auslesen

In PHP ist der Zugriff auf Formulardaten sehr einfach: Das globale Array `$_GET` stellt die mithilfe einer `GET`-Anfrage versandten Inhalte eines Query-Strings zur Verfügung, während `POST`-Daten in dem Array `$_POST` bereitstehen. In PHP gibt es, anders als in Perl, keinen Unterschied zwischen Array und Hash (die Indizes von Arrays können Zahlen oder Strings sein).

Angenommen, ein Formular enthielt die Felder `kunde` und `anschrift` und wurde per `POST`-Anfrage an das aktuelle Skript geschickt. Sie können folgendermaßen auf die beiden Felder zugreifen:

```
$kunde = $_POST['kunde'];
$anschrift = $_POST['anschrift'];
```

Ein Problem ergibt sich allerdings, wenn ein Benutzer eines der Felder nicht ausgefüllt hat. Ein vollständiger Test für das Feld `kunde`, der zu früheren und aktuellen PHP-Versionen kompatibel ist und der Variablen `$kunde` in jedem Fall einen Wert zuweist, sieht beispielsweise so aus:

```
$kunde = 'NO_NAME';  // oder anderen Standardwert einfügen
if (isset($_POST['kunde']) && $_POST['kunde'] != '') {
   $kunde = $_POST['kunde'];
}
```

Die Funktion `isset()` überprüft, ob eine Variable überhaupt definiert wurde. In älteren PHP-Versionen liefert `isset()` aber unter Umständen für jedes Formularfeld `true`, weil das Array selbst existiert. Deshalb wird zusätzlich verlangt, dass das Feld nicht leer sein darf.

Ein solcher Test ist für alle Formularfelder zu empfehlen. Das bedeutet bei sehr vielen Feldern natürlich eine Menge Schreibarbeit. Deshalb empfiehlt es sich, zum Einlesen von Formularfeldern folgende Funktion zu benutzen, die versucht, den Wert des Feldes mit der Bezeichnung `$field` auszulesen, und andernfalls den Standardwert `$default` verwendet (der letztgenannte Wert ist optional, da er seinerseits den leeren String als Standardwert erhält):

```
function cgi_param($field, $default = '')
{
```

```
    // Standardwert setzen
    $value = $default;
    $http_meth = $_SERVER['REQUEST_METHOD'];  // GET, POST?
    if ($http_meth == 'GET') {
       if (isset($_GET[$field]) && $_GET[$field] != '') {
          $value = $_GET [$field];
       }
    } elseif ($http_meth == 'POST') {
       if (isset ($_POST[$field]) && $_POST[$field] != '') {
          $value = $_POST[$field];
       }
    }
    return $value;
}
```

Die Funktion ähnelt der in Kapitel 14, »CGI«, vorgestellten Perl-CGI-Methode `param()`; diese bietet allerdings nicht die Möglichkeit, einen Vorgabewert festzulegen. Wahrscheinlich haben Sie auch die Zeile bemerkt, in der das globale Array `$_SERVER` verwendet wird, um die CGI-Umgebungsvariable `REQUEST_METHOD` auszulesen. Dies ist die PHP-Standardmethode, um auf die in Kapitel 14, »CGI«, besprochenen Umgebungsvariablen zuzugreifen.

Mit `cgi_param()` reduziert sich der Aufwand zum Auslesen der beiden Felder `kunde` und `anschrift` auf die folgenden zwei Zeilen:

```
$kunde = cgi_param('kunde', 'NO_NAME');
$anschrift = cgi_param('anschrift', 'NO_PLACE');
```

Es ist davon auszugehen, dass Sie diese Funktion ziemlich häufig benötigen werden. Deshalb lohnt es sich, sie in einer externen Importdatei abzulegen. Auch PHP-Anweisungen müssen in einer solchen Datei in einem `<?php ... ?>`-Block stehen. Die Datei erhält üblicherweise die doppelte Dateiendung `.inc.php` und wird folgendermaßen eingebettet (der Pfad ist relativ zum Verzeichnis des aktuellen PHP-Skripts):

```
include_once("php_tools.inc.php");
```

### Datei-Uploads

HTML bietet ein spezielles Formularfeld, das Benutzern den Versand einer lokalen Datei mit dem Formular ermöglicht. Es handelt sich um ein `<input>`-Element mit dem Attribut `type="file"`. Damit der Upload funktionieren kann, muss das Formular mit der HTTP-Methode `POST` und der speziellen MIME-Codierung `multipart/form-data` versandt werden. PHP verarbeitet optional ein zusätzliches Formularfeld mit der Bezeichnung `MAX_FILE_SIZE`, das die maximal zulässige

Dateigröße in Byte enthält und meist als `Hidden`-Formularfeld gesetzt wird. Wenn der angegebene Wert größer ist als die *php.ini*-Einstellung `upload_max_filesize` (siehe dort), wird er ignoriert.

Das folgende Beispiel definiert ein Formular für den Versand einer GIF-Bilddatei (dies wird erst auf dem Server geklärt!) von maximal 400 KByte Größe:

```
<form action="bildupload.php" method="post" enctype="multipart/
form-data">
  <input type="hidden" name="MAX_FILE_SIZE" value="409600" />
  Bitte eine GIF-Bilddatei w&auml;hlen:<br />
  <input type="file" name="bild" />
  <br />
  <input type="submit" value="Abschicken" />
</form>
```

Die Datei lässt sich in PHP über das globale Array `$_FILES` auslesen; der Index ist der Feldname. Das entsprechende Array-Element ist wiederum ein Array mit folgenden Elementen:

- `tmp_name`: der temporäre Dateiname, unter dem der Server die hochgeladene Datei gespeichert hat
- `type`: der MIME-Type der Datei (aus der Dateiendung oder aus dem `Content-Type`-Header des Formularabschnitts ermittelt)
- `size`: Größe der Datei oder 0 bei einem Fehler
- `name`: ursprünglicher Pfad und Dateiname der Datei auf dem Client-Rechner
- `error`: Fehlercode bei einem Upload-Fehler. Seit PHP 4.3.0 sind die angegebenen symbolischen Konstanten definiert, in früheren Versionen nur die zugehörigen numerischen Werte:
  - `UPLOAD_ERR_OK` (0): kein Fehler
  - `UPLOAD_ERR_INI_SIZE` (1): Datei ist größer als die in *php.ini* angegebene Höchstgrenze `upload_max_filesize`.
  - `UPLOAD_ERR_FORM_SIZE` (2): Datei ist größer als die mit dem Formular übertragene Höchstgrenze `MAX_FILE_SIZE`.
  - `UPLOAD_ERR_PARTIAL` (3): Die Datei wurde nur zum Teil hochgeladen.
  - `UPLOAD_ERR_NO_FILE` (4): Es wurde gar keine Datei hochgeladen.

Das folgende PHP-Fragment kopiert die hochgeladene Datei unter dem Namen *bild.gif* in das Verzeichnis des Skripts selbst, falls es sich um eine Bilddatei vom Typ GIF handelt:

```
if (is_uploaded_file($_FILES['bild']['tmp_name'])) {
   if ($_FILES['bild']['type'] == 'image/gif') {
      move_uploaded_file ($_FILES['bild']['tmp_name'],
           "bild.gif");
   } else {
      echo ("Falscher Dateityp!<br />");
   }
} else {
   echo ("Upload-Fehler (Datei zu gro&szlig;)? <br />");
}
```

### Sessions

Eines der interessantesten Web-Features von PHP ist das *Session-Tracking*. Wie Sie wissen, ist HTTP ein zustandsloses Protokoll – jede Client-Anfrage steht für sich allein. Für größere Webanwendungen, die sich über viele Seiten erstrecken, wird deshalb ein Verfahren zur Datenweitergabe an nachfolgende Anfragen benötigt. PHP stellt sehr praktische Funktionen zur Verwaltung von Session-Daten bereit.

Beachten Sie, dass Sie Session-Befehle in einen PHP-Block zu Beginn Ihres Skripts setzen müssen: Session-Daten werden als Cookies übertragen, wenn der Client dies zulässt, ansonsten über eine Session-ID im Query-String. Beides erfordert, dass das eigentliche geparste Dokument noch nicht begonnen hat, sodass mit dem ersten Zeichen Ihres Skripts die Sequenz `<?php` einsetzen muss.

Jedes Skript, das auf Session-Daten zugreifen soll, muss zunächst folgende Anweisung enthalten:

```
session_start();
```

Anschließend können Sie das spezielle Array `$_SESSION` mit Name-Wert-Paaren bestücken oder Werte von einer zuvor besuchten Seite daraus lesen. Aus dem soeben erwähnten Grund muss das Lesen vor dem Schreibzugriff erfolgen:

```
// Stückzahl auslesen
$stueckzahl = $_SESSION['stueck'];
// GET-Formularfeld zahlungsart auslesen
$zahlungsart = $_GET['zahlungsart'];
// Zur zukünftigen Verwendung in Session speichern
$_SESSION['zahlungsart'] = $zahlungsart;
```

### Cookies

Für die meisten Anwendungen ist die Verwendung von Session-Daten gegenüber Cookies vorzuziehen. Der einzige Nachteil von Sessions besteht darin, dass die

Daten beim nächsten Besuch eines Benutzers nicht mehr zur Verfügung stehen. Dies lässt sich oft durch die Speicherung der Daten in einer Datenbank und durch eine persönliche Anmeldung beheben.

Dennoch gibt es Fälle, in denen Cookies praktischer sind – es kann für einen Besucher unter Umständen nützlich sein, Einstellungen, die er auf einer Seite vorgenommen hat, beim nächsten Besuch wieder vorzufinden. Sie sollten allerdings niemals eine Webanwendung programmieren, die von Cookies abhängt: Aufgrund des Missbrauchs, der häufig zu Werbe- und Usertracking-Zwecken mit Cookies getrieben wird, schalten viele Benutzer sie generell ab. In PHP wird ein Cookie mithilfe der folgenden Funktion gesetzt:

```
setcookie($name, $wert[, $verfallsdatum[, $pfad
     [, $domain[, $secure]]]])
```

Hier eine Übersicht über die Parameter:

- `$name`: der Name des Cookies, über den es später wieder abgefragt werden kann
- `$wert`: der Wert, den Sie für den entsprechenden Namen festlegen möchten
- `$verfallsdatum`: Der Verfallszeitpunkt für das Cookie in Sekunden seit `EPOCH`. Sie können einfach `time() + $sekundenzahl` verwenden, da die Funktion `time()` den aktuellen Zeitpunkt in diesem Format liefert. Wenn Sie diesen Wert weglassen, erzeugen Sie automatisch ein Session-Cookie, das nur während der aktuellen Client-Sitzung gilt.
- `$pfad`: der URL-Pfad, unter dem das Cookie zur Verfügung steht
- `$domain`: die Domain, unter der das Cookie gültig ist
- `$secure`: Wenn Sie diesen Wert auf `1` setzen, wird das Cookie nur über eine gesicherte HTTPS-Verbindung übertragen.

Wie Sie wissen, wird ein Cookie als HTTP-Header gesetzt. Deshalb muss dieser Befehl – genau wie die Session-Anweisungen – vor jeglichem HTML-Code in der PHP-Datei stehen. Die folgende Anweisung setzt ein sieben Tage gültiges Cookie namens `lastvisit`, dessen Wert die aktuelle Uhrzeit ist:

```
setcookie ("lastvisit", time(), time() + 7 * 24 * 60 * 60);
```

Die Cookies, die der Client bei der Anfrage mitgeschickt hat, können Sie aus dem globalen Array `$_COOKIE` lesen. Dieses Beispiel liest das Cookie `lastvisit` und speichert seinen Wert in einer gleichnamigen Variablen:

```
$lastvisit = $_COOKIE['lastvisit'];
```

Diese Variable lässt sich im weiteren Verlauf des Skripts etwa so verwenden:

```
$lastvisitformat = date ("d.m.Y, H:i", $lastvisit);
echo ("Ihr letzter Besuch: $lastvisitformat <br />");
```

Die Funktion `date ($format, $sek_seit_epoch)` erzeugt formatierte Datumsangaben. Die Buchstaben in `$format` entsprechen zum größten Teil `strftime()`, allerdings ohne Prozentzeichen. Einen Überblick über die `strftime()`-Formatangaben finden Sie übrigens in Kapitel 11, »Logging«.

### MySQL-Zugriff – klassisch

PHP kann mit zahlreichen verschiedenen Datenbanken zusammenarbeiten. Hier wird die Verwendung von MySQL in PHP-Skripten vorgestellt, weil dies die beliebteste Kombination ist. Wie bereits erwähnt, gibt es zwei verschiedene PHP-Schnittstellen für den Zugriff auf MySQL-Datenbanken: die klassische `mysql`-Schnittstelle und die Neuentwicklung `mysqli`. Letztere existiert nur in PHP 5, arbeitet nur mit MySQL ab Version 4.1 zusammen und ermöglicht die objektorientierte Programmierung von Datenbankanwendungen.

Hier wird zunächst einmal die Vorgehensweise für die klassische Schnittstelle vorgestellt; im weiteren Verlauf lernen Sie dieselben Anweisungen für `mysqli` kennen.

Als Erstes müssen Sie eine Verbindung zum MySQL-Server herstellen. Dies funktioniert mithilfe der klassischen Schnittstelle so:

```
$connID = mysql_connect($host[, $user[, $passwort]]);
```

Angenommen, der Datenbank-Server läuft auf demselben Host wie Apache, der Benutzername ist `dbuser` und das Passwort `strengGeheim`. Dann wird der Befehl folgendermaßen aufgerufen:

```
$connID = mysql_connect ("localhost", "dbuser",
     "strengGeheim");
```

Sie sollten es bei einem Produktivsystem dringend vermeiden, anonyme Zugriffe auf den Datenbank-Server zuzulassen. Es sollte also keine Webanwendung geben, bei der ein Zugriff auf MySQL ohne Benutzernamen und Passwort möglich ist.

Die Verbindungskennung `$connID` ist übrigens im Erfolgsfall ein positiver Integer, andernfalls erhält die Variable den Wert 0. Deshalb könnte die konkrete Verwendung folgendermaßen aussehen:

```
$connID = mysql_connect ("localhost", "dbuser",
     "strengGeheim");
if ($connID) {
```

```
   // Datenbankverarbeitung ...
} else {
   echo ("Fehler: Kein Datenbankzugriff m&ouml;glich!<br />");
}
```

Als Nächstes müssen Sie die Datenbank auswählen, auf die zugegriffen wird. Dies geschieht mithilfe des Befehls `mysql_select_db()`. Der erste Parameter ist der Name der gewünschten Datenbank. Optional können Sie die Verbindungskennung angeben – dies ist aber nur erforderlich, wenn mehrere Datenbankverbindungen bestehen. Beispiel:

```
mysql_select_db("buecher");
// Falls mehrere Datenbankverbindungen geöffnet sind:
mysql_select_db("buecher", $connID);
```

Nun können Sie der Datenbank SQL-Abfragen senden. Dies geschieht mithilfe der Funktion `mysql_query($abfrage[, $connID])`. Die Verbindungskennung braucht auch hier wieder nur angegeben zu werden, wenn Sie mehrere Datenbankverbindungen gleichzeitig verwenden. Der Rückgabewert ist eine Ergebniskennung, deren Inhalt Sie mit weiteren Funktionen auslesen können, oder `false`, wenn die Abfrage fehlschlägt. Im letztgenannten Fall liefert die Funktion `mysql_error()` eine Fehlerbeschreibung.

Hier ein Beispiel für eine Auswahlabfrage:

```
$result = mysql_query(
      "SELECT * FROM buecher WHERE verlag LIKE "Galileo%");
```

Die einzelnen Zeilen des Ergebnisses lassen sich mithilfe von `mysql_fetch_row($result)` oder `mysql_fetch_assoc($result)` auslesen. Trotz der unterschiedlichen Bezeichnungen liefern beide ein Array. Der Unterschied ist, dass die Indizes bei `mysql_fetch_row()` numerisch und bei `mysql_fetch_assoc()` die eigentlichen Feldnamen sind. Sie können beide Funktionen in einer `while()`-Schleife aufrufen, weil sie `false` zurückgeben, falls kein weiterer Datensatz mehr vorhanden ist. Hier ein Beispiel für `mysql_fetch_row()`:

```
echo ("<table border=\"2\"><tr>");
echo ("<th>Autor</th><th>Titel</th><th>Verlag</th>");
echo ("</tr>");
while ($record = mysql_fetch_row($result)) {
   $autor = $record[0];
   $titel = $record[1];
   $verlag = $record[2];
   echo ("<tr>");
   echo ("<td>$autor</td>");
   echo ("<td>$titel</td>");
```

```
   echo ("<td>$verlag</td>");
   echo ("</tr>");
}
echo ("</table>");
```

Mit `mysql_fetch_assoc()` sieht der relevante Ausschnitt der `while()`-Schleife dagegen so aus:

```
while ($record = mysql_fetch_assoc($result)) {
   $autor = $record['autor'];
   $titel = $record['titel'];

   $verlag = $record['verlag'];
   // ... wie gehabt ...
}
```

Bei den unterschiedlichen Arten von Änderungsabfragen (`UPDATE`, `INSERT`, `DELETE` usw.) gibt es natürlich kein Ergebnis in Form zurückgegebener Datensätze. Sie können über die Funktion `mysql_affected_rows()` erfahren, wie viele Datensätze geändert (beziehungsweise hinzugefügt oder gelöscht) wurden. Beispiel:

```
// Neues Buch hinzufügen
$result = mysql_query(
     "INSERT INTO buecher VALUES ('9783827326362',
     'Kofler', 'MySQL 5', 'Addison-Wesley')");
// Hat es funktioniert?
if (mysql_affected_rows() == 1) {
   echo ("Buch erfolgreich hinzugef&uuml;gt.<br />");
} else {
   echo ("Buch konnte nicht hinzugef&uuml;gt werden.<br />");
}
```

Wenn Sie eine Abfrage durchführen, die besonders viele Datensätze liefert, sollten Sie anschließend den Speicher leeren, den das Ergebnis belegt hat:

```
mysql_free_result($result);
```

Zum Schluss sollten Sie die Datenbankverbindung ordnungsgemäß schließen:

```
mysql_close($connID);   // $connID entfällt, wenn eindeutig
```

#### Die mysqli-Schnittstelle

Die neuen `mysqli_*`-Anweisungen können sowohl prozedural als auch objektorientiert verwendet werden. Wenn Sie die prozedurale Schreibweise verwenden, unterscheiden sie sich nicht sehr stark von den gerade vorgestellten klassischen `mysql_*`-Funktionen. Die wichtigste Besonderheit ist, dass Sie die Verbindungs-

kennung nicht mehr weglassen können. Deshalb ist der objektorientierte Zugriff zu bevorzugen – hier wird für jede Verbindung ein eigenes MySQL-Objekt erzeugt.

Hier sehen Sie zunächst sämtliche Anweisungen aus dem `mysql`-Beispiel unter Verwendung der prozeduralen Form der `mysqli`-Schnittstelle:

```
$connID = mysqli_connect("localhost", "dbuser",
     "strengGeheim", "buecher");
$result = mysql_query($connID,
     "SELECT * FROM buecher WHERE verlag LIKE "Galileo%");
echo ("<table border=\"2\"><tr>");
echo ("<th>Autor</th><th>Titel</th><th>Verlag</th>");
echo ("</tr>");
while ($record = mysqli_fetch_row($result)) {
   $autor = $record[0];
   $titel = $record[1];
   $verlag = $record[2];
   echo ("<tr>");
   echo ("<td>$autor</td>");
   echo ("<td>$titel</td>");
   echo ("<td>$verlag</td>");
   echo ("</tr>");
}
echo ("</table>");
mysqli_free_result($result);
mysqli_close($connID);
```

Die objektorientierte Variante von `mysqli` verursacht weniger Schreibarbeit und ist erheblich praktischer. Da die Verbindung und das Ergebnis Objekte sind, entfällt beim Aufruf der verschiedenen Methoden das Präfix `mysqli_` und auch die Angabe der Verbindungskennung. Hier sehen Sie das gesamte Beispiel noch einmal in objektorientierter Syntax:

```
$conn = new mysqli("localhost", "dbuser", "strengGeheim",
     "buecher");
$result = $conn->query(
     "SELECT * FROM buecher WHERE verlag LIKE "Galileo%");
echo ("<table border=\"2\"><tr>");
echo ("<th>Autor</th><th>Titel</th><th>Verlag</th>");
echo ("</tr>");
while ($record = $result->fetch_row()) {
   $autor = $record[0];
   $titel = $record[1];
   $verlag = $record[2];
   echo ("<tr>");
```

```
   echo ("<td>$autor</td>");
   echo ("<td>$titel</td>");
   echo ("<td>$verlag</td>");
   echo ("</tr>");
}
echo ("</table>");
$result->free();
$conn->close();
```

Bitte beachten Sie bei Änderungsabfragen mithilfe der objektorientierten Schreibweise von `mysqli`, dass `affected_rows` hier eine Eigenschaft und keine Methode ist – hinter dem Namen dürfen keine leeren Klammern stehen. Beispiel:

```
// Buch einfügen, mysqli
$result = $conn->query(
             "INSERT INTO buecher VALUES ('9783827326362',
              'Kofler', 'MySQL 5', 'Addison-Wesley')");
// Hat es geklappt?
if ($conn->affected_rows == 1) {
   echo ("Buch erfolgreich hinzugef&uuml;gt.<br />");
} else {
   echo ("Buch konnte nicht hinzugef&uuml;gt werden.<br />");
}
```

## 15.2 mod_perl

`mod_perl`, entwickelt seit 1997 von Doug McEachern, integriert den Perl-Interpreter als Modul in den Apache-Webserver. Natürlich werden Perl-Skripte auf diese Weise erheblich schneller ausgeführt als über die CGI-Schnittstelle. Außerdem ergeben sich daraus noch zwei weitere nützliche Vorteile:

- Sie können innerhalb der Apache-Konfigurationsdateien Perl-Code verwenden. Auf diese Weise können Sie verschiedene Konfigurationsschritte automatisieren oder flexibel von Bedingungen abhängig machen.
- Es besteht die Möglichkeit, Apache-Module in Perl zu programmieren; `mod_perl` stellt eine praktische Schnittstelle zur Apache-API bereit. Auf diesen speziellen Aspekt des Moduls wird hier allerdings – bis auf ein kurzes Beispiel – nicht näher eingegangen.

Speziell für den Einsatz mit Apache 2 wurde die völlig neue Version `mod_perl` 2.0 geschrieben. Beachten Sie, dass Sie für `mod_perl` 2.0 eine Perl-Version ab 5.6 benötigen, da erst diese Version die für einige Apache-MPM-Module erforderliche Thread-Unterstützung besitzt. Noch empfehlenswerter ist die aktuelle Version Perl 5.10.

### 15.2.1 Installation von mod_perl

Stellen Sie für die mod_perl-Installation sicher, dass Sie mod_perl 2.0 herunterladen – die alte Version arbeitet nicht mit Apache 2 zusammen. Auf der DVD zum Buch finden Sie natürlich das zum Druckzeitpunkt neueste Release der Version 2.0.

#### Installation unter UNIX

Diese Installationsanleitung erklärt, wie Sie mod_perl als dynamisches Apache-Modul kompilieren und installieren können. Sie erfordert, dass Ihr Apache-Webserver mit DSO-Unterstützung kompiliert wurde – das Modul mod_so muss aktiv sein. Wenn Sie Ihren Server lieber mit einer statischen mod_perl-Version neu kompilieren möchten, konsultieren Sie die mod_perl-Originaldokumentation unter *perl.apache.org*.

Selbstverständlich muss Perl auf Ihrem System installiert sein – da dies bei fast allen modernen UNIX-Systemen der Fall ist, wird hier nicht näher auf die Installation eingegangen. Besorgen Sie sich im Zweifelsfall die aktuelle Version aus dem CPAN (*www.cpan.org/ports* für Binaries oder *www.cpan.org/src* für den Quellcode zum Selbstkompilieren). Zur Installation aus dem Quellcode-Paket müssen Sie zunächst die Archivdatei auspacken und anschließend das Konfigurationsskript aufrufen:

```
# tar -xzvf perl-5.14.2.tar.gz
# cd perl-5.14.2
# ./configure
```

An dieser Stelle erleben Sie eine kleine Überraschung: configure ist hier kein Autoconf-Skript, sondern ein interaktives Kommandozeilenprogramm, das Ihnen zahlreiche Fragen zur Konfiguration stellt. Das ist praktisch, weil Sie keine langen Optionslisten anzugeben brauchen; außerdem wird jeder Schritt genau erklärt. Nach der Konfiguration werden wie üblich die Kompilierung und die Installation eingeleitet:

```
# make
# make install
```

Nun geht es an die eigentliche mod_perl-Installation. Zunächst einmal müssen Sie, wie üblich, die GNU-ZIP-komprimierte Archivdatei entpacken. Dies funktioniert mit folgendem Befehl:

```
# tar -xzvf mod_perl-2.0-current.tar.gz
```

Anschließend können Sie in das Verzeichnis wechseln, das durch das Entpacken des Archivs neu angelegt wurde:

```
# cd mod_perl-2.0-current.tar.gz
```

Der nächste Schritt besteht darin, das Perl-Skript `Makefile.PL` zu aktivieren. Seine Aufgabe entspricht bei Perl-Modulen dem `configure`-Skript bei `Autoconf`-Installationen. Leider enthält die Datei selbst keine Optionsliste; einige Hinweise finden Sie aber in der Textdatei `INSTALL`. In den meisten Fällen funktioniert folgender Aufruf ohne Probleme:

```
# perl Makefile.PL MP_APXS=/usr/local/apache2/bin/apxs \
MP_INST_APACHE2=1
```

Natürlich müssen Sie */usr/local/apache2/bin/apxs* je nach `Apache`-Layout an den Pfad Ihrer `apxs`-Version anpassen. Die Option `MP_INST_APACHE2=1` ist wichtig, wenn in Ihren Perl-Verzeichnissen gleichzeitig `mod_perl 1.0` installiert ist – dies ist bei einigen realen Servern immer noch der Fall. Wenn Ihr Apache-Webserver – wie bei dem Installationslayout `Apache` – komplett unter einem gemeinsamen Stammverzeichnis installiert ist, brauchen Sie nicht mit `MP_APXS` den Pfad Ihres `apxs`-Skripts anzugeben, sondern können stattdessen die Option `MP_AP_PREFIX` verwenden, um dieses Verzeichnis zu setzen:

```
# perl Makefile.PL MP_AP_PREFIX=/usr/local/apache2 \
MP_INST_APACHE2=1
```

Als Nächstes erfolgt in jedem Fall die eigentliche Kompilierung:

```
# make
```

Optional können Sie vor der eigentlichen Installation einen Testlauf durchführen, um zu überprüfen, ob alles in Ordnung ist:

```
# make test
```

Wenn der Test keine Probleme bereitet hat oder wenn Sie darauf verzichtet haben, werden die kompilierten Dateien zum Schluss installiert:

```
# make install
```

Nun müssen Sie einige Änderungen an Ihrer Konfigurationsdatei (*httpd.conf*) vornehmen. Zunächst einmal muss natürlich das neu kompilierte DSO-Modul geladen werden. Dazu müssen Sie folgende Direktive setzen:

```
LoadModule perl_module modules/mod_perl.so
```

Natürlich wird das Verzeichnis *modules* wie üblich durch Ihr Modulverzeichnis ersetzt. Als Nächstes lohnt es sich, die folgende Zeile einzufügen (die Direktive `PerlModule` wird durch `mod_perl` definiert und dient dem Import eines in Perl geschriebenen Moduls in *httpd.conf*):

```
PerlModule Apache2
```

Andernfalls müssen Sie nämlich jedes `mod_perl`-Skript mit der folgenden Importanweisung versehen:

```
use Apache2();
```

Da es im Internet (und im Repertoire vieler Website-Betreiber) noch einige Skripte für `mod_perl` 1.0 gibt, kann es praktisch oder sogar notwendig sein, die folgende Direktive einzufügen – sie sorgt für die Kompatibilität mit diesen älteren Skripten:

```
PerlModule Apache::compat
```

Nun müssen Sie natürlich noch dafür sorgen, dass Apache Skripte mit der Dateiendung `.pl` überhaupt durch `mod_perl` verarbeiten lässt. Beispielsweise können Sie folgendermaßen ein externes Verzeichnis für `mod_perl`-Skripte einbinden:

```
# Verzeichnis für einfache mod_perl-Skripte
Alias /perl/ /usr/local/apache2/perl/
<Location /perl/>
    SetHandler perl-script
    PerlResponseHandler ModPerl::Registry
    PerlOptions +ParseHeaders
    Options +ExecCGI
</Location>
```

Damit haben Sie ein Verzeichnis, in dem klassische CGI-Skripte ohne großartige Änderungen durch `mod_perl` ausgeführt werden – beispielsweise läuft das kleine Taschenrechnerskript aus Kapitel 14, »CGI«, ohne Schwierigkeiten im angegebenen Verzeichnis. Die `Perl*`-Direktiven werden im Folgenden genauer erläutert. Wenn Sie dafür sorgen möchten, dass Apache 2 Dateien mit der Endung `.pl` in allen Verzeichnissen als `mod_perl`-Skripte betrachtet, können Sie folgende Variante in Ihre Konfigurationsdatei schreiben:

```
# .pl allgemein als mod_perl betrachten
AddHandler perl-script .pl
<Files *.pl>
  PerlResponseHandler ModPerl::Registry
  PerlOptions +ParseHeaders
  Options +ExecCGI
</Files>
```

Darüber hinaus ist es oft erforderlich, eine `mod_perl`-Startup-Datei zu definieren. Diese Datei wird beim Start oder Neustart von Apache aufgerufen. Sie wird in der Regel zur Definition von `Include`-Pfaden verwendet, in denen Perl nach Zusatz-

modulen suchen soll. Auch das Vorausladen von Perl-Modulen, die Sie regelmäßig benötigen, kann hier stattfinden. Eine Startup-Datei wird mithilfe der Direktive `PerlRequire` eingebettet, beispielsweise so:

```
PerlRequire /usr/local/apache2/perl/startup.pl
```

Das folgende kurze Beispiel für eine Startup-Datei definiert zwei Verzeichnisse als `Include`-Pfad – `mod_perl` sucht bei jeder `use`- oder `require`-Anweisung zusätzlich in diesen Verzeichnissen nach Dateien oder Modulen, die eingebunden werden sollen:

```
#!/usr/bin/perl
use lib qw(/usr/local/apache2/perl /usr/local/mywebtools);
1;
```

Zum Abschluss der Installationsanweisung soll noch gezeigt werden, wie Sie anstelle eines einfachen Skripts ein Perl-Modul erzeugen können, das eine Client-Anfrage beantwortet. Das hat wohlgemerkt noch nicht viel mit den vielfältigen Möglichkeiten zur Modulprogrammierung mit `mod_perl` zu tun.

Das Skript für dieses Beispiel muss unter dem Dateinamen *MPTest.pm* in ein Unterverzeichnis namens *MyApache* unterhalb eines Verzeichnisses in Ihrem `Include`-Pfad kopiert werden. Falls Sie die soeben beschriebene Startup-Datei eingerichtet haben, können Sie es beispielsweise unter */usr/local/apache2/perl/MyApache/MPTest.pm* ablegen. Das Modulskript selbst sieht folgendermaßen aus:

```
package MyApache::MPTest;

use strict;
use warnings;

use Apache::RequestRec ();
use Apache::RequestIO ();

use Apache::Const -compile => qw(OK);

sub handler {
    my $r = shift;

    $r->content_type('text/html');
    print << "END_HTML";
<html>
<head>
<title>Nur ein Test</title>
```

```
</head>
<body>
Hallo! Hier ist ${ENV{'SERVER_SOFTWARE'}}!
</body>
</html>
END_HTML

    return Apache::OK;
}
1;
```

Mit folgenden Direktiven können Sie das neue Perl-Modul anschließend in Ihre Konfigurationsdatei einbinden:

```
<Location /mptest>
    SetHandler perl-script
    PerlResponseHandler  MyApache::MPTest
</Location>
```

Wenn Sie nun in Ihren Browser eine URL wie *www.mynet.de/mptest* eingeben, erhalten Sie eine Website mit einer Ausgabe wie dieser:

```
Hallo! Hier ist Apache/2.3.15 (UNIX)!
```

Damit sind die Installation und Grundkonfiguration beendet; auch den ersten Test haben Sie erfolgreich bestanden. Von nun an können Sie `mod_perl` produktiv einsetzen.

#### Installation unter Windows

Unter Windows verläuft die Installation von `mod_perl` in der Regel ein wenig anders als auf einem UNIX-System. Das ist vor allem dann der Fall, wenn Sie die Perl-Distribution ActivePerl von ActiveState Software verwenden, die Sie auf der DVD zum Buch finden oder kostenlos unter *www.activestate.com* herunterladen können. Diese Perl-Variante bietet nämlich das praktische Zusatzprogramm PPM (Perl Package Manager), das auf einfache Weise den Download und die Installation von Perl-Modulen für Windows ermöglicht.

Die Installation von ActivePerl wird durch einen Doppelklick auf die Windows-Installer-Datei (zurzeit *ActivePerl-5.14.2.1402-MSWin32-x86-295342.msi*) in Gang gesetzt.[4] Anschließend müssen Sie nacheinander folgende Schritte durchführen:

4 Informationen zur Verfügbarkeit des Windows Installers in den unterschiedlichen Windows-Versionen finden Sie in Kapitel 4, »Apache kompilieren und installieren«.

- Akzeptieren Sie die Lizenz (die ActiveState Community License ist mit der BSD-Lizenz vergleichbar, bietet also sehr viele Freiheiten).
- Als Nächstes können Sie den Installationsumfang wählen. Wenn Sie sich um Festplattenplatz keine Sorgen zu machen brauchen, sollten Sie den übergeordneten Punkt ActivePerl anklicken und Entire Feature will be installed on local hard drive auswählen. Andernfalls können Sie sich dazu entschließen, Perl ISAPI/PerlScript (Erweiterung für ISAPI-Webserver-Programmierung und Windows Scripting Host) sowie Examples wegzulassen. Außerdem lässt sich auf diesem Bildschirm das Installationsverzeichnis (meist *C:\Perl*) wählen.
- Den Auswahlpunkt Enable PPM3 to send profile info to ASPN können Sie deaktivieren, wenn Sie möchten. Zwar werden keine persönlichen Daten übertragen, aber möglicherweise machen die statistischen Informationen Sie bereits nervös genug.
- Sie sollten dagegen auf jeden Fall die beiden Punkte Add Perl to the PATH environment variable und Create Perl file extension association auswählen. Der erste Punkt sorgt dafür, dass das *bin*-Verzeichnis Ihrer Perl-Installation im `PATH` liegt; damit wird der Interpreter überall gefunden. Die zweite Einstellung bewirkt, dass Perl-Skripte per Doppelklick mit dem Perl-Interpreter ausgeführt werden.
- Zu guter Letzt können Sie nun Install anklicken, um die eigentliche Installation durchzuführen. Diese dauert recht lange, weil der Installationsprozess aus der umfangreichen POD-Dokumentation HTML-Dokumente generiert.

Als Nächstes erfolgt die eigentliche Installation von `mod_perl`. Das Programm PPM lädt die erforderlichen Dateien automatisch herunter und installiert sie anschließend. Falls Sie über einen HTTP-Proxy auf das Web zugreifen, müssen Sie diesen zunächst einstellen. PPM übernimmt keine Proxy-Einstellungen aus Webbrowsern, wertet aber die Umgebungsvariable `HTTP_proxy` aus. Diese können Sie für die aktuelle Konsolensitzung beispielsweise folgendermaßen setzen:

```
> set HTTP_proxy=http://proxy.mynet.de:3128
```

Selbstverständlich müssen Sie den Hostnamen und die Portnummer des Proxys anpassen.

Nun können Sie den `ppm`-Befehl zur Installation aufrufen. Da `mod_perl` für Windows leider nicht in den Standard-Repositorys vorliegt, in denen PPM automatisch sucht, müssen Sie die vollständige URL eingeben:

```
> ppm install http://theoryx5.uwinnipeg.ca/ppms/mod_perl.ppd
```

Im weiteren Verlauf der Installation wird Ihnen mitgeteilt, dass das Apache-Modul `mod_perl.so` benötigt und heruntergeladen wird:

```
The Apache2 module mod_perl.so is needed to complete the
installation, and should be placed in your Apache2 modules
directory. I will now fetch and install this for you.

Fetching http://theoryx5.uwinnipeg.ca/ppms/x86/mod_perl.so ...
done!
```

Nun werden Sie gefragt, wohin das Apache-Modul installiert werden soll. Wenn die Vorgabe in den eckigen Klammern in Ordnung ist, können Sie einfach ↵ drücken. Andernfalls müssen Sie den vollständigen Pfad Ihres *modules*-Verzeichnisses eingeben. Beispiel:

```
Where should mod_perl.so be placed? [C:/Apache2/modules]
C:/Program Files/Apache Foundation/Apache2.2/modules
```

Damit ist die Installation abgeschlossen. Der Installer teilt Ihnen noch mit, welche Änderungen Sie an Ihrer *http.conf* vornehmen müssen. Unbedingt erforderlich sind die beiden folgenden Zeilen:

```
LoadFile "C:/Perl/bin/perl58.dll"
LoadModule perl_module modules/mod_perl.so
```

Anstelle von *C:/Perl/bin* müssen Sie selbstverständlich den korrekten Pfad Ihrer Perl-Installation angeben; die Bibliothek *perl58.dll* ist der eigentliche Perl-Interpreter. Die restlichen Anpassungen erfolgen analog zu der Beschreibung im UNIX-Unterabschnitt.

### 15.2.2 Perl-Zugriff auf MySQL-Datenbanken

In diesem Buch ist leider nicht genügend Platz für umfangreiche `mod_perl`-Programmiertipps, und außerdem lassen sich die Informationen aus Kapitel 14, »CGI«, auch auf einfache `mod_perl`-Skripte übertragen. Dennoch soll an dieser Stelle kurz darauf hingewiesen werden, wie Sie aus Perl-Skripten heraus auf eine MySQL-Datenbank zugreifen können. Als Beispiel dient wieder die bereits bei der MySQL-Installation beschriebene kleine Bücherdatenbank.

Zunächst einmal wird für alle Datenbankzugriffe in Perl-Skripten das Standardmodul `DBI.pm` benötigt. Zu Anfang Ihres Skripts müssen Sie es folgendermaßen einbinden:

```
use DBI;
```

Nun wird mithilfe der DBI-Methode `connect()` ein Datenbankverbindungsobjekt erstellt. Das erste Argument der Methode ist `'dbi:mysql:Datenbank;Hostname'`, gefolgt von Benutzername und Passwort. Dies sieht für das `buecher`-Beispiel so aus:

```
my $conn = DBI->connect('dbi:mysql:buecher;localhost',
     'dbuser', 'strengGeheim');
```

Der nächste Schritt besteht darin, ein Query-Objekt zu erzeugen und die darin enthaltene SQL-Abfrage auszuführen. Beispiel:

```
my $query = $conn->prepare
     ("SELECT * FROM buecher WHERE VERLAG LIKE 'Galileo%'");
$query->execute();
```

Die einzelnen Zeilen des Abfrageergebnisses lassen sich ähnlich auslesen wie in PHP: Die Query-Methode `fetchrow_array()` liefert ein Array mit den Feldern eines Datensatzes zurück und kann in einer `while()`-Schleife eingesetzt werden. Zusammen mit der Konstruktion einer HTML-Tabelle sieht das Ganze etwa so aus:

```
print "<table border=\"1\">\n";
print "<tr>\n";
print "  <th>ISBN</th>\n";
print "  <th>Autor</th>\n";
print "  <th>Titel</th>\n";
print "  <th>Verlag</th>\n";
print "</tr>\n";
while (my @record = $query->fetchrow_array()) {
   print "<tr>\n";
   foreach my $field(@record) {
      print "  <td>$field</td>\n";
   }
   print "</tr>\n";
}
print "</table>\n";
```

Zum Schluss sollten Sie die Datenbankverbindung schließen:

```
$dbi->disconnect();
```

### 15.2.3 Perl in der Apache-Konfigurationsdatei

Wie bereits erwähnt, können Sie direkt in den Apache-Konfigurationsdateien wie *httpd.conf* Perl-Code einsetzen. Dies erlaubt z. B. Schleifen oder Fallentscheidungen in der Konfiguration, was besonders praktisch ist, wenn Sie viele virtuelle Hosts betreiben. Der Code muss in den speziellen Container `<Perl> ... </Perl>` geschrieben werden. Innerhalb dieses Containers ist die normale Schreibweise der Konfigurationsdirektiven nicht zulässig; stattdessen müssen sie als Perl-Anweisungen formuliert werden. Dazu gelten folgende Regeln:

- Einfache Direktiven, die nur einen Wert besitzen, werden als skalare Variablen (mit dem bekannten Präfix `$`) definiert. Beispiel:

```
$DocumentRoot = "/usr/local/apache2";
```

- Direktiven, die mehrere Werte akzeptieren, können Sie als Arrays schreiben, in denen jedes Element einem der Werte entspricht; alternativ ist auch ein einfacher String möglich, in dem die einzelnen Werte durch Leerzeichen getrennt werden. Beispiel:

```
@Options = qw(Indexes FollowSymLinks ExecCGI);    # Array
$Options = "Indexes FollowSymLinks ExecCGI";      # String
```

- Wenn eine bestimmte Direktive mehrmals mit unterschiedlichen Werten vorkommen kann, wird sie als Array definiert, in dem jeder Aufruf dieser Direktive ebenfalls ein Array ist. Hier ein Beispiel:

```
@AddCharset = (
   ['ISO-8859-1' => qw(.iso8859-1 .latin1)],
   ['ISO-8859-2' => qw(.iso8859-2 .latin2 .cen)],
   ['ISO-8859-3' => qw(.iso8859-3 .latin3)]
);
```

- Möchten Sie innerhalb eines `<Perl>`-Containers einen weiteren Container einfügen, dann muss dieser als Hash geschrieben werden, in dem die Direktiven die Schlüssel bilden. Die Werte werden je nach ihrer Art als Strings oder als Arrays geschrieben. Beispiel:

```
$Location{"/test"} = {
   Order => 'deny,allow',
   Deny => 'from all',
   Allow => 'from 192.168.0 localhost'
};
```

Das folgende Beispiel erlaubt denselben vier verschiedenen Netzwerken den Zugriff auf drei verschiedene URL-Pfade:

```
$networks = "192.168.0 192.168.7 192.168.52 196.13";
@paths = qw(/test /info /news/2004);
foreach $path(@paths) {
   $Location{$path} = {
      Order => 'deny,allow',
      Deny => 'from all',
      Allow => "from $networks"
   }
}
```

Beachten Sie, dass alle hier aufgelisteten Beispiele für Apache-Versionen bis einschließlich 2.2 gedacht sind. Es gibt zurzeit noch keine stabile `mod_perl`-Version, die Apache 2.4 unterstützt.

## 15.3 Tomcat

*Tomcat* ist ein weiteres Projekt der Apache Software Foundation. Es handelt sich um einen in Java geschriebenen, voll ausgestatteten Webserver. Natürlich ist er nicht ganz so leistungsfähig und modular erweiterbar wie der Apache-Webserver; für Sites, die fast nur Java-Server-Anwendungen verwenden, ist er trotzdem eine bedenkenswerte Alternative.

Für das Thema dieses Buches ist es interessanter, dass Sie Tomcat auch in Zusammenarbeit mit dem Apache-Webserver einsetzen können. Dazu dient ab Version Apache 2.2 der in Kapitel 13, »Proxy- und Cache-Funktionen«, angesprochene AJP-Proxy oder auch einfach ein Reverse-HTTP-Proxy. Zusätzlich steht das vom Tomcat-Team entwickelte Connector-Modul `mod_jk` zur Verfügung. Alle diese Verfahren werden im Folgenden kurz beschrieben.

Sobald Sie Apache 2 und Tomcat verknüpft haben, dient Tomcat als leistungsfähige Engine für Java Servlets und JavaServer Pages (JSP). Diese beiden Technologien ermöglichen Ihnen den Einsatz der unzähligen Java-Klassenbibliotheken für Ihre Webanwendungen. In diesem Abschnitt werden die Installation von Tomcat und dessen Anbindung an den Apache-HTTP-Server erläutert; anschließend wird ein kleines, lauffähiges JSP-Beispiel vorgestellt.

### 15.3.1 Tomcat installieren

Hier erfahren Sie, wie der Tomcat-Server unter UNIX und Windows installiert wird. Außerdem wird die Anbindung an Ihren Hauptwebserver über `mod_proxy_html` oder `mod_proxy_ajp` beziehungsweise `mod_jk` beschrieben.

Selbstverständlich benötigen Sie zunächst einmal eine Java Virtual Machine, um Tomcat installieren zu können. Um Tomcat auszuführen, brauchen Sie nur das *Java Runtime Environment* (JRE); allerdings können Sie damit noch keine eigenen Servlets kompilieren. Besser ist es deshalb, das vollständige *Java 2 Software Development Kit* (SDK) zu installieren. Sie finden die aktuelle Version des Java 2 SDK in der Variante J2SE (Standard Edition) für verschiedene Plattformen auf der DVD zum Buch oder können sie von *http://java.sun.com* herunterladen. Für die aktuelle Tomcat-Version 7.0 wird Java 6.0 oder neuer benötigt.

Die Installation von Java ist ziemlich einfach und wird auf der genannten Website ausführlich erläutert. Wichtig ist, dass Sie nach der Installation die Umgebungsvariable `CLASSPATH` setzen; sie enthält eine durch Semikola getrennte Liste von Verzeichnissen, in denen die Java Virtual Machine nach Java-Klassen suchen soll.

#### Installation unter UNIX

Auf der DVD zum Buch und im Web unter *http://tomcat.apache.org* wird Tomcat als Binärpaket angeboten. Da der Server in Java geschrieben ist, kann dieses eine Paket für jede Plattform verwendet werden. Die Kompilierung dieses Softwarepakets ist ein wenig komplizierter als bei C-basierter Standardsoftware, deshalb sollten Sie die Binärvariante vorziehen.

Entpacken Sie das Archiv in ein Verzeichnis Ihrer Wahl (z.B. */usr/local*). Die eigentliche Herausforderung unter UNIX besteht darin, den Server als Daemon zu starten. Sie können dafür das mitgelieferte Programm `jsvc` verwenden. Falls Sie Tomcat nach */usr/local* entpackt haben, sodass sich die Dateien unter */usr/local/Tomcat-7.0.22* (beziehungsweise Ihrer aktuellen Version) befinden, können Sie folgende Befehle dafür benutzen:

```
# cd /usr/local/Tomcat-7.0.22
# ./bin/jsvc -Djava.endorsed.dirs=./common/endorsed \
-cp ./bin/bootstrap.jar -outfile ./logs/catalina.out \
-errfile ./logs/catalina.err \
org.apache.catalina.startup.Bootstrap
```

Im Verzeichnis `bin/jsvc/native` finden Sie außerdem das Skript `tomcat.sh`. Sie können es als Startskript verwenden, um Tomcat beim Booten automatisch zu starten. In älteren Tomcat-5-Versionen müssen Sie darin zuerst den Klassennamen `BootstrapService` in `Bootstrap` umbenennen, weil das Skript ursprünglich für Tomcat 4.1 geschrieben wurde.

Falls Sie `mod_jk` als Connector verwenden möchten, müssen Sie dieses Modul nun beschaffen und installieren. Wie üblich wird hier wieder nur auf die DSO-Variante eingegangen. Wenn Sie Apache mit einem statischen `mod_jk` neu kompilieren möchten – wofür es eigentlich keinen vernünftigen Grund gibt –, dann halten Sie sich an die Anleitung in der Online-Dokumentation von Tomcat (*http://tomcat.apache.org/tomcat-7.0-doc*).

Für einige der wichtigsten UNIX-Plattformen (Linux, Mac OS X, FreeBSD und Solaris) sowie für Windows und NetWare gibt es vorkompilierte JK-DSOs. Sie finden sie natürlich auf der DVD zum Buch, aber auch auf der entsprechenden Download-Site des Tomcat-Projekts (*http://tomcat.apache.org/download-connectors.cgi*). Da dieser Link, wie bei allen Apache-Websites üblich, auf den jeweils besten Mirror

verweist, kann hier keine genauere Download-Adresse angegeben werden. Halten Sie sich zur Installation an die beigelegten *README*- oder *INSTALL*-Textdateien – vor allem müssen Sie das Modul `mod_jk.so` in Ihr Apache-Modulverzeichnis (z. B. `/usr/local/apache2/modules`) kopieren. Anschließend müssen Sie die Konfigurationsänderungen vornehmen, die im weiteren Verlauf des Kapitels beschrieben werden und die auch für die Quellcode-Distribution gelten.

Für alle anderen UNIX-Systeme (beispielsweise auch für nicht unterstützte Linux-Distributionen) müssen Sie das Quellcode-Paket verwenden. Es befindet sich ebenfalls auf der DVD zum Buch oder zum Download unter dem Link *JK Source Release* auf der JK-Download-Site (*http://tomcat.apache.org/download-connectors.cgi*). Entpacken Sie zunächst das Archiv:

```
# tar xzvf tomcat-connectors-1.2.32-src.tar.gz
```

Wechseln Sie nun in das neu erstellte Verzeichnis (wahrscheinlich ändert sich der Name ein wenig, wenn Sie eine neuere Version herunterladen):

```
# cd tomcat-connectors-1.2.32-src
```

Im Unterverzeichnis *docs* finden Sie eine kurze HTML-Datei, die Ihnen erklärt, welche Elemente Sie in Ihr Tomcat-Installationsverzeichnis kopieren müssen. Anschließend können Sie in das Unterverzeichnis *native/apache-2.0* wechseln und dort das eigentliche Apache-Modul kompilieren:

```
# cd native/apache-2.0
# ./configure --with-apxs2=/usr/local/apache2/bin/apxs
# make
# cd ../build/jk/apache2
# apxs -n jk -i mod_jk.so
```

Egal, ob Sie das Binär- oder das Quellcode-Paket von `mod_jk` installiert haben, müssen Sie nun in der Konfigurationsdatei *httpd.conf* folgende Zeile einfügen, um das Modul zu laden:

```
LoadModule jk_module modules/mod_jk.so
```

Die Datei *mod_jk.conf* benötigt im Grunde nur eine Zeile. Diese verweist auf die eigentliche Konfigurationsdatei *workers.properties*, die die Laufzeitkonfiguration für Apache 2 enthält. Beispiel:

```
JkSet config.file \
   "/usr/local/apache2/conf/workers.properties"
```

Die vorgefertigte Datei *workers.properties* enthält verschiedene Konfigurationseinträge, auf die hier nicht näher eingegangen werden kann. Wichtig ist dage-

gen vor allem der wie folgt gekennzeichnete Bereich (natürlich leitet die Raute einen Kommentar ein, genau wie in der Apache-Konfiguration):

```
# Map the Tomcat examples webapp to the Web server uri space
[uri:/examples/*]
group=lb
```

`[uri]`-Sektionen in *workers.properties* definieren eine Verknüpfung zwischen einer URL unterhalb des Tomcat-Verzeichnisbaums mit einem gleichnamigen URL-Pfad des Apache-Webservers. Leider enthält die aktuelle Tomcat-Version gar kein Verzeichnis namens *examples* mehr; der Eintrag entstammt einer älteren Version und wurde nicht angepasst. Fügen Sie stattdessen die folgenden Zeilen hinzu, wenn Sie die mit Tomcat gelieferten Servlet- und JSP-Beispiele einer URL Ihrer Apache-2-Website zuordnen möchten:

```
[uri:/servlets-examples/*]
group=lb
[uri:/jsp-examples/*]
group=lb
```

Sie können nun über eine Adresse wie *www.mynet.de/servlets-examples* beziehungsweise *www.mynet.de/jsp-examples* auf die Tomcat-Beispiele zugreifen. Entsprechende Blöcke in der Konfigurationsdatei können Sie auch selbst erstellen. Das folgende Beispiel veröffentlicht den Inhalt des Verzeichnisses `/myservlets`:

```
[uri:/myservlets/*]
group=lb
```

Dieses Verzeichnis müssen Sie zuvor innerhalb Ihrer Tomcat-Installation als Unterverzeichnis von *webapps* erzeugen. Anschließend muss es über die Tomcat-Konfiguration als sogenannter *Context* eingerichtet werden. Dazu sind folgende Schritte erforderlich:

1. Die webbasierte Tomcat-Administrationsoberfläche wird standardmäßig nicht mitinstalliert. Kopieren Sie sie von der DVD, oder laden Sie sie von *http://tomcat.apache.org/download-70.cgi* herunter, indem Sie dort dem passenden Link unter Administration Web Application folgen. Entpacken Sie die Datei in Ihr Tomcat-Installationsverzeichnis.
2. Sie benötigen einen speziellen Tomcat-User, um auf die webbasierte Installation zugreifen zu können. Im *conf*-Verzeichnis Ihrer Tomcat-Installation finden Sie die Datei *tomcat-users.xml*, in der die Benutzerverwaltung stattfindet. Fügen Sie hier folgende Zeile hinzu (oder ändern Sie das Passwort nach Belieben, falls der Eintrag bereits besteht):

```
<user username="admin" password="strengGeheim" roles="admin,
manager"/>
```

Anstelle von `admin` können Sie auch einen anderen Benutzernamen verwenden; wichtig sind nur die Rollen `admin` und `manager`. Anstelle von `strengGeheim` sollten Sie selbstverständlich ein vernünftiges Passwort (eine kryptische Zeichenkombination wie `dZvmG65Q`) verwenden.

3. Starten Sie Ihren Browser, und öffnen Sie die Adresse *http://localhost:8080/admin*.
4. Geben Sie den Benutzernamen (in der Regel `admin`; siehe 2.) und das entsprechende Passwort ein.
5. Es erscheint die Administrationsoberfläche von Tomcat. Hier müssen Sie sich im linken Frame durch die verschachtelte Struktur klicken: Öffnen Sie Tomcat Server • Service (Catalina) • Host (localhost).
6. Auf der Hostseite sehen Sie rechts oben das Popup-Menü Host Actions. Wählen Sie Create New Context. Geben Sie als Document Base den einfachen Verzeichnisnamen *myservlets* (ohne Slash) und als Path */myservlets* ein. Die restlichen Optionen können Sie je nach Bedarf setzen.
7. Klicken Sie zum Schluss auf Save und dann auf Commit Changes. Den Tomcat-Server brauchen Sie anschließend nicht neu zu starten, wohl aber Apache 2.

Nun können Sie das Verzeichnis benutzen, um Ihre Webanwendungen zu veröffentlichen. Da Tomcat ein vollwertiger Webserver ist, können Sie neben JSP-Dateien und Servlet-Klassen natürlich auch einfache HTML-Dateien verwenden. Wenn Sie das Verhalten der Webserver-Komponente anpassen möchten, können Sie sich die verschiedenen Abteilungen der webbasierten Konfiguration anschauen. Das meiste dürfte Ihnen vom Apache-Webserver bekannt vorkommen. Alternativ können Sie in der Online-Dokumentation von Tomcat nachlesen, um die eigentlichen Konfigurationsdateien zu bearbeiten. Das Format ist XML, sodass sie konsistenter sind als die über viele Jahre »organisch gewachsene« *httpd.conf*.

#### Installation unter Windows

Das Tomcat-Installationspaket für Windows ist ein selbst entpackendes Archiv in Form einer einzelnen EXE-Datei. Sie können sie entweder von *tomcat.apache.org* herunterladen oder auf der DVD zum Buch finden; die aktuelle Fassung heißt *apache-tomcat-7.0.22.exe*. Bevor Sie die Installation starten, sollten Sie eventuell vorhandene ältere Versionen deinstallieren. Dies geschieht im Applet Programme der Systemsteuerung (siehe Kapitel 4, »Apache kompilieren und instal-

lieren«). Anschließend können Sie die Installer-Datei doppelklicken. Nun sind nacheinander folgende Schritte erforderlich:

- Sie müssen der Lizenzvereinbarung der ASF zustimmen. Klicken Sie I AGREE an.
- Als Nächstes wird der Installationstyp ausgewählt. Zu empfehlen ist normalerweise FULL; auf diese Weise wird Tomcat auch gleich als Dienst installiert.
- Nun muss der Installationspfad ausgewählt werden; eine gute Wahl ist oft *C:\Tomcat*.
- Nun werden Sie nach dem TCP-Port für HTTP/1.1-Verbindungen gefragt. Wenn Sie Tomcat als einzigen Webserver verwenden möchten (womit dieses Buch nicht das richtige für Sie sein dürfte), wird Port 80 empfohlen, was Sie normalerweise akzeptieren sollten. Wenn Apache 2 bereits an Port 80 lauscht, lautet die – in der Regel empfehlenswerte – Vorgabe dagegen 8080.
- Die nächsten Eingaben sind ein Benutzername und ein Passwort für den Tomcat-Administrator: Tomcat für Windows wird gleich mit der bereits erwähnten webbasierten Konfigurationsschnittstelle ausgeliefert; mit den hier angegebenen Anmeldedaten können Sie darauf zugreifen. Als Benutzernamen können Sie normalerweise `admin` beibehalten; das Passwort sollte wie üblich lang und komplex genug sein.
- Nun müssen Sie dem Installer den Pfad Ihrer Java Virtual Machine oder des Java SDK mitteilen. Dieser wird oft automatisch gefunden. Andernfalls können Sie einfach das Stammverzeichnis Ihrer Java-Installation wählen, z.B. *C:\Program Files\Java\jre6*.

Nun erfolgt bei Bedarf die Installation von `mod_jk`. Die Datei für Windows finden Sie auf der DVD zum Buch sowie im Binär-Download-Bereich unter *tomcat.apache.org/download-connectors.cgi*. Hier befindet sich unter *win32* der Link zur aktuellen Datei *tomcat-connectors-1.2.32-windows-i386-httpd-2.0.x.zip*, die Sie natürlich auch auf der DVD zum Buch finden.

Diese Datei können Sie mit WinZipentpacken und in Ihr *modules*-Verzeichnis kopieren. Konfiguration und Anpassung von Tomcat und `mod_jk` funktionieren analog zu UNIX, nur die Pfade müssen Sie natürlich anpassen.

### 15.3.2 Tomcat per Proxy einbinden

Alternativ zur Verwendung von `mod_jk` können Sie Verzeichnisse, die Tomcat veröffentlicht, auch mithilfe der in Kapitel 13, »Proxy- und Cache-Funktionen«, beschriebenen Reverse-Proxy-Funktionalität in den Adressraum Ihres Apache-Webservers einbinden.

Die einfachste Methode besteht darin, den HTTP-Proxy-Dienst zu verwenden. Dazu wird zunächst – wie bereits beschrieben – ein neuer Kontext erstellt, also ein mittels Tomcat freigegebenes Verzeichnis. Angenommen, Ihr neuer Kontext heißt */connect_test*, und Tomcat verwendet Port 8080. Dann genügt es, Folgendes in den Server-Kontext der *httpd.conf* einzutragen:

```
ProxyPass /connect_test http://localhost:8080/connect_test
ProxyPassReverse /connect_test http://localhost:8080/connect_test
```

Gegebenenfalls müssen Sie den Zugriff auf den URL-Pfad */connect_test* noch mithilfe von `Order`-, `Allow`- und `Deny`-Direktiven freigeben. Nach dem Neustart von Apache sollten Sie das Verzeichnis sowohl unter der Tomcat-eigenen URL *http://localhost:8080/connect_test* als auch unter der eingebundenen Apache-URL *http://localhost/connect_test* erreichen.

Eine Alternative ist der in Apache 2.2 eingeführte AJP-Proxy. Bei der Verknüpfung mit einer einzelnen Tomcat-Instanz ist er zwar nicht schneller als der HTTP-Proxy, aber beim Load-Balancing über mehrere Tomcat-Server ist seine Performance besser. Für einen einzelnen Server genügt es, das Protokoll `http://` durch `ajp://` zu ersetzen. Beispiel:

```
ProxyPass /connect_test ajp://localhost:8080/connect_test
```

Eine einfache Load-Balancing-Lösung, die denselben Inhalt von zwei verschiedenen Tomcat-Servern einbindet, könnte dagegen so aussehen:

```
<Proxy balancer://tomcats>
   BalancerMember ajp://tomcat1.mynet.de:8080
   BalancerMember ajp://tomcat2.mynet.de:8080
</Proxy>
ProxyPass /myservlets balancer://tomcats/myservlets/
```

### 15.3.3 Java-Webprogrammierung

Wie bereits erwähnt, unterstützt Tomcat zwei verschiedene Modelle der serverseitigen Java-Programmierung: JavaServer Pages und Java Servlets. Die beiden Technologien besitzen unterschiedliche Schwerpunkte: Ein Servlet ist eine vollständige, kompilierte Java-Klasse (abgeleitet von `javax.servlet.http.HttpServlet`), während JSP – ähnlich wie PHP-Skripte – gewöhnliche HTML-Seiten sind, in die an der passenden Stelle Java-Code eingefügt werden kann. Dementsprechend sind JSPs eher für Webseiten mit wenigen dynamischen Inhalten geeignet, während Servlets eher für Anwendungen gedacht sind, die viel Arbeit hinter den Kulissen erledigen müssen.

Beachten Sie, dass Sie zum Kompilieren von Servlets Zugriff auf eine Reihe spezieller Java-Klassen haben müssen. Am einfachsten ist es, die Archivdatei *servlet-api.jar* im Unterverzeichnis *common/lib* Ihrer Tomcat-Installation zur Umgebungsvariablen `CLASSPATH` hinzuzufügen; für manche spezielleren Anwendungen müssen Sie möglicherweise noch weitere JAR-Dateien aus diesem Verzeichnis einbinden.

#### Ein Servlet-Beispiel

Der Quellcode eines Servlets sieht beispielsweise folgendermaßen aus:

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class ZweiHochServlet extends HttpServlet
{
   public void doGet (HttpServletRequest req,
        HttpServletResponse resp)
        throws ServletException, IOException
   {
      resp.setContentType ("text/html");
      PrintWriter out = resp.getWriter();
      out.println ("<html>");
      out.println ("<head>");
      out.println ("<title>2 hoch 0 bis 2 hoch 15</title>");
      out.println ("</head>");
      out.println ("<body>");
      out.println ("<h1>2<sup>0</sup>
           bis 2<sup>15</sup></h1>");
      for (int i = 1; i <= 15; i++) {
          out.println ("2<sup>" + i + "</sup> = " +
               Math.pow (2, i) + "<br />");
      }
      out.println ("</body>");
      out.println ("</html>");
   }
}
```

Die Methode `doGet()` ist eine typische Callback-Methode. Sie wird – wie der Name schon sagt – aufgerufen, wenn eine HTTP-`GET`-Anfrage für das Servlet eintrifft. Dabei erhält die Methode vom Webserver automatisch ein Anfrageobjekt vom Typ `HttpServletRequest` sowie ein Antwortobjekt der Klasse `HttpServletResponse`. Letzteres bietet unter anderem die Methoden `setContentType()` für den Header `Content-Type` sowie `getWriter()` für Schreibzugriffe auf die HTTP-Antwort.

Näheres über das Deployment, also die Einrichtung und Veröffentlichung, von Servlets entnehmen Sie bitte der Tomcat-Online-Dokumentation. Am besten verwenden Sie dafür den webbasierten *Tomcat Manager*.

### Ein JSP-Beispiel

Der volle Nutzen von JavaServer Pages in der aktuellen Fassung 2.0 ergibt sich erst durch den Einsatz von Hilfsmitteln wie JavaBeans (Komponenten) oder Custom TagLibs (Definition eigener Tags in einem separaten XML-Namensraum). Beides führt erheblich zu weit, um hier besprochen zu werden. Lesen Sie hierzu beispielsweise das betreffende Kapitel in [ULLE 2008] oder die Online-Einführung auf der Tomcat-Website. Hier finden Sie lediglich ein kurzes Beispiel, das Datum und Uhrzeit ausgibt:

```
<html>
  <head>
    <title>JSP - Datum und Uhrzeit</title>
  </head>
  <body>
    <h1>Datum und Uhrzeit mit JSP</h1>
    Heute ist <%= new java.util.Date() %>
  </body>
</html>
```

Die spezielle Schreibweise `<%= ... %>` sorgt dafür, dass der Wert des eingefügten Ausdrucks nicht nur berechnet, sondern auch gleich ausgegeben wird.

Sie können das Skript unter einem Dateinamen mit der Endung `.jsp` in einen Context-Ordner Ihres Tomcat-Servers kopieren, den Sie – wie beschrieben – über `mod_jk` mit Apache verbunden haben. Geben Sie dann die entsprechende URL in Ihren Browser ein (z.B. *www.mynet.de/my-jsps/jetzt.jsp*), und Sie erhalten eine Seite, die eine Ausgabe wie diese enthält:

```
Heute ist Mon Oct 31 13:13:24 CEST 2011
```

### Java-Zugriff auf MySQL

Auch Java-basierte Webanwendungen sind in der Praxis natürlich nur dann nützlich, wenn eine Datenbank verwendet werden kann. Datenbankzugriffe erfolgen in Java mithilfe der Schnittstelle JDBC, die in jedem gewöhnlichen Java SDK enthalten ist. JDBC ist für die jeweilige konkrete Datenbank auf einen Treiber angewiesen. Für MySQL ist dies *MySQL Connector/J*; er ist auf der DVD zum Buch und auf der Website *www.mysql.com* zu finden. Entpacken Sie zunächst die Archivdatei. Im obersten Verzeichnis ist eine Datei namens *mysql-connector-java-5.1.18-bin.jar* (eventuell mit anderer Versionsnummer) enthalten, die Sie in Ihren `CLASSPATH` einbetten müssen.

Eine Java-Klasse, die JDBC benutzen soll, muss zunächst einmal das entsprechende Package importieren:

```
import java.sql.*;
```

Anschließend wird MySQL Connector/J geladen. Dies geschieht über folgendes Konstrukt:

```
try {
   Class.forName ("com.mysql.jdbc.Driver").newInstance();
}
catch (ClassNotFoundException e) {
   System.out.println
        ("MySQL Connector/J nicht gefunden.");
}
```

Nun wird eine Verbindung zum Datenbank-Server und zur gewünschten Datenbank hergestellt. Zu diesem Zweck gibt es die JDBC-Klasse `DriverManager` mit der Methode `getConnection()`. Die drei Argumente dieser Methode sind die Strings Datenbank-URL, Benutzername und Passwort. Jeder JDBC-Treiber besitzt ein individuelles URL-Schema, bei MySQL Connector/J sieht es so aus:

```
jdbc:mysql://Host/Datenbank
```

Für eine Verbindung zur Datenbank `buecher` auf dem lokalen Rechner müssten Sie also Folgendes schreiben (vorausgesetzt, Benutzername und Passwort sind wieder `dbuser` und `strengGeheim`):

```
Connection conn = DriverManager.getConnection
     ("jdbc:mysql://localhost/buecher", "dbuser",
     "strengGeheim");
```

Der nächste Schritt ist die Erzeugung eines `Statement`-Objekts, über das Sie der Datenbank Abfragen senden können:

```
Statement st = conn.createStatement();
```

Nun können Sie SQL-Abfragen durchführen. Für Auswahlabfragen, die Datensätze zurückgeben, wird die Statement-Methode `excuteQuery()` verwendet, für alle anderen `execute()`. Hier ein Beispiel für eine Einfügeabfrage:

```
try {
   st.execute ("INSERT INTO buecher VALUES
        (\"9783836211345\", \"Laborenz\", \"CSS-Praxis\",
        \"Galileo Computing\")");
}
catch (SQLException e)
{
```

```
   // SQL-Abfrage-Ausnahme behandeln
}
```

Der `try/catch`-Block ist erforderlich, weil die Methode `execute()` bei irgendeinem Datenbankproblem eine `SQLException` auslösen kann.

`executeQuery()` liefert eine Instanz der Klasse `ResultSet` zurück. Das folgende Beispiel zeigt, wie Sie sie auslesen und ausgeben können; stellen Sie sich `out`, wie im Vorfeld beschrieben, als `HttpResponse-PrintWriter`-Objekt vor:

```
try {
   ResultSet rs = st.executeQuery
     ("SELECT * FROM buecher WHERE verlag LIKE \"Galileo%\"");
   while (rs.next()) {
      out.println
           ("<tr><td>" + rs.getString ("isbn") + "</td>");
      out.println ("<td>" + rs.getString ("autor") + "</td>");
      out.println ("<td>" + rs.getString ("titel") + "</td>");
      out.println
           ("<td>" + rs.getString ("verlag") + "</td></tr>");
   }
}
catch (SQLException e) {
   // SQL-Ausnahme behandeln
}
```

Die Ausgabebefehle für `<table>` und `</table>` habe ich weggelassen; der Tradition englischsprachiger Fachbücher folgend: »This is left as an exercise to the reader.«

Wie Sie sehen, wurden alle vier Felder mit der Methode `getString()` ausgelesen – dies passt zu den SQL-Datentypen `CHAR` und `VARCHAR`, als die sie definiert wurden. Für andere SQL-Datentypen müssen Sie andere `get*`-Methoden verwenden – Java ist im Unterschied zu Perl oder PHP eine typisierte Sprache. Tabelle 15.4 zeigt eine Übersicht über die wichtigsten SQL-Datentypen, ihre Java-Pendants und die passenden JDBC-`get*`-Methoden.

| SQL-Datentyp | Java-Datentyp | get-Methode |
|---|---|---|
| CHAR | String | getString() |
| VARCHAR | String | getString() |
| INT | int | getInt() |
| FLOAT | double | getDouble() |
| DOUBLE | double | getDouble() |
| DATE | java.sql.Date | getDate() |

**Tabelle 15.4** Zuordnung von SQL- und Java-Datentypen mit den entsprechenden »get*«-Methoden

| SQL-Datentyp | Java-Datentyp | get-Methode |
|---|---|---|
| TIME | java.sql.Time | getTime() |
| TIMESTAMP | java.sql.Timestamp | getTimestamp() |
| BLOB | java.sql.Blob | getBlob() |

**Tabelle 15.4** Zuordnung von SQL- und Java-Datentypen mit den entsprechenden »get*«-Methoden (Forts.)

Zum Schluss sollten Sie zuerst die `ResultSet`- und dann die `Statement`-Instanz schließen:

```
rs.close();
st.close();
```

## 15.4 mod_lua

Apache 2.3/2.4 wird erstmals mit einem Skriptsprachen-Modul ausgeliefert: `mod_lua` stellt die leichtgewichtige, aber durchaus leistungsfähige Skriptsprache Lua zur Verfügung. Diese wurde an der Katholischen Universität in Rio de Janeiro, Brasilien, entwickelt, und kommt als Scripting- und Konfigurationssprache in vielen verschiedenen Anwendungsprogrammen und Spielen zum Einsatz. Lua (der Name ist das portugiesische Wort für Mond) kann jedoch auch als Standalone-Sprache verwendet werden. Weitere Informationen, Downloads und die ausführliche Dokumentation der Sprache finden Sie auf der Website *www.lua.org*.

Um `mod_lua` überhaupt kompilieren zu können, benötigen Sie die Developer-Version von *liblua*. Unter Ubuntu heißt das entsprechende Paket beispielweise *liblua50-dev*.

Sie können `mod_lua` über einen Handler für Webanwendungsskripte verwenden; dies funktioniert im einfachsten Fall wie folgt, wobei der Handler hier standardmäßig für alle Dateien mit der Endung `.lua` gesetzt wird:

```
AddHandler lua-script .lua
```

Zusätzlich stellt Apache eine Reihe von Direktiven für weitere Anwendungszwecke bereit, die weiter unten vorgestellt werden.

### 15.4.1 Ein einfaches Beispiel

Das folgende Beispiel für ein einfaches Lua-Skript stellt ein HTML-Formular dar. Der Benutzer kann zwei Zahlen und eine Rechenoperation eingeben; nach einem Klick auf den Button Berechnen wird das Skript erneut geladen, und die Berechnung wird durchgeführt:

```
require 'string'

function handle(r)
  r.content_type = "text/html"
  r:puts("<html>\n")
  r:puts("<head>\n")
  r:puts("<title>Lua-Rechner</title>\n")
  r:puts("</head>\n")
  r:puts("<body>\n")
  r:puts("<h1>Lua-Rechner</h1>\n")

  num1 = nil
  num2 = nil
  op = nil

  if r.method == 'GET' then
    for k, v in pairs(r:parseargs()) do
      if k == 'num1' then
        num1 = v
      elseif k == 'num2' then
        num2 = v
      elseif k == 'op' then
        op = v
      end
    end
  end

  if num1 and num2 and op then
    err = false
    if op == '+' then
      result = num1 + num2
    elseif op == '-' then
      result = num1 - num2
    elseif op == '*' then
      result = num1 * num2
    elseif op == '/' then
      if num2 == '0' then
        err = true
        r:puts("Division durch 0 ist verboten!")
      end
      result = num1 / num2
    end
    if err == false then
      r:puts(
        string.format(
```

```
            "%.3f %s %.3f = %.3f",
            num1,
            op,
            num2,
            result
          )
        )
      end
    end
    r:puts('<form action="test2.lua" method="get">')
    r:puts('First number: <input type="text" name="num1" /><br />')
      r:puts('Second number: <input type="text" name="num2" /><br />')
    r:puts('Operation:')
    r:puts('<select name="op">')
    r:puts('<option value="+">+</option>')
    r:puts('<option value="-">-</option>')
    r:puts('<option value="*">*</option>')
    r:puts('<option value="/">/</option>')
    r:puts('</select>')
    r:puts('<input type="submit" value="Berechnen" />')
    r:puts('</form>')
    r:puts('</body>')
    r:puts('</html>')
end
```

Eine Funktion namens `handle()` wird als Webserver-Handler-Funktion automatisch aufgerufen. Der Parameter `r` ist das Request-Objekt, das Attribute wie `content_type` (auszugebender MIME-Type) und `method` (HTTP-Methode) sowie Methoden wie `puts` (Ausgabe) oder `parseargs()` (`GET`-Parameter in ein Array zerlegen) bereitstellt. Das eingebundene Package `string` stellt zusätzliche String-Funktionalität bereit; im Beispiel wird die Methode `string.format()` verwendet, die für `printf()`-String-Formatierung sorgt.

### 15.4.2 mod_lua-Direktiven

In diesem Unterabschnitt werden alle von `mod_lua` bereitgestellten Direktiven beschrieben. Es handelt sich um den Funktions- und Dokumentationsstand in Version 2.3.15, der letzten Beta-Version vor der Veröffentlichung von 2.4.0. Das bedeutet, dass sich hier noch einiges ändern kann; lesen Sie gegebenenfalls die Originaldokumentation auf der Apache-Website.

#### LuaCodeCache

Cache für vorkompilierten Lua-Code

| Seit Version | 2.3 |
|---|---|
| Modul | mod_lua |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (All) |
| Syntax | LuaCodeCache stat\|forever\|never |
| Standardwert | stat |

Aus Performance-Gründen kann Apache einmal kompilierte Lua-Skripte im Arbeitsspeicher cachen, was durch diese Direktive konfiguriert wird. Der Standardwert `stat` bedeutet, dass bei jedem Zugriff auf direkt aufgerufene Lua-Skripte (nicht auf eingebettete) per `stat`-Systemaufruf anhand des Datei-Änderungsdatums überprüft wird, ob die Kopie im Speicher aktuell genug ist. Bei `forever` wird gecached, solange Apache läuft, und bei `never` gar nicht. Somit ist `forever` nur für Produktivsysteme geeignet, `never` nur für Entwicklungsrechner und `stat` für beide.

#### LuaHook*-Direktiven

Lua-Skripte für diverse Hooks

| Seit Version | 2.3 (optionaler Parameter early\|late seit 2.3.15) |
|---|---|
| Modul | mod_lua |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (All) |
| Syntax | LuaHook* Skript-Pfad Funktionsname [early\|late] |
| Standardwert | nicht gesetzt |

Die `LuaHook*`-Direktiven bieten die Möglichkeit, Lua-Skripte für die diversen Apache-Hooks (Phasen der Anfrageverarbeitung) anzugeben, die jeweils in der entsprechenden Phase ausgeführt werden. Die Bedeutung der einzelnen Hooks wird in Kapitel 17, »Apache erweitern«, erläutert.

Der erste Parameter ist der Pfad des auszuführenden Lua-Skripts, der zweite der Name der Funktion innerhalb des jeweiligen Skripts, die ausgeführt werden soll. Seit Version 2.3.15 können Sie optional `early` oder `late` als dritten Parameter angeben, um zu bestimmen, wann das Skript in der jeweiligen Phase ausgeführt werden soll: Bei `early` wird es so früh wie möglich ausgeführt, bei `late` so spät wie möglich.

Im Einzelnen werden folgende Hooks unterstützt (angegeben sind jeweils der offizielle Name des Hooks und die zuständige Direktive):

- `access_checker` – `LuaHookAccessChecker`
- `auth_checker` – `LuaHookAuthChecker`
- `check_user_id` – `LuaHookCheckUserId`
- `fixups` - `LuaHookFixups`
- `insert_filter` – `LuaHookInsertFilter`
- `map_to_storage` – `LuaHookMapToStorage`
- `translate_name` – `LuaHookTranslateName`
- `type_checker` – `LuaHookTypeChecker`

Einige dieser Direktiven sind vorgesehen, aber zumindest in Version 2.3.15 noch nicht implementiert, beispielsweise `LuaHookInsertFilter`.

Besonders für die Hook-Direktiven stellt Lua das Package `apache2` bereit, das Sie wie folgt in Ihre Skripte integrieren können:

```
require 'apache2'
```

In diesem Package sind insbesondere die folgenden Konstanten interessant, die Ihre Lua-Hook-Handler je nach Ergebnis zurückgeben sollten:

- `apache2.OK` – Anfrage erfolgreich verarbeitet
- `apache2.DECLINED` – nicht für diese Anfrage zuständig
- `apache2.DONE` – Anfrage braucht nicht mehr weiterverarbeitet zu werden

Je nach Hook können auch andere Rückgabewerte sinnvoll sein; für `LuaHookAuthChecker` (Authentifizierungsprüfung) beispielsweise der numerische Wert 401 (HTTP-Status »Authentifizierung erforderlich«).

Das folgende Beispiel ruft möglichst früh im `translate_name`-Hook (Interpretation der Anfrage-URI) die Funktion `show_article` im Lua-Skript *article.lua* auf:

```
LuaHookTranslateName /usr/local/apache2/lua/article.lua \
show_article early
```

Das entsprechende Skript könnte beispielsweise so aussehen:

```
require "string"
require "apache2"

function show_article(r)
  if string.find(r.uri, "/article-%d+") then
    r.filename = "/usr/local/apache2/lua/show_article.lua"
    return apache2.OK
  else
```

```
    return apache2.DECLINED
  end
end
```

Wenn die aufgerufene URI */article-<Nummer>* lautet, wird das Skript *show_article.lua* aufgerufen. Andernfalls erklärt sich das Skript per `apache2.DECLINED` für nicht zuständig. Um den Lua-Interpreter klein und effizient zu halten, haben die Entwickler übrigens keine regulären Ausdrücke für Mustervergleiche implementiert, sondern ein eigenes Verfahren. Die Methode `string.find(string, muster)` sucht nach dem angegebenen Muster; der Platzhalter `%d` steht darin für eine Ziffer, und der Quantifier `+` wie in regulären Ausdrücken für ein oder mehrere Vorkommen. Mehr über String-Mustervergleiche in Lua erfahren Sie unter *http://www.lua.org/pil/20.1.html*.

#### LuaInherit

Umgang mit Lua-Hooks der übergeordneten Konfigurationsebene

| Seit Version | 2.4 |
|---|---|
| Modul | `mod_lua` |
| Kontext | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`All`) |
| Syntax | `LuaInherit none\|parent-first\|parent-last` |
| Standardwert | `parent-first` |

Bestimmt, ob und wann `LuaHook*`-Skripte aus übergeordneten Konfigurationsebenen ausgeführt werden, wenn die aktuelle Ebene denselben Hook enthält. Der Standardwert `parent-first` besagt, dass das übergeordnete Skript zuerst ausgeführt wird, `parent-last` führt es nach dem aktuellen aus und `none` gar nicht.

#### LuaMapHandler

Einem Pfad einen Lua-Handler zuordnen

| Seit Version | 2.3 |
|---|---|
| Modul | `mod_lua` |
| Kontext | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`All`) |
| Syntax | `LuaMapHandler URL-Muster Skript-Pfad [Funktionsname]` |
| Standardwert | nicht gesetzt |

Wenn Sie diese Direktive setzen, wird das als zweiter Parameter angegebene Lua-Skript aufgerufen, sobald der entsprechende URL-Pfad (erster Parameter) aufge-

rufen wird. Optional können Sie als dritten Parameter einen Funktionsnamen angeben; andernfalls wird `handle()` erwartet.

Das folgende Beispiel ruft für den Pfad */info* die Methode `print_info()` im Skript *info.lua* auf:

```
LuaMapHandler /info /usr/local/apache2/lua/info.lua \
print_info
```

Optional können Sie im URL-Muster reguläre Ausdrücke (PCRE) verwenden. Geklammerte Teiltreffer können dann als `$1` bis `$9` für den Skriptnamen und den Funktionsnamen verwendet werden. Das folgende Beispiel wandelt zusammengesetzte Pfade entsprechend um – */skript/funktion* wird zu einem Aufruf von `funktion()` in *skript.lua*:

```
LuaMapHandler /(\w+)/(\w+) $1.lua $2
```

## 15.5 Weitere Programmierschnittstellen

Hier wird noch kurz die ISAPI-Schnittstelle vorgestellt, weil das Modul `mod_isapi` zum Lieferumfang von Apache für Windows gehört; im Anschluss finden Sie einige Hinweise auf Bezugsquellen für weitere Programmiersprachenmodule.

### 15.5.1 ISAPI-Anwendungen mit mod_isapi

Das Modul `mod_isapi` steht nur für die Windows-Version von Apache zur Verfügung. Die ISAPI-Technologie ermöglicht Webanwendungen in Form kompilierter DLLs (Laufzeitbibliotheken) und wurde ursprünglich für Microsoft Internet Information Services entwickelt. `mod_isapi` ist eine freie Implementierung dieser Technologie und gehört zum Lieferumfang von Apache für Windows.

Zunächst einmal müssen Sie dafür sorgen, dass Apache DLL-Dateien als ISAPI-Erweiterungen versteht. Dazu können Sie den Handler `isapi-isa` folgendermaßen setzen (Näheres siehe Kapitel 7, »Header und MIME-Einstellungen«):

```
AddHandler isapi-isa .dll
```

Die Apache-Implementierung von ISAPI beherrscht nicht den gesamten von Microsoft definierten Standard. Unter anderem können Sie keine ISAPI-Filter verwenden (brauchen Sie normalerweise auch nicht; das in Kapitel 16, »SSI und Filter«, vorgestellte Apache-Filtermodell ist besser). Tabelle 15.5 enthält eine Kurzübersicht über die implementierten ISAPI-Funktionsaufrufe; diese können aus dem Programm heraus mithilfe der Methode `ServerSupportFunction()` aufgerufen werden.

| Funktion | Erläuterung |
|---|---|
| HSE_REQ_SEND_URL_REDIRECT_RESP | Umleitung zu einer externen URL (FQDN) |
| HSE_REQ_SEND_URL | Umleitung zu einer relativen URL |
| HSE_REQ_SEND_RESPONSE_HEADER | HTTP-Header senden |
| HSE_REQ_DONE_WITH_SESSION | Session-Ende (Platzhalter für Kompatibilitätszwecke) |
| HSE_REQ_MAP_URL_TO_PATH | URL in Dateipfad umwandeln |
| HSE_APPEND_LOG_PARAMETER | Ausgabe für Log-Datei erzeugen (kann durch \"%{isapi-parameter}n\" in Log-Formate übernommen werden) |
| HSE_REQ_IS_KEEP_CONN | Keepalive (persistente Verbindung) |
| HSE_REQ_SEND_RESPONSE_HEADER_EX | Header senden; Keepalive ignorieren |
| HSE_REQ_IS_CONNECTED | false bei abgebrochener Verbindung |

**Tabelle 15.5** Von Apache unterstützte ISAPI-»ServerSupportFunction( )«-Aufrufe

### ISAPIAppendLogToErrors

HSE_APPEND_LOG_PARAMETER in ErrorLog aufnehmen

| | |
|---|---|
| **Modul** | mod_isapi |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| **Syntax** | ISAPIAppendLogToErrors on\|off |
| **Standardwert** | off |

Diese Direktive bestimmt, dass der Wert von HSE_APPEND_LOG_PARAMETER (siehe dort) automatisch in die ErrorLog-Datei aufgenommen wird.

### ISAPIAppendLogToQuery

HSE_APPEND_LOG_PARAMETER in Query-Feld schreiben

| | |
|---|---|
| **Modul** | mod_isapi |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| **Syntax** | ISAPIAppendLogToQuery on\|off |
| **Standardwert** | on |

Wenn diese Direktive eingeschaltet ist (Standard), wird HSE_APPEND_LOG_PARAMETER in das Query-Feld geschrieben, das Sie über das Kürzel %q in Log-Formate einfügen können.

### ISAPICacheFile

ISAPI-DLLs, die beim Start geladen werden sollen

| | |
|---|---|
| **Modul** | mod_isapi |
| **Kontext** | Server, <VirtualHost> |
| **Syntax** | ISAPICacheFile Dateipfad [Dateipfad ...] |
| **Standardwert** | nicht gesetzt |

Mithilfe dieser Direktive können Sie eine Liste von ISAPI-DLLs angeben, die beim Start von Apache geladen werden und anschließend aktiv bleiben sollen. Wenn die Pfadangaben nicht absolut sind, werden sie als relativ zur ServerRoot betrachtet. Beispiel:

```
ISAPICacheFile "C:/ISAPI_ISA/HalloWelt.dll"
```

### ISAPIFakeAsync

Simulation asynchroner ISAPI-Funktionalität für Callbacks

| | |
|---|---|
| **Modul** | mod_isapi |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| **Syntax** | ISAPIFakeAsync on\|off |
| **Standardwert** | off |

Diese Direktive simuliert die asynchrone Funktionalität, die für ISAPI-Callback-Funktionen benötigt wird.

### ISAPILogNotSupported

Anfragen nach nicht unterstützten ISAPI-Erweiterungen in ErrorLog aufnehmen

| | |
|---|---|
| **Modul** | mod_isapi |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| **Syntax** | ISAPILogNotSupported on\|off |
| **Standardwert** | off |

Wenn Sie diese Direktive auf on setzen, werden Aufrufe von Server-SupportFunction() für Funktionen, die mod_isapi nicht kennt, in die ErrorLog-Datei geschrieben.

#### ISAPIReadAheadBuffer

Größe des Read-Ahead-Puffers für ISAPI-Erweiterungen

| Modul | mod_isapi |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | ISAPIReadAheadBuffer Größe |
| Standardwert | 49152 |

Diese Direktive legt die Größe des Read-Ahead-Puffers fest, der ISAPI-Erweiterungen beschleunigen kann.

### 15.5.2 Sonstige Technologien

Neben ISAPI und den anderen ausführlich vorgestellten Schnittstellen gibt es noch zahlreiche weitere lohnende Ergänzungen von Drittanbietern. Hier finden Sie lediglich eine kurze Liste mit Bezugsquellen; sofern die Module frei verfügbar sind, befinden sie sich auch auf der beiliegenden DVD.

- mod_python: Die Skriptsprache Python (benannt nach Monty Python, nicht etwa nach einem Reptil!) von Guido van Rossum ist zwar nicht so verbreitet wie Perl, aber dennoch lohnt sich ein näherer Blick darauf. Sie besticht durch eine etwas eigenwillige, aber überaus mächtige Syntax sowie unzählige Erweiterungen, die durch eine fleißige Community zusammengetragen werden. Natürlich können Sie Python-Skripte einfach als CGI-Anwendungen ausführen; mod_python realisiert den Interpreter aber analog zu mod_perl oder PHP als Apache-Modul. Die Website des Projekts ist *www.modpython.org*.
- mod_ruby: *Ruby* ist ebenfalls eine freie Skriptsprache, geschrieben von Yukihiro Matsumoto (auch »Matz« genannt). Ihre Besonderheit besteht darin, dass sie absolut konsequent objektorientiert ist; wirklich jeder Aufruf wird nach dem Schema instanz.methode() durchgeführt. mod_ruby ist die Verwirklichung der Sprache als Apache-Modul; den Download und weitere Informationen finden Sie unter *www.modruby.org*.
- mod_mono: Microsofts .NET-Plattform hat innerhalb von knapp vier Jahren die Windows-Programmierung neu definiert. *Mono* ist eine freie Implementierung des .NET-Frameworks für Linux und einige andere UNIX-Systeme. mod_mono ermöglicht die Ausführung von ASP.NET-Anwendungen auf dem Apache-Webserver. Sie finden Mono selbst unter *www.go-mono.com*; das Modul mod_mono ist unter *www.apacheworld.org/modmono* erhältlich.

- `mod_aspdotnet`: Dieses nur für Windows verfügbare Apache-2-Modul ist ein Unterprojekt des Apache-HTTP-Servers. Es stellt eine Schnittstelle zur ASP.NET-Engine von Microsoft her, sodass ASP.NET-Webanwendungen (`.aspx`) von Apache veröffentlicht werden können.

## 15.6 Zusammenfassung

Wie Sie wissen, gibt es unzählige Programmiersprachen, und die meisten von ihnen können Sie zur Programmierung von Webserver-Anwendungen verwenden. Während Sie in Kapitel 14, »CGI«, die klassische CGI-Schnittstelle kennengelernt haben, die die Anbindung völlig beliebiger Sprachen ermöglicht, wurden in diesem Kapitel drei Sprachen mit spezifischer (und vor allem beschleunigter) Verbindung zum Apache-Webserver vorgestellt. In Abschnitt 15.5.2, »Sonstige Technologien«, wurden noch einige weitere angesprochen, die (nicht nur meiner Ansicht nach) etwas weniger bedeutend sind.

PHP ist eine überaus beliebte Skriptsprache für Webanwendungen. Sie ist leicht zu erlernen, verfügt bereits im Kern über zahlreiche leistungsfähige Funktionen und kann zudem mithilfe von Drittanbieter-Extensions erweitert werden. Das wichtigste Argument für PHP ist aber wahrscheinlich die praktische Zusammenarbeit mit der freien Datenbank MySQL, deren Installation und Verwendung hier ebenfalls kurz beschrieben wurden.

Perl ist der Klassiker für CGI-Anwendungen mit Millionen frei verfügbarer Skripte und Tausenden von nützlichen Modulen. Deshalb lag es nahe, den Perl-Interpreter als Apache-Modul zur Verfügung zu stellen, um die Ausführung von Perl-Server-Anwendungen zu beschleunigen. Das Ergebnis dieser Bemühung, `mod_perl`, kann aber noch erheblich mehr: Sie können Perl-Anweisungen in die Apache-Konfigurationsdateien schreiben, um bestimmte Vorgänge zu automatisieren, und Sie können sogar vollständige Apache-Module in Perl schreiben.

Java ist eine sehr weit verbreitete, mit unzähligen eingebauten Klassenbibliotheken versehene Programmiersprache, für die zudem unzählige externe Erweiterungen verfügbar sind. Deshalb liegt es nahe, Webserver-Anwendungen in dieser Sprache zu schreiben – besonders die Anbindung an bestehende Enterprise- oder E-Commerce-Infrastrukturen ist eine beliebte Aufgabe der Java-Technologie. Tomcat, genau wie der HTTP-Server ein freies Projekt der Apache Software Foundation, ist ein Server für beide bekannten Arten von Java-Webserver-Anwendungen: Servlets und JSP. Für Apache-Anwender ist natürlich nicht so sehr die Webserver-Funktionalität von Tomcat interessant, sondern vor allem die Tatsache, dass sie den Server über Reverse-Proxy-Funktionalität oder über das Modul

mod_jk mit Apache 2 verbinden und somit als Servlet- und JSP-Container nutzen können.

Apache 2.4 ist die erste Apache-Version, die ab Werk einen eingebauten Programmiersprachen-Interpreter enthält: mod_lua bettet die leichtgewichtige, aber durchaus leistungsfähige Skriptsprache Lua ein. Neben der klassischen Skript-Handler-Fähigkeit bietet mod_lua auch die Möglichkeit, Lua-Skripte für spezielle Verarbeitungs-Hooks einzusetzen – hier handelt es sich im Grunde um einen Ersatz für selbst geschriebene Apache-Module.

*»Das Wasser, das zu rein ist, hat keine Fische.«*
*– Zen-Weisheit*

# 16 SSI und Filter

Ein weiterer Klassiker zur Erstellung dynamischer Inhalte neben dem in Kapitel 14, »CGI«, vorgestellten CGI sind *Server Side Includes* (SSI), die das schnelle Einbetten variabler Daten wie beispielsweise Umgebungsvariablen oder externer Dateien ermöglichen. Bei Apache wird die SSI-Funktionalität durch das Modul `mod_include` bereitgestellt.

Mit dem Erscheinen von Apache 2.0 wurde SSI in das neu geschaffene Konzept der *Filter* integriert. Filter sind beliebige externe Programme (oder alternativ Apache-Module), die die Ein- oder Ausgabe des Servers manipulieren:

- *Eingabefilter* können den Inhalt der Client-Anfrage vor der eigentlichen Verarbeitung durch Apache beliebig ändern.
- *Ausgabefilter* sind dagegen in der Lage, die vom Server erzeugte Antwort vor der Auslieferung an den Client zu verändern.

Konsequenterweise wurde `mod_include` in Apache 2 als Spezialfall eines Ausgabefilters realisiert. Dennoch wird im ersten Abschnitt dieses Kapitels die ältere SSI-Technologie beschrieben, während im zweiten Abschnitt Einrichtung und Programmierung eigener Filter erläutert werden.

## 16.1 Server Side Includes (SSI)

Mithilfe von Server Side Includes können Sie auf einfache Weise dynamische Daten in Ihre Websites einfügen, beispielsweise den Inhalt von Umgebungsvariablen oder Datum und Uhrzeit. In diesem Abschnitt wird zunächst dargelegt, wie Sie Apache 2 für die Ausführung von SSI konfigurieren können. Anschließend werden alle SSI-Elemente vorgestellt und zum Schluss sämtliche `mod_include`-Konfigurationsdirektiven.

Seit Version 2.3.13 verwendet SSI für `#if`-Abschnitte die Ausdruckssyntax von `ap_expr` (siehe die Beschreibung der Direktive `<If>` in Kapitel 6, »Grundkonfigu-

ration«). Es gibt allerdings einen Kompatibilitätsmodus, der sich durch die in derselben Version eingeführte Direktive `SSILegacyExprParser` aktivieren lässt.

### 16.1.1 SSI aktivieren

Die erste Voraussetzung dafür, dass in einem bestimmten Kontext Ihrer Site SSI ausgeführt werden, wurde bereits in Kapitel 6, »Grundkonfiguration«, erwähnt: Sie müssen die Funktionalität mithilfe der Direktive `Options` freischalten. Im Server-Kontext müssen Sie `Includes` zu den vorhandenen Optionen hinzufügen; in einem untergeordneten Kontext können Sie Folgendes schreiben:

```
Options +Includes
```

In der Optionsliste `All` ist `Includes` übrigens bereits enthalten. Eine sicherere Alternative ist die Option `IncludesNOEXEC`, bei der SSI zwar aktiviert werden, die aber keine Ausführung externer Programme mittels `exec` zulässt. Setzen Sie für diese Variante im gewünschten Kontext folgende Direktive:

```
Options +IncludesNOEXEC
```

Als Nächstes müssen Sie für die speziellen HTML-Dokumente, in denen Server Side Includes ausgeführt werden sollen, den Ausgabefilter `INCLUDES` setzen. Normalerweise werden Dateien mit der Endung `.shtml` als SSI-Dokumente konfiguriert. Dies geschieht mithilfe der folgenden beiden Direktiven (`AddOutputFilter` wird weiter unten näher erläutert):

```
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
```

Natürlich steht es Ihnen auch frei, alle HTML-Dokumente in Bezug auf Server Side Includes zu untersuchen. Bedenken Sie aber, dass infolge dieses zusätzlichen Arbeitsschritts Ihr Server durchaus langsamer werden kann.

In Apache-Versionen vor 2.0 gab es das Filterkonzept noch nicht. Deshalb wird SSI in älteren Versionen als Handler mit dem Namen `server-parsed` aktiviert. Apache 2 unterstützt diese Variante aus Kompatibilitätsgründen, sie sollte aber nicht mehr verwendet werden. Diese Schreibweise sieht folgendermaßen aus:

```
AddType text/html .shtml
SetHandler server-parsed .shtml
```

### 16.1.2 SSI-Elemente

Jedes SSI-Element wird standardmäßig von einem HTML-Kommentar mit dem zusätzlichen Startzeichen # umschlossen, sieht also schematisch folgendermaßen aus:

```
<!--#element [Argument=Wert ...] -->
```

Mithilfe der weiter unten vorgestellten Direktiven `SSLStartTag` und `SSLEndTag` können Sie die Sequenzen ändern, die das Server Side Include umschließen. Dies ist aber eigentlich nur nötig, wenn Sie gleichzeitig eine serverseitig dynamische Technologie eines Drittherstellers verwenden, die diese Zeichenfolgen verwendet.

Für die meisten Elemente sind einige benannte Argumente im Format `Argument=Wert` verfügbar. Die Werte werden üblicherweise in Anführungszeichen gesetzt; solange sie keine Leerzeichen enthalten, ist dies aber optional.

SSI definiert folgende Elemente:

- `config`
  Definition von Ausgabeformaten
- `echo`
  Ausgabe von Variablen
- `exec`
  Ausführen eines externen Programms
- `fsize`
  Ausgabe der Größe einer Datei
- `flastmod`
  Ausgabe des letzten Änderungsdatums einer Datei
- `include`
  Einfügen einer externen Datei
- `printenv`
  Ausgabe aller Umgebungsvariablen
- `set`
  Setzen einer Variablen
- `if`, `elif`, `else`, `endif`
  bedingte Ausführung beziehungsweise bedingte HTML-Blöcke

### config

Dieses SSI-Element legt fest, wie die Ausgabe bestimmter anderer Elemente formatiert wird. Sie können eines oder mehrere der folgenden Argumente angeben:

- `errmsg="Meldungstext"`
  Dies legt die Meldung fest, die in das HTML-Argument geschrieben wird, wenn bei der SSI-Verarbeitung ein Fehler auftritt.

- `sizefmt="byte"|"abbrev"`
  Es gibt an, wie das Element `fsize` die Größe einer Datei anzeigen soll. `"byte"` zeigt den Wert als einfache Zahl in Byte an; `"abbrev"` verwendet je nach Größe Kilobyte oder Megabyte mit der Abkürzung `K` beziehungsweise `M`.
- `timefmt="..."`
  Dies ist ein `strftime()`-kompatibles Datums- und Uhrzeitformat für die Ausgabe diverser Zeitvariablen. Eine Übersicht über die verfügbaren `strftime()`-Optionen finden Sie in Kapitel 11, »Logging«.

Hier sehen Sie für jede Variante ein Beispiel:

```
<!--#config errmsg="Bei der SSI-Verarbeitung ist ein Fehler
aufgetreten." -->
<!--#config sizefmt="abbrev" -->
<!--#config timefmt="%d.%m.%Y, %h:%i" -->
```

#### echo

Mithilfe dieses Elements können Sie den Wert einer Umgebungsvariablen ausgeben. Das Element besitzt zwei Attribute:

- `var="Variable"`
  Dieses Attribut gibt die Variable an, deren Wert Sie ausgeben möchten. Zusätzlich zu den in Kapitel 14, »CGI«, vorgestellten Umgebungsvariablen definiert SSI die folgenden eigenen Variablen:
  - `DATE_GMT`
    Datum und Uhrzeit in Greenwich Mean Time
  - `DATE_LOCAL`
    Datum und Uhrzeit der lokalen Zeitzone
  - `DOCUMENT_NAME`
    Dateiname des angeforderten Dokuments
  - `DOCUMENT_URI`
    URL-Pfad des angeforderten Dokuments
  - `LAST_MODIFIED`
    Datum und Uhrzeit der letzten Änderung des angeforderten Dokuments
  - `QUERY_STRING_UNESCAPED`
    Query-String der Anfrage; die URL-codierten `%`-Sequenzen werden aufgelöst, während Shell-relevante Zeichen mit vorangestellten Backslashes versehen werden.

  Wenn Sie versuchen, eine Variable zu verwenden, die nicht gesetzt ist, bestimmt die Direktive `SSIUndefinedEcho`, was geschieht.

- `encoding="none" | "url" | "entity"`
  Dieses optionale Argument bestimmt, auf welche Weise der Ausgabewert codiert wird. Wenn Sie es verwenden möchten, muss `encoding` vor dem Argument `var` stehen. Die drei möglichen Werte bedeuten Folgendes:
  - `"none"`
    Es wird keine Codierung verwendet. Dies ist aus Sicherheitsgründen normalerweise nicht zu empfehlen – unter Umständen könnte ein Angreifer auf diese Weise schädlichen Code einschmuggeln.
  - `"url"`
    Der Wert wird URL-codiert, das heißt, die meisten Sonderzeichen werden durch ein Prozentzeichen und den hexadezimalen Zeichencode ersetzt.
  - `"entity"`
    Diese Einstellung sorgt dafür, dass HTML-relevante Sonderzeichen durch die entsprechenden Entity-Referenzen ersetzt werden (z.B. `<` für < oder `&` für &). Da `echo` letztendlich statischen HTML-Code für den Browser erzeugt, ist dies normalerweise die richtige Wahl; deshalb ist es auch der Standardwert von `encoding`.

Das folgende Beispiel gibt den Namen des Browsers aus, mit dem der entfernte Benutzer die Anfrage gesendet hat:

```
<!--#echo var="HTTP_USER_AGENT" -->
```

#### exec

Dieses Element führt ein externes Programm oder CGI-Skript aus und fügt dessen Ausgabe an der aktuellen Position in das HTML-Dokument ein. Beachten Sie, dass `exec` nur verfügbar ist, wenn Sie die Option `Includes` gesetzt haben, nicht aber bei `IncludesNOEXEC`. Sie können eines der folgenden beiden Argumente verwenden:

- `cgi="URL-Pfad"`
  Das angegebene CGI-Skript wird ausgeführt. Wenn der Pfad relativ ist, bezieht er sich auf das Verzeichnis des aktuellen Dokuments. Wenn das Skript anstelle eines Dokuments nur einen `Location`-Header zur Weiterleitung produziert, wird dieser in einen Hyperlink umgewandelt.
- `cmd="Dateipfad"`
  Der angegebene Befehl wird unter UNIX durch */bin/sh* ausgeführt, gilt also als Shell-Kommando; unter Windows wird der enthaltene Befehl ebenfalls durch das Betriebssystem ausgeführt.

In den meisten Fällen sollten Sie anstelle von `exec cgi` das Element `include` mit dem Attribut `virtual` verwenden. Hier zwei Beispiele:

```
<!--#exec cgi="/cgi-bin/test.cgi" -->
<!--#exec cmd="ls /usr/local/apache2/htdocs" -->
```

#### fsize

Dies gibt die Größe der angegebenen Datei in dem durch `sizefmt` definierten Format an. Das Element ist sehr praktisch für Linklisten oder Bildergalerien. Es gibt zwei mögliche Argumente:

- `file="Dateipfad"`
  Diese Option gibt den Pfad einer Datei an; er muss relativ zum Verzeichnis des aktuellen Dokuments angegeben werden.
- `virtual="URL-Pfad"`
  Bei dieser Variante ist der Wert ein URL-Pfad; falls er nicht mit `/` beginnt, ist er relativ zum aktuellen Dokument. Das bedeutet, dass Apache daraus zunächst eine URL konstruiert und diese als Datei betrachtet, deren Größe ermittelt wird.

Beispiel:

```
<!--#fsize file="bild.jpg" -->
```

#### flastmod

Dieses Element liefert Datum und Uhrzeit der letzten Änderung der angegebenen Datei; das Format folgt der `timefmt`-Einstellung. Sie können genau wie bei `fsize` eines der Argumente `file` oder `virtual` verwenden. Beispiel:

```
<!--#flastmod virtual="/cgi-bin/test.cgi" -->
```

#### include

Das Element `include` fügt den Inhalt der angegebenen Datei an der aktuellen Position ein. Dies ist beispielsweise nützlich, um stets wiederkehrende Elemente wie Kopf- oder Fußzeilen auszulagern. Wenn für das Verzeichnis, in dem die einzufügende Datei liegt, die Option `IncludesNOEXEC` gesetzt ist, können Sie nur Dokumente mit Text-MIME-Types einfügen. Andernfalls sind auch CGI-Skripte möglich; in diesem Fall wird deren Ausgabe eingebettet. Sie können wiederum eines der Argumente `file` oder `virtual` wählen; der Dateipfad bei `file` muss hier relativ zum aktuellen Verzeichnis sein und darf auch nicht den Bestandteil `../` für einen Wechsel in das übergeordnete Verzeichnis enthalten. Hier sehen Sie zwei Beispiele:

```
<!--#include file="start.html" -->
<!--#include virtual="/cgi-bin/test.cgi" -->
```

### printenv

Dieses Element gibt eine vollständige Liste aller Umgebungsvariablen aus; es besitzt keine Argumente. Beispiel:

```
<!--#printenv -->
```

### set

Mithilfe dieser SSI-Anweisung können Sie eine eigene Variable definieren. Sie müssen die beiden folgenden Argumente verwenden:

- `var="Variablenname"`
  Dies ist der Name der Variablen.
- `value="Variablenwert"`
  Dieses Argument gibt den Wert an, den Sie der Variablen zuweisen möchten.

Hier ein Beispiel:

```
<!--#set var="gruss" value="Hallo Welt!" -->
```

### if, elif, else, endif

Mithilfe dieser Elemente können Sie in Ihre Server Side Includes Fallentscheidungen einfügen. Schematisch betrachtet, gilt folgender Ablauf (eckige Klammern bezeichnen optionale Bestandteile):

```
<!--#if expr="Bedingung" -->
  ... beliebiger HTML/SSI-Code ...
[<!--#elif expr="Bedingung" -->
  ... beliebiger HTML/SSI-Code ...
[<!--#elif expr="Bedingung" -->
  ... beliebiger HTML/SSI-Code ...
] ... ]
[<!--#else -->
   ... beliebiger HTML/SSI-Code ...
]
<!--#endif -->
```

`if` enthält die erste Bedingung; der nachfolgende HTML- und/oder SSI-Code wird ausgeführt, wenn sie zutrifft. Anschließend können Sie optional ein `elif`-Element (»else if«) einfügen, dessen Bedingung nur überprüft wird, falls die `if`-Bedingung nicht zutrifft. Entsprechend lassen sich beliebig viele `elif`-Bedingungen ineinander verschachteln. Ein anschließendes, ebenfalls optionales `else` (ohne Bedingung) leitet Code ein, der nur übernommen wird, wenn keine der Bedingungen wahr ist. Die gesamte Sequenz muss schließlich durch `endif` abgeschlossen werden.

Die Bedingungen für `if` und `elif` werden als Werte des Arguments `expr` gesetzt. Seit Version 2.3.13 wird die Syntax des allgemeinen Ausdrucksparsers `ap_expr` verwendet, die in Kapitel 6, »Grundkonfiguration«, in der Beschreibung der Direktive `<If>` erläutert wurde.

Bis einschließlich Apache 2.3.12 kommt eine ähnliche, aber nicht ganz so leistungsfähige Syntax zum Einsatz, die sich in neueren Versionen mit der Direktive `SSILegacyExprParser on` aktivieren lässt. Ausdrücke in dieser Syntax können aus folgenden Komponenten bestehen:

- **Literale**
  Jede einfache Zeichenkombination wird wörtlich genommen.
- **Variablen**
  Zeichenfolgen, denen Sie ein Dollarzeichen voranstellen, werden als Variablen ausgewertet. Dies können sowohl Umgebungsvariablen als auch selbst definierte Variablen sein. Beispiele: `$test` oder `$SERVER_NAME`.
- **Reguläre Ausdrücke**
  Zeichenfolgen, die von zwei Slashes umschlossen werden, werden als Perl-kompatible reguläre Ausdrücke interpretiert. Beispiel: `/^[a-z]/` – alles, was mit einem Kleinbuchstaben beginnt.
- **Vergleichsoperatoren**
  Definiert sind `=`, `!=`, `<`, `>`, `<=` und `>=`; sie dürften selbsterklärend sein.
- **Zugriffstest**
  Ein Konstrukt nach dem Schema `-A URL-Pfad` trifft zu, wenn der Zugriff auf den angegebenen URL-Pfad mit den derzeitigen Konfigurationseinstellungen möglich ist. Diese Option ist nur verfügbar, wenn die unten beschriebene Direktive `SSIAccessEnable` auf `on` gesetzt wird. Andernfalls wird der Ausdruck einfach als String betrachtet.
- **Logische Verknüpfungen**
  `&&` (logisches Und), `||` (logisches Oder) und `!` (logisches Nicht)
- **Klammern**
  Um die Priorität der Auswertung von Teilausdrücken zu beeinflussen, können Sie Klammern setzen.

Das folgende Beispiel führt eine Art »Content-Negotiation light« durch: Es überprüft den Anfrage-Header `Accept-Language` (die bevorzugte Sprache, die der Client wünscht) und gibt eine Begrüßung in der ersten Sprache aus dieser Liste aus. Wenn die Sprache nicht den Bedingungen entspricht, wird Englisch verwendet:

```
<!--#if expr="req('ACCEPT_LANGUAGE') =~ /^de/" -->
  Guten Tag!
<!--#elif expr="req('ACCEPT_LANGUAGE') =~ /^en/" -->
```

```
  Good Afternoon!
<!--#elif expr="req('ACCEPT_LANGUAGE') =~ /^fr/" -->
  Bonjour!
<!--#elif expr="req('ACCEPT_LANGUAGE') =~ /^it/" -->
  Buongiorno!
<!--#else -->
  Good Afternoon!
<!--#endif -->
```

In der Legacy-Syntax sieht dieselbe Sequenz so aus:

```
<!--#if expr="$HTTP_ACCEPT_LANGUAGE = /^de/" -->
  Guten Tag!
<!--#elif expr="$HTTP_ACCEPT_LANGUAGE = /^en/" -->
  Good Afternoon!
<!--#elif expr="$HTTP_ACCEPT_LANGUAGE = /^fr/" -->
  Bonjour!
<!--#elif expr="$HTTP_ACCEPT_LANGUAGE = /^it/" -->
  Buongiorno!
<!--#else -->
  Good Afternoon!
<!--#endif -->
```

Die regulären Ausdrücke, die jeweils den Beginn des Header-Wertes untersuchen, sind notwendig, weil der Header meist mehrere Sprachen enthält und die einzelnen Sprachen zudem optionale Länderkürzel enthalten können (zum Beispiel `en-GB` für britisches und `en-US` für amerikanisches Englisch). Näheres über die `Accept-*`-Header finden Sie in Kapitel 2, »Funktionsweise von Webservern«, sowie in Kapitel 7, »Header und MIME-Einstellungen«.

### 16.1.3 mod_include-Direktiven

`mod_include` stellt einige Konfigurationsdirektiven bereit, die verschiedene Einstellungen des SSI-Verhaltens ermöglichen. In diesem Unterabschnitt werden sie vorgestellt.

#### SSIStartTag

Startmarkierung eines Include-Elements

| | |
|---|---|
| **Seit Version** | 2.0.30 |
| **Modul** | `mod_include` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSIStartTag Tag` |
| **Standardwert** | `<!--#` |

Diese Direktive bestimmt, mit welcher Zeichenfolge eine SSI-Anweisung beginnt. In der Regel können Sie die Voreinstellung `<!--#` beibehalten – es sei denn, Sie verwenden eine weitere serverseitige Programmiertechnologie, die dieselbe Sequenz benötigt. Das folgende Beispiel legt eine PHP-ähnliche Startzeichenfolge fest:

```
SSIStartTag "<?ssi "
```

### SSIEndTag

Endmarkierung eines Include-Elements

| | |
|---|---|
| **Seit Version** | 2.0.30 |
| **Modul** | `mod_include` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SSIEndTag Tag` |
| **Standardwert** | `-->` |

Diese Direktive bestimmt, mit welcher Zeichenfolge ein SSI-Element endet. Genau wie `SSIStartTag` brauchen Sie sie nur zu ändern, wenn es Konflikte mit anderen Sprachen geben könnte. Das folgende Beispiel korrespondiert mit dem oben vorgestellten Start-Tag `"<?ssi "`:

```
SSIEndTag "?>"
```

### SSIErrorMsg

SSI-Fehlermeldung

| | |
|---|---|
| **Seit Version** | 2.0.30 |
| **Modul** | `mod_include` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`All`) |
| **Syntax** | `SSIErrorMsg Fehlermeldung` |
| **Standardwert** | `"[an error occured while processing this directive]"` |

Die Direktive `SSIErrorMsg` legt fest, welche Fehlermeldung angezeigt werden soll, wenn bei der SSI-Verarbeitung ein Fehler auftritt. Die Direktive ist eine Alternative zum SSI-Element `<!--#config errmsg="Meldungstext" -->`.

Beispiel:

```
SSIErrorMsg "Fehler bei der SSI-Verarbeitung."
```

### SSITimeFormat

Format für Datum und Uhrzeit in Includes

| Seit Version | 2.0.30 |
|---|---|
| Modul | mod_include |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (All) |
| Syntax | SSITimeFormat Formatstring |
| Standardwert | "%A, %d-%b-%Y %H:%M:%S %Z" |

Mit dieser Direktive können Sie festlegen, wie Datums- und Uhrzeitangaben in SSI formatiert werden sollen; dies entspricht der SSI-Anweisung `<!--#config timefmt="Format" -->`. Der Wert ist ein `strftime()`-kompatibles Datums- und Uhrzeitformat; eine Liste finden Sie in Kapitel 11, »Logging«. Hier ein Beispiel:

```
SSITimeFormat "%d.%m.%Y, %h:%i"
```

### SSIUndefinedEcho

SSI-Anzeige bei Ausgabe einer undefinierten Variablen

| Seit Version | 2.0.34 |
|---|---|
| Modul | mod_include |
| Kontext | Server, <VirtualHost> |
| Syntax | SSIUndefinedEcho String |
| Standardwert | "(none)" |

Diese Direktive bestimmt, was ausgegeben wird, wenn eine Variable bei Verwendung im `echo`-Element nicht definiert ist. Beispiel:

```
SSIUndefinedEcho "[nicht vorhanden]"
```

### XBitHack

SSI-Direktiven mit gesetztem Execute-Bit verarbeiten

| Modul | mod_include |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Options) |
| Syntax | XBitHack on\|off\|full |
| Standardwert | off |

Mit dieser Direktive können Sie festlegen, dass Server Side Includes in gewöhnlichen HTML-Dokumenten ausgeführt werden sollen, falls für diese Dokumente

das Execute-Bit (Dateirecht »Ausführen«) gesetzt ist. Sie können ein HTML-Dokument auf einem UNIX-System also durch `chmod +x` zum SSI-Dokument machen.

Die drei möglichen Werte bedeuten Folgendes:

- `off`
  Das Execute-Bit wird nicht beachtet.
- `on`
  In jedem Dokument mit dem MIME-Type `text/html`, das der Eigentümer ausführen darf, wird SSI ausgeführt.
- `full`
  Bei dieser Einstellung wird zusätzlich das Execute-Bit der Gruppe untersucht. Ist es gesetzt, dann wird ein `Last-Modified`-Header mit dem Datum der letzten Änderung der Datei gesendet, sodass Clients und Proxys die Datei zwischenspeichern können.

#### SSIAccessEnable

Zugriffstest in SSI-Ausdrücken ermöglichen

| | |
|---|---|
| **Seit Version** | 2.2 |
| **Modul** | `mod_include` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`All`) |
| **Syntax** | `SSIAccessEnable on\|off` |
| **Standardwert** | `off` |

Wenn Sie `SSIAccessEnable`[1] auf `on` setzen, kann im betreffenden Kontext der oben beschriebene Zugriffstest `-A URL-Pfad` in SSI-Ausdrücken verwendet werden.

#### SSILegacyExprParser

Kompatibilitätsmodus für die klassische Ausdruckssyntax

| | |
|---|---|
| **Seit Version** | 2.3.13 |
| **Modul** | `mod_include` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`All`) |
| **Syntax** | `SSILegacyExprParser on\|off` |
| **Standardwert** | `off` |

1 Beachten Sie, dass die Direktive in der derzeitigen Apache-Online-Dokumentation fälschlicherweise als `SSIEnableAccess` bezeichnet wird.

Wenn Sie diese Direktive explizit auf `on` setzen, aktivieren Sie bei Versionen ab 2.3.13 in `#if`- und `#elif`-Anweisungen den herkömmlichen Modus für Ausdrücke anstelle von `ap_expr`. Damit lassen sich alte SSI-Seiten weiterverwenden. Mittelfristig sollten Sie diese jedoch auf `ap_expr` umstellen.

## 16.2 Filterkonfiguration

Eine der wichtigsten Neuerungen in der Apache-Version 2.0 sind die Filter. Dieses Konzept ermöglicht es, eingehende Daten vor der eigentlichen Anfrageverarbeitung zu verändern (Eingabefilter) oder ausgehende Daten vor der endgültigen Auslieferung an den Client zu modifizieren (Ausgabefilter). Filter können entweder als Module realisiert werden oder externe Programme sein, die mithilfe des Moduls `mod_ext_filter` aktiviert werden. In diesem Abschnitt erfahren Sie alles Wissenswerte über Filter:

- Direktiven zur Festlegung eingebauter beziehungsweise durch Module realisierter Filter
- Verwendung der Standardfiltermodule (`mod_deflate`; Beispielmodule zur Umwandlung von Groß- und Kleinschreibung)
- Definition und Implementierung externer Programme als Filter

### 16.2.1 Grundlegende Filterdirektiven

In diesem Abschnitt werden die Konfigurationsdirektiven vorgestellt, mit denen Sie Filter konfigurieren können, die durch Module bereitgestellt werden. Ähnlich wie die in Kapitel 7, »Header und MIME-Einstellungen«, vorgestellten Direktiven, die diverse MIME-Aspekte steuern, sind einige dieser Filterdirektiven im Core enthalten, während andere durch das Standardmodul `mod_mime` bereitgestellt werden.

Im Lieferumfang von Apache 2 sind nur wenige Filter enthalten, die Sie auf diese Weise einschalten können, unter anderem folgende: Der durch `mod_include` bereitgestellte Ausgabefilter `INCLUDE` wurde bereits ausführlich beschrieben. Der Filter `DEFLATE`, den das Modul `mod_deflate` zur Verfügung stellt und der ausgehende Daten automatisch komprimiert, wird dagegen weiter unten näher betrachtet.

Der einzige Eingabefilter, den die Apache-Standarddistribution zurzeit mitbringt, heißt ebenfalls `DEFLATE` – `mod_deflate` kann nämlich nicht nur Server-Antworten komprimieren, sondern auch Client-Anfragen dekomprimieren. Diese Anfragen

enthalten einen entsprechenden `Content-Encoding`-Header. Normale Browser verwenden allerdings keine Komprimierung; der Body einer `POST`-Anfrage ist für gewöhnlich auch nicht allzu groß. In der Praxis kommt das Verfahren deshalb nur bei den `PUT`-Anfragen einiger WebDAV-Clients zum Einsatz.

Es ist zu erwarten, dass in Zukunft weitere Standardfilter eingeführt werden. Die Technologie ist nämlich überaus praktisch, weil sie das Ändern oder Ersetzen von Inhalten ermöglicht, ohne den inneren Ablauf der Anfrageverarbeitung zu verändern. Es sind unzählige Anwendungsmöglichkeiten denkbar:

- *Eingabefilter* könnten die Client-Anfragen z.B. auf Angriffsversuche überprüfen, Zeichensätze modifizieren oder fehlende Einträge in Query-Strings ergänzen.
- *Ausgabefilter* könnten beliebige Modifikationen am HTML-Code vornehmen, Dokumente in ein völlig anderes Format konvertieren (denken Sie an PDF oder an die zahllosen Möglichkeiten im XML-Bereich), Inhalte hervorheben oder gar übersetzen.

Einige Möglichkeiten werden weiter unten anhand von Perl-Beispielskripten vorgestellt, die Sie als externe Filter einsetzen können.

Übrigens können sogar mehrere Filter nacheinander die Ein- oder Ausgabe verarbeiten; eine solche Abfolge von Filtern wird als *Filter Chain* bezeichnet.

#### SetOutputFilter

Setzt Ausgabefilter zur Verarbeitung von Inhalten vor der Ausgabe an den Client.

| | |
|---|---|
| **Seit Version** | 2.0 |
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `SetInputFilter Filter[;Filter...]` |
| **Standardwert** | nicht gesetzt |

Die Direktive `SetOutputFilter` bestimmt, dass alle Dateien im aktuellen Konfigurationskontext durch den oder die angegebenen Ausgabefilter verarbeitet werden sollen. Mit folgendem Beispiel werden alle Dateien unter dem URL-Pfad */compress* vor der Auslieferung `DEFLATE`-komprimiert:

```
<Location /compress>
  SetOutputFilter DEFLATE
</Location>
```

Wenn Sie mehrere Filter angeben möchten, müssen Sie ihre Namen durch Semikola ohne Leerzeichen voneinander trennen. Das folgende Beispiel führt in allen Dateien Server Side Includes aus und führt anschließend die DEFLATE-Komprimierung durch:

```
SetOutputFilter INCLUDES;DEFLATE
```

Diese Reihenfolge der beiden Filter ist übrigens sehr wichtig, weil mod_include in dem scheinbaren Zeichensalat, den mod_deflate ausgibt, natürlich keine SSI-Elemente mehr finden kann. Sie sollten es sich daher zur Regel machen, mod_deflate immer als letzten Filter in der Filter Chain zu setzen.

#### AddOutputFilterByType

Ausgabefilter nach MIME-Type zuweisen

| | |
|---|---|
| **Seit Version** | 2.0.33 |
| **Modul** | mod_filter (bis 2.2 core) |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| **Syntax** | AddOutputFilterByType Filter[;Filter...] MIME-Type [MIME-Type ...] |
| **Standardwert** | nicht gesetzt |

Diese Direktive konfiguriert einen oder mehrere Ausgabefilter für Dateien mit bestimmten MIME-Types. Die Direktive wurde nachträglich eingefügt, weil sie die Verwendung von Filtern mit Dateien ermöglicht, die nicht die »richtige« Dateiendung besitzen – auf diese Weise können Sie beispielsweise in der Ausgabe von CGI- oder PHP-Skripten auch noch Server Side Includes ausführen.

Das folgende Beispiel aktiviert den Ausgabefilter DEFLATE für alle Dateien mit den MIME-Types image/gif, image/jpeg und image/png – also für alle in der Praxis üblichen Bilddateitypen:

```
AddOutputFilterByType DEFLATE image/gif image/jpeg image/png
```

Wie bei der Angabe von MIME-Types üblich, können Sie auch hier Platzhalter einsetzen – die folgende Variante komprimiert wirklich alle Bilder:

```
AddOutputFilterByType DEFLATE image/*
```

Beachten Sie zu guter Letzt, dass die mit dieser Direktive festgelegten Filter nicht auf Dokumente angewendet werden, deren MIME-Type nicht ermittelt werden kann und deshalb durch DefaultType festgelegt wird. Dies gilt selbst dann, wenn der DefaultType in einer AddOutputFilterByType-Direktive enthalten ist.

### SetInputFilter

Legt die Eingabefilter zur Verarbeitung der Client-Anfrage fest.

| | |
|---|---|
| **Seit Version** | 2.0 |
| **Modul** | core |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| **Syntax** | SetInputFilter Filter[;Filter...] |
| **Standardwert** | nicht gesetzt |

Diese Direktive legt analog zu SetOutputFilter einen Eingabefilter für alle Dateien im aktuellen Kontext fest. Das folgende Beispiel aktiviert die DEFLATE-Dekomprimierung für den URL-Pfad */dav*:

```
<Location /dav>
  SetInputFilter DEFLATE
</Location>
```

### AddOutputFilter

Ausgabefilter nach Dateiendung festlegen

| | |
|---|---|
| **Seit Version** | 2.0.26 |
| **Modul** | mod_mime |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| **Syntax** | AddOutputFilter Filter[;Filter...] Dateiendung [Dateiendung ...] |
| **Standardwert** | nicht gesetzt |

Mithilfe dieser in mod_mime definierten Direktive können Sie einen oder mehrere Filter für Dateien mit bestimmten Dateiendungen festlegen. Wie bei mod_mime üblich, betrifft dies auch Dateien mit mehreren Endungen – das Verfahren wurde bereits in Kapitel 7, »Header und MIME-Einstellungen«, beschrieben. Das folgende Beispiel schaltet SSI in allen HTML-Dokumenten ein:

```
AddOutputFilter INCLUDES .htm .html
```

Die folgenden beiden Direktiven können Sie verwenden, wenn Sie die Verwendung von DEFLATE und INCLUDES durch zusätzliche Dateiendungen (.def beziehungsweise .ssi) steuern möchten:

```
AddOutputFilter DEFLATE .def
AddOutputFilter INCLUDES .ssi
```

Dies ist insbesondere für DEFLATE praktisch, denn durch ein solches Verfahren können Sie über eine Type-Map steuern, dass nur diejenigen Clients ein DEFLATE-komprimiertes Dokument erhalten sollen, die einen Accept-Encoding-Header mit dem Bestandteil deflate gesendet haben.

#### RemoveOutputFilter

Ausgabefilterzuordnung für Dateien mit bestimmten Endungen entfernen

| | |
|---|---|
| **Seit Version** | 2.0.26 |
| **Modul** | mod_mime |
| **Kontext** | <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| **Syntax** | RemoveOutputFilter<br>Dateiendung [Dateiendung ...] |
| **Standardwert** | nicht gesetzt |

Diese Direktive ermöglicht es Ihnen, die Zuordnung von Ausgabefiltern für bestimmte Dateiendungen wieder zu entfernen. Dies ist nützlich, um die aus einem übergeordneten Kontext stammende Voreinstellung aufzuheben. Das folgende Beispiel entfernt die Ausgabefilter für HTML-Dokumente:

```
RemoveOutputFilter .htm .html
```

#### AddInputFilter

Eingabefilter nach Dateiendung festlegen

| | |
|---|---|
| **Seit Version** | 2.0.26 |
| **Modul** | mod_mime |
| **Kontext** | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| **Syntax** | AddOutputFilter Filter[;Filter...] Dateiendung [Dateiendung ...] |
| **Standardwert** | nicht gesetzt |

Mit AddInputFilter können Sie einen oder mehrere Eingabefilter für bestimmte Dateiendungen festlegen. Die Syntax entspricht AddOutputFilter. Ein sinnvolles Beispiel kann hier nicht gegeben werden, weil der einzige Standardeingabefilter (DEFLATE) nicht aufgrund einer Dateiendung aktiviert werden sollte – Sie sollten ihn für DAV-Verzeichnisse mittels SetInputFilter aktivieren.

**RemoveInputFilter**

Filterzuordnung für Dateien mit bestimmten Endungen entfernen

| Seit Version | 2.0.26 |
|---|---|
| Modul | mod_mime |
| Kontext | <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | RemoveInputFilter Dateiendung [Dateiendung ...] |
| Standardwert | nicht gesetzt |

Diese Direktive entfernt die Zuordnung von Eingabefiltern für bestimmte Dateiendungen. Genau wie bei RemoveOutputFilter dient dies der gezielten Entfernung von Filtereinstellungen in untergeordneten Kontexten.

### 16.2.2 Freie Modifikation der Filter Chain mit mod_filter

Das für die Version 2.2 völlig neu entwickelte Apache-Modul mod_filter ermöglicht es, die Abfolge der Filter – die sogenannte *Filter Chain* – frei zu definieren und von den Inhalten beliebiger Anfrage-Header, Antwort-Header oder Umgebungsvariablen abhängig zu machen.

Das Einfügen der Filter in die Filter Chain erfolgt in drei Stufen: Zuerst wird ein Filter deklariert, dann wird ein FilterProvider für den Filter registriert, der vor allem die Bedingungen für den Einsatz des Filters regelt. Zum Schluss wird aus den definierten Providern mit der gleichnamigen Direktive die FilterChain gebildet, die die Reihenfolge der Filterdurchläufe festlegt.

Bevor die Direktiven des Moduls im Einzelnen beschrieben werden, hier ein Beispiel – es setzt einen Filter namens RESIZE, sofern die zurzeit ausgelieferte Ressource ein Bild ist, und fügt den Filter DECREASE_COLORS hinzu, falls es sich um ein GIF handelt:

```
FilterProvider res RESIZE resp=Content-Type $image
FilterProtocol res "change=yes"
FilterProvider dec DECREASE_COLORS resp=Content-Type $image/gif
FilterProtocol dec "change="yes"
FilterChain res dec
```

Bitte beachten Sie, dass es sich bei RESIZE und DECREASE_COLORS nicht um existierende Filter handelt, sondern um fiktive, die nur das Prinzip erläutern. In der Praxis müssten diese Filter separat hinzugefügt werden – entweder durch eigene Module (siehe Kapitel 17, »Apache erweitern«) oder durch die weiter unten besprochene Einbindung externer Programme als Filter mittels mod_ext_filter.

Dieses Modul ist sehr neu und unterliegt daher laufend Änderungen. Bitte konsultieren Sie die Website zum Buch, um sich auf dem Laufenden zu halten.

### FilterDeclare

Einen Filter deklarieren

| Seit Version | 2.1-beta |
|---|---|
| Modul | mod_filter |
| Kontext | <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Options) |
| Syntax | FilterDeclare Filtername [Typ] |
| Standardwert | nicht gesetzt |

FilterDeclare deklariert einen Filter zur Verwendung in den weiter unten beschriebenen Direktiven FilterProvider, FilterProtocol und FilterChain.

Das erste Argument ist ein beliebiger Filtername, das zweite ein Filtertyp, das heißt eine symbolische Konstante vom Typ ap_filter_type. Die möglichen Werte sind: RESOURCE, CONTENT_SET, PROTOCOL, TRANSCODE, CONNECTION und NETWORK. Die Bedeutung dieser Typen wird weiter unten in der Beschreibung der Direktive ExtFilterDefine erläutert.

Wenn der neue Filter den Standardtyp RESOURCE besitzt, ist diese Direktive optional – in diesem Fall übernimmt FilterProvider die Deklaration implizit.

### FilterProvider

Einen Filter registrieren und Bedingungen für seinen Einsatz festlegen

| Seit Version | 2.1-beta |
|---|---|
| Modul | mod_filter |
| Kontext | <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Options) |
| Syntax | FilterProvider Filtername Providername [req\|res\|env]=Feld Vergleichswert |
| Standardwert | nicht gesetzt |

Mithilfe dieser Direktive wird der Provider für den angegebenen Filter festgelegt; dies kann ein Modul oder ein mittels mod_ext_filter eingebundenes externes Programm sein. Zusätzlich werden die Bedingungen festgelegt, unter denen der entsprechende Filter zum Einsatz kommen soll.

Das erste Argument ist ein beliebiger Filtername; unter den oben beschriebenen Umständen müssen Sie diesen zuerst mittels `FilterDeclare` deklarieren.

Als Nächstes wird ein Provider angegeben, also der von einem Modul oder externen Programm festgelegte offizielle Filtername. Beispiele für eingebaute Filter sind `INCLUDES` (SSI durch `mod_include`) oder `DEFLATE`.

Das dritte Argument gibt den Typ und Namen des Feldes an, das überprüft wird, um zu bestimmen, ob der Filter ausgeführt wird. Die möglichen Typen sind `req` (Anfrage-Header), `res` (Antwort-Header) und `env` (Umgebungsvariable). Die möglichen Header sind in Kapitel 2, »Funktionsweise von Webservern«, verzeichnet; die vorgegebenen Umgebungsvariablen sowie die Möglichkeiten zur Definition eigener Variablen finden Sie dagegen in Kapitel 14, »CGI«. Wenn Sie keinen Typ angeben, wird `res` angenommen. Das wahrscheinlich häufigste Vergleichsfeld ist `res=Content-Type`, um den Dateityp der auszuliefernden Ressource zu überprüfen.

Zu guter Letzt wird noch ein Wert benötigt, mit dem das Feld verglichen werden soll. Dieser Vergleichswert kann eines der folgenden Formate besitzen:

- `String` – Der Wert des Feldes muss genau der angegebene String sein.
- `$String` – Der String muss im Feld enthalten sein (Substring).
- `/RegExp/` – Der angegebene reguläre Ausdruck muss auf das Feld zutreffen.
- `=Integer` – Der Wert muss gleich der angegebenen Ganzzahl sein.
- `<Integer` – Der Wert muss kleiner als die angegebene Zahl sein.
- `<=Integer` – Es wird ein Wert benötigt, der kleiner oder gleich der Zahl ist.
- `>Integer` – Hier muss der Wert größer als die angegebene Zahl sein.
- `>=Integer` – Trifft zu, wenn der Wert größer oder gleich der Zahl ist.
- `*` – Es genügt, wenn das Feld überhaupt existiert.

Optional können Sie dem Vergleichswert mitsamt seinem Vorzeichen noch ein Ausrufezeichen voranstellen – in diesem Fall wird der Filter nur ausgeführt, falls die Bedingung nicht zutrifft. Das folgende Beispiel definiert den Filter `kompress` mit dem Provider `DEFLATE`, falls der `Content-Type` ein Bild ist (mit `image` beginnt):

```
FilterProvider kompress DEFLATE resp=Content-Type /^image/
```

Beachten Sie, dass Sie für ein und denselben Filter auch verschiedene Provider festlegen können. Auf diese Weise stellen Sie sicher, dass dieser spezifische Filter an einer bestimmten Stelle in der Filter Chain ausgeführt wird, obwohl je nach Voraussetzungen unterschiedliche konkrete Aufgaben erledigt werden. Hier ein Beispiel, das unterschiedliche »Verkleinerer« für die verschiedenen Bilddateitypen registriert:

```
FilterProvider resize gif_resizer resp=Content-Type image/gif
FilterProvider resize jpeg_resizer resp=Content-Type image/jpeg
FilterProvider resize png_resizer resp=Content-Type image/png
```

### FilterProtocol

HTTP-Protokolländerungen für Filter vornehmen

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | mod_filter |
| **Kontext** | <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (Options) |
| **Syntax** | FilterProtocol Filtername [Provider] Schlüssel=Wert [Schlüssel=Wert ...] |
| **Standardwert** | nicht gesetzt |

Manche Filter ändern die Eigenschaften einer Ressource so, dass die eigentlich bereits erzeugten HTTP-Antwort-Header nicht mehr zu ihnen passen; eines der wichtigsten Beispiele ist die `Content-Length`, also die Größe der Ressource. Diese Direktive ermöglicht daher die Angabe von Protokolländerungen für bestimmte Filter beziehungsweise Filter-Provider.

Das erste Argument ist ein mittels `FilterProvider` registrierter Filtername. Das optionale zweite Argument ist ein Filter-Provider; Sie brauchen es nur anzugeben, falls Sie für einen Filternamen unterschiedliche Provider registriert haben und die Protokolländerung nur für einen von ihnen gelten soll.

Alle weiteren Argumente haben die Form `Schlüssel=Wert` und können aus folgendem Vorrat möglicher Einstellungen schöpfen:

- `change=yes` – Der Inhalt der Ressource wird geändert, wahrscheinlich auch die `Content-Length`.
- `change=1:1` – Der Inhalt ändert sich, aber die Größe bleibt erhalten (z.B. bei bestimmten Zeichensatzkonvertierungen).
- `byteranges=no` – Der Filter benötigt die vollständige Ressource; mit Teilbereichen kommt er nicht zurecht (wichtig bei den meisten Binärdateien wie Bildern oder anwendungsspezifischen Daten).
- `proxy=no` – Der Filter sollte nicht ausgeführt werden, wenn Apache als Proxy arbeitet.
- `proxy=transform` – Die Änderungen durch den Filter beeinflussen die Ressource so, dass der Header `Cache-Control: no-transform` (siehe Kapitel 2, »Funktionsweise von Webservern«) nicht mehr gültig ist.

- `cache=no` – Die Ausgabe des Filters kann nicht in einem Cache abgelegt werden (weil jeder Durchlauf einen anderen Inhalt generiert).

### FilterChain

Festlegung einer Filter Chain

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_filter` |
| **Kontext** | `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`Options`) |
| **Syntax** | `FilterChain [+-=@!]Filtername [[+-=@!]Filtername ...]` |
| **Standardwert** | nicht gesetzt |

Diese Direktive baut aus Filtern, die zuvor mittels `FilterProvider` registriert wurden, eine Filter Chain auf. Die Argumente sind die entsprechenden Filternamen; optional können Sie folgende Zeichen verwenden:

- `+Filtername` – Der Filter wird am Ende der Filter Chain eingefügt; wenn Sie das Pluszeichen weglassen, geschieht dasselbe.
- `-Filtername` – Der angegebene Filter wird aus der Filter Chain entfernt.
- `=Filtername` – Die bisherige Filter Chain wird durch den angegebenen Filter ersetzt.
- `@Filtername` – Der Filter wird am Anfang der Filter Chain eingefügt.
- `!` – Die Filter Chain wird komplett geleert.

Hier ein einfaches Beispiel, das die (fiktiven) Filter `resize` und `kompress` einfügt:

```
FilterChain resize kompress
```

### FilterTrace

Debug-Informationen für `mod_filter` erzeugen

| | |
|---|---|
| **Seit Version** | 2.1-beta |
| **Modul** | `mod_filter` |
| **Kontext** | `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `FilterTrace Filtername Level` |
| **Standardwert** | 0 |

Diese Direktive schreibt auf Wunsch Debug-Informationen für `mod_filter` in die `ErrorLog`-Datei. Beim Ausprobieren neuer Filter Chains ist es eine gute Idee, diese Funktion vorübergehend zu aktivieren. Die möglichen Werte sind:

- 0 – keine Debug-Informationen
- 1 – Alle Buckets und Brigades (Ressourceninhalte), die den Filter durchlaufen, werden im `ErrorLog` verzeichnet.
- 2 – Alle Daten, die die Filter Chain durchlaufen, werden in eine temporäre Datei geschrieben, bevor die Provider sie verarbeiten. Diese Funktion ist noch nicht implementiert.

### 16.2.3 Der Komprimierungsfilter mod_deflate

Die wichtigste Aufgabe des Moduls `mod_deflate` ist die Bereitstellung des bereits erwähnten Filters `DEFLATE`. Wenn dieser Filter als Ausgabefilter eingesetzt wird, komprimiert er den Body der HTTP-Antwort, bevor er an den Client gesendet wird. Als Eingabefilter dekomprimiert er dagegen gegebenenfalls den Body der Anfrage.

Die Deflate-Komprimierung ist ein patentfreies, offenes Kompressionsverfahren, das auf den beiden Komponenten LZ77 und Huffman-Codierung basiert. Der Algorithmus wird durch die Bibliothek `zlib` bereitgestellt. Die meisten neueren Browser beherrschen die Dekomprimierung solcher Dateien und teilen dies dem Server durch einen entsprechenden `Accept-Encoding`-Header mit.

`mod_deflate` definiert einige Direktiven, die Sie für die Feineinstellung des Kompressionsvorgangs verwenden können. Diese werden im Folgenden vorgestellt.

#### DeflateBufferSize

Größe der Fragmente, die `zlib` komprimieren soll

| | |
|---|---|
| **Modul** | `mod_deflate` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `DeflateBufferSize Größe` |
| **Standardwert** | `8096` |

Die Deflate-Funktion komprimiert keine beliebig großen Dateien am Stück, sondern nimmt sich jeweils einzelne Blöcke einer bestimmten Größe vor. Die Direktive `DeflateBufferSize` bestimmt die Größe dieser Blöcke. In aller Regel brauchen Sie an der Voreinstellung von 8 Kilobyte nichts zu ändern.

### DeflateCompressionLevel

Komprimierungsfaktor für `mod_deflate`

| Seit Version | 2.0.45 |
|---|---|
| Modul | `mod_deflate` |
| Kontext | Server, `<VirtualHost>` |
| Syntax | `DeflateCompressionLevel Größe` |
| Standardwert | `zlib`-Standard |

Diese sehr neue Direktive bestimmt die Kompressionsrate, mit der `mod_deflate` Dateien komprimieren soll. Die möglichen Werte reichen von 1 bis 9, wobei 1 für die geringste und 9 für die höchste Rate steht. Beachten Sie aber, dass ein hoher Komprimierungsfaktor zwar den Traffic minimiert, aber dafür sehr rechenintensiv ist und Performance kostet. Beispiel:

```
DeflateCompressionLevel 5
```

### DeflateFilterNote

Komprimierungsfaktor für Log-Dateien veröffentlichen

| Seit Version | 2.0.45 |
|---|---|
| Modul | `mod_deflate` |
| Kontext | Server, `<VirtualHost>` |
| Syntax | `DeflateFilterNote [Typ] Name` |
| Standardwert | nicht gesetzt |

Diese Direktive erzeugt eine Variable mit dem angegebenen Namen und speichert darin bei jeder `DEFLATE`-Verwendung den Komprimierungsfaktor. Diesen Wert können Sie in eine selbst definierte Log-Datei schreiben. Das folgende Beispiel erzeugt eine Variable namens `faktor` und definiert anschließend eine separate Log-Datei für Kompressionsvorgänge:

```
DeflateFilterNote faktor
CustomLog logs/deflate_log \
'"%r" %b (%{faktor}n) "%{User-agent}i"'
```

Die einzelnen Platzhalter für Log-Formate wurden bereits in Kapitel 11, »Logging«, ausführlich beschrieben.

Optional können Sie als erstes Argument der Direktive einen der folgenden drei Typen angeben, um nicht nur die Kompressionsrate, sondern auch noch zwei andere Aspekte der Komprimierung auszulesen:

- `Input` – Anzahl der Bytes im Eingabestrom (entspricht der Dokumentgröße vor der Komprimierung)
- `Output` – Anzahl der Bytes im Ausgabestrom (Dokumentgröße nach der Komprimierung)
- `Ratio` – die Kompressionsrate, also `Output / Input * 100`. Dies ist der Standardfall, wenn Sie die Typangaben weglassen.

Die folgenden Konfigurationsanweisungen definieren eine Log-Datei, in der alle diese Informationen gesammelt werden:

```
DeflateFilterNote Input eingang
DeflateFilterNote Output ausgang
DeflateFilterNote Ratio faktor
CustomLog logs/deflate_log \
'"%r" %{ausgang}n/%{eingang}n (%{faktor}n%%)'
```

#### DeflateMemLevel

Speicherzuteilung für `zlib`-Komprimierung

| | |
|---|---|
| **Modul** | mod_deflate |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `DeflateMemLevel Wert` |
| **Standardwert** | `9` |

Mit dieser Direktive können Sie einen Richtwert für die Speichermenge festlegen, die Sie `zlib` zuteilen möchten. 1 steht für die geringste Speichermenge, 9 für die höchste. Mit einer größeren Speichermenge komprimiert `zlib` stärker, mit einer kleineren geht es schneller.

#### DeflateWindowSize

Fenstergröße für `zlib`-Komprimierung

| | |
|---|---|
| **Modul** | `mod_deflate` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `DeflateWindowSize Wert` |
| **Standardwert** | `15` |

Diese Direktive bestimmt die Größe des sogenannten *Komprimierungsfensters*, das von `zlib` verwendet wird. Sie können einen Wert von 1 bis 15 verwenden; je größer er ist, desto stärker ist der Komprimierungsfaktor. Sie brauchen die Voreinstellung 15 also nur zu vermindern, falls spezielle Probleme auftreten sollten.

### 16.2.4 mod_charset_lite

Dieses Modul ist eine abgespeckte Version des Moduls `mod_charset`, das speziell für die russische Version von Apache kreiert wurde. `mod_charset_lite` ermöglicht die dynamische Konvertierung von Dokumenten in einen anderen Zeichensatz, bevor sie ausgeliefert werden. Dies betrifft aber eben nicht nur den MIME-Header `Content-Type` mit seiner `charset`-Information, sondern auch den tatsächlichen Datenbestand der Dateien selbst.

Eine Liste von Zeichensätzen, die als Werte für `CharsetDefault` oder `CharsetSourceEnc` infrage kommen, finden Sie in Anhang E. Ob eine bestimmte Konvertierung möglich ist, können Sie auf einem UNIX-System mit dem Konsolenprogramm `iconv` testen. Es besitzt folgende Syntax:

```
# iconv -f Quellzeichensatz -t Zielzeichensatz [Datei]
```

Wenn Sie keine Datei angeben, wird bis EOF von der Standardeingabe gelesen.[2] Hier ein Beispiel:

```
# iconv -f iso-8859-1 -t utf-8 test.txt
```

Normalerweise werden die Funktionen von `mod_charset_lite` nicht durch Filtereinstellungen, sondern durch die drei im Folgenden vorgestellten Direktiven angesprochen. Dennoch definiert es zusätzlich den Eingabefilter `XLATEIN`[3] und den Ausgabefilter `XLATEOUT`, die Sie bei Bedarf mithilfe von Filterdirektiven setzen können.

#### CharsetSourceEnc

Ausgangszeichensatz, der umgewandelt werden soll

| | |
|---|---|
| **Modul** | `mod_charset_lite` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `CharsetSourceEnc Zeichensatz` |
| **Standardwert** | nicht gesetzt |

Mit dieser Direktive wird der Quellzeichensatz der Dokumente angegeben, die konvertiert werden sollen.

Beispiel:

```
CharsetSourceEnc utf-8
```

2 Falls Sie den Text von Hand eintippen, können Sie EOF durch Strg + D (UNIX) beziehungsweise Strg + Z (Windows) erzeugen.

3 Das hat nichts mit »Latein« zu tun, sondern steht für Translate-Input.

### CharsetDefault

Zeichensatz, in den Dateien umgesetzt werden sollen

| Modul | mod_charset_lite |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | CharsetDefault Zeichensatz |
| Standardwert | nicht gesetzt |

Diese Direktive gibt an, in welchen Zeichensatz die Dokumente im aktuellen Kontext konvertiert werden sollen. Beispiel:

```
CharsetDefault iso-8859-1
```

### CharsetOptions

Konfiguration der Zeichensatzumwandlung

| Modul | mod_charset_lite |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (FileInfo) |
| Syntax | CharsetOptions Option [Option ...] |
| Standardwert | DebugLevel=0 NoImplicitAdd |

Mithilfe dieser Direktive können Sie zwei Optionen für das Verhalten von `mod_charset_lite` setzen:

- `DebugLevel=[0-9]`. Diese Direktive bestimmt, wie viele Debug-Informationen `mod_charset_lite` generieren und in die `ErrorLog`-Datei schreiben soll. Konkret sind folgende Werte definiert (siehe `modules/experimental/mod_charset_lite.c` im Apache-Quellcode):
  - `9`: detaillierte Informationen über den Arbeitsablauf
  - `4`: Informationen über jede einzelne Anfrage
  - `2`: nur Warnungen
  - `0`: keine Informationen

  Beachten Sie, dass `4` und `9` den Server massiv verlangsamen können und nur kurzfristig zum Aufspüren von Konfigurationsproblemen benutzt werden sollten.
- `ImplicitAdd | NoImplicitAdd`. Die Einstellung `ImplicitAdd` wendet die `mod_charset_lite`-Filter `XLATEIN` und `XLATEOUT` automatisch an, sobald ein Dokument mit einem passenden Quellzeichensatz gefunden wird. Wenn Sie selbst mittels `AddInputFilter` beziehungsweise `AddOutputFilter` eine Filter Chain

einrichten möchten, sollten Sie besser `NoImplicitAdd` verwenden und die Filter manuell hinzufügen, weil sie beispielsweise nach der Ausführung von `DEFLATE` nicht mehr funktionieren können.

### 16.2.5 mod_xml2enc

Zusammen mit `mod_proxy_html` (siehe Kapitel 13, »Proxy- und Cache-Funktionen«) wurde auch das Modul `mod_xml2enc` in Apache 2.4 integriert; zuvor war es ebenfalls ein Drittanbieter-Modul. Es stellt Internationalisierung und Zeichensatzkorrekturen für Module zur Verfügung, die auf `libxm2` basieren. Von den mitgelieferten Modulen ist dies zurzeit nur `mod_proxy_html`, aber Modulentwickler können auch für ihre eigenen `libxm2`-basierten Module auf die Funktionen dieses Moduls zugreifen. Dazu kann es als Filter-Provider mit dem Namen `xml2enc` in die Filterkette eingefügt werden.

Anders als `mod_charset_lite` arbeitet `mod_xml2enc` nicht mit einem zuvor bekannten Zeichensatz, sondern ermittelt ihn durch Untersuchung des zu verarbeitenden Dokuments selbst. Dazu werden die folgenden Regeln in der angegebenen Reihenfolge angewendet, wobei der erste Treffer gilt:

1. Wenn der Header `Content-Type` den Parameter `charset` enthält, wird der darin angegebene Zeichensatz verwendet.
2. Wenn die Daten eine XML-Encoding-Deklaration oder Byte Order Mark (BOM) enthalten, wird diese Information verwendet.
3. Wenn ein HTML-Meta-Tag vorhanden ist, das den Zeichensatz angibt, wird dieser verwendet.
4. Andernfalls wird der durch `xml2EncDefault` festgelegte Wert verwendet.

#### xml2EncAlias

Aliasnamen für bekannte Zeichensätze angeben

| | |
|---|---|
| **Seit Version** | 2.4; für ältere 2.x-Versionen als Drittanbieter-Modul |
| **Modul** | `mod_xml2enc` |
| **Kontext** | Server |
| **Syntax** | `xml2EncAlias Zeichensatz Alias [Alias ...]` |
| **Standardwert** | nicht gesetzt |

Mit dieser Direktive können Aliasnamen für bekannte Zeichensätze gesetzt werden, damit sie auch unter Namen erkannt werden, die `libxml2` nicht kennt. Hypothetisches Beispiel (in Wirklichkeit werden alle drei Namen unterstützt):

```
xml2EncAlias iso-latin-1 iso-8859-1 iso-8859-15
```

#### xml2EncDefault

Anzunehmender Standardzeichensatz, wenn er nicht anders erkannt werden kann

| Seit Version | 2.4; für ältere 2.x-Versionen als Drittanbieter-Modul |
|---|---|
| Modul | mod_xml2enc |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (All) |
| Syntax | xml2EncAlias Zeichensatz Alias [Alias ...] |
| Standardwert | utf-8 |

In den allgemeinen Erläuterungen zu mod_xml2enc wurde bereits beschrieben, wie das Modul versucht, den Zeichensatz eines Dokuments zu bestimmen. Diese Direktive gibt den Fallback-Wert an, der verwendet wird, wenn keine andere Möglichkeit besteht, den Zeichensatz zu ermitteln. Das folgende Beispiel stellt ISO-Latin-1 anstelle von UTF-8 ein:

```
xml2EncDefault iso-8859-1
```

#### xml2StartParse

Unstrukturierte Daten am Dokumentbeginn überspringen

| Seit Version | 2.4; für ältere 2.x-Versionen als Drittanbieter-Modul |
|---|---|
| Modul | mod_xml2enc |
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (All) |
| Syntax | xml2StartParse Element [Element ...] |
| Standardwert | nicht gesetzt |

Wenn ein Backend-Server vor den eigentlichen XML- oder HTML-Daten Sonderzeichen oder sonstige nicht standardkonforme Daten ausgibt, können Sie mit dieser Direktive das XML- oder HTML-Element (auf Wunsch auch mehrere) festlegen, bei dem der Parser mit der Arbeit beginnen soll. Wenn Sie wissen, dass Sie valides HTML oder XML erhalten, brauchen Sie diese Direktive nicht. Der mod_proxy_html- und mod_xml2enc-Entwickler Nick Kew liefert in seinem Blog unter *http://bahumbug.wordpress.com/2006/10/12/mod_proxy_html-revisited/* ein Praxisbeispiel, das zur Einführung dieser Direktive führte.

### 16.2.6 Inhalt ersetzen mit mod_substitute

Die neueste Errungenschaft im Bereich der Filter ist das seit Revision 2.2.7 verfügbare Modul mod_substitute. Es ermöglicht das Ersetzen beliebiger Inhalte auszuliefernder Dokumente mithilfe regulärer Ausdrücke oder wahlweise litera-

ler Strings. Die einzige Direktive des Moduls ist `Substitute`; eine Anleitung zur Einrichtung als Ausgabefilter sowie einige Anwendungsbeispiele finden Sie in der Beschreibung der Direktive.

### Substitute

Konfiguration der Zeichensatzumwandlung

| | |
|---|---|
| **Modul** | `mod_substitute` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`FileInfo`) |
| **Syntax** | `Substitute s/Regexp/Ersatztext/[infq]` |
| **Standardwert** | nicht gesetzt |

Die Schreibweise `s/Regexp/Ersatztext/[Flags]` kennen Sie vielleicht aus der Programmiersprache Perl, aus der sie für diese Direktive entlehnt wurde. Genau wie dort können Sie auch in dieser Direktive anstelle von `/` ein anderes Trennzeichen benutzen, beispielsweise `|` oder `!` – wichtig ist nur, dass an allen drei Stellen dasselbe Zeichen verwendet wird. Dies ist beispielsweise nützlich, wenn im regulären Ausdruck oder im Ersatztext selbst ein `/` vorkommt, denn andernfalls müssten Sie das fragliche Zeichen durch einen vorangestellten `\` escapen.

Falls im regulären Ausdruck oder im Ersatztext Leerzeichen vorkommen, müssen Sie das gesamte Argument wie üblich in Anführungszeichen setzen.

Das erste Element des Arguments ist ein Perl-kompatibler regulärer Ausdruck; einige Beispiele dafür haben Sie bereits in Kapitel 8, »Weiterleitungen und Indizes«, im Rahmen von `mod_rewrite` gesehen. In Anhang G finden Sie zudem eine Übersicht der wichtigsten RegExp-Konstrukte.

Wenn Teile des regulären Ausdrucks in runden Klammern stehen, können die entsprechenden Treffer-Strings im Ersatztext als `$1`, `$2` usw. eingesetzt werden – die Zählung der geklammerten Bereiche erfolgt jeweils von links nach rechts und bei verschachtelten Klammern von außen nach innen. Davon abgesehen, ist der Ersatztext ein einfacher, literaler String. Bis einschließlich Version 2.2.9 darf der Ersatztext *nicht leer* sein. Wenn Sie also mithilfe von `Substitute` Inhalte komplett entfernen möchten, müssen Sie in diesen Versionen mindestens ein Leerzeichen als Ersatztext angeben. In Version 2.2.10 wurde diese Beschränkung aufgehoben.

Hinter dem Ersatztext und seinem Trennzeichen können optional beliebig viele der folgenden Flags oder Modifier stehen:

- `i` – Groß- und Kleinschreibung werden nicht beachtet.
- `n` – Das Suchmuster wird nicht als regulärer Ausdruck, sondern als einfacher String betrachtet. Wann immer es möglich ist, sollten Sie von dieser Option

Gebrauch machen, da die String-Suche wesentlich schneller ist als der RegExp-Vergleich.

- `f` – Auf das Ergebnis der Ersetzung können weitere Ersetzungsregeln angewendet werden (Standard).
- `q` – Das Ergebnis der Ersetzung ist endgültig; weitere Ersetzungen sind nicht mehr möglich. Wenn keine weiteren Ersetzungen mehr geplant sind, sollten Sie diese Option wählen, da sie den Prozess beschleunigt. Die Modifier `f` und `q` schließen einander logischerweise aus.

In der Praxis wird im gewünschten Verzeichniskontext zunächst der Ausgabefilter `SUBSTITUTE` gesetzt, beispielsweise mithilfe von `AddOutputFilterByType` für Dokumente mit einem bestimmten MIME-Type. Anschließend folgen eine oder mehrere `Substitute`-Direktiven. Das folgende Beispiel stellt durch zwei Ersetzungsregeln für HTML-Dateien sicher, dass `<br>`-Tags (Zeilenumbrüche) den XHTML-Endslash erhalten, aus `<br>` also beispielsweise `<br />` wird:

```
<Location /xhtml>
  AddOutputFilterByType SUBSTITUTE text/html
  Substitute "s|<br\s*/?>|<br />|iq"s
</Location>
```

Als Trennzeichen wurde hier jeweils `|` verwendet, weil der `/` im regulären Ausdruck und im Ersatztext verwendet wird. Der reguläre Ausdruck lautet `<br\s*/?>`. Die ersten drei Zeichen sind literal – dieser Teil findet also Strings, die mit `<br` beginnen. Danach folgt `\s*`, also beliebig viel Whitespace, denn das betreffende Tag kann `<br>`, `<br/>` oder `<br />` lauten. `/?` ist ein optionaler `/`, weil es möglicherweise Tags gibt, die dieses Zeichen bereits enthalten. Zum Schluss steht die literale, schließende Spitzklammer. Der Ersatztext lautet in jedem Fall `<br />`, und zum Schluss wird die Option `i` gesetzt, weil HTML-Tags in manchen älteren Dokumenten großgeschrieben wurden.

#### Substitute-Praxisbeispiele

Hier noch zwei weitere Beispiele für den Einsatz von `Substitute` – das erste wandelt HTML in Plaintext um, das zweite erfüllt die Aufgabe von `mod_rewrite` für Hyperlinks im Dokument. Letzteres dürfte einer der wichtigsten Verwendungszwecke von `mod_substitute` werden.

Die Umwandlung von HTML in Text folgt hier einem naiven Ansatz: Jeglicher Text zwischen zwei spitzen Klammern wird entfernt; dies könnte beispielsweise Probleme mit JavaScript- oder CSS-Blöcken machen, die dann ohne ihr `<script>`- beziehungsweise `<style>`-Gerüst stehenbleiben.

Hier die Konfiguration für einen URL-Pfad namens */html2text*:

```
<Location /html2text>
  AddOutputFilter SUBSTITUTE .html
  AddType text/plain .html
  Substitute s/<[^>]+>//
</Location>
```

Der Ausgabefilter wird hier mittels `AddOutputFilter` für die Dateiendung `.html` gesetzt, nicht mithilfe von `AddOutputFilterByType`. Denn da das Ergebnis der Ersetzung reiner Text ist, wird anschließend der MIME-Type `text/plain` für diese Endung festgelegt.

Der reguläre Ausdruck in der `Substitute`-Direktive beginnt mit einer öffnenden Spitzklammer (`<`), gefolgt von einem oder mehreren Zeichen, die keine schließende Spitzklammer sind (`[^>]+`). Am Ende steht die schließende Klammer (`>`). Der Ersatztext ist ein leerer String; denken Sie daran, dass Sie in Version 2.2.7 bis 2.2.9 stattdessen ein Leerzeichen angeben müssen, das wiederum Anführungszeichen für das Argument bedingt:

```
Substitute "s/<[^>]+>/ /"
```

Betrachten Sie das folgende kleine Beispieldokument:

```
<html>
<head>
<title>Test</title>
</head>
<body>
<h1>Text aus HTML</h1>
Dieser Text wurde aus einem HTML-Dokument extrahiert.
</body>
</html>
```

Der Filter erzeugt daraus Folgendes:

```
Test


Der Test
Dies ist so ein Dokument, das könnt ihr mal lesen.
```

Da der Filter zeilenweise arbeitet, bleiben einige Leerzeilen zwischen dem Text stehen.

Im zweiten Beispiel sollen interne Hyperlinks auf Seiten wie *test.html* durch *content.php?page=test* ersetzt werden – normalerweise ein typischer Anwendungs-

fall für mod_rewrite, aber in diesem Fall eben bereits auf die Links angewendet und nicht erst auf die entsprechende Client-Anfrage. Für den URL-Pfad */content* sähe das Ganze so aus:

```
<Location /content>
  AddOutputFilter SUBSTITUTE .html .php
  Substitute "s/<a(.*?)href=\"([a-z0-9_-]+).html\"(.*?)>/
<a$1href=\"content.php?page=$2\"$3>/i" [4]

</Location>
```

Auch hier wird der Ausgabefilter mit der Direktive AddOutputFilter festgelegt, weil er auf Dateien mit den beiden Endungen .html und .php angewendet werden soll. Der reguläre Ausdruck zur Ersetzung lautet:

```
<a(.*?)href=\"([a-z0-9_-]+).html\"(.*?)>
```

Zwischen <a und href steht das Konstrukt (.*?), da hier noch weitere Attribute stehen könnten, die in den Ersetzungs-String mitgenommen werden müssen. .* steht dabei wie gewohnt für beliebig viele beliebige Zeichen; allerdings ist dieser Ausdruck gemäß den Regeln für Perl-kompatible reguläre Ausdrücke »greedy« (zu Deutsch gierig); er würde den href-Teil einfach mitnehmen. Das nachgestellte Fragezeichen bildet die Non-greedy-Variante; ein solcher Teilausdruck reicht dann nur noch bis zum Beginn des nächsten möglichen Treffers.

Die Anführungszeichen für den Wert des Attributs href müssen durch einen Backslash escaped werden, um nicht mit den Anführungszeichen des Substitute-Arguments selbst zu kollidieren (diese sind hier allerdings optional, da weder der reguläre Ausdruck noch der Ersatztext Leerzeichen enthalten).

Für den Dateinamen vor der Endung .html wurden nur spezielle Zeichen ausgewählt, um sicherzustellen, dass nur lokale Hyperlinks ohne Verzeichnispfad ersetzt werden; konkret handelt es sich um Buchstaben[5], Ziffern, Unterstriche und Bindestriche (Minuszeichen).

Hinter dem schließenden Anführungszeichen des Attributs href folgt noch ein (.*?)-Ausdruck, der eventuelle nachstehende Attribute aufnimmt. Zum Schluss folgt die schließende spitze Klammer.

Der Ersatztext verwendet die Platzhalter $1 bis $3, um die Zeichen vor href="...", den Dateinamen sowie die Zeichen nach href="..." an den passenden Stellen einzufügen.

4 In Wirklichkeit darf diese Zeile keinen Umbruch enthalten; aufgrund des Satzspiegels geht es hier leider nicht anders.

5 Genauer gesagt, Kleinbuchstaben, aber diese Einschränkung wird durch den Modifier i aufgehoben.

Der folgende kleine Ausschnitt aus einem HTML-Dokument mag als Testparcours dienen:

```
<a href="about.html">Über uns</a>
<a style="text-decoration: none"
href="imprint.html">Impressum</a>
<a href="news.html" style="background-color: #FFFF00">News</a>
<a href="http://www.galileocomputing.de">Externer Link</a>
```

Nach der Ersetzung lauten die Links folgendermaßen:

```
<a href="content.php?page=about">Über uns</a>
<a style="text-decoration: none"
href="content.php?page=imprint">Impressum</a>
<a href="content.php?page=news" style="background-color:
#FFFF00">News</a>
<a href="http://www.galileocomputing.de">Externer Link</a>
```

## 16.3 Externe Filter programmieren

Eine der Stärken des Filterkonzepts besteht darin, dass Sie mithilfe des Moduls `mod_ext_filter` beliebige externe Programme als Ein- oder Ausgabefilter festlegen können. In diesem Abschnitt werden zunächst die Direktiven dieses Moduls vorgestellt; anschließend lernen Sie einige in Perl geschriebene Beispiele kennen. Beachten Sie, dass Sie `mod_substitute` vorziehen sollten, wenn Ihr spezifischer Anwendungsfall dies zulässt – da das Modul ein Bestandteil von Apache selbst ist, funktioniert es erheblich schneller und stabiler als externe Filterskripte.

### 16.3.1 mod_ext_filter

Dieses Modul besitzt nur zwei Direktiven: `ExtFilterDefine` ermöglicht die Festlegung eines externen Programms als Filter, während `ExtFilterOptions` einige spezielle Verarbeitungsoptionen festlegt.

#### ExtFilterDefine

Einen externen Filter definieren

| Seit Version | 2.0 (seit 2.1 auch Eingabefilter) |
|---|---|
| Modul | `mod_ext_filter` |
| Kontext | Server |

| Syntax | `ExtFilterDefine Filtername Parameter [Parameter ...]` |
|---|---|
| Standardwert | nicht gesetzt |

Mit dieser Direktive können Sie ein externes Programm als Filter definieren. Wie weiter unten genauer beschrieben wird, erhält dieses Programm die Daten über die Standardeingabe und muss den gefilterten, das heißt beliebig modifizierten, Inhalt über die Standardausgabe wieder ausgeben. Bisher können Sie nur Ausgabefilter definieren; für die Zukunft sind auch Eingabefilter vorgesehen.

Das erste Argument ist ein beliebiger Name für den Filter, den Sie in Direktiven wie `SetOutputFilter` oder `AddOutputFilter` verwenden können, um diese Filterdefinition tatsächlich einzusetzen. Selbstverständlich darf jeder Name nur einmal verwendet werden. Hinter dem Namen können Sie einige der folgenden Parameter angeben:

- `cmd="Befehl"`
  Dies ist der einzige Pflichtparameter; er gibt das Programm an, das ausgeführt werden soll. Auch Kommandozeilenparameter können Sie an dieser Stelle übergeben. Beispiel:

  ```
  cmd="/usr/local/filters/myfilter param1 param2"
  ```

  Wenn Sie keine Binärprogramme, sondern Skripte verwenden, müssen Sie unter Windows zuerst den Pfad des Interpreters angeben und diesem anschließend den Skriptnamen als Parameter übergeben. Beispiel:

  ```
  cmd="C:/Perl/bin/perl.exe C:/filter/myfilter.pl"
  ```

- `mode=output|input`
  Dies ist der Modus des Filters (Ein- oder Ausgabefilter). In Apache 2.0 konnte nur `mode=output` verwendet werden; dies ist nach wie vor der Standardwert.
- `intype=Typ`
  Mit diesem Parameter können Sie den MIME-Type der Dokumente angeben, die von dem Filter verarbeitet werden sollen. Wenn Sie ihn weglassen, werden in dem Kontext, in dem Sie den Filter setzen, alle Dateien verarbeitet.

  Beispiel:

  ```
  intype=text/html
  ```

- `outtype=Typ`
  Dieser Parameter legt den MIME-Type fest, der dem Client im `Content-Type`-Header mitgeteilt werden soll. Standardmäßig bleibt er unverändert; wenn Ihr

Filter den Typ ändert, also beispielsweise Text in HTML oder XML in PDF konvertiert, müssen Sie den entsprechenden Typ angeben. Beispiel:

```
outtype=text/plain
```

- `PreservesContentLength`
  Diesen argumentlosen Parameter können Sie angeben, falls Ihr Filter die Byte-Anzahl (`Content-Length`) des Inhalts nicht ändert. In diesem Fall braucht er nicht neu berechnet zu werden, was Ressourcen spart. Weitaus die meisten Filter ändern allerdings die Länge des Dokuments.
- `ftype=Filtertyp`
  Der Filtermechanismus kennt einige leicht unterschiedliche Arten von Filtern; wenn Ihr Filter besondere Aufgaben erfüllen soll, müssen Sie hier möglicherweise einen anderen Typ festlegen. Der Standardtyp ist 10 (`AP_FTYPE_RESOURCE`), was in schätzungsweise 99 Prozent aller Fälle die richtige Wahl ist. Die in Klammern angegebenen symbolischen Konstanten sind im Apache-Quellcode in der Datei `include/util_filter.h` definiert; hier müssen Sie aber leider die numerischen Werte verwenden:
  - `10` (`AP_FTYPE_RESOURCE`): Diese Filter modifizieren den eingegebenen Inhalt. Ein Beispiel ist SSI.
  - `20` (`AP_FTYPE_CONTENT_SET`): Filter dieses Typs modifizieren den gesamten Inhalt, nachdem die `AP_FTYPE_RESOURCE`-Operationen abgeschlossen sind. Beispiel: `DEFLATE`.
  - `30` (`AP_FTYPE_PROTOCOL`): Solche Filter arbeiten auf der Ebene des Anwendungsprotokolls (z. B. HTTP).
  - `40` (`AP_FTYPE_TRANSCODE`): Dieser Filtertyp ist für Transfer-Codierungen wie z. B. Chunks zuständig.
  - `50` (`AP_FTYPE_CONNECTION`): Diese Filter klinken sich in die Verbindung zwischen Client und Server ein.
  - `60` (`AP_FTYPE_NETWORK`): Filter dieses Typs kümmern sich um die Datenübertragung zwischen Client und Server.
- `disableenv=Variable`
  Wenn Sie diesen Parameter angeben, deaktiviert das Vorhandensein der angegebenen Umgebungsvariablen den Filter.
- `enableenv=Variable`
  Dieser Parameter sorgt dafür, dass der Filter nur dann arbeitet, wenn die angegebene Variable gesetzt ist.

**ExtFilterOptions**

Optionen für `mod_ext_filter`

| Seit Version | 2.0 |
|---|---|
| Modul | `mod_ext_filter` |
| Kontext | `<Directory>`, `<Location>`, `<Files>` |
| Syntax | `ExtFilterOptions Option [Option ...]` |
| Standardwert | `DebugLevel=0 NoLogStderr` |

Mit dieser Direktive können Sie die beiden folgenden speziellen Optionen für die Ausführung externer Filter festlegen:

- `DebugLevel=[0-9]`
  Legt fest, wie ausführlich die Debug-Informationen sein sollen, die `mod_ext_filter` generiert. Die Voreinstellung 0 besagt, dass gar keine Debug-Meldungen erzeugt werden. Wenn Sie unerklärliche Probleme mit dem Modul haben, können Sie `DebugLevel` auf einen höheren Wert setzen; gleichzeitig müssen Sie `LogLevel` auf `Debug` setzen, damit die Meldungen überhaupt im `ErrorLog` landen. Die Datei `modules/filters/mod_ext_filter.c` definiert folgende konkreten Werte:
  - `1`: Meldung bei jeder Aktivierung des Filters
  - `2`: wichtige Fehlermeldungen
  - `9`: Bericht über jedes Detail
- `LogStdErr | NoLogStdErr`
  Mit diesen Optionen können Sie entscheiden, ob Meldungen, die das Modul auf die Standardfehlerausgabe (`stderr`) schreibt, in das `ErrorLog` übernommen werden sollen. Bei der Voreinstellung `NoLogStdErr` ist dies nicht der Fall; `LogStdErr` schaltet diese Option ein. Auch dies brauchen Sie nur bei Problemen mit dem Modul zu tun.

Das folgende Beispiel setzt das `DebugLevel` auf 1 und aktiviert die Übernahme von Fehlermeldungen:

```
ExtFilterOptions DebugLevel=1 LogStdErr
```

### 16.3.2 Beispiele für externe Filter

In diesem Abschnitt werden einige mehr oder weniger nützliche Beispiele für Filter vorgestellt. Sie sind in Perl geschrieben und bieten eine Basis für eigene Experimente. Beachten Sie, dass Sie häufig genutzte Filter aus Performance-Gründen

besser in C oder in einer anderen kompilierten Sprache implementieren sollten; besonders arbeitsintensive Filter sollten sogar als Module geschrieben werden.

### Ein X für ein U

Hier sehen Sie zum Einstieg einen sehr einfachen Filter: Er wandelt in der übergebenen Datei einfach jedes »u« in ein »x« um. Beachten Sie, dass er in dieser Form nur für einfache Textdateien (`text/plain`) geeignet ist – in HTML-Dokumenten macht es durchaus einen Unterschied, ob `<input>` oder `<inpxt>` verwendet wird; Binärdateien wie etwa Bilder werden durch eine solche Änderung sogar vollkommen unbrauchbar. Hier zunächst einmal das Skript:

```
#!/usr/bin/perl -w
while (<>) {
   s/u/x/gi;
   print;
}
```

Speichern Sie die Datei beispielsweise unter dem Namen `x4u.pl`; natürlich nicht in einem Verzeichnis, auf das Apache externen Benutzern Zugriff gewährt. Unter UNIX müssen Sie daran denken, die Datei ausführbar zu machen. Das folgende Konfigurationsbeispiel geht davon aus, dass Sie das Skript in dem Verzeichnis */usr/local/apache2/filters* gespeichert haben:

```
ExtFilterDefine x4u cmd="/usr/local/apache2/filters/x4u.pl" \
mode="output" intype="text/plain"
<Location /x4u>
  SetOutputFilter x4u
</Location>
```

Alle einfachen Textdateien unter dem URL-Pfad */x4u* werden nun durch diesen Filter bearbeitet. Aus »Hundert User« würde dadurch beispielsweise »Hxndert Xser«. Besonders nützlich ist das natürlich nicht, aber es zeigt das Prinzip.

Unter Windows müssen Sie, wie bereits erwähnt, den Pfad des Perl-Interpreters angeben. Wenn sich Ihr Perl-Interpreter unter *C:\Perl\bin* befindet und die Filter im Verzeichnis *C:\filter* gespeichert sind, können Sie folgende Konfiguration benutzen:

```
ExtFilterDefine x4u \
cmd="C:/Perl/bin/perl.exe C:/filter/x4u.pl" \
intype="text/plain"
<Location /x4u>
  SetOutputFilter x4u
</Location>
```

### Text in HTML konvertieren

Der folgende Filter konvertiert einfache Textdateien in HTML-Dokumente. Dabei verwendet er nicht einfach `<pre> ... </pre>`, was automatische Zeilenumbrüche bei zu langen Zeilen verhindern würde, sondern stellt die Festbreitenschrift mittels `<tt> ... </tt>` ein und wandelt führende Leerzeichen sowie Abfolgen mehrerer Leerzeichen in ` `-Folgen um. Zusätzlich müssen bestimmte Sonderzeichen wie `<` und `>` in die entsprechenden Escape-Sequenzen umgewandelt werden. In der Regel ist dieses Ausgabeformat für Textdateien geeigneter als das Originalformat, in dem die Browser sie anzeigen.

Hier sehen Sie zunächst das eigentliche Skript, `txt2html.pl`:

```
#!/usr/bin/perl -w
# Dokumentkopf ausgeben
print << "ENDHEADER";
<html>
<head>
<title>Automatisch erzeugtes HTML</title>
<meta http-equiv=\"content-type\" content=\"text/html;
charset=iso-8859-1\" />
</head>
<body>
<tt>
ENDHEADER
# Umwandlungsanweisungen
while (my $line = <>) {
   $line =~ s/&/&amp;/g;
   $line =~ s/</&lt;/g;
   $line =~ s/>/&gt;/g;
   $line =~ s/"/&quot;/g;
   $line =~ s/\n/<br \/>/g;
   $line =~ s/(\s+)/' '.(' ' x (length ($1) - 1))/eg;
   $line =~ s/^(\s*)/' ' x length ($1)/eg;
   print $line;
}
print << "ENDFOOTER";
</tt>
</body>
</html>
ENDFOOTER
```

Unter UNIX wird dieser Filter beispielsweise folgendermaßen aktiviert; wichtig ist hier der Parameter `outtype`, da sich der MIME-Type des Dokuments ändert:

```
ExtFilterDefine txt2html \
cmd="/usr/local/apache2/filters/txt2html.pl" \
mode="output" intype="text/plain" outtype="text/plain"
<Location /texte>
  SetOutputFilter txt2html
</Location>
```

Für Windows gelten auch hier wieder die obigen Anmerkungen.

#### Quellcode von HTML-Dokumenten anzeigen

Dieser Filter wandelt HTML-Dokumente so um, dass der Browser sie nicht auswertet, sondern den eigentlichen HTML-Code anzeigt. Dies geschieht durch die Umwandlung der HTML-relevanten Sonderzeichen in Entity-Referenzen. Beachten Sie, dass das &-Zeichen zuerst umgewandelt werden muss, damit die anderen Entity-Referenzen nicht versehentlich doppelt behandelt werden.

Dies ist der Code des Filterskripts:

```
#!/usr/bin/perl -w
print << "ENDHEADER";
<html>
<head>
<title>HTML-Code-Ansicht</title>
</head>
<body>
<pre>
ENDHEADER
while (<>) {
   s/&/&amp;/g;
   s/</&lt;/g;
   s/>/&gt;/g;
   s/"/&quot;/g;
   print;
}
print << "ENDFOOTER";
</pre>
</body>
</html>
ENDFOOTER
```

Sie können den Filter beispielsweise folgendermaßen in Betrieb nehmen:

```
ExtFilterDefine sourceview \
cmd="/usr/local/apache2/filters/sourceview.pl" \
mode="output" intype="text/html"
```

```
<Location /html_tutorial/sources>
  SetOutputFilter sourceview
</Location>
```

### Konfigurierbarer Farbwechsler

Das letzte Beispiel wandelt die Grundfarben eines HTML-Dokuments um, die im `<body>`-Tag definiert werden. Sie können die Hintergrundfarbe, die Textfarbe, die Linkfarbe, die Farbe für besuchte Links und die Farbe für aktivierte Links nacheinander als Kommandozeilenparameter angeben; für nicht gesetzte Werte verwendet das Skript Voreinstellungen. Beachten Sie, dass der Lesebefehl hier `<STDIN>` anstelle von `<>` lautet, weil Perl andernfalls versuchen würde, die Kommandozeilenargumente nacheinander als Dateien zu öffnen und daraus zu lesen. Der Code für diesen Filter sieht folgendermaßen aus:

```
#!/usr/bin/perl -w
use strict;
my $bgcolor = $ARGV[0] || '#FFFFFF';
my $textcolor = $ARGV[1] || '#000000';
my $linkcolor = $ARGV[2] || '#0000FF';
my $vlinkcolor = $ARGV[3] || '#9900FF';
my $alinkcolor = $ARGV[4] || '#FF0000';
while (<STDIN>) {
   s/bgcolor="[^"]+"/bgcolor="$bgcolor"/i;
   s/text="[^"]+"/text="$textcolor"/i;
   s/link="[^"]+"/link="$linkcolor"/i;
   s/vlink="[^"]+"/vlink="$vlinkcolor"/i;
   s/alink="[^"]+"/alink="$alinkcolor"/i;
   print;
}
```

Die folgenden Konfigurationsanweisungen richten den Filter ein und stellen dabei Farben ein, die einigermaßen zusammenpassen:

```
ExtFilterDefine colors \
cmd="/usr/local/apache2/filters/colors.pl #000099 #0099FF \
#CCCCCC #FFFFFF #99FF99" mode="output" intype="text/html"
<Location /colors>
  SetOutputFilter colors
</Location>
```

Natürlich tritt der Nutzen dieses Filters allmählich in den Hintergrund, weil die meisten HTML-Dokumente ihre Farben mit CSS definieren. Aber auch dafür lässt sich ein passendes Skript programmieren.

## 16.4 Zusammenfassung

Das Filterkonzept, eines der wichtigsten neuen Merkmale von Apache 2, kann die Flexibilität und Leistungsfähigkeit Ihrer Websites erheblich verbessern. Durch die Kombination eingebauter und selbst programmierter Filter zu Filter Chains lassen sich nahezu beliebige Manipulationen an den eingehenden und auszugebenden Inhalten vornehmen. Das neue Apache-2.2-Modul `mod_filter` erhöht die Flexibilität der Filterdefinition noch weiter.

Einer der ältesten Mechanismen für dynamische Inhalte, Server Side Includes (SSI), wurde in Apache 2 ebenfalls als Filter realisiert. Das ermöglicht früher undenkbare Kombinationen wie die Verarbeitung von SSI in CGI-Skripten. Server Side Includes sind ideal geeignet, um sehr schnell und einfach dynamische Daten in ansonsten statische Websites einzufügen.

Eine andere interessante Erweiterung im Zusammenhang mit Filtern ist das Modul `mod_ext_filter`: Es ermöglicht die schnelle und einfache Programmierung von Filtern, weil Sie im Grunde jedes beliebige externe Programm, das von der Standardeingabe lesen und auf die Standardausgabe schreiben kann, als Filter benutzen können. Die wenigen Beispiele in diesem Kapitel sind nur ein erster Ausblick auf die vielfältigen Möglichkeiten; im Web finden Sie unzählige weitere Beispiele. Das brandneue Modul `mod_substitute` kann für einige Anwendungsgebiete von `mod_ext_filter` innerhalb von Apache selbst eingesetzt werden und bietet somit eine performancebezogene und stabilere Alternative.

*»Sobald einer in einer Sache Meister geworden ist,*
*sollte er in einer neuen Sache Schüler werden.«*
*– Gerhard Hauptmann*

# 17 Apache erweitern

In diesem Kapitel werden verschiedene Möglichkeiten vorgestellt, den Apache-Webserver um neue Funktionen zu erweitern. Zunächst erhalten Sie einen Einblick in die Möglichkeiten von `mod_dav`, der mit Apache mitgelieferten WebDAV-Erweiterung. Anschließend werden diverse Drittanbieter-Module und ihre Bezugsquellen genannt. Zum Schluss erfahren Sie in einem Tutorial, wie Sie selbst Apache-Module programmieren können.

## 17.1 WebDAV

*Web-based Distributed Authoring and Versioning* (WebDAV) ist eine Technologie, die den Zugriff auf Dateien und Verzeichnisse im Web beinahe so flexibel gestaltet wie die Arbeit mit lokalen Dateien und Ordnern. Es handelt sich um ein in RFC 2518 standardisiertes Verfahren, das einige Erweiterungsmethoden zu HTTP definiert.

Für den Zugriff auf ein DAV-Verzeichnis benötigen Sie einen passenden Client. Unter Windows ist die Unterstützung durch die sogenannten *Webordner* bereits eingebaut; auch grafische Website-Editoren wie Macromedia Dreamweaver und Adobe GoLive verfügen über diese Funktion. WebDAV ist nämlich erheblich praktischer für die Pflege einer Website als FTP.

Apache 2 enthält mit `mod_dav` eine eingebaute Implementierung des WebDAV-Protokolls. In diesem Abschnitt werden alle Direktiven dieses Moduls vorgestellt.

Die DAV-Implementierung in Apache arbeitet auch hier nach dem Prinzip, das in diesem Buch bereits für `mod_proxy` und `mod_cache` beschrieben wurde: `mod_dav` stellt die Grundfunktionalität zur Verfügung; Provider-Module implementieren eine bestimmte konkrete Variante. Mit Apache 2 wird zurzeit nur das Provider-Modul `mod_dav_fs` geliefert, das ebenfalls aktiviert werden muss. Es verwirklicht DAV-Verzeichnisse auf der Basis des lokalen Dateisystems.

### 17.1.1 Konfigurationsbeispiel

Bevor in Abschnitt 17.1.2, »DAV-Direktiven«, die einzelnen Konfigurationsanweisungen von `mod_dav` und `mod_dav_fs` besprochen werden, sehen Sie hier zunächst ein Komplettbeispiel. Bei WebDAV müssen Sie besonders auf die Sicherheit achten. Schließlich wird den entsprechenden Usern der Schreibzugriff erlaubt. Sie sollten für das entsprechende Verzeichnis also unbedingt Authentifizierungseinstellungen verwenden. Details zu den verschiedenen Verfahren finden Sie in Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«.

Das folgende Beispiel aktiviert ein DAV-Verzeichnis unter *www.mynet.de/dav*; der Zugriff ist (außer für harmlose HTTP-Methoden) auf die Benutzer aus der Textdatei *.htdav* zur Basic-Authentifizierung beschränkt:

```
DavLockDB /usr/local/apache2/var/DavLock

<Location /dav>

  Dav On

  AuthType Basic
  AuthName WebDAV
  AuthUserFile .htdav

  <LimitExcept GET HEAD OPTIONS>
    Require valid-user
  </LimitExcept>

</Location>
```

### 17.1.2 DAV-Direktiven

Im Folgenden finden Sie die Referenz der Direktiven, die die Module `mod_dav` und `mod_dav_fs` bereitstellen.

#### DAV

WebDAV-HTTP-Methoden aktivieren

| | |
|---|---|
| **Modul** | `mod_dav` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `Dav On\|Off\|Provider-Name` |
| **Standardwert** | `Off` |

Ein Verzeichnis oder URL-Pfad, für den die WebDAV-Funktionalität eingeschaltet werden soll, benötigt diese Direktive. Sie kann nur in Verzeichniskontexten eingesetzt werden. Um DAV in einem bestimmten Container zu aktivieren, können Sie folgende Anweisung verwenden:

```
Dav On
```

Dies schaltet die Funktionalität von `mod_dav_fs` ein. Wenn später andere Provider-Module zur Verfügung stehen, muss anstelle von `On` der Name des jeweiligen Providers verwendet werden; für den Dateisystem-Provider können Sie alternativ auch Folgendes schreiben:

```
Dav filesystem
```

#### DavDepthInfinity

Ermöglicht `PROPFIND`-Anfragen mit beliebiger Tiefe.

| | |
|---|---|
| **Modul** | `mod_dav` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `DavDepthInfinity on\|off` |
| **Standardwert** | `off` |

Wenn Sie diese Direktive aktivieren, werden `PROPFIND`-Anfragen mit dem Header `Depth: Infinity` zugelassen. Mit anderen Worten: Clients wird erlaubt, zum Auffinden von Ressourcen beliebig tiefe Verzeichnisstrukturen zu durchsuchen. Sie sollten Ihre User genau kennen (oder selbst der einzige User sein), bevor Sie hier den Wert `On` setzen. Da das Verfahren recht ressourcenhungrig ist, lässt es sich nämlich für DoS-Attacken missbrauchen.

#### DavMinTimeout

Mindestzeit für Sperren einer DAV-Ressource

| | |
|---|---|
| **Modul** | `mod_dav` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>` |
| **Syntax** | `DavMinTimeout Sekunden` |
| **Standardwert** | `0` |

Wenn ein DAV-Client eine Ressource anfordert, wird diese standardmäßig für eine bestimmte Zeit gesperrt. Diese Direktive ermöglicht es Ihnen, eine Mindestsperrzeit einzurichten, die auch dann verwendet wird, wenn der Client eine kürzere Zeit angefordert hat. Das folgende Beispiel stellt fünf Minuten ein:

```
DavMinTimeout 300
```

#### DavLockDB

Pfad der DAV-Lock-Datenbank

| Modul | mod_dav_fs |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | DavLockDB Dateipfad |
| Standardwert | nicht gesetzt |

Diese Direktive wird vom Modul mod_dav_fs bereitgestellt. Es handelt sich um den Pfad für eine SDBM-Datei, die verwendet wird, um Informationen über ausgecheckte und gesperrte Dateien zu speichern. Sie müssen den vollständigen Pfad der Datei angeben, allerdings ohne Dateiendung. Die User-ID, unter der Apache ausgeführt wird, benötigt Schreibrechte für das entsprechende Verzeichnis. Wenn Sie einen relativen Pfad angeben, gilt er relativ zur ServerRoot. Beispiel:

```
DavLockDB var/DavLockDB
```

#### DavGenericLockDB

Pfad einer verallgemeinerten DAV-Lock-Datenbank

| Seit Version | 2.1-beta |
|---|---|
| Modul | mod_dav_lock |
| Kontext | Server, <VirtualHost> |
| Syntax | DavGenericLockDB Dateipfad |
| Standardwert | nicht gesetzt |

Das in Apache 2.1 neu eingeführte Modul mod_dav_lock implementiert nur diese eine Direktive. Sie stellt eine verallgemeinerte Locking-Schnittstelle für Provider-Module zur Verfügung; innerhalb von Apache ist bisher kein solches Modul enthalten. mod_dav_fs verwendet sein eigenes Locking-Verfahren und benötigt diese Direktive nicht.

## 17.2 Weitere Module

In diesem Abschnitt werden zunächst Module vorgestellt, die die Multiprotokoll-Unterstützung von Apache 2 verwenden. Es wird also die Tatsache, dass außer HTTP noch andere Protokolle implementiert werden können, näher untersucht. Im Anschluss daran finden Sie eine Liste von Drittanbieter-Modulen. Diese sind auf der DVD zum Buch enthalten.

### 17.2.1 Multiprotokoll-Unterstützung

Eine der interessantesten Neuerungen in Apache 2 ist die grundsätzliche Unterstützung beliebiger TCP-Anwendungsprotokolle. Es lassen sich also Module implementieren, die Internetdienste wie E-Mail, News oder FTP unterstützen. Auch Module, die bereits an anderen Stellen in diesem Buch vorgestellt wurden, konnten auf diese Weise leicht in Apache 2.0 implementiert werden – beispielsweise SSL oder die Proxy-Unterstützung. Der besondere Vorteil ist, dass Server, die als Apache-Module realisiert werden, von sämtlichen Apache-Features wie Filtern oder Authentifizierungseinstellungen profitieren können.

Mindestens drei derartige Module sind bereits im Entwicklungsstadium verfügbar:

- `mod_ftpd`
  Dieses Modul implementiert einen einfachen FTP-Server. Sie finden es unter *www.outoforder.cc/projects/apache/mod_ftpd* sowie auf der beiliegenden DVD.
- `mod_pop3`
  Diese experimentelle Implementierung eines POP3-Mail-Servers als Apache-Modul wird von der Apache Group selbst bereitgestellt. Sie finden das Modul zurzeit nur in den CVS-Repositorys der ASF; Webzugriff ist über *cvs.apache.org/viewcvs.cgi/httpd-pop3* möglich.
- `mod_daytime`
  Um die Multiprotokoll-Unterstützung genauer auszuprobieren, habe ich selbst ein kleines Modul geschrieben. Es beantwortet Anfragen gemäß `daytime`-Protokoll, antwortet also auf jede beliebige Anfrage mit Datum und Uhrzeit. Es wird weiter unten im Abschnitt über die Modulprogrammierung behandelt und ist natürlich ebenfalls auf der DVD enthalten.

Sie sollten zurzeit nicht einmal darüber nachdenken, diese Module in der Praxis einzusetzen. Sie sind bis jetzt nicht mehr als schöne Spielzeuge für einen verregneten Sonntagnachmittag. Dennoch ist davon auszugehen, dass auf der Basis der Multiprotokoll-Unterstützung im Laufe der Zeit auch professionell einsetzbare Module entstehen werden.

Zum Lieferumfang von Apache 2 gehört noch ein weiteres Modul, das ein alternatives Protokoll implementiert: `mod_echo` stellt einen gewöhnlichen `echo`-Server zur Verfügung. Das `echo`-Protokoll kann zum Testen einer Netzwerkverbindung eingesetzt werden; wie der Name schon sagt, wird die Zeichenfolge, die Sie dem `echo`-Server senden, absolut identisch wieder zurückgeschickt.

Wenn Sie Apache mit `mod_echo` kompiliert haben, können Sie beispielsweise folgende Konfiguration verwenden – sie richtet den `echo`-Dienst für den Standardport 7 ein, auf dem dieses Protokoll traditionell ausgeführt wird:

```
# mod_echo laden, falls DSO
LoadModule echo_module modules/mod_echo.so

# An Port 7 lauschen
Listen 7

# Virtueller Host für Echo
<VirtualHost *:7>
  ProtocolEcho On
</VirtualHost>
```

**ProtocolEcho**

`mod_echo` ein-/ausschalten

| Seit Version | 2.0 |
|---|---|
| Modul | `mod_echo` |
| Kontext | Server, `<VirtualHost>` |
| Syntax | `ProtocolEcho On\|Off` |
| Standardwert | nicht gesetzt |

Wie Sie gerade in der Beispielkonfiguration gesehen haben, schaltet diese Direktive den `echo`-Dienst ein. Sinnvollerweise sollten Sie, wie oben gezeigt, einen virtuellen Host dafür verwenden. Aus Sicherheitsgründen empfiehlt es sich, `mod_echo` nur zum Testen, aber nicht für den Produktiveinsatz zu benutzen.

### 17.2.2 Weitere Drittanbieter-Module

In Tabelle 17.1 sehen Sie eine Liste von Drittanbieter-Modulen, die auf der beiliegenden DVD mitgeliefert werden. Es handelt sich ausschließlich um Module, die von Apache 2 unterstützt werden. Einige wurden bereits an anderen Stellen in diesem Buch erwähnt. Es handelt sich um annähernd alle Module, die unter *http://modules.apache.org* verzeichnet sind; nur einige absolute Exoten und offiziell nicht mehr weiterentwickelte Module wurden nicht berücksichtigt.

| Modul | Lizenz | Beschreibung |
|---|---|---|
| `mod_acronym` | Apache | Ausgabefilter zum Ersetzen von Akronymen |
| `mod_accessibility` | Reserved | Filter oder Proxy, um die Barrierefreiheit von Websites on-the-fly zu verbessern |
| `mod_access_dnsbl` | Apache | Zugriffskontrolle über DNSBL-Black- und Whitelists |

**Tabelle 17.1** Drittanbieter-Module (alle auf der DVD zu finden)

| Modul | Lizenz | Beschreibung |
|---|---|---|
| mod_activex_filter | GPL | ActiveX-Filterung für Apache-Proxy |
| mod_antispam | Apache | Schutz vor Referer-Spams (wird vor allem oft in Blog-Trackbacks hinterlassen) |
| mod_apache_snmp | Apache, GPL | SNMP-Modul für Apache (Simple Network Management Protocol, verbreitetes Protokoll zur Netzwerküberwachung und -steuerung |
| mod_authnz_external | Apache | Provider für Authentifizierung über externe Skripte, um z.B. passwd-Dateien anzusprechen |
| mod_authnz_ibmdb2 | Apache | IBM DB2 als Authentifizierungs-Provider |
| mod_authn_dbi | Apache | Authentifizierung gegen Datenbanken per libdbi (Provider-Modul) |
| mod_authn_sasl | Apache | SASL-Sockets als Authentifizierungs-Provider |
| mod_authn_yubikey | Apache | Provider zur Authentifizierung über YubiKey von Yubico (USB-Schlüssel mit Open-Source-Authentifizierungssystem) |
| mod_authz_fsacl | GPL | Zugriffskontrolle über Access Control Lists (ACL) des Betriebssystems |
| mod_authz_unixgroup | Open | Zugriffskontrolle anhand der Mitgliedschaft in UNIX-Gruppen |
| mod_auth_dacs | eigene | integriertes Single-Sign-on-System für Authentifizierung und Zugriffskontrolle (benötigt keine anderen mod_auth*-Module) |
| mod_auth_memcookie | Apache | Re-Authentifizierung per Cookie |
| mod_auth_msfix | GPL | Bugfix für WebDAV mit WinXP-Clients |
| mod_auth_mysql | Apache | Authentifizierung gegen MySQL-Datenbank |
| mod_auth_oracle | GPL | Authentifizierung gegen Oracle-Datenbank |
| mod_auth_pam | Apache | PAM-Authentifizierung |
| mod_auth_pamacea | GPL | PAM-Authentifizierung |
| mod_auth_pamport | GPL | PAM-Schnittstelle |
| mod_auth_sim | Apache | Session-Tracking mit MySQL/Cookies |
| mod_auth_timeout | Apache | Idle-Timeout für Basic-Authentifizierung |
| mod_auth_user_dir | GPL | Authentifizierung gegen Home-Verzeichnisse |
| mod_bandwidth | eigene | Bandbreiten- und Verbindungssteuerung pro virtuellem Host oder Verzeichnis |
| mod_bash | Apache | eingebettete Bash-Shell |
| mod_bt | Apache | BitTorrent-Tracker mit API für Perl und PHP |
| mod_bwshare | Artistic License | Bandbreitendrosselung und -überwachung pro Client-IP-Adresse |

**Tabelle 17.1** Drittanbieter-Module (alle auf der DVD zu finden) (Forts.)

| Modul | Lizenz | Beschreibung |
|---|---|---|
| mod_caldav | LGPL | Modul zur Verwaltung von Terminen und Kalenderdaten über WebDAV (CalDAV) |
| mod_captcha | GPL | integrierter Generator für CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart; Bilder mit verzerrtem Text, um Formulare vor automatisiertem Zugriff zu schützen) |
| mod_cband | GPL | Bandbreitenbegrenzung für Virtual Hosts |
| mod_cbroker | eigene | HTTP/CORBA-Gateway |
| mod_cfg_ldap | BSD | Nachschlagen von Virtual-Host-Informationen in einem LDAP-Verzeichnis |
| mod_cguard | Apache | Einschränkung der Anzahl offener Verbindungen |
| mod_cidr_lookup | Apache | Setzen von HTTP-Headern und Umgebungsvariablen aufgrund von CIDR-Netzmasken |
| mod_clamav | GPL | Virenscanner |
| mod_concat | Apache | Zusammenfügen mehrerer Dateien in einer URL-Anfrage |
| mod_countm | BSD | User-Counter (erzeugt JPEG oder PNG) |
| mod_cplusplus | Apache | C++-Wrapper für Apache |
| mod_cppserv | LGPL | Servlet- und JSP-ähnliche Webanwendungen in C++ |
| mod_crowd | Public Domain | webbasierte Single-Sign-on-Lösung |
| mod_csv | GPL | Ausgabe von CSV-Daten für die Tabellenkalkulation |
| mod_dav_acl | LGPL | Zugriffskontrolllisten (ACLs) für WebDAV-Verzeichnisse |
| mod_dav_svn | Apache | WebDAV-Zugriff auf Subversion-Repositorys (Bestandteil von Subversion) |
| mod_debug | GPL | Liefert einen speziellen HTTP-Header mit Informationen über die Anfrage. |
| mod_diagnostics | Apache | Debugger für Filter |
| mod_domaintree | Apache | Massen-VHosting, bei dem Hostnamen einem Verzeichnisbaum zugeordnet werden |
| mod_dsp | Apache | Webentwicklung mit Oracle PL/SQL und Oracle-Authentifizierung |
| mod_fcgid | GPL | FastCGI-Unterstützung |
| mod_footer | BSD | Fußzeilen mit Variablen hinzufügen |
| mod_ftpd | Apache | FTP-Server |

**Tabelle 17.1** Drittanbieter-Module (alle auf der DVD zu finden) (Forts.)

| Modul | Lizenz | Beschreibung |
|---|---|---|
| mod_globule | BSD | Replikation von Dokumenten eines Webservers; Clients werden automatisch zu einem der Slaves umgeleitet |
| mod_gridsite | Modified BSD | Grid-Authentifizierung |
| mod_highlight | Apache | Syntax-Highlighting als Apache-Ausgabefilter |
| mod_i18n | Apache | Internationalisierung per gettext als Ausgabefilter |
| mod_idcheck | GPL | Cookie-basierte Single-Sign-on-Lösung |
| mod_ifier | GPL mit Apache-Ausnahme | Filterung und Zurückweisung von Client-Anfragen |
| mod_injection | BSD | nachträgliches Einfügen von Text in HTML-Dokumente |
| mod_ipenv | GPL | Setzen von Umgebungsvariablen anhand von Client-IP-Adressen und -Hostnamen |
| mod_ip_count | Apache | Anzahl der Anfragen pro IP beschränken |
| mod_itp | MIT-Lizenz | ITP-Protokoll (nicht Internet Time Protocol, sondern Transfer-Protokoll des PDA Simputer) |
| mod_line_edit | GPL | schneller Vielzweck-Textfilter |
| mod_log50x | Apache | Schreibt 5xx-Fehler in ErrorLog. |
| mod_log_access | BSD | anpassbare Log-Dateien mit automatischer Log-Rotation |
| mod_log_firstbyte | Open Source | Logging der Zeit zwischen der Anfrage und dem ersten ausgelieferten Byte |
| mod_log_mysql | Apache | Logging in MySQL-Datenbanken |
| mod_log_spread2 | Apache | Logging mithilfe des Reliable Message Frameworks spread |
| mod_lua | Apache | dynamische Websites mit der Skriptsprache Lua (seit 2.4 in der offiziellen Distribution; siehe Kapitel 15, »Technologien zur Webprogrammierung«) |
| mod_memcached_cache | Apache | Memcache-Speicherung als Provider für mod_cache (guter Ersatz für das abgeschaffte mod_mem_cache) |
| mod_mono | Apache | ASP.NET-Unterstützung mit Mono |
| mod_murka | Apache | XML-/XSLT-basierte Webpublishing-Umgebung |
| mod_oai | CC | Unterstützung für das Open-Archives-Initiative-Protokoll (OAI) |

**Tabelle 17.1** Drittanbieter-Module (alle auf der DVD zu finden) (Forts.)

| Modul | Lizenz | Beschreibung |
|---|---|---|
| mod_pgheader | GPL | Body-Tags durch Kopf-/Fußzeilen ersetzen |
| mod_mysql_pool | GPL | MySQL-Connection-Pooling |
| mod_pg_pool | GPL | PostgreSQL-Connection-Pooling |
| mod_prolog | GPL | eingebetteter Prolog-Interpreter |
| mod_proxy_fcgi | Apache | Proxy-Modul für das Fast-CGI-Protokoll (seit 2.4 in der offiziellen Distribution; siehe Kapitel 13, »Proxy- und Cache-Funktionen«, und 14, »CGI«) |
| mod_proxy_html | GPL | Korrektur von HTML-Hyperlinks für den Proxy-Betrieb (seit 2.4 in der offiziellen Distribution; siehe Kapitel 13, »Proxy- und Cache-Funktionen«) |
| mod_proxy_xml | GPL | Korrektur von Links in diversen XML-Dokumenttypen für den Proxy-Betrieb |
| mod_publisher | keine | Allzweck-Verarbeitungsfilter für Auszeichnungssprachen |
| mod_python | Apache | Python-Interpreter |
| mod_qos | GPL | Festlegen verschiedener Prioritäten für unterschiedliche Anfragen (Quality of Service) |
| mod_schema | LGPL | W3C/XML-Schema-Validator |
| mod_scheme | Apache | Scheme-Interpreter |
| mod_sesehe | Apache | Ändern oder Entfernen des HTTP-Headers `Server` (SEcure SErver HEader) |
| mod_slotlimit | GPL | Ressourcenmanagement mit Dynamic-Slot-Allocation-Algorithmus |
| mod_spamhaus | GPL | Webformular-Spam, URL-Injection und DDOS-Attacken blockieren |
| mod_spin | GPL | Template-Engine |
| mod_sqil | Apache | Mapper, der Daten aus relationalen Datenbanken als XML-Strukturen abbildet |
| mod_sql | GPL | XML-Namensraum mit SQL-Abfragen |
| mod_ssi_func | Apache | Erweitert SSI um mathematische Funktionen, Zufallsgenerator und Variablen-Wertzuweisung. |
| mod_ssl_error | Apache | Fehler bei der X.509-Zertifikats-Validierung abfangen |
| mod_suexec_helper | GPL | Stellt die Direktiven `UserId` und `GroupId` bereit, deren Werte als Argumente an `Suexec` weitergegeben werden. |

**Tabelle 17.1** Drittanbieter-Module (alle auf der DVD zu finden) (Forts.)

| Modul | Lizenz | Beschreibung |
|---|---|---|
| mod_suid2 | Apache | alternatives Suexec-Modul |
| mod_swf2html | BSD | Textsuche in SWF-Dateien per Adobe Flash Search Engine SDK |
| mod_tee | GPL | Verdopplung der Ausgabe |
| mod_telemetry | GPL | Logging aller URLs, auf die zugegriffen wird, sowie der Zeit, die die Beantwortung jeder Anfrage benötigt |
| mod_tidy | frei | Ausgabefilter zur HTML-Validierung |
| mod_traf_thief | Apache | Traffic prozentweise auf URL umleiten |
| mod_txt | GPL | Textdateien in HTML mit Kopf-/Fußzeile umwandeln |
| mod_upload | GPL | Eingabefilter für Datei-Uploads (multipart/form-data) |
| mod_uploader | zlib/libpng | ein weiteres Upload-Modul |
| mod_validator | GPL | Markup-Validierungsmodul (HTML, XML, SGML) |
| mod_vd | giftware | schnelle mod_vhost_alias-Alternative |
| mod_vhost_ldap | Apache | virtuelle Hosts über LDAP |
| mod_vhost_mysql2 | Apache | dynamisch konfigurierte virtuelle Hosts über MySQL-Datenbank |
| mod_vhs | Apache | Daten für virtuelle Hosts aus LDAP, MySQL, PAM usw. |
| mod_websh | Artistic License | Tcl-Erweiterung für Rapid Web Application Development |
| mod_wiki | GPL | ein Wiki als Apache-Modul |
| mod_wxjs | LGPL | JavaScript/wxWidgets als serverseitige Sprache |
| mod_xcap | LGPL | Anwendungs-Konfigurationsdaten aus XML-Dateien |
| mod_xhtml | GPL | Namensraummodul für XHTML |
| mod_xmlns | GPL | Unterstützung für XML-Namensräume |
| mod_xslt | GPL | XSLT-Prozessor |
| modXLDAPAuth | Apache | erweitertes X.509-/LDAP-Authentifizierungs-modul |
| Vhost Limit Module | Open Source | Beschränkung der Clients pro virtuellem Host |
| xmlns_esi | GPL | Parser für Edge Side Includes |

**Tabelle 17.1** Drittanbieter-Module (alle auf der DVD zu finden) (Forts.)

## 17.3 Programmierung eigener Module

Die flexibelste aller Erweiterungsmöglichkeiten besteht darin, selbst Module für den Apache-Webserver zu programmieren. Die Programmiersprache ist ANSI C, unterstützt durch die mächtige APR-Bibliothek und die Modul-API. Dieser Abschnitt kann und will keine umfangreiche Referenz der Apache Portable Runtime und der Modul-API sein, bietet aber ein Tutorial, das die wesentlichen Schritte erklärt und aus dem zwei funktionierende Module hervorgehen. Bevor es losgeht, wird noch kurz das mit Apache mitgelieferte Modul `mod_example` vorgestellt, das die Funktionsweise der Apache-Modul-API demonstriert.

### 17.3.1 mod_example – Erforschen der Modul-API

Sie sollten `mod_example` nicht auf einem Produktivsystem verwenden – es liefert ohnehin keine Dienstleistung, die den Usern Ihrer Website Vorteile bringen könnte. Es wurde vor allem geschrieben, damit Entwickler seinen Quellcode studieren können – es handelt sich um eine Art Live-Referenz zur Modulprogrammierung.

Um `mod_example` in einem Kontext zu aktivieren, müssen Sie seinen Handler setzen. Beispiel:

```
<Location /example>
  SetHandler example-handler
</Location>
```

Eine URL, für die dieser Handler eingestellt wurde, beantwortet Client-Anfragen mit einer Beispielausgabe im HTML-Format; diese gibt Aufschluss über die Aufruf- und Verarbeitungsreihenfolge der einzelnen Bestandteile des Moduls selbst.

#### Example

Aktiviert die `mod_example`-Funktionalität.

| | |
|---|---|
| **Modul** | `mod_example` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`All`) |
| **Syntax** | `Example` |
| **Standardwert** | nicht gesetzt |

Diese Direktive besitzt keine Argumente. Der gesamte Aufruf sieht folgendermaßen aus:

```
Example
```

Wenn sie in einem Container gesetzt wird, in dem der `example-handler` aktiviert wurde, wird dies unter der Überschrift `Example directive declared here: YES/NO` angezeigt.

### 17.3.2 Arbeitsweise von Modulen

Um zu verstehen, wie Module arbeiten, sollten Sie wissen, in welcher Reihenfolge Apache 2 Anfragen beantwortet. Im Wesentlichen lassen sich die folgenden vier Phasen unterscheiden:

- **Parsing der Anfrage**
  In dieser ersten Phase wird der URL-String der Anforderung einer Reihe von Tests und Umwandlungsprozessen unterzogen. Zu guter Letzt muss die Entsprechung der angeforderten URL im lokalen Dateisystem herauskommen. Auch Eingabefilter arbeiten in dieser Phase.
- **Sicherheitsphase**
  Die zweite Phase der Anfrageverarbeitung überprüft die Authentifizierung und/oder die Berechtigung des Clients anhand seiner Adresse, je nachdem, ob `Satisfy All` (Standard) oder `Satisfy Any` eingestellt ist.
- **Vorbereitung**
  An dieser Stelle wird vor allem der MIME-Type der auszuliefernden Ressource bestimmt (`type_checker`); außerdem besteht die sogenannte *Fixup-Möglichkeit* zur Erledigung von Aufgaben, die zuvor noch nicht durchgeführt werden konnten.
- **Handler**
  In der letzten Phase wird die eigentliche Ausgabe generiert. Die Antwort kann noch immer beliebig modifiziert werden, bevor sie an den Client versendet wird; in dieser Phase arbeiten nämlich auch die Ausgabefilter. Die Vorbereitung der endgültigen HTTP-Antwort erledigen die Handler (siehe Kapitel 7, »Header und MIME-Einstellungen«).

Die ersten drei Phasen werden von der Funktion `ap_process_request_internal()` gesteuert. Deshalb können Core-Funktionen hier Unterabfragen erzeugen, die ihrerseits wieder in der ersten Ausführungsphase beginnen. Die letzte Phase wird dagegen von `ap_invoke_handler()` ausgeführt.

Damit die Funktionalität von Modulen überhaupt aktiviert werden kann, stellt Apache 2 an zahlreichen Stellen der Verarbeitung sogenannte *Hooks* bereit – Einstiegspunkte für Funktionen, die sich an bestimmten Stellen in den Verarbeitungsprozess einklinken. Damit Ihre Module in einer bestimmten Phase aktiv werden, können Sie die jeweiligen Hooks darin registrieren. Die wichtigsten Hooks sind folgende:

- `pre_config`: Initialisierungsaufgaben, die noch vor der Verarbeitung der Konfigurationsdateien erfolgen
- `check_config` (seit 2.3): Dieser neue Hook greift während der Verarbeitung der Konfigurationsdateien. Damit können Module gegenseitige Abhängigkeiten in ihrer Konfiguration überprüfen und eventuelle Probleme auf die Konsole protokollieren.
- `post_config`: Initialisierungsarbeiten, die erst nach der Konfigurationsverarbeitung stattfinden können
- `open_logs`: Öffnen der Log-Dateien
- `child_init`: Initialisierung der Worker-Prozesse beziehungsweise -Threads
- `translate_name`: Parsing der URL in einen internen URL-Pfad
- `map_to_storage`: Umwandlung der bereits geparsten URL in einen Dateinamen
- `header_parser`: Auswertung der Anfrage-Header
- `type_checker`: Ermittlung des Medientyps für den `Content-Type`-Header
- `fixups`: Erledigung von Last-Minute-Aufgaben
- `insert_filter`: Implementierung der Filter Chain
- `handler`: Erzeugung der endgültigen HTTP-Antwort

### 17.3.3 Die Modulentwicklung

Das in diesem Abschnitt entwickelte Modul heißt `mod_holiday` und besitzt die Aufgabe, alle Anfragen für einen URL-Pfad in einem bestimmten, frei wählbaren Monat an eine konfigurierbare externe URL weiterzuleiten.[1] Dazu definiert das Modul zwei Direktiven: `HolidayMonth` verlangt eine numerische Monatsangabe zwischen 1 und 12 und legt so den Monat fest; `HolidayURL` definiert die URL für die Weiterleitung. Beide gelten nur in Verzeichniskontexten. Hier ein Konfigurationsbeispiel für den Monat Juli:

```
<Location /company_info>
   # Im Juli nicht verfügbar wegen Urlaub
   HolidayMonth 7
   HolidayURL http://ausserbetrieb.mynet.de
</Location>
```

Der Code eines Moduls beginnt selbstverständlich mit einer Reihe von `#include`-Direktiven, damit die Funktionen der Apache Portable Runtime und des Webser-

1 Regelmäßige Leser der Zeitschrift iX fühlen sich vielleicht an das Modul `mod_happy_hour` aus meinem Artikel zur Modulprogrammierung in Ausgabe 2/2005 erinnert – es stimmt; dieses Modul funktioniert sehr ähnlich.

vers selbst zur Verfügung stehen. Eine typische Liste solcher Präprozessor-Direktiven sieht beispielsweise folgendermaßen aus:

```
#include "httpd.h"
#include "http_config.h"
#include "http_core.h"
#include "http_log.h"
#include "http_main.h"
#include "http_protocol.h"
#include "http_request.h"
#include "util_script.h"
#include "http_connection.h"

#include "apr_strings.h"
```

Welche Funktionen die jeweiligen Header-Dateien bereitstellen, erklären ihre Namen beinahe selbst. Ansonsten können Sie natürlich den Quellcode lesen, um es herauszufinden – es gibt keine bessere Informationsquelle über die Apache-Programmierung als den Quellcodebaum selbst.

Sie können auch die Entwicklerdokumentation im HTML-Format aus den Quellcodes generieren. Dazu benötigen Sie die Software Doxygen (unter *http://www.doxygen.org* zu finden); anschließend können Sie im obersten Verzeichnis der Sourcecode-Hierarchie nach dem `configure`-Durchlauf folgenden Befehl eingeben:

```
make dox
```

`mod_holiday` kommt mit erheblich weniger Includes aus:

```
#include <stdlib.h>
#include <time.h>

#include "httpd.h"
#include "http_config.h"
#include "http_protocol.h"
```

Die Header-Dateien `stdlib.h` und `time.h` aus der C-Standardbibliothek werden hier bequemerweise verwendet, um Datumsfunktionen beziehungsweise konventionelle C-Datenstrukturen zu verwenden. Für ein Modul, das produktiv eingesetzt werden soll, sollten Sie sich stattdessen mit entsprechenden APR-Funktionen vertraut machen.

Ein weiterer Block, den jedes Modul enthalten muss, ist die Deklaration der Modulstruktur. Sie steht für gewöhnlich am Ende seiner Hauptdatei. An dieser Stelle werden die Funktionen für die unterschiedlichen Konfigurationskontexte

registriert. Wenn Sie einen bestimmten Kontext nicht benötigen, können Sie ihn durch `NULL` ersetzen.

Hier sehen Sie die Struktur von `mod_holiday`:

```
module AP_MODULE_DECLARE_DATA holiday_module =
{
    STANDARD20_MODULE_STUFF,
    hd_dir_config,  /* Konfiguration pro Verzeichnis */
    NULL,           /* Merger der Verzeichniskonfigurationen */
    NULL,           /* Konfiguration pro Server */
    NULL,           /* Merger der Server-Konfigurationen */
    hd_commands,    /* Befehlstabelle (Direktiven) */
    register_hooks  /* Hooks (Aufruf der Modulfunktionen) */
};
```

Die symbolische Konstante `STANDARD20_MODULE_STUFF` muss bei jedem Apache-2-Modul an erster Stelle in der Liste stehen. Sie ist in *include/httpd_config.h* definiert und enthält alle Informationen, die jedes Modul braucht.

Die folgenden Zeilen enthalten jeweils den Namen einer aufzurufenden Funktion oder `NULL`, falls das entsprechende Feature nicht benötigt wird. Es handelt sich um folgende Funktionen:

- **Konfiguration pro Verzeichnis**
  Die hier aufgerufene Funktion erstellt die Grundkonfiguration des Moduls für Verzeichniskontexte.
- **Merger der Verzeichniskonfigurationen**
  Bei vielen Modulen wird die Konfiguration in einem Verzeichnis von übergeordneten Verzeichnissen beeinflusst; hier können Sie gegebenenfalls eine Funktion aufrufen, die die gewünschten »Verschachtelungsregeln« implementiert. Denken Sie z.B. an komplexe Direktiven wie `Options` (siehe Kapitel 6, »Grundkonfiguration«), bei denen Sie sich jeweils entscheiden können, ob Sie die Optionen des übergeordneten Kontextes modifizieren oder ganz neue Optionen setzen möchten.
- **Konfiguration pro Server**
  Definiert die Grundkonfiguration für einen Server oder virtuellen Host.
- **Merger der Server-Konfigurationen**
  Regelt entsprechend die Überlagerung der Hauptserver-Konfiguration durch virtuelle Hosts.
- **Befehlstabelle**
  In dieser Struktur werden die Konfigurationsdirektiven des Moduls deklariert.

- **Hooks**
  Die weiter oben bereits erwähnten Hooks bestimmen, an welchen Stellen der Anfrageverarbeitung das Modul in Aktion tritt. In der hier angegebenen Funktion werden wiederum die Funktionen aufgerufen, die die eigentliche Arbeit des Moduls erledigen.

`mod_holiday` kann nur in Verzeichniskontexten arbeiten. Aufgrund der Einfachheit der möglichen Konfigurationseinstellungen – genau ein Monat und genau eine URL – beeinflussen die Konfigurationen verschachtelter Kontexte einander nicht. Deshalb wird nur eine Funktion für die Verzeichniskonfiguration benötigt, aber kein Merger und auch keine entsprechenden Funktionen für den Server-Kontext. Zusätzlich definiert die Datenstruktur `hd_commands` die Konfigurationsdirektiven, und die Funktion `register_hooks()` sorgt dafür, dass die Arbeitsfunktionen des Moduls während der Anfrageverarbeitung ausgeführt werden.

Da sich der Code mitunter auf die Modulstruktur bezieht, empfiehlt sich direkt nach den Präprozessor-Direktiven deren Forward-Deklaration:

```
module AP_MODULE_DECLARE_DATA holiday_module;
```

Als Nächstes müssen Datenstrukturen für die Konfigurationseinstellungen definiert werden. Viele Module verwenden zwei getrennte Strukturen für die Verzeichnis- und die Server-Konfiguration; `mod_holiday` hat dagegen nur eine Verzeichniskonfiguration und benötigt daher auch nur eine Struktur. Diese sieht wie folgt aus:

```
typedef struct {
    int hd_month;        /* Monat  */
    const char *hd_url;  /* URL    */
} hd_conf;
```

Die beiden Variablen innerhalb der Struktur speichern den Monat beziehungsweise die URL für die Weiterleitung.

An dieser Stelle folgen die in der Modulstruktur registrierten Funktionen für die Grundkonfiguration. Das vorliegende Modul benötigt nur eine Funktion, nämlich für die Konfiguration pro Verzeichnis. Sie setzt den Monat auf 0 und die URL auf den leeren String `""`, da standardmäßig keine Urlaubsweiterleitung erfolgen soll:

```
static void *hd_dir_config (apr_pool_t *p, char *d)
{
    hd_conf *a = (hd_conf *) apr_pcalloc (p, sizeof (hd_conf));
    a->hd_month = 0;
    a->hd_url = "";
}
```

Die APR-Funktion `apr_pcalloc()` reserviert Speicher für die jeweiligen Konfigurationsdaten. Der entsprechende Speicher-Pool ist ein Zeigerargument der Funktion, damit die aufrufende Umgebung Zugriff auf die erzeugten Konfigurationsdaten erhält. APR-Pools werden automatisch im Speicher gehalten und erst zerstört, wenn sie nicht mehr benötigt werden.

Die Konfigurationsdirektiven werden durch die folgende Wertzuweisung definiert:

```
static const command_rec hd_commands[] =
{
    AP_INIT_TAKE1 ("HolidayMonth", set_hd_month, NULL,
                   OR_FILEINFO, "Holiday Month"),
    AP_INIT_TAKE1 ("HolidayURL", set_hd_url, NULL, OR_FILEINFO,
                   "Set URL to be redirected to"),
    { NULL }
};
```

Der Bezeichner vor der jeweiligen Klammer bestimmt den Direktiventyp, das heißt, er zeigt, wie viele Argumente die jeweilige Direktive erwartet. `AP_INIT_TAKE1` steht für eine Direktive, die genau einen Parameter benötigt.

In Tabelle 17.2 sehen Sie eine Übersicht aller zulässigen Varianten. In den Klammern hinter dem Direktiventyp stehen folgende Informationen:

- der Name der Direktive
- die Funktion, die beim Auftreten dieser Direktive aufgerufen wird
- eventuelles Zusatzargument (zur Unterscheidung, falls dieselbe Funktion für mehrere Direktiven verwendet wird)
- Override-Konstante – Bestimmt, welcher `AllowOverride`-Wert gesetzt sein muss, damit die Direktive in *.htaccess*-Dateien stehen darf. Die Namen entsprechen in etwa den möglichen Werten von `AllowOverride` (siehe Kapitel 6, »Grundkonfiguration«) mit vorangestelltem `OR` und komplett in Großbuchstaben: `OR_NONE` (numerischer Wert 0), `OR_LIMIT` (1), `OR_OPTIONS` (2), `OR_FILEINFO` (4), `OR_AUTHCFG` (8), `OR_INDEXES` (16). `OR_ALL` kombiniert alle diese Werte durch ein bitweises Oder (daher das Präfix `OR_`) mit dem Ergebnis 31.
- Kurzbeschreibung (wird unter anderem durch `mod_info` veröffentlicht)

| Direktiventyp | Bedeutung |
|---|---|
| `AP_INIT_NO_ARGS` | keine Argumente |
| `AP_INIT_RAW_ARGS` | Argumente als unformatierter Gesamt-String (Modul muss die Daten selbst parsen) |

**Tabelle 17.2** Die möglichen Direktiventypen und ihre Argumente

| Direktiventyp | Bedeutung |
|---|---|
| AP_INIT_TAKE_ARGV | ebenfalls unformatiert |
| AP_INIT_TAKE1 | genau ein Argument |
| AP_INIT_ITERATE | beliebig viele gleichartige Argumente |
| AP_INIT_TAKE_2 | genau zwei Argumente |
| AP_INIT_TAKE_12 | ein oder zwei Argumente |
| AP_INIT_ITERATE_2 | ein festes Argument und Iteration ab dem zweiten |
| AP_INIT_TAKE_13 | ein oder drei Argumente |
| AP_INIT_TAKE_23 | zwei oder drei Argumente |
| AP_INIT_TAKE_123 | ein bis drei Argumente |
| AP_INIT_TAKE_3 | genau drei Argumente |
| AP_INIT_FLAG | On oder Off – wird für die Funktion automatisch in 1 oder 0 umgewandelt |

**Tabelle 17.2** Die möglichen Direktiventypen und ihre Argumente (Forts.)

Hier als Beispiel die Deklaration der Direktive `HolidayMonth`:

```
AP_INIT_TAKE1 ("HolidayMonth", set_hd_month, NULL,
               OR_FILEINFO, "Holiday Month"),
```

Wenn diese Direktive in einer Konfigurationsdatei steht, wird also `set_hd_month` aufgerufen. Da diese Funktion nicht durch mehrere Direktiven aufgerufen wird, ist der nächste Wert `NULL`. Die symbolische Konstante `OR_FILEINFO` bestimmt, dass `AllowOverride` im entsprechenden Kontext den Wert `FileInfo` besitzen muss, um diese Direktive in einer *.htaccess*-Datei verwenden zu können. Das letzte Argument ist wie erwähnt die Beschreibung.

Die Funktion `set_hd_month()`, die für die Verarbeitung dieser Direktive zuständig ist, sieht folgendermaßen aus:

```
static const char *set_hd_month
     (cmd_parms *cmd, void *mconfig, const char *w)
{
    hd_conf *conf = (hd_conf *) mconfig;

    conf->hd_month = atoi (w);
    return NULL;
}
```

Der erste Parameter einer solchen Funktion zur Direktivenverarbeitung ist das dritte Argument des `AP_INIT_TAKE1()`-Aufrufs – der selbst definierte String zur Direktivenunterscheidung; hier hat er aus bereits erläuterten Gründen den Wert `NULL`. Als zweiter Parameter wird ein Zeiger auf die bisherige Konfiguration des

aktuellen Kontextes verwendet – diese Struktur wird durch die Direktive verändert. Da jedes Modul seine eigenen Konfigurationsdatenstrukturen besitzt, hat der Zeiger hier zunächst den Datentyp `void *`, also »Zeiger auf beliebige Daten«. Das dritte Argument schließlich ist der direkt aus der Konfigurationsdatei gelesene Wert für die Direktive. Bei anderen `AP_INIT_*`-Varianten müssen hier natürlich noch mehr Parameter folgen.

Der erste Schritt besteht darin, die übergebenen Konfigurationsdaten per Type-Casting in das richtige Format umzuwandeln, in diesem Fall in die weiter oben definierte Datenstruktur `hd_conf`.

Die eigentliche Aufgabe der Funktion folgt in der nächsten Zeile: `hd_month` wird auf den Wert aus der Konfigurationsdatei gesetzt. Da Direktivenargumente immer Strings sind, wird er mithilfe der Standardfunktion `atoi()` in einen Integer-Wert konvertiert. In der Praxis sollten Sie an einer solchen Stelle Datentyp und Wertebereich überprüfen; falls sie falsch sind, können Sie anstelle von `NULL` einen Fehlermeldungs-String zurückgeben.

Die Funktion zur Verarbeitung der anderen Direktive funktioniert im Prinzip genauso, außer dass kein Type-Casting für das Argument erforderlich ist, weil die URL ein String ist:

```
static const char *set_hd_url
     (cmd_parms *cmd, void *mconfig, const char *w)
{
    hd_conf *conf = (hd_conf *) mconfig;

    conf->hd_url = w;
    return NULL;
}
```

Die in der Modulstruktur angegebene Funktion `register_hooks()` bestimmt, durch welche Hooks das Modul aktiviert wird und welche Funktionen in diesen Fällen aufgerufen werden:

```
static void register_hooks(apr_pool_t *p)
{
    ap_hook_fixups  (hd_fixups, NULL, NULL, APR_HOOK_MIDDLE);
}
```

Der Bezeichner bestimmt den jeweiligen Hook; es handelt sich um die weiter oben vorgestellten Namen mit vorangestelltem `ap__hook_`. Beachten Sie, dass die Liste nur eine Auswahl der möglichen Hooks enthält. Wenn Ihr Modul in einer ungewöhnlichen Phase arbeiten soll, können Sie sogar eigene Hooks definieren – dies würde hier allerdings zu weit führen. `mod_holiday` verwendet den Hook

`fixups` – das ist spät genug, sodass bereits der URL-Pfad ermittelt wurde, und noch früh genug, um die HTTP-Antwort in eine Weiterleitung umzuwandeln.

Das erste Argument ist die aufzurufende Funktion; das letzte bestimmt, wann das Argument an die Reihe kommt, wenn der entsprechende Hook mehrere Aufgaben erfüllen muss: Wählen Sie `AP_HOOK_FIRST`, wenn der Hook Ihre Funktion als erste aufrufen soll (genauer gesagt: im Laufe derjenigen Funktionen, für die `AP_HOOK_FIRST` angegeben wurde). Wenn der Hook Ihr Modul als eines der letzten aktivieren soll, können Sie `AP_HOOK_LAST` verwenden. `AP_HOOK_MIDDLE` schließlich steht für »irgendwann« – wählen Sie diese Option, wenn es keine besondere Rolle spielt, wann der Hook die entsprechende Aufgabe Ihres Moduls verarbeitet.

Die Arbeitsfunktion von `mod_holiday` heißt `hd_fixups()`. Sie wird durch folgende Deklaration eingeleitet:

```
static int hd_fixups (request_rec *r)
```

Das Argument ist ein Zeiger auf eine Datenstruktur vom Typ `request_rec`; diese enthält sämtliche Aspekte der HTTP-Anfrage. Im Einzelnen sind folgende Bestandteile am interessantesten:

- `server_rec *server` – ein Zeiger auf die Datenstruktur mit allen relevanten Informationen über den aktuellen Server oder virtuellen Host
- `char *protocol` – Protokoll und Version der Anfrage; meist `"HTTP/1.1"`
- `const char *hostname` – der Hostname aus dem HTTP-Header `Host`
- `int status` – der bisherige HTTP-Status (siehe Kapitel 2, »Funktionsweise von Webservern«). Viele Module, z.B. das vorliegende, ändern den Status noch einmal.
- `int method_number` – Beschreibt die HTTP-Anfrage-Methode; die API definiert symbolische Konstanten wie `M_GET` oder `M_POST`, um diesen Wert einfacher zu überprüfen.
- `apr_table_t *headers_in` – eine Tabelle mit den HTTP-Anfrage-Headern. Sie können sie mittels `apr_table_get(r->headers_in, "<Headername>")` lesen; der Schreibzugriff erfolgt durch `apr_table_setn(r->headers_in, "<Headername>", "<Wert>")`.
- `apr_table_t *headers_out` – die entsprechende Tabelle mit den Antwort-Headern; der Zugriff erfolgt genau wie bei `headers_in`.
- `const char *content_type` – ein String mit dem MIME-Type der Antwort, z.B. `"text/html"`
- `const char *unparsed_uri` – String-Entsprechung des ursprünglich angeforderten URL-Pfads

Die Funktion deklariert als Erstes einige Hilfsvariablen:

```
hd_conf *conf;      /* Konfiguration des aktuellen Kontextes */
time_t now;         /* aktuelles Datum/Uhrzeit               */
struct tm *tmp;      /* Struktur für Datumsbestandteile      */
```

Nun wird die aktuelle Verzeichniskonfiguration ausgelesen und in der Variablen `conf` gespeichert. Dies erledigt die Funktion `ap_get_config()`. Die Argumente sind die gewünschte Konfigurationsstruktur (hier das Feld `per_dir_config` der Anfrage-Struktur) und die Adresse des Moduls (`&holiday_module`).

Nun wird die aktuelle Systemzeit ermittelt; die Funktion `localtime()` wandelt sie aus dem Standardformat (Sekunden seit `EPOCH`) in eine Datenstruktur mit ihren einzelnen Bestandteilen (Tag, Monat, Jahr, Stunde, Minute, Sekunde usw.) um:

```
now = time (NULL);
tmp = localtime (&now);
```

Es folgt eine `if`-Fallentscheidung zur Überprüfung, ob das Modul überhaupt aktiv werden muss. Dies ist nur der Fall, wenn der aktuelle Monat mit dem `HolidayMonth` übereinstimmt und die `HolidayURL` gesetzt ist:

```
if ((tmp->tm_mon + 1) != conf->hd_month ||
     !strcmp (conf->hd_url, "")) {
    return DECLINED;
}
```

Der Rückgabewert `DECLINED` bedeutet, dass das Modul nicht zuständig ist und keine Änderungen vorgenommen hat.

Die merkwürdige Addition `tmp->tm_mon + 1` ist notwendig, weil die Datenstruktur `struct tm` die Monate von 0 bis 11 durchnummeriert; in der Direktive gilt dagegen die benutzerfreundliche normale Zählung von 1 bis 12.

Falls die Funktion nach dieser Fallentscheidung noch nicht verlassen wurde, steht nun fest, dass das Modul sich an die Arbeit machen soll: Der Antwort-Header `Location` wird auf die `HolidayURL` gesetzt, und der Rückgabewert ist `HTTP_MOVED_TEMPORARILY` (307):

```
apr_table_setn (r->headers_out, "Location", conf->hd_url);
return HTTP_MOVED_TEMPORARILY;
```

Die Apache-API definiert für jeden Statuscode eine symbolische Konstante. Viele Module geben `HTTP_OK` (200) zurück; ein anderes bekanntes Beispiel ist `HTTP_NOT_FOUND` (404).

Listing 17.1 zeigt den vollständigen Quellcode des Beispiels mit ausführlichen Kommentaren. Einige Stellen sind mit TODO gekennzeichnet; wenn Sie Lust haben, können Sie einen verregneten Sonntagnachmittag damit verbringen, die entsprechenden Erweiterungen einzufügen.

```
#include <stdlib.h>
#include <time.h>

#include "httpd.h"
#include "http_config.h"
#include "http_protocol.h"

/* Forward-Deklaration der Moduldatenstruktur     */
module AP_MODULE_DECLARE_DATA holiday_module;

/* Konfigurationsdatenstruktur                     */
typedef struct {
    int hd_month;             /* Monat             */
    const char *hd_url;       /* URL               */
} hd_conf;

/* Grundkonfiguration für Verzeichnisse */
static void *hd_dir_config (apr_pool_t *p, char *d)
{
    hd_conf *a = (hd_conf *) apr_pcalloc (p, sizeof (hd_conf));
    a->hd_month = 0;
    a->hd_url = "";
}

/* Einstellen des Monats;
   durch Direktive HolidayMonth aufgerufen         */
static const char *set_hd_month (cmd_parms *cmd, void *mconfig,
const char *w)
{
    /* Zeiger auf bisherige Konfiguration des
       aktuellen Kontextes                          */
    hd_conf *conf = (hd_conf *) mconfig;

    /* String in Integer umwandeln und speichern   */
    conf->hd_month = atoi (w);
    return NULL;
    /* TODO: Datentyp und Wertebereich überprüfen;
       bei Fehler nicht NULL, sondern Meldung
       zurückgeben                                  */
}
```

```
/* Einstellen der Weiterleitungs-URL;
   durch Direktive HolidayURL aufgerufen */
static const char *set_hd_url (cmd_parms *cmd, void *mconfig, const
char *w)
{
    hd_conf *conf = (hd_conf *) mconfig;

    conf->hd_url = w;
    return NULL;
    /* TODO: Formale URL-Gültigkeit überprüfen;
       bei Fehler nicht NULL, sondern Meldung
       zurückgeben                                  */
}

/* Hier arbeitet das Modul:
   Falls der Monat gleich dem HolidayMonth ist,
   erfolgt die Weiterleitung zur HolidayURL        */
static int hd_fixups (request_rec *r)
{
    /* TODO: time.h- und stdlib.h-Funktionen und
             -Datenstrukturen durch entsprechende
             APR-Funktionalität ersetzen.
             Siehe: http://apr.apache.org          */

    /* Hilfsvariablen   */
    hd_conf *conf;   /* Konfiguration des aktuellen Kontextes */
    time_t now;      /* aktuelles Datum/Uhrzeit               */
    struct tm *tmp;  /* Struktur für Datumsbestandteile       */

    /* Die aktuelle Konfiguration auslesen         */
    conf =
       ap_get_module_config(r->per_dir_config, &holiday_module);

    now = time (NULL);
    tmp = localtime (&now);

    /* Falscher Monat oder leere URL?
       Dann sind wir nicht zuständig               */
    if ((tmp->tm_mon + 1) != conf->hd_month
          || !strcmp (conf->hd_url, "")) {
        return DECLINED;
    }

    /* Location-Header und Statuscode 307
       für die Weiterleitung setzen                */
```

```
    apr_table_setn (r->headers_out, "Location", conf->hd_url);
    return HTTP_MOVED_TEMPORARILY;
}

/* Vereinbarung der Konfigurationsdirektiven       */
static const command_rec hd_commands[] =
{
    AP_INIT_TAKE1 ("HolidayMonth", set_hd_month,
                   NULL, OR_FILEINFO, "Holiday Month"),
    AP_INIT_TAKE1 ("HolidayURL", set_hd_url, NULL, OR_FILEINFO,
                   "Set URL to be redirected to"),
    { NULL }
};

/* Hooks: Wann wird die Kernfunktion
   des Moduls aufgerufen?
   Hier fixups für Last-Minute-Umleitung           */
static void register_hooks(apr_pool_t *p)
{
    ap_hook_fixups  (hd_fixups, NULL, NULL, APR_HOOK_MIDDLE);
}
/* Datenstruktur des Moduls */
module AP_MODULE_DECLARE_DATA holiday_module =
{
    STANDARD20_MODULE_STUFF,
    hd_dir_config,  /* Konfiguration pro Verzeichnis         */
    NULL,           /* Merger der Verzeichniskonfigurationen */
    NULL,           /* Konfiguration pro Server              */
    NULL,           /* Merger der Server-Konfigurationen      */
    hd_commands,    /* Befehlstabelle (Direktiven)           */
    register_hooks  /* Hooks (Aufruf der Modulfunktionen)    */
};
```

**Listing 17.1** Das Apache-Modul »mod_holiday«

Nachdem das Modul fertig ist, kann es kompiliert werden. Dafür ist das bereits in Kapitel 4, »Apache kompilieren und installieren«, angesprochene Apache-Tool `apxs` zuständig. Der Aufruf sieht für `mod_holiday` so aus:

```
# apxs -cia mod_holiday.c
```

Die Aufgaben der drei Kommandozeilenparameter `-c`, `-i` und `-a` sind das Kompilieren, Installieren beziehungsweise Aktivieren des Moduls. Letzteres fügt die folgende Zeile in die *httpd.conf* ein:

```
LoadModule holiday_module modules/mod_holiday.so
```

Anschließend können Sie einen Beispieleintrag mit den Direktiven `HolidayMonth` und `HolidayURL` hinzufügen und Apache neu starten. Falls der aktuelle Monat Ihrer Angabe entspricht, erfolgt die gewünschte Weiterleitung, andernfalls wird der normale Inhalt der angeforderten URL serviert.

### 17.3.4 mod_daytime – ein Beispiel zur Multiprotokoll-Unterstützung

In diesem Abschnitt wird ein weiteres Modul vorgestellt, das eine andere Aufgabe löst und etwas komplizierter aufgebaut ist: `mod_daytime` implementiert einen Server-Dienst für das Daytime-Protokoll gemäß RFC 867: Jede TCP-Anfrage wird mit Datum und Uhrzeit beantwortet, danach beendet der Server die Verbindung. Das Grundgerüst des Moduls macht einige Anleihen bei dem ähnlich gestrickten, weiter oben vorgestellten Standardmodul `mod_echo`.

Das RFC macht keine Vorschrift über die Formatierung der Ausgabe; dieses Modul verwendet der Einfachheit halber `ctime(3)`. Auch hier sollte für eine robustere und Apache-konformere Praxisversion anstelle dieser `time.h`-Funktion die entsprechende APR-Funktionalität genutzt werden.

`mod_daytime` stellt nur eine einzige Konfigurationsdirektive zur Verfügung: `ProtocolDaytime` ist nur im Server- oder `<VirtualHost>`-Kontext gültig und kann die Werte `On` oder `Off` annehmen. Bei `On` wird jede Anfrage im entsprechenden Kontext durch dieses Protokoll beantwortet – selbst gültige HTTP-Anfragen. Daher empfiehlt es sich, einen virtuellen Host für den Daytime-Standardport 13 einzurichten und Anfragen für diesen Port durch dieses Modul verarbeiten zu lassen:

```
# Lauschen an Port 13
Listen 13

<VirtualHost *:13>
  ProtocolDaytime On
</VirtualHost>
```

Nach einem Neustart von Apache können Sie eine Telnet-Verbindung zu Port 13 herstellen und irgendetwas eingeben; als Antwort erhalten Sie die aktuelle Systemzeit:

```
$ telnet localhost 13
Thu Nov 10 21:46:31 2011
Connection closed by foreign host.
```

Die Modulstruktur unterscheidet sich von `mod_holiday`, weil es nur eine Server-Konfiguration gibt:

```
module AP_MODULE_DECLARE_DATA daytime_module = {
    STANDARD20_MODULE_STUFF,
    NULL,                           /* Verzeichnis-Konfig.    */
    NULL,                           /* Verzeichnis-Merger     */
    create_daytime_server_config,   /* Server-Konfiguration   */
    NULL,                           /* Server-Merger          */
    daytime_cmds,                   /* Befehlstabelle         */
    register_hooks                  /* Hooks registrieren     */
};
```

Der passende Hook für ein solches Modul, das ein anderes Protokoll implementiert, liegt zeitlich sehr früh in der Konfiguration. `mod_daytime` verwendet `ap_hook_process_connection()`, die Phase des Verbindungsaufbaus:

```
static void register_hooks(apr_pool_t *p)
{
    ap_hook_process_connection(process_daytime_connection,
                               NULL, NULL, APR_HOOK_MIDDLE);
}
```

Die Direktive `ProtocolDaytime` besitzt den Typ `AP_INIT_FLAG`, kann also die Werte `On` oder `Off` annehmen:

```
static const command_rec daytime_cmds[] =
{
    AP_INIT_FLAG("ProtocolDaytime", daytime_on, NULL, RSRC_CONF,
                 "Run a daytime server on this host"),
    { NULL }
};
```

Die eigentliche Arbeitsfunktion des Moduls ist etwas komplexer als bei `mod_holiday`, da nicht nur ein Header manipuliert wird (die gibt es bei diesem Protokoll und in diesem Verarbeitungsstadium auch noch gar nicht), sondern auszugebende Daten erzeugt werden. Apache verwaltet seine Ein- und Ausgabedaten in sogenannten *Buckets*, also Speicherblöcken, die wiederum in Brigades organisiert werden. `mod_daytime` erzeugt eine entsprechende Bucket Brigade für die Ausgabedaten und schreibt das `ctime()`-formatierte Datum hinein:

```
bb = apr_brigade_create(c->pool, c->bucket_alloc);

now = time (NULL);

ap_fprintf(c->output_filters, bb, "%s\r\n", ctime(&now));
ap_fflush(c->output_filters, bb);
```

`c` ist übrigens die Verbindungsdatenstruktur (Datentyp `conn_rec`), die einer Modulfunktion in diesem Verarbeitungsstadium als Argument übergeben wird.

Alle weiteren Informationen erhalten Sie aus den Kommentaren in Listing 17.2, das den vollständigen Sourcecode des Moduls enthält.

```
#include "ap_config.h"
#include "ap_mmn.h"
#include "httpd.h"
#include "http_config.h"
#include "http_connection.h"

#include "apr_buckets.h"
#include "util_filter.h"

#include <time.h>

module AP_MODULE_DECLARE_DATA daytime_module;

/* Konfigurationsdatenstruktur                    */
typedef struct {
    int dt_enabled;
} daytime_config;

/* Erstellen der Konfiguration pro Server          */
static void *create_daytime_server_config
     (apr_pool_t *p, server_rec *s)
{
    /* Pool erzeugen, Speicher reservieren         */
    daytime_config *conf = apr_pcalloc(p, sizeof *conf);

    /* Standardmäßig ist daytime abgeschaltet.      */
    conf->dt_enabled = 0;

    return conf;
}

/* Verarbeiten der Konfigurationsdirektive
   ProtocolDaytime                                 */
static const char *daytime_on
      (cmd_parms *cmd, void *dummy, int arg)
{
    /* Aktuelle Konfiguration auslesen             */
    daytime_config *conf = ap_get_module_config
                (cmd->server->module_config, &daytime_module);
    /* Den Wert speichern - AP_INIT_FLAG wandelt
       On|Off automatisch in 1|0 um.                */
    conf->dt_enabled = arg;
```

```
    return NULL;
}

/* Verarbeiten der daytime-Verbindung */
static int process_daytime_connection(conn_rec *c)
{
    /* Hilfsvariablen                                */
    /* Speicherstrukturen für Ausgabedaten           */
    apr_bucket_brigade *bb;
    apr_bucket *b;
    /* Aktuelle Systemzeit                           */
    time_t now;

    /* Aktuelle Konfiguration auslesen               */
    daytime_config *conf = ap_get_module_config
               (c->base_server->module_config, &daytime_module);

    /* Abbruch, falls daytime deaktiviert            */
    if (!conf->dt_enabled) {
        return DECLINED;
    }

    /* Ausgabedatenpool erzeugen                     */
    bb = apr_brigade_create(c->pool, c->bucket_alloc);

    /* Aktuelle Systemzeit ermitteln                 */
    now = time (NULL);

    /* Datum formatiert in die Ausgabe schreiben     */
    ap_fprintf(c->output_filters, bb, "%s\r\n", ctime(&now));
    /* Ausgabepuffer leeren; sofort senden           */
    ap_fflush(c->output_filters, bb);

    /* Anfrage war in Ordnung; HTTP_OK wäre hier
       allerdings fehl am Platz.                      */
    return OK;
}

/* Die einzige Konfigurationsdirektive               */
static const command_rec daytime_cmds[] =
{
    AP_INIT_FLAG("ProtocolDaytime", daytime_on, NULL, RSRC_CONF,
                 "Run a daytime server on this host"),
    { NULL }
};
```

```
/* Hooks registrieren                                */
static void register_hooks(apr_pool_t *p)
{
    ap_hook_process_connection
      (process_daytime_connection, NULL, NULL, APR_HOOK_MIDDLE);
}

/* Modulstruktur */
module AP_MODULE_DECLARE_DATA daytime_module = {
    STANDARD20_MODULE_STUFF,
    NULL,                          /* Verzeichnis-Konfig.    */
    NULL,                          /* Verzeichnis-Merger     */
    create_daytime_server_config,  /* Server-Konfiguration   */
    NULL,                          /* Server-Merger          */
    daytime_cmds,                  /* Befehlstabelle         */
    register_hooks                 /* Hooks registrieren     */
};
```

**Listing 17.2** »mod_daytime« – ein Beispiel für Multiprotokoll-Unterstützung

## 17.4 Zusammenfassung

In diesem Kapitel haben Sie einen ersten Einblick in einige interessante Erweiterungsmöglichkeiten für Apache erhalten. Der Betrieb als WebDAV-Server eröffnet Ihnen neue Möglichkeiten für die Online-Zusammenarbeit, insbesondere an Websites.

Neben den zahlreichen im Lieferumfang enthaltenen Modulen stehen auch noch unzählige Drittanbieter-Module zur Verfügung, die Apache um teils naheliegende, teils völlig ungeahnte neue Funktionen erweitern. Auf der beiliegenden DVD finden Sie eine Auswahl dieser Module zum Ausprobieren.

Zu guter Letzt können Sie sich Ihre Module natürlich auch selbst schreiben; die Grundlagen dafür wurden in diesem Kapitel erläutert. Auf der Website zum Buch finden Sie nach und nach erweiternde Tutorials mit neuen Modulen.

*»Sicherheit erreicht man nicht, indem man Zäune errichtet, sondern indem man Türen öffnet.«*
*– Urho Kekkonen*

# 18 Sicherheit

Dieses kurze Kapitel versammelt die wichtigsten Informationen, die Sie zur Absicherung Ihres Apache-Webservers benötigen. Damit ist der Schutz vor Crackerattacken und sonstigen unberechtigten Zugriffen gemeint. Die Beschränkung Ihrer Websites auf bestimmte User wird hier dagegen nicht behandelt; dieses Thema wurde bereits ausführlich in Kapitel 9, »Authentifizierung«, erläutert.

## 18.1 Sicherheit der Server-Umgebung

Sicherheitskonzepte, die nichts mit dem Betrieb von Apache selbst zu tun haben, würden den Rahmen dieses Buches sprengen. Selbstverständlich sollten Sie aber folgende zusätzliche Sicherheitsmaßnahmen in Erwägung ziehen:

- **Eine Firewall**
  Natürlich können Sie einen Webserver, dessen Aufgabe die Veröffentlichung von Informationen für den allgemeinen, anonymen Zugriff ist, nicht vollständig hinter einer Firewall verbergen. Vielmehr sollten Sie über eine intelligente Verteilung der betroffenen Rechner im Netzwerk nachdenken. Möglicherweise können Sie Proxy-Dienste verwenden, um eine Firewall zwischen dem eigentlichen Webserver und den Backend-Servern, etwa für Anwendungen oder Datenbanken, zu überbrücken.

  Übrigens sollten Sie neben einer netzwerkweiten Firewall auch auf dem Webserver-Host selbst eine Software einrichten, die Zugriffe auf bestimmte Ports unterbindet oder spezielle Pakete zurückweist. Wenn Sie Linux verwenden, ist die Firewall in Form von Netfilter (konfigurierbar über das Kommando `iptables`) bereits im Kernel enthalten. Ein Ersatz für eine netzwerkweite Firewall-Lösung ist dies zwar nicht, aber immerhin eine zusätzliche Absicherung.
- **Ein Intrusion Detection System** (IDS)
  Angriffe oder Angriffsversuche sind nicht immer leicht zu erkennen. Am besten ist es, Sie installieren eine Software, die unerwünschte Änderungen

erkennt und Alarm schlägt. Eine gute Wahl für Linux ist beispielsweise das Integritätsprüfungs-Tool `tripwire`. Sie finden dieses Open-Source-Programm auf der DVD zum Buch. Weiter unten wird zudem `mod_security` vorgestellt – es handelt sich um die Verwirklichung eines IDS (und weiterer Sicherheitsmaßnahmen) als Apache-Modul.

- **chroot-Umgebung**
  Auf UNIX-Systemen gibt es noch eine weitere interessante Sicherheitsmaßnahme: Sie können ein Programm in einen `chroot`-»Käfig« sperren. Das bedeutet, dass der Software vorgegaukelt wird, ihr lokales Unterverzeichnis sei das Wurzelverzeichnis des Dateisystems. Beachten Sie allerdings, dass auch dies keine absolute Sicherheit gewährleistet – es gibt immer wieder Sicherheitslücken in den eingesperrten Programmen oder in den `chroot`-Implementierungen, die es einem Angreifer ermöglichen, aus der `chroot`-Umgebung zu entkommen.

  Die Einrichtung einer solchen Umgebung ist vor allem deshalb nicht ganz einfach, weil Sie herausfinden müssen, welche Verzeichnisse und Dateien Apache außer seinem eigenen Installationsumfang benötigt, um ausgeführt werden zu können. Die genaue Konfiguration hängt stark von Ihrem konkreten System ab. Auf der Website zum Buch finden Sie Links zu Informationen für diverse Linux- und UNIX-Versionen.

  In Apache 2.2.10 wurde die Direktive `ChrootDir` hinzugefügt, die nach dem Start von Apache, aber vor dem Akzeptieren von Anfragen ein chroot in das angegebene Verzeichnis durchführt. Sie wird in Abschnitt 17.2.2 beschrieben.

- »**Cracker-Tools**«
  Im ersten Moment hört es sich vielleicht merkwürdig an, aber eine der besten Schutzmaßnahmen vor Angriffen besteht darin, die Werkzeuge der potenziellen Angreifer zu benutzen, um Sicherheitslücken zu erkennen und nach Möglichkeit zu schließen. Eines der wertvollsten Tools in dieser Sparte ist *Nessus*.[1] Dieses freie Programm funktioniert unter zahlreichen UNIX-Varianten; für Windows ist ein Port namens *NeWT* verfügbar. Sie finden beide Versionen auf der DVD.

- **Geschulte Mitarbeiter!**
  Die beste technische Sicherheitsmaßnahme nützt nichts, wenn die User nicht darüber Bescheid wissen. Menschliches Versagen ist immer die größte Sicherheitslücke; einige der erfolgreichsten Cracker haben nicht nur durch ausgeklügelte technische Maßnahmen Zugriff auf geschützte Systeme erlangt, sondern vor allem durch »Social Engineering« – siehe beispielsweise [MITN 2003]. Besonders der vernünftige Umgang mit Passwörtern ist weithin unbekannt. Weisen Sie Ihre Mitarbeiter deshalb besonders auf folgende Punkte hin:

1 Damit will ich nicht gesagt haben, die Intention der Nessus-Entwickler sei die Unterstützung von Crackern!

- Passwörter dürfen niemals sinnvolle Wörter sein, sondern sollten möglichst wirre Kombinationen aus Groß- und Kleinbuchstaben, Ziffern und Sonderzeichen sein.
- Passwörter müssen regelmäßig gewechselt werden. Wenn Ihr Betriebssystem eine Möglichkeit bietet, die Benutzer nach einer bestimmten Zeit zur Änderung zu zwingen, sollten Sie diese nutzen.
- Ein Passwort wird niemals aufgeschrieben! Ein sicheres Passwort, das man sich dennoch leicht merken kann, ergibt sich am einfachsten aus den Anfangsbuchstaben eines möglichst sinnlosen Satzes; einige Zeichen sollten allerdings keine Buchstaben sein.
- Administratoren, Mitarbeiter von Internet-Providern, Polizisten oder Staatsbeamte fragen niemals nach dem Passwort. Wer auch immer Sie also danach fragt, ist ein Betrüger.
- Passwörter sind erst der Anfang – die vorgestellten Maßnahmen zur Passwortsicherheit sind wichtig. Für sich allein sind sie aber noch lange keine ausreichende Absicherung; das sollten sowohl Sie als auch die restlichen Benutzer wissen.

## 18.2 Apache-Sicherheit

Dieser Abschnitt stellt zunächst einige allgemeine Tipps zusammen, die Sie über das Buch verteilt bereits erhalten haben; anschließend werden einige eingebaute Direktiven zur Sicherheit vorgestellt.

### 18.2.1 Allgemeine Sicherheitshinweise

In diesem Buch finden Sie die wichtigsten Hinweise zur Sicherheit jeweils an Ort und Stelle. Dennoch ist das Thema so bedeutend, dass die wichtigsten Sicherheitstipps hier noch einmal im Überblick zusammengefasst werden:

- **Im Allgemeinen schützen, im Besonderen freigeben**
  Sie sollten es sich zur Regel machen, sämtliche Direktiven, die mit der Aktivierung von Zugriffsrechten oder der Freischaltung von Optionen zu tun haben, grundsätzlich so restriktiv wie möglich einzustellen und dann nur in Einzelfällen zu lockern. Dazu gehört insbesondere die grundsätzliche Voreinstellung zum Sperren aller Verzeichnisse:

```
<Directory />
Require all denied
</Directory>
```

Beziehungsweise für Apache bis Version 2.2:

```
<Directory />
  Order deny,allow
  Deny from all
  Options None
  AllowOverride None
</Directory>
```

- **ScriptAlias-Verzeichnisse für CGI verwenden**
  Es ist ein relativ großes Sicherheitsrisiko, Verzeichnisse unterhalb der `DocumentRoot` für CGI-Anwendungen freizuschalten. Sie sollten deshalb nach Möglichkeit `ScriptAlias`-Verzeichnisse verwenden. In manchen Situationen lässt es sich allerdings nicht verhindern, CGI innerhalb normaler Site-Verzeichnisse zu aktivieren – das ist beispielsweise der Fall, wenn Sie Hosting-Dienstleistungen mit virtuellen Hosts anbieten. Näheres zu diesem Thema finden Sie in Kapitel 14, »CGI«.

- **Eine separate User- und Group-ID für Apache benutzen**
  Auf UNIX-Systemen muss Apache unter der User-ID `root` gestartet werden, beispielsweise weil der Zugriff auf die TCP-Ports 0 bis 1023 anders nicht möglich ist. Die Child-Prozesse, die Anfragen bearbeiten, werden allerdings unter den IDs ausgeführt, die Sie mithilfe der Direktiven `User` und `Group` festgelegt haben. Achten Sie darauf, dass dieser Benutzer und diese Gruppe speziell für Apache angelegt werden. Das Benutzerkonto sollte keine Möglichkeit zur persönlichen Anmeldung besitzen; sogar das Passwort sollten Sie nach der Eingabe schnellstmöglich vergessen.

  Übrigens besteht auch unter Windows die Möglichkeit, einen separaten Benutzer für den Betrieb von Apache einzurichten. Das Verfahren wurde in Kapitel 5, »Apache in Betrieb nehmen«, ausführlich erläutert.

- **Regelmäßige Updates durchführen**
  Open-Source-Software besitzt zahlreiche Vorteile gegenüber kommerziellen Programmen. Einer der wichtigsten ist die schnellere Erkennung von Sicherheitslücken und die meist ebenso prompte Bereitstellung von Patches. Wichtig ist allerdings, dass Sie diese Patches auch einspielen; zahlreiche Angriffe sind nur deshalb erfolgreich, weil die zuständigen Admins dies monatelang versäumt haben. Sie sollten zumindest stets das neueste Release von Apache installieren, weil darin jeweils alle bisher bekannten Sicherheitslücken geschlossen werden. Die Devise »Never change a running system« gilt hier nicht!

  Fast jedes Unterversions-Upgrade schließt mehrere Sicherheitslücken für unterschiedliche Plattformen. Eine vollständige Liste der Änderungen in allen Versionen von Apache 2.3-beta, einschließlich Sicherheitslücken und Gegen-

maßnahmen dazu, finden Sie in der Datei *www.apache.org/dist/httpd/CHANGES_2.3*. Diese Datei ist übrigens auch Teil jeder Apache-Distribution.

- **Log-Dateien auswerten**
  Unregelmäßigkeiten in Ihren Log-Dateien deuten oft auf Angriffsversuche hin. Sie sollten diese sorgfältig untersuchen und dann gegebenenfalls Schutzmaßnahmen gegen eine bestimmte Art von Attacken einleiten. Alles Wichtige über Log-Dateien steht in Kapitel 11, »Logging«.
- **Sicherheits-Mailinglisten abonnieren**
  Als verantwortungsvoller Administrator sollten Sie Mailinglisten zum Thema Sicherheit lesen, in denen regelmäßig über Sicherheitslücken und eventuelle Patches berichtet wird. Die bekannteste derartige Liste ist Bugtraq (mittlerweile aufgeteilt in zahlreiche Spezialgebiete). Sie finden sie auf der Site *www.securityfocus.com*.

  Eine recht brauchbare wöchentliche Übersicht von Sicherheitsthemen bietet darüber hinaus der Heise Security Newsletter. Sie können ihn unter *www.heise-sec.de* abonnieren.

### 18.2.2 Sicherheitsrelevante Direktiven

Es gibt einige Apache-Konfigurationsdirektiven, die indirekt die Sicherheit betreffen. Sie schützen insbesondere vor Denial-of-Service-Attacken, indem sie verschiedene Aspekte der HTTP-Anfrage sowie der Ressourcennutzung einschränken.

#### ChrootDir

chroot in das angegebene Verzeichnis durchführen

| | |
|---|---|
| **Seit Version** | 2.2.10 |
| **Modul** | `mod_unixd`; bis 2.2 `prefork`, `worker`, `event` |
| **Kontext** | Server |
| **Syntax** | `chroot Verzeichnis` |
| **Standardwert** | nicht gesetzt |

Vor Version 2.2.10 war die in Abschnitt 18.1 beschriebene `chroot`-Funktionalität für Apache nur extern verfügbar. Diese neue Direktive führt nach dem Start von Apache, aber vor dem Akzeptieren ein `chroot` in das angegebene Verzeichnis durch. Allerdings sind zusätzliche Punkte zu beachten, die über das bloße Setzen der Direktive hinausgehen. Halten Sie auf der Website zum Buch Ausschau nach entsprechenden Links.

### LimitInternalRecursion

Maximale Anzahl interner Weiterleitungen

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost> |
| Syntax | LimitInternalRecursion Anzahl [Anzahl] |
| Standardwert | 10 |

Diese Direktive verhindert, dass Weiterleitungen beziehungsweise Unteranfragen in eine Endlosschleife geraten; nach der angegebenen Höchstzahl von Umleitungen wird die Anfrage verworfen. Der erste Wert gilt für Weiterleitungen. Wenn Sie einen zweiten Wert angeben, wird dieser für verschachtelte Unteranfragen verwendet; ansonsten gilt der erste Wert für beide.

### LimitRequestBody

Zulässige Größe für den Body der HTTP-Anfrage

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (All) |
| Syntax | LimitRequestBody Bytes |
| Standardwert | 0 (unbegrenzt) |

Mithilfe dieser Direktive können Sie die Größe einschränken, die der Body von Anfragen mit den HTTP-Methoden POST oder PUT besitzen darf. Der Wert kann zwischen 0 (unbegrenzt; Voreinstellung) und 2.147.483.647 (2 Gigabyte) liegen. Das folgende Beispiel begrenzt die Größe auf 500 Kilobyte:

```
LimitRequestBody 512000
```

### LimitRequestFields

Zulässige Höchstzahl von HTTP-Anfrage-Headern

| Modul | core |
|---|---|
| Kontext | Server |
| Syntax | LimitRequestFields Anzahl |
| Standardwert | 100 |

Diese Direktive limitiert die Anzahl der HTTP-Header-Zeilen, die von einem Client akzeptiert werden. Der Wert darf zwischen 0 (unbegrenzt) und 32.767 liegen.

Im Apache-Quellcode existiert die Konstante `DEFAULT_LIMIT_REQUEST_FIELDS`, die den Vorgabewert (normalerweise 100) bestimmt. Mehr als 100 Header-Felder sind für normale Anfragen übrigens so gut wie ausgeschlossen; eine solche Anfrage weist eher auf einen Angriffsversuch hin. Normalerweise können Sie den Wert sogar noch einschränken. Beispiel:

```
LimitRequestFields 30
```

#### LimitRequestFieldSize

Zulässige Maximalgröße des HTTP-Anfrage-Headers

| Modul | core |
|---|---|
| Kontext | Server |
| Syntax | LimitRequestFieldSize Bytes |
| Standardwert | 8190 |

Diese Direktive beschränkt die Gesamtlänge aller HTTP-Anfrage-Header, die ein Client senden darf. Der Vorgabewert wird durch die Konstante `DEFAULT_LIMIT_REQUEST_FIELDSIZE` bestimmt und beträgt zunächst 8190. Das ist für gewöhnlich mehr als ausreichend.

#### LimitRequestLine

Zulässige Höchstlänge einer HTTP-Anfragezeile

| Modul | core |
|---|---|
| Kontext | Server |
| Syntax | LimitRequestLine Bytes |
| Standardwert | 8190 |

Mit dieser Direktive wird die Länge der ersten Zeile einer HTTP-Anfrage begrenzt. Diese Zeile besitzt normalerweise einen Aufbau wie in folgendem Beispiel (Näheres siehe Kapitel 2, »Funktionsweise von Webservern«):

```
GET / HTTP/1.1
```

Da Pfadnamen selten eine solche Länge erreichen, geht es vor allem um eine Beschränkung für die Länge von Query-Strings, die bekanntlich Teil der angeforderten URL sind. Der Standardwert von 8190 genügt in aller Regel. Er wird durch die Konstante `DEFAULT_LIMIT_REQUEST_LINE` bestimmt.

#### LimitXMLRequestBody

Zulässige Maximalgröße eines HTTP-Anfrage-Bodys im XML-Format

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (All) |
| Syntax | LimitXMLRequestBody Bytes |
| Standardwert | 1000000 |

Diese Direktive schränkt die maximal erlaubte Größe eines HTTP-Anfrage-Bodys im XML-Format ein. Dies ist besonders im Zusammenhang mit WebDAV (siehe Kapitel 17, »Apache erweitern«) wichtig. Wenn Sie den Wert 0 verwenden, wird diese Prüfung deaktiviert.

### RLimitCPU

Begrenzt die CPU-Nutzung durch Prozesse, die von Apache-Child-Prozessen gestartet wurden

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (All) |
| Syntax | RLimitCPU Sekunden\|max |
| Standardwert | unbestimmt |

Mit »Prozessen, die von Apache-Child-Prozessen gestartet wurden«, sind CGI-Skripte und andere serverseitige Anwendungen gemeint. Die RLimit*-Direktiven schützen nicht so sehr vor externen Angriffsversuchen, sondern viel stärker vor schlechter Programmierung.

RLimitCPU limitiert die CPU-Nutzung durch untergeordnete Prozesse. Sie können einen oder zwei Werte angeben. Die erste Angabe ist eine weiche Begrenzung (es wird kein neuer Prozess gestartet, wenn der Wert überschritten wurde); die zweite ist die absolute Obergrenze. Die Angabe erfolgt in Sekunden pro Prozess. Der spezielle Wert max bezeichnet die Höchstgrenze des Betriebssystems.

### RLimitMEM

Begrenzt die Speichernutzung durch Prozesse, die von Apache-Child-Prozessen gestartet wurden

| Modul | core |
|---|---|
| Kontext | Server, <VirtualHost>, <Directory>, <Location>, <Files>, .htaccess (All) |
| Syntax | RLimitMEM Bytes\|max |
| Standardwert | unbestimmt |

Diese Direktive bestimmt, wie viel Speicher abhängige Unterprozesse wie CGI oder SSI nutzen dürfen. Sie können entweder eine bestimmte Byte-Anzahl angeben oder den Wert `max` verwenden, der den Grenzwert des Systems darstellt.

#### RLimitNPROC

Begrenzt die Erzeugung neuer Prozesse durch Prozesse, die wiederum von Apache-Child-Prozessen gestartet wurden

| | |
|---|---|
| **Modul** | `core` |
| **Kontext** | Server, `<VirtualHost>`, `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`All`) |
| **Syntax** | `RLimitNPROC Anzahl\|max [Anzahl\|max]` |
| **Standardwert** | unbestimmt |

Mit dieser Direktive können Sie festlegen, wie viele Child-Prozesse ein abhängiger Prozess starten darf. Mit anderen Worten: Es geht um spezielle Webanwendungen, die ihrerseits `fork()` aufrufen, um Child-Prozesse zu erzeugen. Genau wie bei `RLimitCPU` ist der erste Wert eine weiche Grenze; der optionale zweite Wert ist das absolute Maximum. `max` ist wiederum die Voreinstellung des Betriebssystems.

#### EnableExceptionHook

Aktiviert einen zusätzlichen Hook für die Ausnahmebehandlung

| | |
|---|---|
| **Seit Version** | 2.0.49 |
| **Modul** | `prefork`, `worker` |
| **Kontext** | Server |
| **Syntax** | `EnableExceptionHook On\|Off` |
| **Standardwert** | `Off` |

Diese Direktive steht nur zur Verfügung, wenn Apache mit der Option `--enable-exception-hook` kompiliert wird (siehe Kapitel 4, »Apache kompilieren und installieren«). Wenn Sie den Wert `On` setzen, wird ein zusätzlicher Hook aktiviert, der beim Absturz eines Worker-Prozesses eine Funktion aufrufen kann. Der Umgang mit Hooks in der Modulprogrammierung wurde in Kapitel 17, »Apache erweitern«, erläutert; es gibt bereits zwei Zusatzmodule, die von diesem Hook Gebrauch machen: `mod_whatkilledus` und `mod_backtrace`. Beide dienen dazu, die Umstände des Absturzes aufzuklären. Näheres über den Exception-Hook und diese beiden Module erfahren Sie auf den Seiten von Jeff Trawick unter *http://www.apache.org/~trawick/exception_hook.html*.

### 18.2.3 SuEXEC

Mithilfe von SuEXEC haben Sie die Möglichkeit, CGI-Skripte und Server Side Includes unter einer anderen User- und Group-ID auszuführen als Apache selbst. Einem Angreifer, dem der Einbruch über ein schlecht abgesichertes Skript gelingt, wird auf diese Weise der Zugriff auf den eigentlichen Webserver erschwert. Apache ruft zu diesem Zweck das Hilfsprogramm `suexec` auf, das im Wesentlichen ein `setuid`- und `setgid`-Wrapper für eigene Skripte ist.

Damit `suexec` benutzt werden kann, muss Apache 2 zunächst einmal mit den passenden Einstellungen kompiliert worden sein. Diese `configure`-Optionen wurden bereits in Kapitel 4, »Apache kompilieren und installieren«, erwähnt. Hier sehen Sie sie noch einmal in etwas ausführlicherer Beschreibung:

- **`--enable-suexec[=shared]`**
  Damit wird das Modul `mod_suexec` grundsätzlich aktiviert; die Variante mit `shared` müssen Sie natürlich eingeben, um es als DSO-Modul zu kompilieren.
- **`--with-suexec-bin=/Pfad/von/suexec`**
  Aus Sicherheitsgründen muss der Pfad des Programms `suexec` fest einkompiliert werden. Die Voreinstellung variiert je nach Installationslayout.
- **`--with-suexec-caller=UID`**
  Dies ist die User-ID des Benutzers, der `suexec` aufrufen soll. Es handelt sich um die UID, unter der auch Apache selbst ausgeführt wird.
- **`--with-suexec-userdir=Verzeichnis`**
  Mit dieser Option wird der Name des Verzeichnisses unterhalb der Home-Verzeichnisse von Benutzern angegeben, in dem `suexec`-CGI-Skripte verwendet werden können. Der Standardwert ist `public_html`.
- **`--with-suexec-docroot=Verzeichnis`**
  Hier wird die `DocumentRoot` von Apache angegeben. Diese bildet das allgemeine Verzeichnis, in dem `suexec`-Skripte gestattet sind.
- **`--with-suexec-uidmin=UID`**
  Dies ist der niedrigste numerische Wert für die User-ID, unter der `suexec`-CGIs ausgeführt werden dürfen. Der Standardwert ist 100, was meist in Ordnung ist.
- **`--with-suexec-gidmin=GID`**
  Dieser Wert ist die niedrigste numerische Group-ID, der die Ausführung von `suexec`-Skripten erlaubt wird. Auch diese Option besitzt die Voreinstellung 100.
- **`--with-suexec-logfile=/Pfad/zur/Datei`**
  `suexec` führt eine eigene Log-Datei, in der sämtliche Aktionen und Fehler protokolliert werden. Der Standardwert ist `suexec_log` im Standardverzeichnis für Log-Dateien (`--logfiledir`).

- `--with-suexec-safepath=Verzeichnis`
  Diese Option bestimmt den Inhalt der Umgebungsvariablen `PATH`, der an `suexec`-CGIs übermittelt wird.

Nachdem Sie `suexec` auf diese Weise eingerichtet haben, werden CGI-Skripte und Server Side Includes unter einer alternativen User- und Group-ID ausgeführt, wenn eine der beiden folgenden Voraussetzungen erfüllt ist:

- Sie haben für den Server oder den jeweiligen virtuellen Host die Direktive `SuexecUserGroup` gesetzt.
- Das Skript befindet sich auf der Website unter einem mittels `mod_userdir` eingebundenen Home-Verzeichnis (siehe Kapitel 8, »Weiterleitungen und Indizes«). In diesem Fall wird der Prozess unter der User- und Group-ID des jeweiligen Benutzers selbst ausgeführt.

`suexec` führt nacheinander folgende Sicherheitsüberprüfungen durch, bevor es ein Skript tatsächlich ausführt:

1. User- und Group-ID müssen gültig sein.
2. Das Wrapper-Programm `suexec` muss korrekt aufgerufen worden sein (darum kümmert sich Apache hinter den Kulissen; das Programm ist nicht für den Kommandozeilenaufruf gedacht).[2]
3. Der User muss die Berechtigung besitzen, den SuEXEC-Wrapper aufzurufen; dies darf nur die User-ID, unter der Apache läuft (Option `--with-suexec-caller`).
4. Der Pfad des auszuführenden Skripts darf keine unsichere URL-Referenz sein, das heißt, sein Pfad darf nicht mit / oder ../ beginnen. Das Skript muss unterhalb der `DocumentRoot` oder in einem der festgelegten Userverzeichnisse liegen.
5. Der `suexec`-User muss existieren.
6. Auch die `suexec`-Group muss vorhanden sein.
7. Der `suexec`-User darf nicht `root` sein (dies würde das ganze Konzept ad absurdum führen).
8. Die numerische User-ID muss über dem einkompilierten Limit (`--with-suexec-uidmin`) liegen.
9. Auch die `suexec`-Group darf nicht `root` sein.

2 Es besitzt eine einzige dokumentierte Kommandozeilenoption: `-V` gibt für `root` die Optionen aus, mit denen es kompiliert wurde.

10. Die numerische Group-ID muss oberhalb des Wertes liegen, der mithilfe der Option `--with-suexec-gidmin` konfiguriert wurde.
11. Der Wechsel der User- und Group-ID für den SuEXEC-Wrapper muss möglich sein.
12. Das angegebene Verzeichnis, in dem sich das auszuführende Skript befindet, muss vorhanden sein.
13. Das Verzeichnis des SuEXEC-Skripts muss unter der Apache-`DocumentRoot` oder in einem gültigen Website-Verzeichnis unterhalb eines Home-Verzeichnisses liegen.
14. Das Userverzeichnis darf nur Schreibrechte für den Eigentümer besitzen.
15. Das aufgerufene Skript selbst muss existieren.
16. Nur der Eigentümer darf über Schreibrechte an dem CGI/SSI-Skript verfügen.
17. Bei dem Skript selbst dürfen die `setuid`- beziehungsweise `setgid`-Bits nicht gesetzt sein.
18. `suexec`-Benutzer und -Gruppe müssen Eigentümer des Skripts sein.
19. `PATH` und andere Umgebungsvariablen müssen sich vor dem Start des Skripts auf sichere Werte setzen lassen (siehe `--with-suexec-safepath`).
20. Das Skript muss sich tatsächlich ausführen lassen.

**Suexec**

Suexec-Funktionalität ein- und ausschalten

| | |
|---|---|
| **Seit Version** | 2.3.9 |
| **Modul** | `mod_unixd` |
| **Kontext** | Server |
| **Syntax** | `Suexec On\|Off` |
| **Standardwert** | `On`, falls das `suexec`-Binary vorhanden ist, andernfalls `Off` |

Der Hauptverwendungszweck dieser Direktive besteht darin, die Suexec-Funktionalität abzuschalten, obwohl das `suexec`-Binary vorhanden ist und die anderen oben beschriebenen Voraussetzungen erfüllt sind. In diesem Fall wird folgende Zeile in die Server-Konfiguration geschrieben:

```
Suexec Off
```

Falls Sie versuchen, `Suexec On` zu verwenden, obwohl die Voraussetzungen dafür nicht erfüllt sind, verweigert Apache den Start mit einer Fehlermeldung.

**SuexecUserGroup**

Benutzer- und Gruppen-ID für `suexec`-CGI-Skripte

| | |
|---|---|
| **Seit Version** | 2.0 |
| **Modul** | `mod_suexec` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `SuexecUserGroup Benutzer Gruppe` |
| **Standardwert** | nicht gesetzt |

Diese Direktive gibt die User- und Group-ID an, unter der `suexec`-CGI-Skripte ausgeführt werden sollen. Wenn Sie möchten, können Sie für jeden virtuellen Host einen anderen Benutzer und eine andere Gruppe festlegen. Die gewöhnlichen Apache-Child-Prozesse, die Anfragen beantworten, werden allerdings weiterhin unter der User- und Group-ID ausgeführt, die Sie mithilfe der Direktiven `User` und `Group` festgelegt haben (siehe Kapitel 6, »Grundkonfiguration«). Hier ein Verwendungsbeispiel:

```
SuexecUserGroup suser sgroup
```

## 18.3 mod_security

Eine praktische Möglichkeit, die Sicherheit von Apache 2 weiter zu verbessern, ist die Installation des Moduls `mod_security` von *Ivan Ristic*. Dieser hat seine Erfahrungen mit der Apache-Sicherheit zu dem empfehlenswerten Buch [RIST 2005] verarbeitet. Sie können das Modul auf der Website *www.modsecurity.org* herunterladen; auf der DVD zum Buch finden Sie die zurzeit aktuelle stabile Version 2.6. Das Modul ist ein frei konfigurierbares Intrusion Detection System und Filtersystem für den Webserver.

Die Installation als DSO-Modul unter UNIX beschränkt sich auf eine einzige Zeile:

```
# /usr/local/apache2/bin/apxs -cia mod_security.c
```

Für Windows gibt es ein vorkompiliertes Binary, das Sie einfach nur in Ihr *modules*-Verzeichnis kopieren müssen.

In jedem Fall muss das DSO-Modul nun natürlich in der Konfigurationsdatei aktiviert werden. Das funktioniert nach dem üblichen Schema:

```
LoadModule security_module modules/mod_security.so
```

Das Modul definiert eine Reihe neuer Direktiven zur Überwachung und Abwehr von Angriffsversuchen. Eine vollständige Referenz würde zu weit führen; auf der DVD zum Buch oder auf der `mod_security`-Website finden Sie die Originaldokumentation im PDF-Format.

Hier einige kommentierte Beispiele:

```
<IfModule mod_security.c>
    # Filtermechanismus einschalten
    SecFilterEngine On

    # URL überprüfen (mehrere / oder ../ entfernen)
    SecFilterCheckURLEncoding On

    # Nur Bytes mit den angegebenen Werten erlauben
    # Hier alle Bytes:
    SecFilterForceByteRange 0 255
    # Paranoia - nur ASCII; keine Steuerzeichen:
    # SecFilterForceByteRange 32 127

    # Nur verdächtige Anfragen protokollieren
    SecAuditEngine RelevantOnly

    # mod_security-Logdatei:
    SecAuditLog logs/audit_log

    # Body von POST-Anfragen überprüfen
    SecFilterScanPOST On

    # Verdächtige Anfragen protokollieren und
    # mit Status 500 Internal Script Error verbieten
    SecFilterDefaultAction "deny,log,status:500"
</IfModule>
```

Die nähere Beschäftigung mit `mod_security` lohnt sich. Vor allem die Log-Datei, die das Modul anlegt, sollten Sie regelmäßig auswerten.

## 18.4 Zusammenfassung

Dieses Kapitel kann nur einen ersten Überblick über das ernste und wichtige Thema Sicherheit geben; im ganzen Buch verteilt finden Sie weitere wichtige Hinweise, die die Sicherheit der einzelnen Features betreffen.

Ein gut abgesicherter Apache-Webserver nützt wenig, wenn er in einer schlecht konfigurierten Systemumgebung arbeiten muss. Beachten Sie daher bitte die Hinweise zur Systemsicherheit im ersten Abschnitt dieses Kapitels.

Anschließend sollten Sie sich um die Sicherheit von Apache selbst kümmern und Ihre Konfiguration noch einmal kritisch anhand der Hinweise im zweiten Abschnitt überprüfen. Zusätzlich existieren diverse Konfigurationsdirektiven, die den möglichen Spielraum für Angriffsversuche einschränken, indem sie den Ressourcenzugriff durch Apache begrenzen.

Schließlich lohnt sich auch die Beschäftigung mit dem externen Modul `mod_security`; es fügt ein eigenes Sicherheitskonzept mit integriertem Intrusion Detection System zum Apache-Webserver hinzu.

*»Das Ganze ist mehr als die Summe seiner Teile.«*
*– Aristoteles –*

# 19 Kommentierte Konfigurationsdateien

Nachdem Sie in den meisten der 18 vorangegangenen Kapitel gründliche Informationen über die Apache-Direktiven erhalten, aber nur Ausschnitte von Konfigurationskontexten gesehen haben, ist es nun an der Zeit, sich einige komplette Konfigurationsdateien anzuschauen. Im ersten Abschnitt werden die Datei *httpd.conf* und alle optionalen Zusatzkonfigurationsdateien abgedruckt, wie sie mit Apache ausgeliefert werden.[1] Der zweite Abschnitt enthält einige zusätzliche Beispiele, die in den Originaldateien nicht vorkommen – es handelt sich um die Konfiguration von Authentifizierung, Proxy mit Cache, PHP und URL-Rewriting.

Auf der Buch-DVD finden Sie im Verzeichnis *vmware* eine virtuelle Maschine für VMware sowie ein Linux-Image (Ubuntu), in dem ein Apache mit all diesen Konfigurationsdateien installiert ist. Zusätzlich ist der VMware Player für Linux und Windows enthalten, in dem Sie dieses Image ausführen können, um mit der Konfiguration zu experimentieren.

## 19.1 Die Originalkonfigurationsdateien

Die hier abgedruckten (übersetzten) Dateien stammen aus einer Standardinstallation von Apache 2.2.10 auf einem Linux-System, das mit folgender Konfiguration kompiliert wurde (Ausgabe von `httpd -V`):

```
Server version: Apache/2.2.10 (UNIX)
Server built:   Oct  6 2008 19:21:51
Server's Module Magic Number: 20051115:18
Server loaded:  APR 1.2.12, APR-Util 1.2.12
Compiled using: APR 1.2.12, APR-Util 1.2.12
Architecture:   32-bit
Server MPM:     Prefork
```

1 Druckseitenverschwendung ist dies trotzdem nicht, da die Konfigurationsdateien zu fast 90% aus erläuternden Kommentaren bestehen, die ich komplett ins Deutsche übersetzt habe.

```
  threaded:     no
    forked:     yes (variable process count)
Server compiled with....
 -D APACHE_MPM_DIR="server/mpm/prefork"
 -D APR_HAS_SENDFILE
 -D APR_HAS_MMAP
 -D APR_HAVE_IPV6 (IPv4-mapped addresses enabled)
 -D APR_USE_SYSVSEM_SERIALIZE
 -D APR_USE_PTHREAD_SERIALIZE
 -D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
 -D APR_HAS_OTHER_CHILD
 -D AP_HAVE_RELIABLE_PIPED_LOGS
 -D DYNAMIC_MODULE_LIMIT=128
 -D HTTPD_ROOT="/usr/local/apache2"
 -D SUEXEC_BIN="/usr/local/apache2/bin/suexec"
 -D DEFAULT_PIDLOG="logs/httpd.pid"
 -D DEFAULT_SCOREBOARD="logs/apache_runtime_status"
 -D DEFAULT_LOCKFILE="logs/accept.lock"
 -D DEFAULT_ERRORLOG="logs/error_log"
 -D AP_TYPES_CONFIG_FILE="conf/mime.types"
 -D SERVER_CONFIG_FILE="conf/httpd.conf"
```

Wo die Konfiguration für Apache 2.4 abweicht, wurden die 2.2er-Konfigurationszeilen auskommentiert, mit einem entsprechenden Kommentar versehen und durch die passende Konfiguration für die neue Version ersetzt.

### 19.1.1 Die Grundkonfigurationsdatei httpd.conf

Dies ist die Basiskonfigurationsdatei, die immer geladen wird. Gegen Ende enthält sie einige (standardmäßig auskommentierte) `Include`-Direktiven. Diese können aktiviert werden, um die in den folgenden Abschnitten abgedruckten Spezialkonfigurationsdateien aus dem Unterverzeichnis *extra* einzuschalten.

```
#
# Dies ist die Hauptkonfigurationsdatei des Apache HTTP Servers.
# Sie enthält die Konfigurationsdirektiven, die dem Server seine
# Anweisungen geben.
# Siehe <URL:http://httpd.apache.org/docs/2.2> für
# Detailinformationen, insbesondere
# <URL:http://httpd.apache.org/docs/2.2/mod/directives.html>
# für eine Beschreibung jeder Konfigurationsdirektive.
#
# Lesen Sie NICHT einfach nur die hier enthaltenen Anleitungen, ohne sie
# genau zu verstehen. Sie dienen nur als Hinweise oder Erinnerungen. Wenn
# Sie unsicher sind, konsultieren Sie die Online-Dokumentation.
```

```
# Wir haben Sie gewarnt.
#
# Konfigurations- und Log-Dateinamen: Wenn der angegebene Dateiname
# vieler Server-Steuerdateien mit "/" beginnt (oder "Laufwerk:/" für
# Win32), dann verwendet der Server diesen genauen Pfad. Wenn der Pfad
# *nicht* mit "/" beginnt, wird ihm der Wert von ServerRoot vorangestellt
# -- "logs/foo_log" mit der ServerRoot "/usr/local/apache2/logs/foo_log"
# wird also vom Server als "/usr/local/apache2/logs/foo_log"
# interpretiert.
#
# ServerRoot: die Wurzel des Verzeichnisbaums, unter dem die
# Konfigurations-, Fehler- und Log-Dateien des Servers liegen
#

# Fügen Sie keinen Slash am Ende des Verzeichnispfads ein. Wenn Sie
# ServerRoot auf eine nicht-lokale Festplatte verweisen lassen, stellen
# Sie sicher, dass die Direktive LockFile auf eine lokale Festplatte
# verweist. Wenn Sie dieselbe ServerRoot gemeinsam für mehrere httpd-
# Daemons nutzen möchten, müssen Sie mindestens LockFile und PidFile
# verändern.
#
ServerRoot "/usr/local/apache2"

#
# Listen: Ermöglicht es Ihnen, Apache an spezifische IP-Adressen
# und/oder Ports zu binden, anstatt die Standardeinstellungen zu wählen.
# Siehe auch die Direktive <VirtualHost>.
#
# Ändern Sie die Direktive, wie unten gezeigt, um an bestimmten IP-
# Adressen zu lauschen und Apache daran zu hindern, alle verfügbaren
# IP-Adressen zu verwenden.
#
#Listen 12.34.56.78:80
Listen 80

#
# Unterstützung für Dynamic Shared Objects (DSO)
#
# Um die Funktionalität eines Moduls nutzen zu können, das als DSO kom-
# piliert wurde, müssen Sie entsprechende 'LoadModule'-Zeilen an dieser
# Stelle einfügen, damit die enthaltenen Direktiven tatsächlich
# verfügbar sind, _bevor_ sie verwendet werden. Statisch kompilierte
# Module (die von `httpd -l' aufgelistet werden) brauchen hier nicht
# geladen zu werden.
#
```

```
# Beispiel:
# LoadModule foo_module modules/mod_foo.so
#
LoadModule authn_file_module modules/mod_authn_file.so
LoadModule authn_dbm_module modules/mod_authn_dbm.so
LoadModule authn_anon_module modules/mod_authn_anon.so
LoadModule authn_dbd_module modules/mod_authn_dbd.so
# mod_authn_default wurde in 2.4 entfernt und daher auskommentiert.
#LoadModule authn_default_module modules/mod_authn_default.so
LoadModule authz_host_module modules/mod_authz_host.so
LoadModule authz_groupfile_module modules/mod_authz_groupfile.so
LoadModule authz_user_module modules/mod_authz_user.so
LoadModule authz_dbm_module modules/mod_authz_dbm.so
LoadModule authz_owner_module modules/mod_authz_owner.so
# mod_authz_default wurde in 2.4 entfernt und daher auskommentiert.
#LoadModule authz_default_module modules/mod_authz_default.so
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule dbd_module modules/mod_dbd.so
LoadModule dumpio_module modules/mod_dumpio.so
LoadModule ext_filter_module modules/mod_ext_filter.so
LoadModule include_module modules/mod_include.so
LoadModule filter_module modules/mod_filter.so
LoadModule substitute_module modules/mod_substitute.so
LoadModule deflate_module modules/mod_deflate.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule log_forensic_module modules/mod_log_forensic.so
LoadModule logio_module modules/mod_logio.so
LoadModule env_module modules/mod_env.so
LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule cern_meta_module modules/mod_cern_meta.so
LoadModule expires_module modules/mod_expires.so
LoadModule headers_module modules/mod_headers.so
LoadModule ident_module modules/mod_ident.so
LoadModule usertrack_module modules/mod_usertrack.so
LoadModule unique_id_module modules/mod_unique_id.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule version_module modules/mod_version.so
LoadModule mime_module modules/mod_mime.so
LoadModule dav_module modules/mod_dav.so
LoadModule status_module modules/mod_status.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule asis_module modules/mod_asis.so
LoadModule info_module modules/mod_info.so
LoadModule cgi_module modules/mod_cgi.so
```

```
LoadModule dav_fs_module modules/mod_dav_fs.so
LoadModule vhost_alias_module modules/mod_vhost_alias.so
LoadModule negotiation_module modules/mod_negotiation.so
LoadModule dir_module modules/mod_dir.so
LoadModule imagemap_module modules/mod_imagemap.so
LoadModule actions_module modules/mod_actions.so
LoadModule speling_module modules/mod_speling.so
LoadModule userdir_module modules/mod_userdir.so
LoadModule alias_module modules/mod_alias.so
LoadModule rewrite_module modules/mod_rewrite.so

# In Apache 2.4 kann auch das MPM dynamisch geladen werden;
# hier prefork als Beispiel
LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
# In 2.4 wurde die gemeinsame Funktionalität aller UNIX-MPMs
# in das Modul mod_unixd ausgelagert, das auch geladen
# werden muss.
LoadModule unixd_module modules/mod_unixd.so

<IfModule !mpm_netware_module>
<IfModule !mpm_winnt_module>
#
# Wenn Sie httpd unter einem anderen User oder einer anderen Gruppe
# ausführen möchten, müssen Sie httpd zuerst als root ausführen, danach
# erfolgt der Wechsel.
#
# User/Group: Name (oder #Nummer) des Benutzers und der Gruppe, unter
# deren Rechten httpd ausgeführt werden soll. Es ist im Allgemeinen am
# besten, einen eigenen User und eine eigene Gruppe zum Ausführen von
# httpd zu erstellen, wie bei den meisten Systemdiensten.
#
User daemon
Group daemon

</IfModule>
</IfModule>

# 'Hauptserver'-Konfiguration
#
# Die Direktiven in diesem Abschnitt richten die Werte ein, die vom
# 'Hauptserver' verwendet werden, dieser beantwortet alle Anfragen,
# die nicht von einer <VirtualHost>-Definition verarbeitet werden.
# Diese Werte stellen auch Default-Werte für alle <VirtualHost>-Container
# bereit, die Sie später in der Datei definieren.
#
```

```
# All diese Direktiven können in <VirtualHost>-Containern stehen;
# in diesem Fall werden die Default-Einstellungen für den gerade
# definierten virtuellen Host überschrieben.
#

#
# ServerAdmin: Ihre Adresse, an die Probleme mit dem Server gemailt
# werden sollen. Diese Adresse erscheint auf einigen vom Server erzeugten
# Seiten, etwa in Fehlerdokumenten, z.B. admin@ihre-domain.de.
#
ServerAdmin webmaster@test.local

#
# ServerName gibt den Namen und den Port an, die der Server verwendet, um
# sich selbst zu identifizieren. Diese Werte können oft automatisch
# ermittelt werden, aber wir empfehlen Ihnen, sie explizit festzulegen,
# um Probleme beim Start zu vermeiden.
#
# Wenn Ihr Host keinen registrierten DNS-Namen hat, geben Sie hier seine
# IP-Adresse an.
#
#ServerName www.example.com:80

#
# DocumentRoot: das Verzeichnis, aus dem Ihre Dokumente geliefert werden
# Standardmäßig werden alle Anfragen aus diesem Verzeichnis beantwortet,
# aber symbolische Links und Aliase können verwendet werden, um auf andere
# Verzeichnisse zu verweisen.
#
DocumentRoot "/usr/local/apache2/htdocs"

#
# Jedes Verzeichnis, auf das Apache Zugriff hat, kann im Hinblick darauf
# konfiguriert werden, welche Dienste und Features in diesem Verzeichnis
# (und seinen Unterverzeichnissen) erlaubt und/oder deaktiviert sind.
#
# Zuerst konfigurieren wir als "Default" einen sehr restriktiven Satz
# von Features.
#
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Require all denied
    # In 2.2 anstelle von Require all denied diese zwei Zeilen:
```

```
    #Order deny,allow
    #Deny from all
</Directory>

#
# Beachten Sie, dass Sie einzelne Features von diesem Punkt an explizit
# erlauben müssen, um sie zu aktivieren - falls also etwas nicht so
# funktioniert, wie Sie es erwarten, stellen Sie sicher, dass Sie es
# explizit aktiviert haben.
#

#
# Dies sollte so geändert werden, dass es mit Ihrer DocumentRoot
# übereinstimmt.
#
<Directory "/usr/local/apache2/htdocs">
    #
    # Mögliche Werte für die Direktive Options sind "None", "All" oder
    # eine beliebige Kombination aus:
    #   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI
    # MultiViews
    #
    # Beachten Sie, dass "MultiViews" *explizit* genannt werden muss --
    # mit "Options All" erhalten Sie diese Option nicht.
    #
    # Die Direktive Options ist sowohl kompliziert als auch wichtig.
    # Siehe http://httpd.apache.org/docs/2.2/mod/core.html#options
    # für weitere Informationen.
    #
    Options Indexes FollowSymLinks

    #
    # AllowOverride bestimmt, welche Direktiven in .htaccess-Dateien
    # stehen dürfen.
    # Sie können "All", "None" oder jede Kombination der folgenden
    # Schlüsselwörter verwenden:
    #   Options FileInfo AuthConfig Limit
    #
    AllowOverride None

    #
    # Legt fest, wer Dokumente von diesem Server erhalten kann.
    #
    Require all granted
    # In 2.2 anstelle von Require all granted folgende zwei Zeilen:
```

```
    #Order allow,deny
    #Allow from all

</Directory>

#
# DirectoryIndex: Legt die Datei fest, die Apache ausliefert, wenn ein
# Verzeichnis angefordert wird.
#
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>

#
# Die folgenden Zeilen verhindern, dass Webclients .htaccess- und
# .htpasswd-Dateien lesen können.
#
<FilesMatch "^\.ht">
    Require All denied
    # In 2.2 stattdessen die folgenden Zeilen:
    #Order allow,deny
    #Deny from all
    #Satisfy All
</FilesMatch>

#
# Wenn Sie in einem <VirtualHost>-Container keine ErrorLog-Direktive
# angeben, werden mit diesem virtuellen Host verknüpfte Fehlermeldungen
# hier protokolliert. *Wenn* Sie eine Fehler-Log-Datei für einen
# <VirtualHost>-Container angeben, werden dessen Fehler dort und nicht
# hier protokolliert.
#
ErrorLog "logs/error_log"

#
# LogLevel: Legt den Umfang der Meldungen fest, die im error_log
# protokolliert werden. Die möglichen Werte sind: debug, info, notice,
# warn, error, crit, alert, emerg.
#
LogLevel warn

<IfModule log_config_module>
    #
    # Die folgenden Direktiven definieren einige Format-Nicknames zur
    # Verwendung in einer CustomLog-Direktive (siehe unten).
```

```
    #
    LogFormat \
     "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
       combined
    LogFormat "%h %l %u %t \"%r\" %>s %b" common

    <IfModule logio_module>
      # Sie müssen mod_logio.c einschalten, um %I und %O zu verwenden
      LogFormat \
    "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" \
       combinedio
    </IfModule>

    #
    # Pfad und Format der Access-Log-Datei (Common Logfile Format).
    # Wenn Sie innerhalb eines <VirtualHost>-Containers keine Access-
    # Log-Dateien angeben, werden sie hier protokolliert. *Wenn* Sie
    # dagegen Access-Log-Dateien pro <VirtualHost> definieren, werden
    # deren Transaktionen dort protokolliert und *nicht* in dieser Datei.
    #
    CustomLog "logs/access_log" common

    #
    # Wenn Sie eine Log-Datei mit Zugriffs-, Client- und Referer-
    # Informationen bevorzugen (Combined Logfile Format), können Sie die
    # folgende Direktive verwenden.
    #
    #CustomLog "logs/access_log" combined
</IfModule>

<IfModule alias_module>
    #
    # Redirect: Ermöglicht es Ihnen, Clients über Dokumente zu
    # informieren, die im Namensraum Ihres Servers existierten, nun aber
    # nicht mehr da sind. Der Client führt eine neue Anfrage für das
    # Dokument mit seiner neuen Adresse durch.
    # Beispiel:
    # Redirect permanent /foo http://www.example.com/bar

    #
    # Alias: Ordnet Webpfaden Dateisystempfade zu und wird verwendet,
    # um auf Inhalte zuzugreifen, die nicht unter der DocumentRoot
    # liegen.
    # Beispiel:
    # Alias /webpath /full/filesystem/path
```

```
    #
    # Wenn Sie einen abschließenden / in /webpath einfügen, dann verlangt
    # der Server, dass dieser auch in der URL vorhanden ist. Sie werden
    # wahrscheinlich auch einen <Directory>-Abschnitt benötigen, um den
    # Zugriff auf den Dateisystempfad zu erlauben.

    #
    # ScriptAlias: Dies bestimmt, welche Verzeichnisse Server-Skripte
    # enthalten. Script-Aliase sind im Prinzip dasselbe wie Aliase, bis
    # auf die Tatsache, dass Dokumente im Zielverzeichnis als Anwendungen
    # behandelt werden, die bei der Anfrage vom Server ausgeführt werden,
    # und nicht als Dokumente, die an den Client gesendet werden. Für
    # ScriptAlias gelten dieselben Regeln bezüglich des abschließenden
    # "/" wie für Alias.
    #
    ScriptAlias /cgi-bin/ "/usr/local/apache2/cgi-bin/"

</IfModule>

<IfModule cgid_module>
    #
    # ScriptSock: auf Thread-basierten Servern den Pfad zu dem UNIX-
    # Socket angeben, das zur Kommunikation mit dem CGI-Daemon von
    # mod_cgid verwendet wird.
    #
    #Scriptsock logs/cgisock
</IfModule>

#
# "/usr/local/apache2/cgi-bin" sollte in Ihr tatsächliches ScriptAlias-
# CGI-Verzeichnis geändert werden, falls Sie dies konfiguriert haben.
#
<Directory "/usr/local/apache2/cgi-bin">
    AllowOverride None
    Options None
    Require all granted
    # Alternative zu Require für 2.2:
    #Order allow,deny
    #Allow from all
</Directory>
#
# DefaultType (nur 2.2): Der Standard-MIME-Type, den der Server für ein
# Dokument verwendet, wenn er auf andere Weise keinen ermitteln kann,
# beispielsweise aus Dateinamenerweiterungen. Wenn Ihr Server
# hauptsächlich Text- oder HTML-Dokumente enthält, ist "text/plain" ein
```

```
# guter Wert. Wenn ein Großteil Ihres Contents binär ist, etwa Anwendungen
# oder Bilder, sollten Sie stattdessen "application/octet-stream"
# verwenden, um Browser an dem Versuch zu hindern, Binärdateien
# anzuzeigen, als seien sie Text.
#
#DefaultType text/plain

<IfModule mime_module>
    #
    # TypesConfig verweist auf die Datei, die die Liste der Zuordnungen
    # von Dateinamenerweiterungen zu MIME-Types enthält.
    #
    TypesConfig conf/mime.types

    #
    # AddType ermöglicht es Ihnen, die in TypesConfig angegebene
    # Konfigurationsdatei für bestimmte Dateitypen zu ergänzen
    # oder zu überschreiben.
    #
    #AddType application/x-gzip .tgz
    #
    # AddEncoding ermöglicht es Ihnen, bestimmte Browser-Informationen
    # im laufenden Betrieb dekomprimieren zu lassen. Hinweis: Nicht alle
    # Browser unterstützen dies.
    #
    #AddEncoding x-compress .Z
    #AddEncoding x-gzip .gz .tgz
    #
    # Wenn die AddEncoding-Direktiven weiter oben auskommentiert sind,
    # dann sollten Sie diese Erweiterungen sinnvollerweise so definieren,
    # dass sie Medientypen angeben:
    #
    AddType application/x-compress .Z
    AddType application/x-gzip .gz .tgz

    #
    # AddHandler ermöglicht es Ihnen, bestimmte Dateierweiterungen den
    # "Handlern" zuzuordnen: Aktionen, die nicht mit Dateitypen ver-
    # knüpft sind. Diese können entweder in den Server einkompiliert
    # oder mithilfe der Direktive Action (siehe unten) hinzugefügt
    # werden.
    #

    # Um CGI-Skripte außerhalb von ScriptAlias-Verzeichnissen zu ver-
    # verwenden: (Sie müssen auch "ExecCGI" zur Direktive "Options"
```

```
    # hinzufügen.)
    #
    #AddHandler cgi-script .cgi

    # Für Type-Maps (Negotiating-Ressourcen):
    #AddHandler type-map var

    #
    # Filter ermöglichen es Ihnen, Inhalte zu verarbeiten, bevor sie an
    # den Client gesendet werden.
    #
    # Um .shtml-Dateien für Server Side Includes (SSI) zu parsen:
    # (Sie müssen auch "Includes" zur Direktive "Options" hinzufügen.)
    #
    #AddType text/html .shtml
    #AddOutputFilter INCLUDES .shtml
</IfModule>

#
# Das Modul mod_mime_magic erlaubt es dem Server, diverse Hinweise aus
# dem Inhalt der Dateien selbst zu verwenden, um deren Typ zu ermitteln.
# Die Direktive MIMEMagicFile teilt dem Modul mit, wo sich die Hinweis-
# definitionen befinden.
#
#MIMEMagicFile conf/magic

#
# Anpassbare Fehlerantworten gibt es in drei Sorten:
# 1) reiner Text 2) lokale Weiterleitungen 3) externe Weiterleitungen
#
# Einige Beispiele:
#ErrorDocument 500 "Der Server hat Peng Peng gemacht."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
#

#
# EnableMMAP und EnableSendfile: Auf Systemen, die dies unterstützen,
# werden Memory-Mapping oder der Systemaufruf sendfile verwendet, um
# Dateien auszuliefern. Dies verbessert im Allgemeinen die Server-
# Performance, muss aber ausgeschaltet werden, wenn Content aus
# Netzwerkdateisystemen ausgeliefert wird oder wenn die Unterstützung für
# diese Funktionen auf Ihrem System aus anderen Gründen nicht
# funktioniert.
```

```
#
#EnableMMAP off
#EnableSendfile off

# Zusätzliche Konfiguration
#
# Die Konfigurationsdateien im Verzeichnis conf/extra können eingefügt
# werden, um zusätzliche Features hinzuzufügen oder um die Standard
# konfiguration des Servers zu ändern, oder Sie können ihre Inhalte
# einfach hierhin kopieren und sie je nach Bedarf ändern.

# Server-Pool-Verwaltung (MPM-specifisch)
#Include conf/extra/httpd-mpm.conf

# Mehrsprachige Fehlermeldungen
#Include conf/extra/httpd-multilang-errordoc.conf

# Fancy-Verzeichnislistings
#Include conf/extra/httpd-autoindex.conf

# Spracheinstellungen
#Include conf/extra/httpd-languages.conf

# Benutzer-Home-Verzeichnisse
#Include conf/extra/httpd-userdir.conf

# Echtzeitinformationen über Anfragen und Konfiguration
#Include conf/extra/httpd-info.conf

# Virtuelle Hosts
#Include conf/extra/httpd-vhosts.conf

# Lokaler Zugriff auf das Apache HTTP Server Manual
#Include conf/extra/httpd-manual.conf

# Distributed authoring and versioning (WebDAV)
#Include conf/extra/httpd-dav.conf

# Diverse Standardeinstellungen
#Include conf/extra/httpd-default.conf

# Sichere Verbindungen (SSL/TLS)
#Include conf/extra/httpd-ssl.conf
#
```

```
# Hinweis: Das Folgende muss vorhanden sein, um den Start ohne SSL
#       auf Plattformen zu unterstützen, die kein /dev/random-Äquivalent
#          besitzen, sondern ein statisch einkompiliertes mod_ssl.
#
<IfModule ssl_module>
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
</IfModule>
```

**Listing 19.1** Die Hauptkonfigurationsdatei »httpd.conf«

Es folgen die Konfigurationsdateien aus dem Unterverzeichnis *extra* in der Reihenfolge der `Include`-Direktiven aus der Datei *httpd.conf*.

### 19.1.2 Server-Pool-Verwaltung (httpd-mpm.conf)

Diese Datei enthält die Spezialeinstellungen für die verschiedenen Multiprocessing-Module, jeweils sauber durch `<IfModule>`-Container umschlossen. Wenn Sie Änderungen vornehmen möchten, können Sie die Datei inkludieren und die für Ihr jeweiliges MPM zutreffenden Direktiven editieren – dies ist meist dann der Fall, wenn Sie eine besonders gute Server-Performance oder Hochlastfähigkeit benötigen. Genaueres über die verwendeten Direktiven erfahren Sie in Kapitel 6, »Grundkonfiguration«.

```
#
# Server-Pool-Verwaltung (MPM-spezifisch)
#

#
# PidFile: die Datei, in der der Server seine Prozess-
# Identifikationsnummer ablegen soll, wenn er startet
#
# Beachten Sie, dass dies für die meisten MPMs das Standard-PidFile ist.
#
<IfModule !mpm_netware_module>
    PidFile "logs/httpd.pid"
</IfModule>

#
# Die Accept-Serialisierungs-Lockdatei MUSS AUF EINER LOKALEN FESTPLATTE
# GESPEICHERT WERDEN.
#
<IfModule !mpm_winnt_module>
<IfModule !mpm_netware_module>
LockFile "logs/accept.lock"
```

```
</IfModule>
</IfModule>

#
# Nur einer der unten stehenden Abschnitte ist für Ihren installierten
# httpd relevant. Verwenden Sie "apachectl -l", um das aktive MPM
# zu ermitteln.
#

# prefork MPM
# StartServers: Anzahl der zu startenden Server-Prozesse
# MinSpareServers: Mindestzahl unbeschäftigter Reserve-Server-Prozesse
# MaxSpareServers: Höchstzahl unbeschäftigter Reserve-Server-Prozesse
# MaxClients: Höchstzahl von Server-Prozessen, die gestartet werden
#             dürfen
# MaxRequestsPerChild: Höchstzahl der Anfragen, die ein Server-Prozess
#                      beantwortet
<IfModule mpm_prefork_module>
    StartServers          5
    MinSpareServers       5
    MaxSpareServers      10
    MaxClients          150
    MaxRequestsPerChild   0
</IfModule>

# worker MPM
# StartServers: Anfangszahl zu startender Server-Prozesse
# MaxClients: Höchstzahl gleichzeitiger Client-Verbindungen
# MinSpareThreads: Mindestzahl unbeschäftigter Reserve-Worker-Threads
# MaxSpareThreads: Höchstzahl unbeschäftigter Reserve-Worker-Threads
# ThreadsPerChild: Konstante Anzahl von Worker-Threads in jedem Prozess
# MaxRequestsPerChild: Höchstzahl von Anfragen, die ein Server-Prozess
#                      beantwortet
<IfModule mpm_worker_module>
    StartServers          2
    MaxClients          150
    MinSpareThreads      25
    MaxSpareThreads      75
    ThreadsPerChild      25
    MaxRequestsPerChild   0
</IfModule>

# BeOS MPM
# StartThreads: Wie viele Threads sollen zu Anfang erzeugt werden?
# MaxClients:   Höchstzahl erlaubter Threads (1 Thread == 1 Client)
```

```
# MaxRequestsPerThread: Höchstzahl von Anfragen, die jeder Thread
#                       verarbeitet
<IfModule mpm_beos_module>
    StartThreads            10
    MaxClients              50
    MaxRequestsPerThread 10000
</IfModule>

# NetWare MPM
# ThreadStackSize: erlaubte Stapelgröße für jeden Worker-Thread
# StartThreads: Anzahl der Worker-Threads, die beim Server-Start starten
# MinSpareThreads: Mindestzahl unbeschäftigter Threads für
#                  Anfragespitzen
# MaxSpareThreads: Höchstzahl unbeschäftigter Threads
# MaxRequestsPerChild: Höchstzahl von Anfragen, die ein Thread
#                      beantwortet
#                     Es wird empfohlen, auf NetWare den Standardwert 0
#                    für diese Direktive zu setzen. Dies erlaubt es dem
#                      Thread, eine unbestimmte Anzahl von Anfragen zu
#                      beantworten.
<IfModule mpm_netware_module>
    ThreadStackSize      65536
    StartThreads           250
    MinSpareThreads         25
    MaxSpareThreads        250
    MaxThreads            1000
    MaxRequestsPerChild      0
    MaxMemFree             100
</IfModule>

# OS/2 MPM
# StartServers: Anzahl der aufrechtzuerhaltenden Prozesse
# MinSpareThreads: Mindestzahl unbeschäftigter Threads pro Prozess,
#                  um mit Anfragespitzen umzugehen
# MaxSpareThreads: Höchstzahl unbeschäftigter Threads pro Prozess
# MaxRequestsPerChild: Höchstzahl von Verbindungen pro Server-Prozess
<IfModule mpm_mpmt_os2_module>
    StartServers           2
    MinSpareThreads        5
    MaxSpareThreads       10
    MaxRequestsPerChild    0
</IfModule>

# WinNT MPM
# ThreadsPerChild: Konstante Anzahl der Worker-Threads im Server-Prozess
```

```
# MaxRequestsPerChild: Höchstzahl von Anfragen, die ein Server-Prozess
#                      beantwortet
<IfModule mpm_winnt_module>
    ThreadsPerChild      150
    MaxRequestsPerChild    0
</IfModule>
```

**Listing 19.2** Server-Pool-Verwaltung (»httpd-mpm.conf«)

### 19.1.3 Mehrsprachige Fehlermeldungen (httpd-multilang-errordoc.conf)

Die folgende Datei ist nicht nur nützlich für Besucher aus vielen verschiedenen Ländern, sondern demonstriert obendrein auch die Konfiguration der Content-Negotiation anhand sprachabhängiger Fehlermeldungsdateien. Einzelheiten zu den hier verwendeten Direktiven finden Sie in Kapitel 7, »Header und MIME-Einstellungen«.

```
#
# Die unten stehende Konfiguration implementiert mehrsprachige
# Fehlerdokumente durch Content-Negotiation.
#
# Benötigte Module: mod_alias, mod_include, mod_negotiation
#
# Wir verwenden Alias, um jede /error/HTTP_<error>.html.var-Antwort
# zu unserer Sammlung mehrsprachiger Fehlermeldungen hinzuzufügen.
# Wir verwenden Includes, um den entsprechenden Text zu ersetzen.
#
# Sie können das Erscheinungsbild der Nachricht anpassen, ohne eine
# der Standard-HTTP_<error>.html.var-Dateien zu ändern, indem Sie
# folgende Zeile hinzufügen:
#
#   Alias /error/include/ "/Ihr/Include/Pfad/"
#
# Dies ermöglicht es Ihnen, Ihren eigenen Satz von Dateien zu
# erzeugen, indem Sie mit den Dateien in /usr/local/apache2/include/
# beginnen und sie in /Ihren/Include/Pfad/ kopieren, sogar pro
# virtuellem Host. Die Standard-Include-Dateien geben Ihre Apache-
# Versionsnummer und Ihre ServerAdmin-E-Mail-Adresse aus, ungeachtet
# der Einstellung von ServerSignature.

Alias /error/ "/usr/local/apache2/error/"

<Directory "/usr/local/apache2/error">
    AllowOverride None
```

```
    Options IncludesNoExec
    AddOutputFilter Includes html
    AddHandler type-map var
    Require all granted
    # 2.2-Alternative zu Require:
    #Order allow,deny
    #Allow from all
    LanguagePriority en cs de es fr it ja ko nl pl pt-br ro sv tr
    ForceLanguagePriority Prefer Fallback
</Directory>

ErrorDocument 400 /error/HTTP_BAD_REQUEST.html.var
ErrorDocument 401 /error/HTTP_UNAUTHORIZED.html.var
ErrorDocument 403 /error/HTTP_FORBIDDEN.html.var
ErrorDocument 404 /error/HTTP_NOT_FOUND.html.var
ErrorDocument 405 /error/HTTP_METHOD_NOT_ALLOWED.html.var
ErrorDocument 408 /error/HTTP_REQUEST_TIME_OUT.html.var
ErrorDocument 410 /error/HTTP_GONE.html.var
ErrorDocument 411 /error/HTTP_LENGTH_REQUIRED.html.var
ErrorDocument 412 /error/HTTP_PRECONDITION_FAILED.html.var
ErrorDocument 413 /error/HTTP_REQUEST_ENTITY_TOO_LARGE.html.var
ErrorDocument 414 /error/HTTP_REQUEST_URI_TOO_LARGE.html.var
ErrorDocument 415 /error/HTTP_UNSUPPORTED_MEDIA_TYPE.html.var
ErrorDocument 500 /error/HTTP_INTERNAL_SERVER_ERROR.html.var
ErrorDocument 501 /error/HTTP_NOT_IMPLEMENTED.html.var
ErrorDocument 502 /error/HTTP_BAD_GATEWAY.html.var
ErrorDocument 503 /error/HTTP_SERVICE_UNAVAILABLE.html.var
ErrorDocument 506 /error/HTTP_VARIANT_ALSO_VARIES.html.var
```

**Listing 19.3** Sprachabhängige Fehlermeldungen durch Content-Negotiation (»httpd-multilang-errordoc.conf«)

### 19.1.4 Fancy-Index (httpd-autoindex.conf)

In Kapitel 8, »Weiterleitungen und Indizes«, wurde ausführlich beschrieben, wie Sie Apache konfigurieren können, um für Verzeichnisse ohne `DirectoryIndex`-Datei ein automatisches Verzeichnislisting zu generieren. Die folgende Konfigurationsdatei richtet `Fancy`-Indizes mit Icons und optionalen Beschreibungen für die wichtigsten Dateitypen ein.

```
#
# Direktiven, die die Anzeige serverseitig generierter Verzeichnis-
# listings steuern.
#
# Benötigt die Module: mod_autoindex, mod_alias
```

```
#
# Um das Listing eines Verzeichnisses zu sehen, muss die Direktive
# Options für das Verzeichnis "Indexes" enthalten, und das
# Verzeichnis darf keine Datei enthalten, die den in der Direktive
# DirectoryIndex aufgelisteten Dateien entspricht.
#

#
# IndexOptions: Steuert das Erscheinungsbild serverseitig
# generierter Verzeichnislistings.
#
IndexOptions FancyIndexing HTMLTable VersionSort

# Wir fügen das Alias /icons/ für FancyIndex-Verzeichnislistings
# ein. Wenn Sie kein FancyIndexing verwenden, können Sie dies
# auskommentieren.
#
Alias /icons/ "/usr/local/apache2/icons/"

<Directory "/usr/local/apache2/icons">
    Options Indexes MultiViews
    AllowOverride None
    Require all granted
    # 2.2:
    #Order allow,deny
    #Allow from all
</Directory>

#
# AddIcon*-Direktiven teilen dem Server mit, welche Icons für die
# verschiedenen Dateien oder Dateinamenerweiterungen angezeigt
# werden sollen. Sie werden nur für FancyIndex-Verzeichnisse
# angezeigt.
#
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
```

```
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

#
# DefaultIcon ist das Icon, das für Dateien angezeigt werden soll,
# für die nicht explizit ein Icon gesetzt ist.
#
DefaultIcon /icons/unknown.gif

#
# AddDescription ermöglicht es Ihnen, hinter einer Datei in
# serverseitig erzeugten Indizes eine kurze Beschreibung einzufügen.
# Diese wird nur in FancyIndex-Verzeichnissen angezeigt.
# Format: AddDescription "Beschreibung" Dateiname
#
#AddDescription "GZIP compressed document" .gz
#AddDescription "tar archive" .tar
#AddDescription "GZIP compressed tar archive" .tgz

#
# ReadmeName ist der Name der README-Datei, nach der der Server
# standardmäßig sucht und die an Verzeichnislistings angehängt wird.
#
# HeaderName ist der Name einer Datei, die den Verzeichnislistings
# vorangestellt werden soll.
ReadmeName README.html
HeaderName HEADER.html

#
```

```
# IndexIgnore ist ein Satz von Dateinamen, die Verzeichnisindizes
# ignorieren und nicht in das Listing aufnehmen sollen. Shell-
# Wildcards sind erlaubt.
#
IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t
```

**Listing 19.4** Verzeichnis-Indizes mit »Fancy«-Indexing (»httpd-autoindex.conf«)

### 19.1.5 Sprach- und Zeichensatzeinstellungen (httpd-languages.conf)

Die hier gezeigte Datei konfiguriert die Auslieferung von Sprach- und Zeichensatz-Headern (`Content-Language` und `charset`-Feld im `Content-Type`) anhand bestimmter, zusätzlicher Dateierweiterungen, wie in Kapitel 7, »Header und MIME-Einstellungen«, beschrieben.

```
#
# Einstellungen für das Hosting verschiedener Sprachen
#
# Benötigte Module: mod_mime, mod_negotiation

# DefaultLanguage und AddLanguage ermöglichen es Ihnen, die Sprache
# eines Dokuments anzugeben. Sie können dann Content-Negotiation
# einsetzen, um einem Browser ein Dokument in einer Sprache zu
# liefern, die der User verstehen kann.
#

# Eine Standardsprache festlegen. Das bedeutet, dass alle Daten,
# die ohne eine bestimmte Sprachkennzeichnung ausgeliefert werden,
# als diese Sprache markiert werden. Sie sollten dies in der Regel
# NICHT festlegen, wenn Sie nicht sicher sind, dass es in allen
# Fällen korrekt ist.
#
# * Es ist generell besser, eine Seite nicht mit einer bestimmten
# * Sprache zu markieren, als sie mit der falschen Sprache zu
# * markieren!
#
# DefaultLanguage nl
#
# Hinweis 1: Das Suffix muss nicht mit dem Sprach-Schlüsselwort
# identisch sein -- wer Dokumente auf Polnisch hat (dessen Internet-
# Standardsprachcode pl ist), sollte beispielsweise
# "AddLanguage pl .po" verwenden, um Verwechslungen mit dem üblichen
# Suffix für Perl-Skripte zu vermeiden.
#
# Hinweis 2: Die unten stehenden Beispieleinträge demonstrieren,
# dass die zweibuchstabige 'Sprach'-Abkürzung in manchen Fällen
```

```
# nicht identisch mit dem zweibuchstabigen 'Länder'-Code für das
# entsprechende Land ist, z.B. 'Dänemark/dk' und 'Dänisch/da'.
#
# Hinweis 3: Im Fall von 'ltz' verletzen wir das RFC, indem wir
# einen dreibuchstabigen Bezeichner verwenden. Es sind 'Arbeiten im
# Gange', dies zu reparieren und die Referenzdaten für rfc1766
# aufzuräumen.
#
# Dänisch (da) - Deutsch (de) - Englisch (en) - Esperanto (eo)
# Estnisch (et) - Französisch (fr) - Hebräisch (he)
# Italienisch (it) - Japanisch (ja) - Katalanisch (ca)
# Koreanisch (ko) - Kroatisch (hr) - Luxemburgisch* (ltz)
# Neugriechisch (el) - Niederländisch (nl) - Norwegisch Nynorsk (nn)
# Norwegisch (no) - Polnisch (pl) - Portugiesisch (pt)
# Brasilianisches Portugiesisch (pt-BR) - Russisch (ru)
# Schwedisch (sv) - Spanisch (es) - Tschechisch (cs)
# Traditionelles Chinesisch (zh-TW) - Türkisch (tr)
# Vereinfachtes Chinesisch (zh-CN)
#
AddLanguage ca .ca
AddLanguage cs .cz .cs
AddLanguage da .dk
AddLanguage de .de
AddLanguage el .el
AddLanguage en .en
AddLanguage eo .eo
AddLanguage es .es
AddLanguage et .et
AddLanguage fr .fr
AddLanguage he .he
AddLanguage hr .hr
AddLanguage it .it
AddLanguage ja .ja
AddLanguage ko .ko
AddLanguage ltz .ltz
AddLanguage nl .nl
AddLanguage nn .nn
AddLanguage no .no
AddLanguage pl .po
AddLanguage pt .pt
AddLanguage pt-BR .pt-br
AddLanguage ru .ru
AddLanguage sv .sv
AddLanguage tr .tr
```

```
AddLanguage zh-CN .zh-cn
AddLanguage zh-TW .zh-tw

# LanguagePriority ermöglicht es Ihnen, bestimmten Sprachen bei
# einem Unentschieden während der Content-Negotiation Vorrang zu
# geben.
#
# Listen Sie einfach die Sprachen in absteigender Rangfolge auf. Wir
# haben sie hier mehr oder weniger alphabetisch aufgelistet. Sie
# sollten dies wahrscheinlich ändern.
#
LanguagePriority en ca cs da de el eo es et fr he hr it ja ko ltz nl
nn no pl pt pt-BR ru sv tr zh-CN zh-TW

#
# ForceLanguagePriority ermöglicht es Ihnen, eine Ergebnisseite
# anstelle von MULTIPLE CHOICES auszuliefern (Prefer) [im Fall eines
# Unentschieden] oder aber NOT ACCEPTABLE (Fallback) [in dem Fall,
# dass keine akzeptierten Sprachen den verfügbaren Varianten
# entsprechen].
#
ForceLanguagePriority Prefer Fallback

#
# Gängige Dateinamenerweiterungen für Zeichensätze. Sie sollten
# in der Regel Überschneidungen mit den Spracherweiterungen
# vermeiden, es sei denn, Sie haben Ihre Einstellungen für jede
# Sprache sorgfältig getestet.
# Siehe http://www.iana.org/assignments/character-sets für die
# offizielle Liste der Zeichensatznamen und ihrer jeweiligen RFCs.
#
AddCharset us-ascii.ascii .us-ascii
AddCharset ISO-8859-1  .iso8859-1  .latin1
AddCharset ISO-8859-2  .iso8859-2  .latin2 .cen
AddCharset ISO-8859-3  .iso8859-3  .latin3
AddCharset ISO-8859-4  .iso8859-4  .latin4
AddCharset ISO-8859-5  .iso8859-5  .cyr .iso-ru
AddCharset ISO-8859-6  .iso8859-6  .arb .arabic
AddCharset ISO-8859-7  .iso8859-7  .grk .greek
AddCharset ISO-8859-8  .iso8859-8  .heb .hebrew
AddCharset ISO-8859-9  .iso8859-9  .latin5 .trk
AddCharset ISO-8859-10  .iso8859-10  .latin6
AddCharset ISO-8859-13  .iso8859-13
AddCharset ISO-8859-14  .iso8859-14  .latin8
AddCharset ISO-8859-15  .iso8859-15  .latin9
```

```
AddCharset ISO-8859-16  .iso8859-16  .latin10
AddCharset ISO-2022-JP .iso2022-jp .jis
AddCharset ISO-2022-KR .iso2022-kr .kis
AddCharset ISO-2022-CN .iso2022-cn .cis
AddCharset Big5.Big5   .big5 .b5
AddCharset cn-Big5 .cn-big5
# Für Russisch wird mehr als ein Zeichensatz verwendet
# (hängt hauptsächlich vom Client ab):
AddCharset WINDOWS-1251 .cp-1251   .win-1251
AddCharset CP866   .cp866
AddCharset KOI8  .koi8
AddCharset KOI8-E  .koi8-e
AddCharset KOI8-r  .koi8-r .koi8-ru
AddCharset KOI8-U  .koi8-u
AddCharset KOI8-ru .koi8-uk .ua
AddCharset ISO-10646-UCS-2 .ucs2
AddCharset ISO-10646-UCS-4 .ucs4
AddCharset UTF-7   .utf7
AddCharset UTF-8   .utf8
AddCharset UTF-16  .utf16
AddCharset UTF-16BE .utf16be
AddCharset UTF-16LE .utf16le
AddCharset UTF-32  .utf32
AddCharset UTF-32BE .utf32be
AddCharset UTF-32LE .utf32le
AddCharset euc-cn  .euc-cn
AddCharset euc-gb  .euc-gb
AddCharset euc-jp  .euc-jp
AddCharset euc-kr  .euc-kr
AddCharset EUC-TW  .euc-tw
AddCharset gb2312  .gb2312 .gb
AddCharset iso-10646-ucs-2 .ucs-2 .iso-10646-ucs-2
AddCharset iso-10646-ucs-4 .ucs-4 .iso-10646-ucs-4
AddCharset shift_jis   .shift_jis .sjis
```

**Listing 19.5** Sprach- und Zeichensatzeinstellungen (»httpd-languages.conf«)

### 19.1.6 Benutzerverzeichnisse (httpd-userdir.conf)

Auf Mehrbenutzersystemen in Rechenzentren, Universitäten und Unternehmen ist es praktisch, jedem Mitarbeiter eine persönliche Website zu gönnen. Die folgende Datei ermöglicht dies mithilfe von `mod_userdir` unter URLs nach dem Schema *http://Hostname/~username*. Details zu diesem Thema finden Sie in Kapitel 8, »Weiterleitungen und Indizes«.

```
# Einstellungen für User-Home-Verzeichnisse
#
# Benötigtes Modul: mod_userdir

#
# UserDir: der Name des Verzeichnisses, das an das Home-Verzeichnis
# eines Benutzers angehängt wird, wenn eine ~user-Anfrage empfangen
# wird Beachten Sie, dass Sie auch die Standardzugriffskontrolle für
# diese Verzeichnisse setzen müssen wie im Beispiel unten.
#
UserDir public_html

#
# Den Zugriff auf UserDir-Verzeichnisse steuern. Das Folgende ist
# ein Beispiel für eine Site, in der diese Verzeichnisse auf
# Lesezugriffe beschränkt sind.
#
<Directory "/home/*/public_html">
    AllowOverride FileInfo AuthConfig Limit Indexes
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    <Limit GET POST OPTIONS>
        Require all granted
        # 2.2:
        #Order allow,deny
        #Allow from all
    </Limit>
    <LimitExcept GET POST OPTIONS>
        Require all denied
        # 2.2:
        #Order deny,allow
        #Deny from all
    </LimitExcept>
# In 2.4 können die <Limit>- und <LimitExcept>-Blöcke auch
# einfach durch die folgende Zeile ausgedrückt werden:
#Require method GET POST OPTIONS
</Directory>
```

**Listing 19.6** Konfiguration von Benutzerverzeichnissen (»httpd-userdir.conf«)

### 19.1.7 Status- und Anfrageinformationen (httpd-info.conf)

Das Modul `mod_info` liefert zur Laufzeit Informationen über die Server-Konfiguration, während `mod_status` den aktuellen Status des Webservers beschreibt. Die im Folgenden dargestellte Konfigurationsdatei enthält die passenden Einstellungen; Genaueres erfahren Sie in Kapitel 8, »Weiterleitungen und Indizes«.

```
#
# Informationen über die Anfragen, die der Server verarbeitet, und
# über die Konfiguration des Servers erhalten
#
# Benötigte Module: mod_status (für den Handler server-status),
#                   mod_info (für den Handler server-info)

#
# Von mod_status generierte Server-Statusberichte unter der URL
# http://servername/server-status erlauben
# Ändern Sie ".example.com" in Ihre Domain, um dies zu aktivieren.

<Location /server-status>
    SetHandler server-status
    Require host .example.com
    # 2.2:
    #Order deny,allow
    #Deny from all
    #Allow from .example.com
</Location>

#
# ExtendedStatus bestimmt, ob Apache "vollständige"
# Statusinformationen (ExtendedStatus On) oder nur
# Grundinformationen (ExtendedStatus Off) erzeugt, wenn der Handler
# "server-status" aufgerufen wird. Der Standardwert ist Off.
#
#ExtendedStatus On

#
# Remote-Berichte über die Server-Konfiguration unter der URL
# http://servername/server-info (dazu muss mod_info.c geladen sein).
# Ändern Sie ".example.com" in Ihre Domain, um dies zu aktivieren.
#
<Location /server-info>
    SetHandler server-info
    Require host .example.com
    # 2.2:
    #Order deny,allow
    #Deny from all
    #Allow from .example.com
</Location>
```

**Listing 19.7** Informationen über Anfragen und über den Server-Status (»httpd-info.conf«)

### 19.1.8 Virtuelle Hosts (httpd-vhosts.conf)

Diese `Include`-Datei enthält eine Beispielkonfiguration für namensbasierte virtuelle Hosts. Ports, Domain-Namen und andere Einstellungen können leicht angepasst werden; die ausführliche Dokumentation zu diesem Thema finden Sie in Kapitel 12, »Skalierung und Performance-Tuning«.

```
#
# Virtuelle Hosts
#
# Wenn Sie mehrere Domains/Hostnamen auf Ihrem Rechner verwalten
# möchten, können Sie VirtualHost-Container für sie einrichten. Die
# meisten Konfigurationen verwenden nur namensbasierte virtuelle
# Hosts, sodass der Server sich nicht um IP-Adressen zu kümmern
# braucht. Dies wird durch die Sternchen in den unten stehenden
# Direktiven gekennzeichnet.
#
# Bitte lesen Sie die Dokumentation unter
# <URL:http://httpd.apache.org/docs/2.2/vhosts/>
# für weitere Details, bevor Sie versuchen, virtuelle Hosts
# einzurichten.
#
# Sie können die Kommandozeilenoption '-S' verwenden, um die
# Konfiguration Ihrer virtuellen Hosts zu überprüfen.

#
# Namensbasiertes virtuelles Hosting verwenden.
# (ab Version 2.3.11 unnötig/veraltet)
# NameVirtualHost *:80

#
# VirtualHost-Beispiel:
# Fast jede Apache-Direktive kann in einem VirtualHost-Container
# stehen. Der erste VirtualHost-Abschnitt wird für alle Anfragen
# verwendet, die keinem ServerName oder ServerAlias in irgendeinem
# <VirtualHost>-Block entsprechen.
#
#
<VirtualHost *:80>
    ServerAdmin webmaster@dummy-host.example.com
    DocumentRoot "/usr/local/apache2/docs/dummy-host.example.com"
    ServerName dummy-host.example.com
    ServerAlias www.dummy-host.example.com
    ErrorLog "logs/dummy-host.example.com-error_log"
    CustomLog "logs/dummy-host.example.com-access_log" common
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerAdmin webmaster@dummy-host2.example.com
    DocumentRoot "/usr/local/apache2/docs/dummy-host2.example.com"
    ServerName dummy-host2.example.com
    ErrorLog "logs/dummy-host2.example.com-error_log"
    CustomLog "logs/dummy-host2.example.com-access_log" common
</VirtualHost>
```

**Listing 19.8** Beispielkonfiguration für virtuelle Hosts (»httpd-vhosts.conf«)

### 19.1.9 Einbinden der Apache-Dokumentation (httpd-manual.conf)

Apache wird mit einer umfangreichen HTML-Dokumentation geliefert. Um diese im Webspace Ihres Servers anzuzeigen, können Sie die folgende Konfigurationsdatei einbinden. Sie macht unter anderem Gebrauch von `AliasMatch`-Direktiven, die in Kapitel 8, »Weiterleitungen und Indizes«, besprochen werden, sowie von den in Kapitel 7, »Header und MIME-Einstellungen«, beschriebenen Sprachoptionen.

```
#
# Zugriff auf die Dokumentation auf Ihrem Server unter
#  http://yourserver.example.com/manual/
# bereitstellen. Die Dokumentation ist immer verfügbar unter
#  http://httpd.apache.org/docs/2.2/.
#
# Benötigte Module: mod_alias, mod_setenvif, mod_negotiation
#

AliasMatch ^/manual(?:/(?:de|en|es|fr|ja|ko|pt-br|ru|tr))?(/.*)?$ \
           "/usr/local/apache2/manual$1"

<Directory "/usr/local/apache2/manual">
    Options Indexes
    AllowOverride None
    Require all granted
    # 2.2:
    #Order allow,deny
    #Allow from all

    <Files *.html>
        SetHandler type-map
    </Files>
    # .tr ist text/troff in mime.types!
    <Files *.html.tr.utf8>
```

```
        ForceType text/html
    </Files>

    SetEnvIf Request_URI ^/manual/(de|en|es|fr|ja|ko|pt-br|ru|tr)/ \
             prefer-language=$1
    RedirectMatch 301 ^/manual(?:/(de|en|es|fr|ja|ko|
             pt-br|ru|tr))(2,}(/.*)?$ \
                  /manual/$1$2

    LanguagePriority en de es fr ja ko pt-br ru tr
    ForceLanguagePriority Prefer Fallback
</Directory>
```

**Listing 19.9** Einbinden des Apache-Handbuchs in den Webspace (»httpd-manual.conf«)

### 19.1.10 WebDAV-Konfiguration (httpd-dav.conf)

In Kapitel 17, »Apache erweitern«, wurde die WebDAV-Erweiterung mittels `mod_dav` beschrieben. Wenn Sie diese Funktionalität nutzen möchten, können Sie die hier gezeigte Konfigurationsdatei inkludieren.

```
#
# Distributed authoring and versioning (WebDAV)
#
# Benötigte Module: mod_dav, mod_dav_fs, mod_setenvif, mod_alias
#                   mod_auth_digest, mod_authn_file
#

# Das folgende Beispiel erteilt einem Verzeichnis namens "uploads"
# unter dem ServerRoot-Verzeichnis DAV-Schreibzugriff.
#
# Die in httpd.conf angegebene User/Group-Kombination benötigt
# Schreibberechtigungen in dem Verzeichnis, in dem sich DavLockDB
# befindet, sowie in jedem Verzeichnis, in dem "Dav On" eingestellt
# ist.

DavLockDB "/usr/local/apache2/var/DavLock"

Alias /uploads "/usr/local/apache2/uploads"

<Directory "/usr/local/apache2/uploads">
    Dav On

    Require all granted
    # 2.2:
```

```
    #Order Allow,Deny
    #Allow from all

    AuthType Digest
    AuthName DAV-upload

    # Sie können das Programm htdigest verwenden, um die
    # Passwortdatenbank zu erzeugen:
    AuthUserFile "/usr/local/apache2/user.passwd"
    #   htdigest -c "/usr/local/apache2/user.passwd"
    #   DAV-upload admin
    AuthDigestProvider file

    # Universellen Lesezugriff erlauben, aber Schreiben auf den User
    # admin beschränken
    <LimitExcept GET OPTIONS>
        require user admin
    </LimitExcept>
</Directory>

#
# Die folgenden Direktiven verbieten Zugriffe auf Nicht-GET-Anfragen
# für ein Verzeichnis, das den führenden Slash nicht enthält. Dies
# löst ein Problem mit diversen Clients, die Weiterleitungen für
# Ordner mit DAV-Methoden nicht korrekt handhaben.
BrowserMatch "Microsoft Data Access Internet Publishing Provider" \
             redirect-carefully
BrowserMatch "MS FrontPage" redirect-carefully
BrowserMatch "^WebDrive" redirect-carefully
BrowserMatch "^WebDAVFS/1.[0123]" redirect-carefully
BrowserMatch "^gnome-vfs/1.0" redirect-carefully
BrowserMatch "^XML Spy" redirect-carefully
BrowserMatch "^Dreamweaver-WebDAV-SCM1" redirect-carefully
```

**Listing 19.10** WebDAV-Einstellungen (»httpd-dav.conf«)

### 19.1.11 Erweiterte Standardeinstellungen (httpd-default.conf)

Für einige der in Kapitel 6, »Grundkonfiguration«, vorgestellten Direktiven werden oft die Standardwerte beibehalten. Falls Sie sie doch ändern möchten oder müssen, können Sie die folgende Datei als Vorlage verwenden.

```
#
# Diese Konfigurationsdatei enthält Standardeinstellungen für den
# Apache HTTP Server.
```

```
#
# Sie können diese Einstellungen ändern, aber wahrscheinlich
# brauchen Sie dies nicht zu tun.
#

#
# Timeout: die Anzahl von Sekunden, bis Empfangs- und Sende-
#          versuche abgebrochen werden
#
Timeout 300

#
# KeepAlive: ob persistente Verbindungen (oder mehr als eine Anfrage
# pro Verbindung) erlaubt werden oder nicht. Auf "Off" setzen zum
# Deaktivieren.
#
KeepAlive On

#
# MaxKeepAliveRequests: die maximale Anzahl von Anfragen, die
# während einer persistenten Verbindung erlaubt sind. Auf 0 setzen,
# um eine unbegrenzte Anzahl zuzulassen. Wir empfehlen Ihnen einen
# hohen Wert für maximale Performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Anzahl von Sekunden, die auf die nächste Anfrage
# vom selben Client über dieselbe Verbindung gewartet wird.
#
KeepAliveTimeout 5

#
# UseCanonicalName: Bestimmt, wie Apache selbst referenzierte URLs
# sowie die Variablen SERVER_NAME und SERVER_PORT erzeugt.
# Mit dem Wert "Off" verwendet Apache den vom Client angegebenen
# Hostnamen und Port. Bei "On" verwendet Apache den Wert der
# Direktive ServerName.
#
UseCanonicalName Off

#
# AccessFileName: der Name der Datei mit zusätzlichen
# Konfigurationsdirektiven, nach der in jedem Verzeichnis gesucht
# wird. Siehe auch die Direktive AllowOverride.
```

```
#
#
AccessFileName .htaccess

#
# ServerTokens
# Diese Direktive konfiguriert, was als HTTP- Header an den Server
# zurückgegeben wird. Der Standardwert ist 'Full', der Informationen
# über den Betriebssystemtyp und einkompilierte Module sendet.
# Wählen Sie einen der Werte:  Full | OS | Minor | Minimal |
# Major | Prod, wobei Full die meisten Informationen preisgibt und
# Prod die wenigsten.
#
ServerTokens Full

#
# Eine optionale Zeile mit der Server-Version und dem Namen des
# virtuellen Hosts zu serverseitig erzeugten Seiten hinzufügen
# (interne Fehlerdokumente, FTP-Verzeichnislistings, mod_status- und
# mod_info-Ausgabe usw., aber keine CGI-erzeugten Dokumente oder
# benutzerdefinierten Fehlerdokumente).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of:  On | Off | EMail
#
ServerSignature On

#
# HostnameLookups: die Namen von Clients oder nur ihre IP-Adressen
# protokollieren, z.B. www.apache.org (on) oder 204.62.129-132
# (off). Der Standardwert ist off, weil es insgesamt besser für das
# Netz wäre, wenn die Leute dieses Feature bewusst einschalten
# müssen, da seine Aktivierung bedeutet, dass für jede Client-
# Anfrage MINDESTENS eine Lookup-Anfrage an den Nameserver
# fällig wird.
#
HostnameLookups Off
```

**Listing 19.11** Seltener benötigte Apache-Standardeinstellungen (»httpd-default.conf«)

### 19.1.12 Gesicherte Verbindungen (httpd-ssl.conf)

Die Konfiguration von SSL/TLS zur Sicherung der Webkommunikation wurde in Kapitel 10, »Gesicherte Verbindungen«, ausführlich beschrieben. Verwenden Sie die folgende Konfigurationsdatei als Ausgangspunkt für Ihre eigenen SSL-Einstellungen.

```
#
# Dies ist die Apache-Server-Konfigurationsdatei zur Bereitstellung
# der SSL-Unterstützung. Sie enthält die Konfigurationsdirektiven,
# um den Server anzuweisen, wie er Seiten über eine https-Verbindung
# ausliefern soll. Detaillierte Informationen über diese Direktiven
# finden Sie unter
# <URL:http://httpd.apache.org/docs/2.2/mod/mod_ssl.html>
#
# Lesen Sie NICHT einfach nur die hier enthaltenen Anleitungen, ohne
# sie genau zu verstehen. Sie dienen nur als Hinweise oder
# Erinnerungen. Wenn Sie unsicher sind, konsultieren Sie die Online-
# Dokumentation. Wir haben Sie gewarnt.
#

#
# Pseudo-Zufallszahlen-Generator (PRNG):
# Konfiguration einer oder mehrerer Quellen für den PRNG der SSL-
# Bibliothek. Die Seed-Daten sollten eine gute Zufallsqualität
# aufweisen.
# WARNUNG! Auf manchen Plattformen blockiert /dev/random,
# wenn nicht genügend Entropie verfügbar ist. Das bedeutet, dass Sie
# dort nicht das Device /dev/random verwenden können, weil es lange
# Verbindungszeiten verursachen würde (so lange, bis weitere
# Entropie verfügbar ist). Aber üblicherweise stellen diese
# Plattformen zusätzlich das Device /dev/urandom zur Verfügung,
# das nicht blockiert. Falls verfügbar, verwenden Sie also
# stattdessen dieses. Lesen Sie die mod_ssl-Anleitung für
# weitere Details.
#
#SSLRandomSeed startup file:/dev/random  512
#SSLRandomSeed startup file:/dev/urandom 512
#SSLRandomSeed connect file:/dev/random  512
#SSLRandomSeed connect file:/dev/urandom 512


#
# Wenn wir auch SSL bereitstellen, dann müssen wir am
# Standard-HTTP-Port (siehe oben) und am HTTPS-Port lauschen.
#
# Hinweis: Konfigurationen, die IPv6, aber keine IPv4-Mapping-
#          Adressen benötigen, brauchen zwei Listen-Direktiven:
#          "Listen [::]:443" und "Listen 0.0.0.0:443"
#
Listen 443
```

```
##
##  Globaler SSL-Kontext
##
##  Die Konfiguration in diesem Kontext gilt sowohl für den
##  Hauptserver als auch für alle SSL-fähigen virtuellen Hosts.
##

#
#   Einige MIME-Types für den Download von Zertifikaten und CRLs
#
AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl    .crl

#   Passphrasen-Dialog:
#   Den Prozess zur Ermittlung der Passphrase konfigurieren.
#   Das Filter-Dialogprogramm ('builtin' ist ein interner
#   Terminal-Dialog) muss die Passphrase auf stdout bereitstellen.
SSLPassPhraseDialog  builtin

#   Interprozess-Session-Cache:
#   SSL-Session-Cache konfigurieren: zuerst den zu verwendenden
#   Mechanismus und dann das Verfalls-Timeout (in Sekunden).
#SSLSessionCache         "dbm:/usr/local/apache2/logs/ssl_scache"
SSLSessionCache        "shmcb:/usr/local/apache2/logs/ssl_
scache(512000)"
SSLSessionCacheTimeout  300

#   Semaphore:
#   den Pfad zur Locking-Semaphore konfigurieren, die die SSL-
#   Engine intern für die Interprozess-Synchronisation verwendet.
SSLMutex  "file:/usr/local/apache2/logs/ssl_mutex"

##
## Kontext des virtuellen Hosts für SSL
##
<VirtualHost _default_:443>

#   Allgemeine Einstellungen für den virtuellen Host
DocumentRoot "/usr/local/apache2/htdocs"
ServerName www.example.com:443
ServerAdmin you@example.com
ErrorLog "/usr/local/apache2/logs/error_log"
```

```
TransferLog "/usr/local/apache2/logs/access_log"

#   SSLEngine-Schalter:
#   SSL für diesen virtuellen Host ein-/ausschalten
SSLEngine on

#   SSLCipherSuite:
#   Listet die Verschlüsselungen auf, die der Client aushandeln
#   darf. In der mod_ssl-Dokumentation finden Sie die
#   vollständige Liste.
SSLCipherSuite
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

#   Server-Zertifikat:
#   Verweisen Sie mit SSLCertificateFile auf ein PEM-codiertes
#   Zertifikat. Wenn das Zertifikat verschlüsselt ist, werden Sie
#   nach einer Passphrase gefragt. Beachten Sie, dass Sie nach
#   kill -HUP erneut gefragt werden. Beachten Sie, dass Sie sowohl
#   ein RSA- als auch ein DSA-Zertifikat parallel konfigurieren
#   können, wenn Sie beide haben (um auch die Verwendung von
#   DSA-Schlüsseln zu erlauben usw.)
SSLCertificateFile "/usr/local/apache2/conf/server.crt"
#SSLCertificateFile "/usr/local/apache2/conf/server-dsa.crt"

#   Private Key des Servers:
#   Wenn der Schlüssel nicht mit dem Zertifikat kombiniert ist,
#   können Sie diese Direktive verwenden, um auf die Schlüsseldatei
#   zu verweisen. Behalten Sie im Kopf, dass Sie einen RSA- und
#   einen DSA-Private-Key parallel konfigurieren können, wenn Sie
#   beide haben (um auch die Verwendung von DSA-Verschlüsselung
#   zu ermöglichen usw.)
SSLCertificateKeyFile "/usr/local/apache2/conf/server.key"
#SSLCertificateKeyFile "/usr/local/apache2/conf/server-dsa.key"

#   Server-Certificate-Chain:
#   Verweisen Sie mit SSLCertificateChainFile auf eine Datei,
#   die die Verkettung PEM-codierter CA-Zertifikate enthält und so
#   die Certificate-Chain für das Server-Zertifikat bildet.
#   Alternativ kann die Datei dieselbe sein wie SSLCertificateFile,
#   wenn die CA-Zertifikate aus Gründen der Bequemlichkeit direkt
#   mit dem Server-Zertifikat verknüpft sind.
#SSLCertificateChainFile "/usr/local/apache2/conf/server-ca.crt"
```

```
#   Certificate Authority (CA):
#   Setzen Sie den CA-Zertifikat-Verifizierungs-Pfad, in dem CA-
#   Zertifikate für die Client-Authentifizierung zu finden sind,
#   oder alternativ eine große Datei, die sie alle enthält (die
#   Datei muss PEM-codiert sein).
#   Hinweis: Innerhalb von SSLCACertificatePath brauchen Sie
#            Hash-Symlinks, um auf die Zertifikatsdateien zu
#            verweisen. Verwenden Sie das mitgelieferte Makefile, um
#            die Hash-Symlinks nach Änderungen zu aktualisieren.
#SSLCACertificatePath "/usr/local/apache2/conf/ssl.crt"
#SSLCACertificateFile "/usr/local/apache2/conf/ssl.crt/ca-bundle.crt"

#   Certificate Revocation Lists (CRL):
#   Setzen Sie den CA-Revocation-Pfad, in dem CA-CRLs für die
#   Client- Authentifizierung zu finden sind, oder alternativ eine
#   große Datei, die sie alle enthält (die Datei muss PEM-codiert
#   sein).
# Hinweis: Innerhalb von SSLCARevocationPath brauchen Sie Hash-
#          Symlinks, um auf die Zertifikatsdateien zu verweisen.
#          Verwenden Sie das mitgelieferte Makefile, um die Hash-
#          Symlinks nachÄnderungen zu aktualisieren.
#SSLCARevocationPath "/usr/local/apache2/conf/ssl.crl"
#SSLCARevocationFile "/usr/local/apache2/conf/ssl.crl/ca-bundle.crl"

#   Client-Authentifizierung (Typ):
#   Typ und Tiefe (Depth) der Client-Zertifikats-Verifizierung
#   Die Typen sind none, optional, require und optional_no_ca.
#   Die Tiefe ist eine Zahl, die angibt, wie tief die Zertifikats-
#   Ausgabe-Kette untersucht werden soll, bevor die Entscheidung
#   fällt, dass das Zertifikat nicht gültig ist.
#SSLVerifyClient require
#SSLVerifyDepth  10

#   Zugriffskontrolle:
#   Mit SSLRequire können Sie verzeichnisbasierte Zugriffskontrolle
#   auf der Basis beliebig komplexer boolescher Ausdrücke mit
#   Variablenprüfungen und anderen Lookup-Direktiven durchführen.
#   Die Syntax ist eine Mischung aus C und Perl. Siehe die mod_ssl-
#   Dokumentation für weitere Details.
#<Location />
#SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)/ \
#            and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
#            and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
#            and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
```

```
#             and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20       ) \
#           or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
#</Location>


#   SSL-Engine-Optionen:
#   diverse Optionen für die SSL-Engine festlegen.
#   o FakeBasiceAuth:
#     Das X.509 des Clients in eine Basic-Authentifizierung
#     umsetzen. Das bedeutet, dass die Standardauthentifizierungs-/
#     DBM-Methoden für die Zugriffskontrolle genutzt werden können.
#     Der Benutzername ist die einzeilige Version des Client-X.509-
#     Zertifikats. Beachten Sie, dass vom User kein Passwort erfragt
#     wird. Jeder Eintrag in der Userdatei benötigt dieses Passwort:
#     `xxj31ZMTZzkVA'.
#   o ExportCertData:
#     Dies exportiert zwei zusätzliche Umgebungsvariablen:
#     SSL_CLIENT_CERT und SSL_SERVER_CERT. Diese enthalten die
#     PEM-codierten Zertifikate des Servers (existiert immer) und
#     des Clients (existiert nur, wenn Client-Authentifizierung zum
#     Einsatz kommt). Diese können verwendet werden, um die
#     Zertifikate in CGI-Skripte zu importieren.
#   o StdEnvVars:
#     Dies exportiert die Standard-'SSL_*'-Umgebungsvariablen
#     für SSL/TLS. Standardmäßig ist dieser Export aus
#     Performance-Gründen deaktiviert, weil ihre Extraktion eine
#     rechenintensive Operation ist und weil sie für die
#     Auslieferung statischer Inhalte im Allgemeinen nutzlos ist.
#     Deshalb aktiviert man den Export normalerweise nur für CGI-
#     und SSI-Anfragen.
#   o StrictRequire:
#     Dies verweigert den Zugriff, wenn "SSLRequireSSL" oder
#     "SSLRequire" verwendet wird, selbst in einer "Satisfy-any"-
#     Situation, d.h., der Zugriff wird verweigert, und kein anderes
#     Modul kann dies ändern.
#   o OptRenegotiate:
#     Dies ermöglicht die Aushandlung einer optimierten SSL-
#     Verbindung, wenn SSL-Direktiven in Verzeichniskontexten
#     verwendet werden.
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
<FilesMatch "\.(cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory "/usr/local/apache2/cgi-bin">
```

```
    SSLOptions +StdEnvVars
</Directory>

#   SSL-Protokollanpassungen:
#   Der sichere und übliche, aber noch dem SSL/TLS-Standard
#   genügende Shutdown-Ansatz besteht darin, dass mod_ssl einen
#   Close-Notify-Alarm sendet, aber nicht auf den Close-Notify-Alarm
#   eines Clients wartet. Wenn Sie einen anderen Shutdown-Ansatz
#   benötigen, können Sie eine der folgenden Variablen verwenden:
#   o ssl-unclean-shutdown:
#     Dies erzwingt einen unsauberen Shutdown, wenn die Verbindung
#     geschlossen wird, d.h., es wird kein SSL-Close-Notify-Alarm
#     gesendet oder empfangen. Dies verletzt den SSL/TLS-Standard,
#     wird aber für einige hirntote Browser benötigt. Verwenden Sie
#     dies, wenn Sie beim Standardansatz (bei dem mod_ssl den
#     Close-Notify-Alarm sendet) I/O-Fehler erhalten.
#   o ssl-accurate-shutdown:
#     Dies erzwingt einen sauberen Shutdown, wenn die Verbindung
#     geschlossen wird, d.h., es wird ein SSL-Close-Notify-Alarm
#     gesendet, und mod_ssl wartet auf den Close-Notify-Alarm des
#     Clients. Dies ist 100% SSL/TLS-standardkonform, verursacht in
#     der Praxis aber häufig hängende Verbindungen bei hirntoten
#     Browsern. Verwenden Sie dies nur für Browser, bei denen Sie
#     wissen, dass ihre SSL-Implementierung korrekt funktioniert.
#   Hinweis: Die meisten Probleme mit defekten Clients haben auch
#   mit der Keepalive-Fähigkeit von HTTP zu tun, sodass Sie für
#   diese Clients normalerweise auch Keepalive deaktivieren sollten.
#   Verwenden Sie dafür die Variable "nokeepalive". Entsprechend
#   muss man einige Clients zwingen, HTTP/1.0 zu verwenden, um ihre
#   defekte HTTP/1.1-Implementierung zu umgehen. Verwenden Sie dafür
#   die Variablen "downgrade-1.0" und "force-response-1.0".
BrowserMatch ".*MSIE.*" \
         nokeepalive ssl-unclean-shutdown \
         downgrade-1.0 force-response-1.0

#   Logging pro Server:
#   Der Pfad einer angepassten SSL-Log-Datei. Verwenden Sie dies,
#   wenn Sie eine kompakte Nicht-Fehler-SSL-Log-Datei für den
#   virtuellen Host benötigen.
CustomLog "/usr/local/apache2/logs/ssl_request_log" \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>
```

**Listing 19.12** SSL/TLS-Konfiguration (»httpd-ssl.conf«)

## 19.2 Zusätzliche Konfigurationsdateien

Einige der in diesem Buch beschriebenen Fähigkeiten von Apache kommen in den Standardkonfigurationsdateien nicht vor. In den folgenden Abschnitten finden Sie deshalb einige ebenfalls kommentierte Dateien zu den Themen:

- Authentifizierung
- Forward-Proxy mit Caching
- PHP als Apache-Modul
- URL-Rewriting

### 19.2.1 Authentifizierung

Die im Folgenden vorgestellte Datei stellt Authentifizierungseinstellungen über beide Kommunikationsverfahren (Basic und Digest) mit diversen Provider-Modulen bereit. Die in der virtuellen Maschine konfigurierten Usernamen und Passwörter stehen in den Kommentaren. Einzelheiten über die Authentifizierungsmodule finden Sie in Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«.

```
#
# Authentifizierungsbeispiele
#
# Apache wird mit einer Reihe von Modulen zur Authentifizierung
# geliefert. Die Organisation der Authentifizierungsmodule
# wurde in Version 2.2 umgestellt: Bis 2.0 musste jedes einzelne
# Authentifizierungsmodul selbst mit dem Client kommunizieren,
# um Benutzernamen und Passwörter zu erfragen, und gleichzeitig
# den Speicherort der Vergleichsdaten verwalten (z.B. Textdateien,
# DBM-Dateien, LDAP-Verzeichnis usw.).
# Seit 2.2 übernehmen zwei Module allein die Kommunikation mit
# den Clients:
# * mod_auth_basic: Übertragung der Anmeldedaten im Klartext
# * mod_auth_digest: Verschlüsselte Übertragung der Anmeldedaten
#                    (alte Browser können dies nicht)
# Hinzu kommen diverse Provider-Module zur Verwaltung der
# Speicherorte. Diese werden mithilfe der Direktiven
# AuthBasicProvider beziehungsweise AuthDigestProvider angesteuert.
# Beim Zugriff auf eine geschützte Ressource sendet Apache dem
# Client eine Antwort mit dem Status 401 Unauthorized und dem
# Header WWW-Authenticate. Daraufhin zeigt der Client dem User
# einen Anmeldedialog, in den Benutzername und Kennwort eingegeben
# werden. Der Client sendet diese Daten als Authorization-Header
# an den Server. Für den Rest der Sitzung merkt der Client sich
```

```
# die Daten und schickt sie bei jedem Zugriff auf die Ressource
# automatisch, sodass das Verfahren für den User wie ein Login
# aussieht.
#

# Basic-Authentifzierung; Speicherung in Textdateien
# Benötigte Module: mod_auth_basic, mod_authn_file
#
<Directory /usr/local/apache2/htdocs/basic-text>
# AuthType: Gibt den Authentifizierungstyp Basic oder Digest an.
AuthType basic
# AuthBasicProvider: Provider-Modul für Zugriff auf den
#  Speicher für die Vergleichsdaten. file steht für Textdateien,
#  die mit dem Dienstprogramm htpasswd angelegt werden können.
AuthBasicProvider file
# AuthName: Legt den Realm (Anmeldebereich) fest; gleichzeitig
#  wird der hier eingegebene Text im Anmeldedialog angezeigt
AuthName "Textbasierte Basic-Authentifizierung"
# AuthUserFile: Pfad der Passwortdatei. Entweder absolut
#  oder relativ zur ServerRoot.
#  Benutzername: textbasic, Passwort: basic
AuthUserFile auth/.htplaintext-auth
# Require: Welche User aus der Passwortdatei dürfen sich anmelden?
#  - user Username [Username ...] -- einzelne User
#  - group Gruppenname [Gruppenname ...] -- Gruppen
#  - valid-user -- alle User aus der Datei
Require user textbasic
# Satisfy (nur 2.2): Legt fest, ob Host-Beschränkungen mit
#  Order/Allow/Deny alternativ oder zusätzlich gelten.
#  - all: Alle Bedingungen müssen erfüllt sein.
#  - any: Mindestens eine Bedingung muss erfüllt sein.
#Satisfy all
# In 2.4 werden stattdessen <RequireAll>/<RequireAny>-
# Container verwendet.
</Directory>

# Digest-Authentifzierung; Speicherung in Textdateien
# Benötigte Module: mod_auth_digest, mod_authn_file
#
<Directory /usr/local/apache2/htdocs/digest-text>
# AuthType: Gibt den Authentifizierungstyp Basic oder Digest an.
AuthType digest
# AuthDigestProvider: Provider-Modul
#  Benutzername: textdigest, Passwort: digest
AuthDigestProvider file
```

```
# AuthName: Legt den Realm (Anmeldebereich) fest.
AuthName "Textbasierte Digest-Authentifizierung"
# AuthUserFile: Pfad der Passwortdatei, mit htdigest erzeugt
AuthUserFile auth/.htdigest-auth
# Require: Welche User aus der Passwortdatei dürfen sich anmelden?
Require user textdigest
# Satisfy (nur 2.2): Legt fest, ob Host-Beschränkungen mit
#  Order/Allow/Deny alternativ oder zusätzlich gelten.
#Satisfy all
</Directory>

# Basic-Authentifzierung; Speicherung in DBM-Dateien
# (datenbankartige Binärdateien; bei vielen Einträgen
#  performanter als Textdateien)
# Benötigte Module: mod_auth_digest, mod_authn_dbm
#
<Directory /usr/local/apache2/htdocs/basic-dbm>
# AuthType: Gibt den Authentifizierungstyp Basic oder Digest an.
AuthType basic
# AuthBasicProvider: Provider-Modul
AuthDigestProvider dbm
# AuthName: Legt den Realm (Anmeldebereich) fest.
AuthName "DBM-basierte Basic-Authentifizierung"
# AuthUserFile: Pfad der Passwortdatei; mit htdbm erzeugt
#  Benutzername: dbmbasic, Passwort: basic
AuthUserFile auth/.htdbm-auth
# Require: Welche User aus der Passwortdatei dürfen sich anmelden?
Require user dbmbasic
# Satisfy (nur 2.2): Legt fest, ob Host-Beschränkungen mit
#  Order/Allow/Deny alternativ oder zusätzlich gelten.
Satisfy all
</Directory>Listing
```

**Listing 19.13** 19.13: Authentfizierungsbeispiele, httpd-auth.conf

### 19.2.2 Forward-Proxy mit Caching

Passend zu Kapitel 13, »Proxy- und Cache-Funktionen«, konfiguriert die hier gezeigte Datei Apache als Forward-Proxy auf Port 3128, der auch Caching einsetzt.

```
#
# Forward-Proxy mit Caching
#
# Mit dieser Konfigurationsdatei kann Apache als Webproxy für
# Rechner im eigenen Netzwerk verwendet werden, die nicht über
```

```
# eine eigene Internetverbindung verfügen.
# Zur Beschleunigung der Anfragen werden alle einmal geladenen
# Seiten im Cache gespeichert.
#
# Benötigte Module: mod_proxy, mod_proxy_http, mod_cache,
#                   mod_cache_disk (2.2: mod_disk_cache)

# Zusätzlicher Port, an dem Apache auf Proxy-Anfragen lauschen
# soll (3128 ist der Standardport des beliebten Proxys squid)
Listen 3128

# Virtueller Host für Proxy-Port 3128 mit Cache
<VirtualHost _default_:3128>
  # ProxyRequests: Proxy-Funktionalität einschalten
  ProxyRequests On
  # ProxyVia: einen Via-Header generieren, der alle passierten
  # Proxys auflistet
  ProxyVia On
  # CacheEnable: Auswahl des Cache-Providers (disk oder memory)
  #  Hier wird disk verwendet; benötigt mod_disk_cache.
  CacheEnable disk
  # CacheRoot: Wurzelverzeichnis des Caches
  CacheRoot /usr/local/apache2/cache
  # CacheSize: Maximalgröße des Caches in Kilobyte, hier 128 MByte
  CacheSize 131072

  # <Proxy>: Container für Proxy-Anfragen
  # Warnung! Es ist sehr wichtig, den Zugriff auf den Proxy zu
  # beschränken, da er sonst als anonymer Proxy überall im
  # Internet verfügbar ist, sodass Angreifer ihn als Zwischen-
  # station zur Verschleierung ihrer Herkunft nutzen können.
  <Proxy *>
    # Passen Sie diese Werte an Ihr eigenes Netzwerk an.
    Require ip 192.168.0 127.0.0.1
    # 2.2:
    #Order deny,allow
    #Deny from all
    #Allow from 192.168.0 127.0.0.1
  </Proxy>

</VirtualHost>
```

**Listing 19.14** Proxy- und Cache-Konfiguration, httpd-proxy.conf

### 19.2.3 PHP

Hier sehen Sie eine kurze Konfigurationsdatei, die es ermöglicht, PHP (siehe Kapitel 15, »Technologien zur Webprogrammierung«) als Apache-Modul zu betreiben.

```
#
# PHP-Konfiguration
#
# Dies ist keine separate Konfigurationsdatei, sondern ein
# Auszug aus der httpd.conf der virtuellen Maschine --
# alternativ lassen diese Direktiven sich natürlich auch in
# eine Include-Datei verlegen.
#
# Laden des PHP-Moduls - diese Zeile wird automatisch eingefügt,
# wenn PHP mit der Option --apxs2=/Pfad/zu/apxs kompiliert wird,
# d.h. als Apache-Modul
LoadModule php5_module modules/libphp5.so
#
# Anpassen von DirectoryIndex auf index.html und index.php
DirectoryIndex index.html index.php
#
# Den Handler "php5-script" für alle Dateien mit der Endung .php
# konfigurieren
AddHandler php5-script .php
# Alternative (klassische) Methode:
# AddType application/x-httpd-php .php
#
# Ein praktischer Helfer für PHP-Entwickler:
# Ansicht des PHP-Sourcecodes mit Syntax-Highlighting
# Achtung! Diese Ansicht *muss* strikt vor der Außenwelt verborgen
# werden, da PHP-Skripte Datenbankpasswörter und andere sensible
# Informationen enthalten.
# Hier wird der Alias /view-php für die DocumentRoot gesetzt.
# Der Zugriff ist auf localhost beschränkt, und gleichzeitig
# muss eine Authentifizierung mit dem Usernamen php und dem
# Passwort phpsource erfolgen. Beides können Sie an Ihre Bedürfnisse
# anpassen, aber von weniger Sicherheit ist *strikt* abzuraten.
# Auf einem Produktiv-Server, der Inhalte im Internet
veröffentlicht,
# sollten Sie diese Option ganz weglassen.
Alias /view-php /usr/local/apache2/htdocs

<Location /view-php>
  # Zugriff auf localhost beschränken (so nur 2.2)
  #Order deny,allow
```

```
  #Deny from all
  #Allow from 127.0.0.1
  # Authentifizierung - Username php, Passwort phpsource
  AuthType basic
  AuthBasicProvider file
  AuthName "PHP Source View"
  AuthUserFile auth/.htphp
  # Alle Zugangsvoraussetzungen (2.4):
  <RequireAll>
    Require user php
    Require local
  </RequireAll>
  # Zugangsvoraussetzung (2.2):
  #Require user php
  # Sicherstellen, dass Host-Beschränkung *und* Authentifizierung
  # gleichzeitig zutreffen müssen (so nur 2.2)
  #Satisfy all
  # Konfiguration des PHP-Source-Handlers für die Endung .php
  AddHandler php5-source .php
  # Ein Zugriff auf http://mydomain/test.php führt das PHP-Skript
  # aus, während http://mydomain/view-php/test.php nach Anmeldung
  # den Quellcode desselben Skripts anzeigt.
</Location>
```

**Listing 19.15** PHP-Konfiguration – Auszug aus der httpd.conf der virtuellen Maschine

### 19.2.4 mod_rewrite – papaya CMS

Dies ist die (von mir durch ausführliche Kommentare erweiterte) *.htaccess*-Datei des PHP-basierten Open-Source-Content-Management-Systems *papaya CMS*, an dem ich seit April 2007 hauptberuflich mitarbeite. Neben dem hier ausgiebig genutzten Rewriting zeigt Ihnen das entsprechende Beispiel in der virtuellen Maschine auch, wie Sie ein komplexes, fertiges Softwarepaket mit Ihrem Apache verbinden können.

```
#
# .htaccess von papaya CMS 5.x
#
# Die Datei ist in der vorliegenden Form für die Document-
# Root einer Website gedacht. Für den papaya-Betrieb in
# einem Unterverzeichnis müssen die RewriteRules ent-
# sprechend angepasst werden.
# Selbstverständlich kann der Inhalt der Datei auch in die
# httpd.conf oder eine Include-Datei übernommen werden, was
```

```
# ein performance- und sicherheitsfreundliches 'AllowOverride
# none' erlaubt.
#
# papaya verwendet mod_rewrite zur Erstellung suchmaschinen-
# freundlicher URLs. Eine URL im papaya-Frontend (Seite, die
# der Besucher sieht) hat nach außen folgendes Format:
# <Seitenname>[.<Abschnitt>].<Seite>[.<Sprachkürzel>].<Typ>
# Wenn Sie im Backend (Verwaltungsebene) eingeloggt sind,
# kann .preview angefügt werden, um eine Vorschau noch nicht
# veröffentlichter Seiten zu erhalten.
# Anstelle von <Seitenname> ist immer auch 'index' möglich.
# <Abschnitt> und <Seite> sind ganze Zahlen, die beim Anlegen
# der Seiten und Abschnitte als Datenbankschlüssel erstellt
# werden. <Typ> ist eine Dateiendung, die den Ausgabefilter
# angibt. Der Typ .xml ist immer verfügbar (allerdings nur,
# wenn Sie im Backend eingeloggt sind), da alle papaya-Module
# zunächst XML ausgeben; alle anderen Ausgabemodi werden mit
# hilfe von XSLT-Templates erstellt.
#
# Beispiele:
# - uebersicht.123.html
# - bestellung.3.99.de.html.preview
# - index.127.en.pdf
# - index.12.xml

# Rewrite-Funktionalität einschalten
RewriteEngine On

# Die PHP-Funktionalität von papaya generiert eine Session-ID;
# diese RewriteRule sorgt dafür, dass diese nicht angezeigt wird.
RewriteRule ^/?sid[a-z]*([a-zA-Z0-9,-]{20,40})(/.*) $2 [QSA]

# Zugriff auf das papaya-Backend unter dem URL-Pfad /papaya
# Die RewriteConds sorgen dafür, dass die RewriteRule nur dann
# gilt, wenn die Ressource, auf die zugegriffen wird, kein
# existierendes Verzeichnis (!-d) und keine existierende Datei
# (!-f) ist.
# Konkret geht es hier um die Verwaltung der papaya-Module;
# die Seitennamen werden mit dem Präfix module_ versehen und
# intern an module.php weitergeleitet.
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^/?papaya/module\_([a-z\d\_]+)\.[a-z]{3,4}
```

```
    /papaya/module.php?p_module=$1 [QSA,L]

# Zugriff auf Mediendateien in der papaya-Media-DB
RewriteRule ^/?[^./]*\.(thumb|media)\.((.).*) -
    [E=mediaFile:/papaya-files/$3/$2]
RewriteCond %{DOCUMENT_ROOT}%{ENV:mediaFile} -f
RewriteRule ^/?[^./]*\.(thumb|media)\.((.).*) /papaya-files/$3/$2 [L]

RewriteRule ^/?([a-fA-F0-9]/)*[a-zA-Z0-9_-]+\.
    (media|thumb|download|popup|image)(\.(preview))?
        ((\.([a-zA-Z0-9_]+))?(\.[a-zA-Z0-9_]+))  /index.php [QSA,L]

# Auflösung der Seiten-URLs -- zunächst mit dem Seitennamen,
# anschließend mit 'index'. Auch dies geschieht nur, wenn nicht
# auf eine existierende Ressource zugegriffen wird.
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^/?[a-zA-Z0-9_-]+((\.[0-9]+)?\.[0-9]+)
    ((\.[a-z]{2,5})?\.[a-z]+)((\.[0-9]+)?.preview)? /index.php [QSA,L]
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^/?index((\.[a-z]{2,5})?\.[a-z]+)((\.[0-9]+)?.preview)?
    /index.php [QSA,L]

# Für den Fehler 404 (Seite nicht gefunden) ist die Hauptdatei
# index.php zuständig, die sich intern um die Fehlerbehandlung
# kümmert.
ErrorDocument 404 /index.php

# Standardgültigkeitsdauer einer Seite: 1 Monat seit Zugriff
# Im Backend kann für jede einzelne Seite ein eigener Wert
# festgelegt werden.
<IfModule mod_expires.c>
  ExpiresDefault "access plus 1 month"
  ExpiresActive on
</IfModule>

# Es wird kein ETag erzeugt.
FileETag none
```

**Listing 19.16** .htaccess-Datei von papaya CMS

## 19.3 Zusammenfassung

In diesem Kapitel haben Sie zahlreiche Beispiele für vollständige Apache-Konfigurationsdateien gesehen. Lesen Sie in Ruhe die Kommentare, aber zögern Sie auch nicht, die virtuelle Maschine von der Buch-DVD zu starten und damit zu experimentieren. In diesem Buch finden Sie genügend Anregungen für Erweiterungen und Änderungen.

Damit ist nicht nur das Kapitel, sondern auch der Hauptteil des Buches abgeschlossen – es folgen noch einige Anhänge mit nützlichen Informationen zum Nachschlagen. Ich hoffe, dass das Buch Ihnen nicht nur Nutzen für Ihre tägliche Administrations- und Entwicklungsarbeit mit Apache 2 gebracht, sondern auch ein wenig Spaß gemacht hat. Ich freue mich stets über Anregungen, Fragen und (konstruktive) Kritik; zögern Sie nicht, die Site zum Buch unter *http://buecher.lingoworld.de/apache2* zu besuchen.

# Anhang


| | | |
|---|---|---|
| **A** | **Besonderheiten von Apache 2.0** | 891 |
| **B** | **Kurzreferenz der Konfigurationsdirektiven** | 903 |
| **C** | **Sonstige Tabellen** | 935 |
| **D** | **Die Apache-Lizenz 2.0** | 977 |
| **E** | **Reguläre Ausdrücke** | 983 |
| **F** | **VMware Workstation** | 985 |
| **G** | **Rechtliche Aspekte** | 991 |
| **H** | **Literatur** | 995 |

*»Sich selbst darf man nicht für so göttlich halten, dass man seine eigenen Werke nicht gelegentlich verbessern könnte.«*
*– Ludwig van Beethoven*

# A Besonderheiten von Apache 2.0

Vereinzelt sind noch Webserver mit Releases der Apache-Version 2.0 im Einsatz. Diese Version wird in Kürze mit dem finalen Bugfix-Release 2.0.65 in Rente geschickt, ähnlich, wie es bereits 2010 mit Version 1.3 geschah. In diesem Anhang werden die speziellen Eigenschaften von Apache 2.0 beschrieben, denn wenn Sie die Version noch verwenden, sollten Sie schleunigst auf 2.2 oder 2.4 migrieren, und dafür sollten Sie die Unterschiede kennen.

## A.1 Zusätzliche Multiprocessing-Module

In Apache 2.0 wurde das MPM-Konzept erstmals eingeführt, sodass viel experimentiert wurde. Deshalb gibt es in 2.0 einige Multiprocessing-Module, die in 2.2 nicht mehr unterstützt werden. Bitte beachten Sie, dass diese MPMs hochgradig experimentell sind und möglicherweise nicht richtig funktionieren – für einen Produktivserver sind sie keinesfalls geeignet. Im Einzelnen handelt es sich um folgende:

- `leader`
  Dies ist eine experimentelle Variante von `worker`; für einen Produktionsserver sollte sie nicht verwendet werden. Bei diesem Modell wird ein Thread zum »Leader«, die anderen sind »Followers«. Dies ermöglicht eine bessere Synchronisation der Threads.
- `threadpool`
  Eine weitere experimentelle Variante von `worker`, die noch nicht für den Produktiveinsatz zu empfehlen ist. Sie basiert auf der Wiederverwendung unbeschäftigter Threads.
- `perchild`
  Bei diesem Modul erzeugt eine festgelegte Anzahl von Prozessen variable Mengen von Prozessen. Die Besonderheit dieses Laufzeitmodells besteht

darin, dass jeder der Prozesse unter einer anderen User-ID ausgeführt werden kann. Dies erhöht die Sicherheit für Webanwendungen oder beim virtuellen Hosting.

## A.2 Unterschiede bei Modulen

Einige Module hatten in Apache 2.0 einen anderen Namen als in 2.2, andere waren noch gar nicht verfügbar. In Tabelle B.1 sehen Sie eine Übersicht über diejenigen Module, bei denen sich etwas geändert hat.

| Modul in Apache 2.2 | Entsprechung in Apache 2.0 |
|---|---|
| `mod_auth_basic` | Die Klartextkommunikation mit dem Client ist Bestandteil verschiedener `auth*`-Module, z. B. `mod_auth` oder `mod_auth_dbm`. |
| `mod_authn_*` | Die Authentifizierungs-Provider-Module stehen nicht zur Verfügung; stattdessen gibt es monolithische `mod_auth_*`-Module wie `mod_auth_dbm` oder `mod_auth_ldap`. |
| `mod_authz_host` | `mod_access` |
| `mod_authz_owner` | nicht verfügbar |
| andere `mod_authz_*`-Module | in `mod_auth_*` integriert |
| `mod_dav_lock` | nicht verfügbar |
| `mod_dbd`, `mod_authn_dbd` | nicht verfügbar |
| `mod_ident` | Funktionalität im Core integriert |
| `mod_imagemap` | `mod_imap` |
| `mod_proxy_ajp` | nicht verfügbar |
| `mod_proxy_balancer` | nicht verfügbar |
| `mod_substitute` | nicht verfügbar |

**Tabelle A.1** Unterschiede zwischen den Modulen in Apache 2.2 und 2.0

## A.3 Besonderheiten bei der Authentifizierung

Die umfassendste Änderung von Apache 2.0 auf 2.2 betrifft die Neuorganisation der Authentifizierung. In 2.0 gibt es keine Provider-Module (und damit auch nicht die Direktiven `AuthBasicProvider` und `AuthDigestProvider`). Stattdessen muss sich jedes Authentifizierungsmodul selbst sowohl um die Kommunikation mit dem Client (Aufforderung zur Anmeldung und Entgegennahme der Anmeldedaten) kümmern als auch um die Überprüfung dieser Daten anhand der gespeicherten Informationen.

### A.3.1 Die Organisation der Authentifizierung in Apache 2.0

Mit der normalen Apache-2.0-Distribution werden folgende Authentifizierungsmodule ausgeliefert:

- `mod_auth`
  Dies ist die einfachste Form der Authentifizierung. Die Benutzernamen und Passwörter werden in einfachen Textdateien gespeichert.
- `mod_auth_digest`
  Mithilfe dieses Moduls erfolgt die Übertragung der Anmeldeinformationen nicht im Klartext, sondern im MD5-Digest-Format. Dies verbessert die Sicherheit der Anmeldung, da niemand unterwegs das Passwort stehlen kann.
- `mod_auth_dbm`
  Dieses Modul sorgt dafür, dass die (Basic-)Anmeldeinformationen in DBM-Dateien gespeichert werden. Diese sind schneller und zuverlässiger als Textdateien.
- `mod_auth_ldap`
  Dieses Modul überprüft die Basic-Authentifizierung mithilfe der Informationen eines LDAP-Servers. Dies ist auf UNIX-Systemen in der Regel OpenLDAP. In einem Windows-Netzwerk kann aber natürlich auch Microsoft Active Directory verwendet werden, das ebenfalls LDAP-kompatibel ist.
- `mod_auth_anon`
  Dieses Modul ermöglicht eine anonyme Anmeldung, wie sie von Anonymous FTP bekannt ist: Als Benutzername wird `anonymous` angegeben, als Passwort die eigene E-Mail-Adresse. In den meisten Fällen ist dieses Verfahren für einen Webserver nicht sonderlich nützlich.

Neben diesen mit Apache gelieferten Modulen stellen Authentifizierungsmodule auch eine der größten Gruppen unter den Drittanbieter-Modulen dar. Es gibt Module zur Authentifizierung über beinahe jede Datenbank und jeden sonstigen Anmelde-Server oder Verzeichnisdienst.

Abbildung B.1 zeigt beispielhaft das Organisationsschema der Authentifizierungsmodule in Apache 2.0.

**Achtung!**

Apache 2.0.51 enthielt in Bezug auf `Satisfy` eine gravierende Sicherheitslücke, die die Wirkung der Einstellung `All` unter gewissen Umständen aufhebt. Falls Sie diese Version noch betreiben, sollten Sie sie dringend aktualisieren.

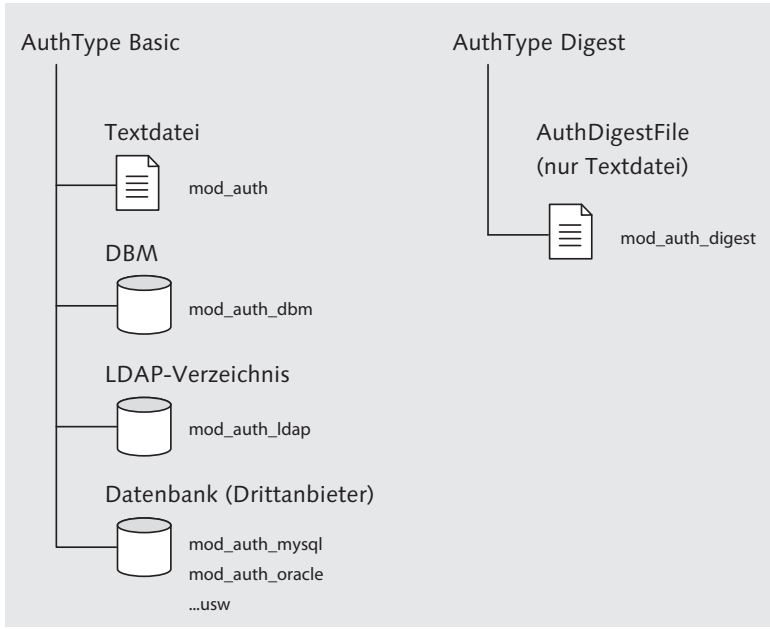



**Abbildung A.1** Organisation der Authentifizierungsmodule in Apache 2.0

## A.3.2 Authentifizierungsdirektiven in Apache 2.0

Die meisten Direktiven zur Authentifizierung sind in Apache 2.2 noch immer verfügbar und wurden in Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«, dokumentiert. In vielen Fällen sind sie in andere Module gewechselt; auch dies wurde in Kapitel 9 vermerkt. Im Folgenden finden Sie daher nur die Dokumentation derjenigen Direktiven, die ausschließlich in Apache 2.0 vorkommen – entweder, weil sie in der neuen Provider-Architektur keinen Sinn mehr ergeben, oder weil ihre Funktionalität durch andere Direktiven übernommen wurde.

### AuthAuthoritative

Bestimmt, ob Authentifizierungsdaten an andere Module weitergereicht werden.

| | |
|---|---|
| **Version** | nur 2.0.x |
| **Modul** | `mod_auth` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthAuthoritative On\|Off` |
| **Standardwert** | `On` |

Wenn Sie mehrere Authentifizierungsmodule einsetzen, könnte es wünschenswert sein, die Authentifizierungsanforderung an ein anderes Modul weiterzuleiten. Standardmäßig ist jedes einzelne `auth*`-Modul *autoritativ* – wenn das Modul die übermittelten Anmeldedaten nicht anerkennt, wird sofort wieder `401 Unauthorized` an den Client gesendet. Sie können explizit verlangen, dass `mod_auth` die Daten des Anmeldeversuchs an andere Module weiterreichen soll, falls die Anmeldung nicht bestätigt werden kann. Dazu muss für die entsprechende Ressource folgende Anweisung gesetzt werden:

```
AuthAuthoritative Off
```

Aus Sicherheitsgründen sollten Sie diese Direktive nur setzen, wenn es einen guten Grund dafür gibt und Sie genau wissen, was Sie tun. In aller Regel ist diese Einstellung nicht erforderlich.

**Das Problem der Reihenfolge**

Ein Problem mit sämtlichen `Auth*Authoritative`-Direktiven in Apache 2.0 besteht darin, dass es in dieser Version keine verbindliche Reihenfolge gibt, in der die Funktionen der diversen Authentifizierungsmodule ausgeführt werden. Für den Aufruf von Modulfunktionen sind die sogenannten *Hooks* zuständig, eine Art registrierbarer Callback-Funktionen. Die gesamte Authentifizierung wird in derselben Phase der Anfrageverarbeitung durchgeführt, sodass dafür nur ein Hook zur Verfügung steht. Die Konsequenz ist, dass die Reihenfolge weitgehend dem Zufall überlassen bleibt.

### AuthDigestFile

Name der Datei mit Benutzernamen und verschlüsselten Passwörtern

| | |
|---|---|
| **Version** | nur bis 2.0.x |
| **Modul** | `mod_auth_digest` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthDigestFile Dateipfad` |
| **Standardwert** | nicht gesetzt |

Mit dieser Direktive wird in Apache 2.0 die Datei angegeben, in der sich die Benutzerdaten für die Digest-Anmeldung befinden (in Apache 2.2 wird diese Funktionalität durch `AuthUserFile` übernommen). Sie wird mithilfe des soeben beschriebenen Tools `htdigest` erstellt. Wie üblich gilt eine relative Pfadangabe als relativ zur `ServerRoot`. Beispiel:

```
AuthDigestFile /usr/local/apache2/misc/.htdigest
```

Hier ein vollständiges Beispiel für den Verzeichnisschutz mit Digest-Authentifizierung; die beiden User, die sich anmelden dürfen, sind `htadmin` und `htmaster`:

```
<Directory /usr/local/apache2/htdocs/topsecret>
   AuthType Digest
   AuthName "Top Secrect"
   AuthDigestFile /usr/local/apache2/misc/.htdigest
   Require user htadmin htmaster
</Directory>
```

Beachten Sie, dass der mittels `AuthName` angegebene Realm bei der Digest-Authentifizierung einen Teil der Anmeldeinformation ausmacht. Deshalb muss der Wert von `AuthName` auf jeden Fall mit dem Wert übereinstimmen, den Sie beim Aufruf des Programms `htdigest` zur Passworterstellung angegeben haben. Wie bereits erwähnt, kann die Digest-Benutzerdatei allerdings mehrere Einträge für denselben Benutzer mit verschiedenen Realms enthalten.

#### AuthDigestGroupFile

Name der Datei mit Gruppennamen

| | |
|---|---|
| **Version** | nur bis 2.0.x |
| **Modul** | `mod_auth_digest` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthDigestGroupFile Dateipfad` |
| **Standardwert** | nicht gesetzt |

Auch bei der Digest-Authentifizierung kann es natürlich vorkommen, dass viele Benutzer Zugriff auf ein geschütztes Verzeichnis erhalten sollen. Mit der Direktive `AuthDigestGroupFile` können Sie in Apache 2.0 eine Gruppendatei dafür einbinden, deren Syntax mit dem gewöhnlichen `AuthGroupFile` identisch ist. Hier ein Komplettbeispiel:

```
<Directory /usr/local/apache2/htdocs/topsecret>
   AuthType Digest
   AuthName "Top Secrect"
   AuthDigestFile /usr/local/apache2/misc/.htdigest
   AuthDigestGroupFile /usr/local/apache2/misc/.htdgroups
   Require group admins
</Directory>
```

In Apache-Versionen ab 2.1 wird je nach Provider eine der Direktiven `AuthGroupFile` beziehungsweise `AuthDBMGroupFile` verwendet.

### AuthDBMAuthoritative

Legt fest, ob Nicht-DBM-Authentifizierungsdaten an andere Module weitergereicht werden.

| | |
|---|---|
| **Version** | nur bis 2.0.x |
| **Modul** | `mod_auth_dbm` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthDBMAuthoritative On\|Off` |
| **Standardwert** | `On` |

Diese Direktive legt fest, ob die Anmeldedaten des Clients an andere Authentifizierungsmechanismen weitergereicht werden sollen, falls die verwendeten Anmeldedaten in der DBM-Datei nicht gefunden werden. Die Voreinstellung `On` besagt, dass die Entscheidung von `mod_auth_dbm` endgültig ist, während die ausdrückliche Angabe des Wertes `Off` zur erwähnten Weitergabe führt. Beachten Sie, dass auch hier das weiter oben für `AuthAuthoritative` beschriebene Reihenfolgeproblem auftritt.

### AuthLDAPAuthoritative

Bestimmt, ob Nicht-LDAP-Authentifizierungsdaten an andere Module weitergereicht werden.

| | |
|---|---|
| **Version** | nur bis 2.0.x |
| **Modul** | `mod_auth_ldap` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthLDAPAuthoritative On\|Off` |
| **Standardwert** | `On` |

Dies ist die Apache-2.0-Fassung von `AuthzLDAPAuthoritative`; Näheres erfahren Sie in Kapitel 9, »Authentifizierung, Autorisierung und Zugriffskontrolle«.

### AuthLDAPEnabled

LDAP-Authentifizierung ein-/ausschalten

| | |
|---|---|
| **Version** | nur bis 2.0.x |
| **Modul** | `mod_auth_ldap` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthLDAPEnabled On\|Off` |
| **Standardwert** | `On` |

Da die LDAP-Authentifizierung beim Laden von `mod_auth_ldap` automatisch aktiviert wird, ermöglicht es Ihnen diese Direktive im Wesentlichen, sie in untergeordneten Kontexten abzuschalten. Dazu müssen Sie explizit folgende Anweisung verwenden:

```
AuthLDAPEnabled Off
```

Das neue Modul `mod_authnz_ldap` benötigt eine solche Direktive nicht; hier genügt es schließlich, `ldap` nicht als AuthBasicProvider anzugeben.

#### AuthLDAPFrontPageHack

LDAP-Authentifizierung mit Microsoft FrontPage verwenden

| | |
|---|---|
| **Version** | nur bis 2.0.x |
| **Modul** | `mod_auth_ldap` |
| **Kontext** | `<Directory>`, `<Location>`, `<Files>`, `.htaccess` (`AuthConfig`) |
| **Syntax** | `AuthLDAPFrontPageHack On\|Off` |
| **Standardwert** | `Off` |

Wenn Sie die (in diesem Buch nicht näher behandelten) FrontPage-Erweiterungen für Apache verwenden, behebt diese Direktive eine kleine Inkompatibilität oder vielmehr ein Missverständnis zwischen `mod_auth_ldap` und FrontPage: `Require valid-user` gilt beim Einsatz von FrontPage für jeden beliebigen Benutzer, der einen Eintrag im LDAP-Verzeichnis besitzt. Wenn Sie `AuthLDAP-Authoritative` auf `Off` setzen und diese Direktive einschalten, verwendet Apache stattdessen die FrontPage-Benutzerdatei. Verwenden Sie in diesem Fall also folgende Direktivenfolge:

```
AuthLDAPAuthoritative off
AuthLDAPFrontPageHack on
```

In Apache 2.2 gibt es diese Direktive nicht mehr; hier genügt folgende Einstellung:

```
AuthzLDAPAuthoritative Off
```

#### LDAPTrustedCA

Datei mit der vertrauenswürdigen Certification Authority für LDAP

| | |
|---|---|
| **Seit Version** | 2.0.41 |
| **Modul** | `mod_ldap` |
| **Kontext** | Server |

| Syntax | `LDAPTrustedCA Verzeichnispfad/Dateiname` |
|---|---|
| Standardwert | nicht gesetzt |

Diese Direktive legt die vertrauenswürdige Certification Authority (CA) für SSL-Verbindungen (Näheres siehe nächstes Kapitel) zum LDAP-Server fest. Wenn Sie Netscape/iPlanet SDK anstelle von OpenLDAP verwenden, sollten Sie den Wert auf `cert7.db` setzen.

#### LDAPTrustedCAType

Typ der CA-Datei

| Seit Version | 2.0.41 |
|---|---|
| Modul | `mod_ldap` |
| Kontext | Server |
| Syntax | `LDAPTrustedCAType Typ` |
| Standardwert | nicht gesetzt |

Über diese Konfigurationseinstellung wird der Typ der vertrauenswürdigen CA angegeben. Es gibt drei mögliche Werte:

- `DER_FILE`: binäres DER-Format
- `BASE64_FILE`: Base64-Format
- `CERT7_DB_PATH`: Datenbankdatei für das Netscape-Zertifikat

Was Sie hier einstellen müssen, hängt von der SSL- und der LDAP-Implementierung ab, die Sie verwenden.

## A.4 Weitere exklusive Direktiven

Auch die im Folgenden dokumentierten beiden Konfigurationsdirektiven, die nichts mit Authentifizierung zu tun haben, sind nur in Apache 2.0 verfügbar.

#### AssignUserID

Einem virtuellen Host eine User- und Group-ID zuordnen

| Modul | `mpm_perchild` |
|---|---|
| Kontext | `<VirtualHost>` |
| Syntax | `AssignUserID User-ID Group-ID` |
| Standardwert | nicht gesetzt |

Diese Direktive stellt eine Funktion des Multiprocessing-Moduls `mpm_perchild` (siehe Abschnitt B.1) bereit, das die Ausführung von Child-Prozessen und virtuellen Hosts unter verschiedenen User-IDs erlaubt. Diese Direktive legt dabei die User- und Group-ID fest, unter der ein bestimmter virtueller Host ausgeführt werden soll. Hier sehen Sie ein Beispiel mit zwei namensbasierten virtuellen Hosts:

```
NameVirtualHost *:80

<VirtualHost *:80>
  ServerName www1.mynet.de
  DocumentRoot /var/www1/htdocs
  AssignUserID webuser1 webgroup
</VirtualHost>

<VirtualHost *:80>
  ServerName www2.mynet.de
  DocumentRoot /var/www2/htdocs
  AssignUserID webuser2 webgroup
</VirtualHost>
```

#### BS2000Account

Nicht privilegierter User-Account auf BS2000-Systemen

| | |
|---|---|
| **Seit Version** | 2.0 |
| **Modul** | `prefork` |
| **Kontext** | Server |
| **Syntax** | `BS2000Account Account` |
| **Standardwert** | nicht gesetzt |

Unter dem Mainframe-Betriebssystem BS2000 wird mithilfe dieser Direktive das Benutzerkonto eingestellt, unter dem Apache Anfragen beantwortet. Dies muss zusätzlich zur bereits beschriebenen Direktive `User` geschehen.

#### CacheSize

Maximaler Festplattenspeicher für den Cache

| | |
|---|---|
| **Modul** | `mod_disk_cache` |
| **Kontext** | Server, `<VirtualHost>` |
| **Syntax** | `CacheSize KBytes` |
| **Standardwert** | `1000000` |

Diese Direktive legt die maximale Größe fest, die das Cache-Verzeichnis insgesamt auf dem Datenträger einnehmen darf. Wenn der Cache zu groß wird, tritt die Garbage-Collection-Funktion von `mod_disk_cache` in Aktion und löscht Dateien, bis der Cache wieder klein genug ist. Es gibt mit anderen Worten keine Garantie dafür, dass diese Größe niemals überschritten wird. Deshalb sollte der gewählte Wert nicht annähernd so hoch sein wie der des verfügbaren Festplattenplatzes.

Der Standardwert beträgt eine Million Kilobyte (circa 977 Megabyte). Das folgende Beispiel setzt eine Größe von 300 Megabyte: `CacheSize 307200`.

*»Das Nachdenken ist eine höchst zwecklose Beschäftigung, solange man nichts von den Tatsachen weiß.«*
*– Bernard Shaw*

# B Kurzreferenz der Konfigurationsdirektiven

In der Tabelle finden Sie eine alphabetische Liste sämtlicher Apache-Konfigurationsdirektiven der Versionen 2.2 und 2.4, die für den Server-Kern und alle mit der Standarddistribution gelieferten Module verfügbar sind. Aus Platzgründen wird bei einigen überlangen Direktivennamen die Silbentrennung verwendet – selbstverständlich müssen die Namen in Wirklichkeit ohne Trennstrich geschrieben werden. Für die Kontexte, in denen die Direktiven stehen dürfen, werden folgende Abkürzungen benutzt:

- s – Server-Konfiguration
- v – `<VirtualHost>`-Container
- d – `<Directory>`-, `<Files>`- und `<Location>`-Container
- h – `.htaccess`-Dateien

In der Spalte »Vers.« (Version) steht nur dann etwas, wenn eine Direktive nur für eine der beiden Versionen (2.2 oder 2.4) verfügbar ist; wenn das Feld leer ist, gilt die Direktive für beide.

Als zusätzliche Information wurde für jede Direktive das Kapitel dieses Buches angegeben, in dem ihre Beschreibung steht. Die genauen Seitenzahlen entnehmen Sie bitte dem Index. Bei einigen Direktiven steht statt einer Kapitelnummer »DVD« – diese sind erst nach dem finalen Satztermin dokumentiert worden und befinden sich im Dokument *ergaenzungen.pdf* im Hauptverzeichnis der beiliegenden DVD.

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| AcceptFilter | s | | 6 | TCP-Filter auf Kernel-Ebene |
| AcceptMutex | s | 2.2 | 6 | Methode zum Sperren gleichzeitiger Socket-Zugriffe |
| AcceptPathInfo | svdh | | 14 | zusätzliche Pfadangaben hinter der eigentlichen URL auswerten |
| AccessFileName | sv | | 6 | Name der Verzeichnis-Konfigurationsdateien |
| Action | svdh | | 14 | CGI-Skript für einen bestimmten Handler/MIME-Type aktivieren |
| AddAlt | svdh | | 8 | Alternativtext für Icons im automatisch erzeugten Index |
| AddAltByEncoding | svdh | | 8 | Alternativtext für Icons nach Kompressionstyp |
| AddAltByType | svdh | | 8 | Alternativtext für Icons nach MIME-Type |
| AddCharset | svdh | | 7 | Dateiendungen einen Zeichensatz zuordnen |
| AddDefaultCharset | svdh | | 7 | Standardzeichensatz für einen Kontext |
| AddDescription | svdh | | 8 | Beschreibung für Objekte im automatischen Index |
| AddEncoding | svdh | | 7 | MIME-Konfiguration |
| AddHandler | svdh | | 7 | Handler für bestimmte Dateiendungen hinzufügen |
| AddIcon | svdh | | 8 | Icons für den automatisch erzeugten Index definieren |
| AddIconByEncoding | svdh | | 8 | Icons im Index nach Kompressionstyp |
| AddIconByType | svdh | | 8 | Icons nach MIME-Type |
| AddInputFilter | svdh | | 16 | Eingabefilter für bestimmte Dateiendungen definieren |
| AddLanguage | svdh | | 7 | Sprache für bestimmte Dateiendungen hinzufügen |
| AddModuleInfo | sv | | 8 | Text für ein bestimmtes Modul zur Server-Info hinzufügen |
| AddOutputFilter | svdh | | 16 | Ausgabefilter nach Dateiendungen hinzufügen |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| AddOutputFilterByType | svdh | | 16 | Ausgabefilter nach MIME-Type hinzufügen |
| AddType | svdh | | 7 | Bestimmte Dateiendungen einem MIME-Type zuordnen |
| Alias | sv | | 8 | externes Verzeichnis einem URL-Pfad zuordnen |
| AliasMatch | sv | | 8 | externe Verzeichnisse nach RegExp URL-Pfaden zuordnen |
| Allow | dh | | 6 | bestimmten Hosts Zugriff gewähren |
| AllowCONNECT | sv | | 13 | SSL-Verbindungen über den Proxy tunneln |
| AllowEncodedSlashes | sv | | 14 | codierte Slashes (%2F) in URLs zulassen |
| AllowMethods | | | 6 | Erlaubte HTTP-Zugriffsmethoden für den aktuellen Kontext |
| AllowOverride | d | | 6 | in .htaccess-Dateien erlaubte Direktivengruppen |
| AllowOverrideList | d | 2.4 | DVD | Liste einzelner Direktiven, die in .htaccess-Dateien erlaubt sind |
| Anonymous | dh | | 9 | Benutzername für Anonymous-Authentifizierung |
| Anonymous_LogEmail | dh | | 9 | E-Mail-Adressen aus der Anonymous-Authentifizierung in ErrorLog schreiben |
| Anonymous_MustGive Email | dh | | 9 | Bestimmt, ob bei Anonymous-Anmeldedaten ein Passwort erforderlich ist. |
| Anonymous_NoUserID | dh | | 9 | Bestimmt, ob bei Anonymous-Anmeldedaten ein Benutzername erforderlich ist. |
| Anonymous_Verify Email | dh | | 9 | Überprüfung, ob die Anonymous-Passworteingabe formal einer E-Mail-Adresse entspricht |
| AuthBasicAuthoritative | dh | | 9 | Weitergabe der Basic-Authentifizierungsdaten an andere Module |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| `AuthBasicProvider` | dh | | 9 | Speichermethode für Basic-Benutzerdaten |
| `AuthDBDUserPWQuery` | dh | | 9 | SQL-Abfrage, die das Passwort für einen gegebenen Benutzernamen liefert |
| `AuthDBDUserRealmQuery` | dh | | 9 | SQL-Abfrage, die das Passwort für Benutzername/Realm liefert |
| `AuthDBMGroupFile` | dh | | 9 | DBM-basierte Gruppendaten |
| `AuthDBMType` | dh | | 9 | Typ der DBM-Benutzerdatei |
| `AuthDBMUserFile` | dh | | 9 | DBM-basierte Benutzerdaten |
| `AuthDefault-Authoritative` | dh | 2.2 | 9 | Weitergabe der Authentifizierung an andere Module |
| `AuthDigestAlgorithm` | dh | | 9 | Algorithmus für die Digest-Berechnung |
| `AuthDigestDomain` | dh | | 9 | gemeinsamer Authentifizierungsbereich für Digest-Authentifizierung |
| `AuthDigestNcCheck` | s | | 9 | Überprüfung des Nonce-Wertes |
| `AuthDigestNonceFormat` | dh | | 9 | Format des Nonce-Wertes |
| `AuthDigestNonceLifetime` | dh | | 9 | Lebensdauer des Nonce-Wertes |
| `AuthDigestProvider` | dh | | 9 | Speichermethode für die Digest-Benutzerdaten |
| `AuthDigestQop` | dh | | 9 | Schutzstufe für die Digest-Authentifizierung |
| `AuthDigestShmemSize` | s | | 9 | Größe des gemeinsamen Speicherbereichs für Digest-Authentifizierung |
| `AuthFormAuthoritative` | d | 2.4 | 9 | Bestimmt, ob die formularbasierte Authentifizierung das letzte Wort hat |
| `AuthFormBody` | d | 2.4 | 9 | Feldname für den Body einer HTTP-Anfrage im Anmeldeformular |
| `AuthFormDisableNoStore` | d | 2.4 | 9 | Den Header `Cache-control: no-store` für die Login-Seite deaktivieren |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| AuthFormFakeBasicAuth | d | 2.4 | 9 | Einen gefälschten Basic-Auth-Header hinzufügen |
| AuthFormLocation | d | 2.4 | 9 | Feldname für die Weiterleitungs-URL im Anmeldeformular |
| AuthFormLoginRequired-Location | d | 2.4 | 9 | URL, zu der weitergeleitet wird, wenn ein Login erforderlich ist |
| AuthFormLoginSuccess-Location | d | 2.4 | 9 | URL, zu der nach erfolgreichem Login weitergeleitet wird |
| AuthFormLogoutLocation | d | 2.4 | 9 | URL, zu der nach Logout weitergeleitet wird |
| AuthFormMethod | d | 2.4 | 9 | Feldname für die HTTP-Methode einer Anfrage im Anmeldeformular |
| AuthFormMimetype | d | 2.4 | 9 | Feldname für den MIME-Type einer HTTP-Anfrage im Anmeldeformular |
| AuthFormPassword | d | 2.4 | 9 | Feldname für das Passwort im Anmeldeformular |
| AuthFormProvider | d | 2.4 | 9 | Provider-Modul für den Abgleich der Formular-Anmeldedaten |
| AuthFormSitePassphrase | d | 2.4 | 9 | Passphrase zum Umgehen der Authentifizierung auf Sites mit viel Traffic |
| AuthFormSize | d | 2.4 | 9 | Maximale Größe der Anmelde-Formulardaten |
| AuthFormUsername | d | 2.4 | 9 | Feldname für den Benutzernamen im Anmeldeformular |
| AuthGroupFile | dh | | 9 | Datei mit Gruppendaten |
| AuthLDAPAuthorizePrefix | dh | 2.4 | DVD | Präfix für die während der Autorisierung gesetzten Umgebungsvariablen |
| AuthLDAPBindDN | dh | | 9 | Benutzername für Zugriff auf das LDAP-Verzeichnis |
| AuthLDAPBindPassword | dh | | 9 | Passwort für Zugriff auf das LDAP-Verzeichnis |
| AuthLDAPCharsetConfig | s | | 9 | Zeichensatzkonfiguration für LDAP-Authentifizierung |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| AuthLDAPCompareDNOn-Server | dh | | 9 | DN für LDAP-Authentifizierung auf dem Server überprüfen |
| AuthLDAPCompareAsUser | dh | | DVD | Die Credentials des angegebenen Users für Autorisierungsvergleiche verwenden |
| AuthLDAPDereference-Aliases | Dh | | 9 | LDAP-Aliase auflösen |
| AuthLDAPGroupAttribute | Dh | | 9 | Festlegung der LDAP-Daten für die Gruppenüberprüfung |
| AuthLDAPInitialBindAs-User | dh | 2.4 | DVD | Bestimmt, ob die LDAP-Anmeldung mit Userdaten oder anonym erfolgt |
| AuthLDAPInitialBind-Pattern | dh | 2.4 | DVD | Transformationsregel des Usernamens für die LDAP-Anmeldung |
| AuthLDAPGroup-AttributeIsDN | dh | | 9 | Festlegung des DN als Kriterium für die Gruppenüberprüfung |
| AuthLDAPMaxSubGroup-Depth | dh | 2.4 | DVD | Maximale Tiefe der zu durchsuchenden Untergruppen |
| AuthLDAPRemote-UserAttribute | dh | | 9 | LDAP-Attribut für die Umgebungsvariable REMOTE_USER angeben |
| AuthLDAPRemoteUserIsDN | dh | | 9 | DN in der Umgebungsvariablen REMOTE_USER speichern |
| AuthLDAPSearchAsUser | dh | 2.4 | DVD | Credentials des angegebenen Users für LDAP-Suchoperationen verwenden |
| AuthLDAPSubGroup-Attribute | dh | 2.4 | DVD | LDAP-Attribut, das Untergruppen kennzeichnet |
| AuthLDAPSubGroupClass | dh | 2.4 | DVD | LDAP-objectClass, die Untergruppen kennzeichnet |
| AuthLDAPUrl | dh | | 9 | URL des LDAP-Verzeichnis-Servers |
| AuthMerging | dh | 2.4 | 9 | Zusammenwirkung von Authentifizierungseinstellungen mit denjenigen aus übergeordneten Kontexten |
| AuthName | dh | | 9 | Name des Authentifizierungsbereichs |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| `<AuthnProviderAlias>` | sv | | 9 | Festlegung modifizierter Authentifizierungs-Provider |
| `AuthnCacheContext` | d | 2.4 | DVD | Kontext-String für den Auth-Cache-Schlüssel |
| `AuthnCacheEnable` | s | 2.4 | DVD | Den Auth-Cache generell aktivieren |
| `AuthnCacheProvideFor` | dh | 2.4 | DVD | Angabe der Auth-Provider, deren Informationen im Cache gespeichert werden |
| `AuthnCacheSOCache` | s | 2.4 | DVD | Auswahl des Shared-Objects-Providers für den Auth-Cache |
| `AuthnCacheTimeout` | dh | 2.4 | DVD | Timeout für Auth-Cache-Einträge |
| `AuthType` | dh | | 9 | Authentifizierungstyp (Basic oder Digest) |
| `AuthUserFile` | dh | | 9 | Datei mit Benutzerdaten |
| `AuthzDBMAuthoritative` | dh | 2.2 | 9 | Weitergabe der DBM-Gruppendaten an andere Module |
| `AuthzDBMType` | dh | | 9 | Typ der DBM-Gruppendatei |
| `AuthzDefault-Authoritative` | dh | 2.2 | 9 | Weitergabe der Zugriffskontrolle an andere Module |
| `AuthzGroupFile-Authoritative` | dh | 2.2 | 9 | Weitergabe der Gruppenzugriffskontrolle an andere Module |
| `AuthzLDAPAuthoritative` | dh | 2.2 | 9 | Weitergabe der LDAP-Zugriffskontrolle an andere Module |
| `AuthzOwnerAuthoritative` | dh | 2.2 | 9 | Weitergabe der eigentümerbasierten Zugriffskontrolle an andere Module |
| `<AuthzProviderAlias>` | s | 2.4 | 9 | Definition benannter `Require`-Blöcke zur späteren Verwendung |
| `AuthzSendForbiddenOn-Failure` | dh | 2.4 | 9 | `403 Forbidden` statt `401 Unauthorized` senden, wenn die Autorisierung fehlschlägt |
| `AuthzUserAuthoritative` | dh | | 9 | Weitergabe der benutzerbasierten Zugriffskontrolle an andere Module |
| `BalancerMember` | d | | 13 | Member zu einer Proxy-Load-Balancing-Gruppe hinzufügen |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| BrowserMatch | svdh | | 14 | Umgebungsvariable setzen, falls Client-Browser einem RegExp entspricht |
| BrowserMatchNoCase | svdh | | 14 | wie BrowserMatch, ohne Unterscheidung von Groß-/Kleinschreibung |
| BufferedLogs | s | | 11 | Logdaten im Speicher puffern |
| BufferSize | svdh | 2.4 | DVD | Maximale Größe des Buffer-Filters in Byte |
| CacheDefaultExpire | sv | | 13 | Standardaufbewahrungsdauer einer Datei im Cache |
| CacheDetailHeader | | 2.4 | 13 | Header mit detaillierten Caching-Informationen hinzufügen |
| CacheDirLength | sv | | 13 | Länge der Cache-Verzeichnisnamen |
| CacheDirLevels | sv | | 13 | Anzahl der Cache-Verzeichnisstufen |
| CacheDisable | sv | | 13 | Caching für bestimmte Unterverzeichnisse deaktivieren |
| CacheEnable | sv | | 13 | Caching für bestimmte URL-Pfade aktivieren |
| CacheFile | s | | 12 | Dateien für schnelleren Zugriff im Arbeitsspeicher geöffnet halten |
| CacheHeader | svdh | 2.4 | 13 | Header mit Caching-Informationen hinzufügen |
| CacheIgnoreCacheControl | sv | | 13 | Caching trotz gegenteiligem Cache-Control-Header |
| CacheIgnoreHeaders | sv | | 13 | Caching, ohne Header zu beachten |
| CacheIgnoreNoLastMod | sv | | 13 | Caching, ohne das Änderungsdatum zu berücksichtigen |
| CacheIgnoreQueryString | sv | | 13 | Query-String der URL beim Caching nicht berücksichtigen |
| CacheKeyBaseURL | sv | | 2.4 | Basis-URL zur Erzeugung des Cache-Schlüssels |
| CacheLastModifiedFactor | sv | | 13 | Faktor zur Berechnung des Änderungsdatums im Cache |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| CacheMaxExpire | sv | | 13 | maximales Alter eines Dokuments für den Cache |
| CacheMaxFileSize | sv | | 13 | maximale Größe einer Datei für den Cache |
| CacheMinExpire | sv | 2.4 | 13 | Mindesthaltbarkeit eines Dokuments im Cache |
| CacheMinFileSize | sv | | 13 | minimale Größe einer Datei für den Cache |
| CacheNegotiatedDocs | sv | | 13 | Caching von Dokumenten, die durch Content-Negotiation erzeugt wurden |
| CacheQuickHandler | sv | 2.4 | 13 | Dateien in der Quick-Handler-Phase aus dem Cache laden |
| CacheReadSize | svdh | 2.4 | DVD | Anzahl von Bytes eines Dokuments, die gelesen und gecached werden, bevor das Dokument gesendet wird |
| CacheReadTime | svdh | 2.4 | DVD | Mindest-Lesezeit, bevor ein Dokument gesendet wird |
| CacheRoot | sv | | 13 | Stammverzeichnis des Festplatten-Caches |
| CacheStaleOnError | svdh | 2.4 | 13 | Dokumente aus dem Cache liefern, obwohl die Originale nicht mehr verfügbar sind |
| CacheStoreExpired | svdh | 2.4 | 13 | Caching abgelaufener Dokumente |
| CacheStoreNoStore | sv | | 13 | Caching trotz Cache-Control: no-store |
| CacheStorePrivate | sv | | 13 | Caching trotz Cache-Control: private |
| CGIMapExtension | dh | | 14 | Kriterium für die Auswahl des CGI-Interpreters (nur Windows) |
| CharsetDefault | svdh | | 16 | Ziel-Zeichensatz der Konvertierung |
| CharsetOptions | svdh | | 16 | Optionen für die Zeichensatzkonvertierung |
| CharsetSourceEnc | svdh | | 16 | Ausgangszeichensatz der Konvertierung |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| CheckCaseOnly | svdh | | 8 | nur Groß- und Kleinschreibung in URLs korrigieren, keine anderen Fehler |
| CheckSpelling | svdh | | 8 | Fehlerkorrektur in URLs |
| ChrootDir | s | | 18 | chroot in das angegebene Verzeichnis durchführen |
| ContentDigest | svdh | | 7 | Content-MD5 erzeugen |
| CookieDomain | svdh | | 11 | Domain des Usertracking-Cookies |
| CookieExpires | svdh | | 11 | Verfallsdatum des Usertracking-Cookies |
| CookieLog | sv | 2.2 | 11 | Log-Datei für Usertracking-Cookies |
| CookieName | svdh | | 11 | Name des Usertracking-Cookies |
| CookieStyle | svdh | | 11 | Typ des Usertracking-Cookies |
| CookieTracking | svdh | | 11 | Cookie-basiertes Usertracking ein-/ausschalten |
| CoreDumpDirectory | s | | 6 | Verzeichnis, in das Apache bei einem Coredump wechselt |
| CustomLog | sv | | 11 | benutzerdefinierte Log-Datei |
| Dav | d | | 17 | WebDAV ein-/ausschalten |
| DavDepthInfinity | svd | | 17 | PROPFIND-Anfragen mit beliebiger Verzeichnistiefe ein-/ausschalten |
| DavGenericLockDB | svd | | 17 | verallgemeinertes Locking für WebDAV |
| DavLockDB | sv | | 17 | Dateisystem-Locking für WebDAV |
| DavMinTimeout | svd | | 17 | Mindestvorhaltezeit für eine WebDAV-Sperre |
| DBDExptime | sv | | 9 | Keepalive-Zeit für leere Datenbankverbindungen |
| DBDKeep | sv | | 9 | Anzahl offen gehaltener Datenbankverbindungen |
| DBDMax | sv | | 9 | absolute Höchstzahl der Datenbankverbindungen |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| DBDMin | sv | | 9 | Mindestzahl der Datenbankverbindungen |
| DBDParams | sv | | 9 | Parameter für die Datenbankverbindung |
| DBDPersist | sv | | 9 | Persistenz der Datenbankverbindungen |
| DBDPrepareSQL | sv | | 9 | SQL Prepared Statements |
| DBDriver | sv | | 9 | Treiber für die Datenbankverbindung |
| DefaultIcon | svdh | | 8 | Standard-Icon für den automatisch erzeugten Index |
| DefaultLanguage | svdh | | 7 | Standardsprache für Dateien in einem Kontext |
| DefaultType | svdh | | 7 | Standard-MIME-Type für Dateien in einem Kontext |
| Define | | 2.4 | 6 | Definition einer Variablen zur Abfrage durch <IfDefine> |
| DeflateBufferSize | sv | | 16 | Größe des zlib-Puffers |
| DeflateCompressionLevel | sv | | 16 | Deflate-Komprimierunsfaktor |
| DeflateFilterNote | sv | | 16 | Komprimierungsergebnisse für Log-Dateien veröffentlichen |
| DeflateMemLevel | sv | | 16 | zlib-Speicherzuteilung |
| DeflateWindowSize | sv | | 16 | Größe des zlib-Fensters |
| Deny | dh | | 6 | bestimmten Hosts Zugriff verweigern |
| <Directory> | sv | | 6 | Container für Direktiven, die für ein Verzeichnis gelten |
| DirectoryIndex | svdh | | 6 | Dateinamen für Verzeichnis-Indexdateien |
| DirectoryIndexRedirect | svdh | 2.4 | DVD | Externe Weiterleitung für Verzeichnis-Indizes |
| <DirectoryMatch> | sv | | 6 | Container für Direktiven, die auf ein Verzeichnismuster gelten |
| DirectorySlash | svdh | | 6 | Weiterleitung bei URLs ohne abschließenden Slash ein-/ausschalten |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| DocumentRoot | sv | | 6 | Stammverzeichnis der Website |
| DumpIOInput | s | | 11 | alle Eingabedaten des Servers in eine Log-Datei schreiben |
| DumpIOLogLevel | s | 2.2 | 11 | LogLevel, ab dem Ein- oder Ausgabedaten in die Log-Datei geschrieben werden |
| DumpIOOutput | s | | 11 | alle Ausgabedaten des Servers in eine Log-Datei schreiben |
| <Else> | svdh | 2.4 | 6 | Alternative, wenn alle <If>- und <ElseIf>-Angaben nicht zutreffen |
| <ElseIf> | svdh | 2.4 | 6 | Prüft eine weitere Bedingung, falls die vorherige <If>- oder <ElseIf>-Angabe nicht zutrifft |
| EnableExceptionHook | s | | 18 | speziellen Hook für die Ausnahmebehandlung aktivieren |
| EnableMMAP | svdh | | 12 | Memory-Mapping aktivieren |
| EnableSendfile | svdh | | 12 | direkte Auslieferung von Dateien aktivieren |
| Error | svdh | 2.4 | 6 | Die Verarbeitung der Konfiguration mit einer Fehlermeldung abbrechen |
| ErrorDocument | svdh | | 8 | benutzerdefinierte Server-Fehlermeldungen |
| ErrorLog | sv | | 11 | Log-Datei für Fehlermeldungen |
| ErrorLogFormat | | 2.4 | 11 | Format für die Error-Log-Dateien |
| Example | svdh | | 17 | Demonstriert die Apache-Modul-API. |
| ExpiresActive | svdh | | 7 | Expires-Header generieren |
| ExpiresByType | svdh | | 7 | Verfallsdatum für bestimmten MIME-Type festlegen |
| ExpiresDefault | svdh | | 7 | Standardverfallsdatum |
| ExtendedStatus | s | | 8 | erweiterte Server-Status-Meldungen |
| ExtFilterDefine | s | | 16 | Definition eines externen Programms als Ausgabefilter |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| ExtFilterOptions | d | | 16 | Optionen für externe Ausgabefilter |
| FileETag | svdh | | 7 | Elemente für die Erzeugung des ETag-Headers |
| <Files> | svdh | | 6 | Container für Direktiven, die für bestimmte Dateien gelten |
| <FilesMatch> | svdh | | 6 | Container für Direktiven, die für RegExp-Dateimuster gelten |
| FilterChain | svdh | | 16 | Festlegung der Filter Chain |
| FilterDeclare | svdh | | 16 | Deklaration eines Filters |
| FilterProtocol | svdh | | 16 | HTTP-Protokolländerungen durch einen Filter |
| FilterProvider | svdh | | 16 | Registrieren eines Providers für Filter |
| FilterTrace | svd | | 16 | Debug-Informationen für Filter erzeugen |
| ForceLanguagePriority | svdh | | 7 | Sprachpräferenzen des Servers erzwingen |
| ForceType | dh | | 7 | allen Dokumenten in einem Kontext einen bestimmten MIME-Type zuweisen |
| ForensicLog | sv | | 11 | forensische Log-Datei |
| GracefulShutdownTimeout | s | | 6 | Zeitlimit für sauberes Beenden des Servers |
| Group | s | | 6 | Group-ID, unter der die Worker-Prozesse ausgeführt werden |
| Header | svdh | | 7 | Manipulation von Ausgabe-Headern |
| HeaderName | svdh | | 8 | Datei, die über dem Verzeichnisindex eingefügt wird |
| HeartbeatAddress | s | 2.4 | DVD | Adresse, an die Heartbeat-Anfragen gesendet werden |
| HeartbeatListen | s | 2.4 | DVD | Adresse, an der auf Heartbeat-Anfragen gelauscht wird |
| HeartbeatStorage | s | 2.4 | DVD | Pfad zur Speicherung von Heartbeat-Daten |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| HostnameLookups | svd | | 11 | IP-Adressen in Log-Dateien im DNS nachschlagen |
| IdentityCheck | svd | | 11 | RFC-1413-Identität überprüfen |
| IdentityCheckTimeout | svd | | 11 | Zeitlimit für RFC-1413-Abfragen |
| <If> | svdh | 2.4 | 6 | Prüfen eines beliebigen ap_expr-Ausdrucks und Ausführen der enthaltenen Konfiguration |
| <IfDefine> | svdh | | 6 | Container für Direktiven, die von einer Kommandozeilenoption abhängen |
| <IfModule> | svdh | | 6 | Container für Direktiven, die von einem Modul abhängen |
| <IfVersion> | svdh | | 6 | Container für Direktiven, die von der Version abhängen |
| ImapBase | svdh | | 6 | Basisverzeichnis für Image-Maps |
| ImapDefault | svdh | | 6 | Standardziel in Image-Maps |
| ImapMenu | svdh | | 6 | Menüdatei für Image-Maps |
| Include | svd | | 6 | Integration externer Konfigurationsdateien |
| IndexIgnore | svdh | | 8 | Dateien, die nicht im Index erscheinen sollen |
| IndexIgnoreReset | svdh | 2.4 | 8 | Zurücksetzen der IndexIgnore-Liste aus dem übergeordneten Kontext |
| IndexOptions | svdh | | 8 | Optionen für den automatisch erzeugten Index |
| IndexOrderDefault | svdh | | 8 | Sortierreihenfolge für den Index |
| IndexStyleSheet | svdh | | 8 | CSS-Datei für den Index |
| InputSed | dh | 2.4 | DVD | sed-Befehl zum Filtern von Anfragedaten |
| ISAPIAppendLogToErrors | svdh | | 15 | HSE_LOG_PARAMETER aus ISAPI-Erweiterungen in ErrorLog schreiben |
| ISAPIAppendLogToQuery | svdh | | 15 | HSE_LOG_PARAMETER aus ISAPI-Erweiterungen in QueryString schreiben |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| ISAPICacheFile | sv | | 15 | ISAPI-DLLs, die beim Start vorausgeladen werden sollen |
| ISAPIFakeAsync | svdh | | 15 | asynchrone ISAPI-Anfragen simulieren |
| ISAPILogNotSupported | svdh | | 15 | nicht unterstützte ISAPI-Funktionen protokollieren |
| ISAPIReadAheadBuffer | svdh | | 15 | Größe des Vorauslesepuffers für ISAPI |
| KeepAlive | sv | | 6 | Persistente HTTP/1.1-Verbindungen ein-/ausschalten |
| KeepAliveTimeout | sv | | 6 | Zeitlimit für persistente HTTP-Verbindungen |
| KeptBodySize | d | 2.4 | DVD | Body einer Anfrage bis zur angegebenen Größe behalten |
| LanguagePriority | svdh | | 7 | Reihenfolge der Sprachpräferenzen |
| LDAPCacheEntries | s | | 9 | Anzahl der Einträge im LDAP-Cache |
| LDAPCacheTTL | s | | 9 | Lebensdauer der Einträge im LDAP-Cache |
| LDAPConnectionPoolTTL | sv | 2.4 | DVD | Verbindungen beenden, die sich länger als angegeben im Verbindungs-Pool befanden |
| LDAPConnectionTimeout | s | | 9 | maximale Socket-Wartezeit für LDAP-Verbindungen |
| LDAPLibraryDebug | s | 2.4 | DVD | Debugging im LDAP-SDK aktivieren |
| LDAPOpCacheEntries | s | | 9 | Anzahl der Cache-Einträge für LDAP-Vergleichsoperationen |
| LDAPOpCacheTTL | s | | 9 | Lebensdauer der Einträge im LDAP-Operations-Cache |
| LDAPReferralHopLimit | dh | 2.4 | DVD | Maximale Anzahl von Referral-Hops, bevor eine LDAP-Verbindung beendet wird |
| LDAPReferrals | dh | 2.4 | DVD | Referral-Hops bei Anfragen an LDAP-Server zulassen |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| LDAPSharedCacheFile | s | | 9 | gemeinsam genutzter Speicher für den LDAP-Cache |
| LDAPSharedCacheSize | s | | 9 | Größe des Speicherbereichs für den LDAP-Cache |
| LDAPTimeout | s | 2.4 | DVD | Timeout für LDAP-Verbindungs- und -Suchanfragen |
| LDAPTrustedClientCert | svdh | | 9 | Client-Zertifikate für LDAP-Verbindungen |
| LDAPTrustedGlobalCert | s | | 9 | Zertifikate der CAs für LDAP-Verbindungen |
| LDAPTrustedMode | svdh | | 9 | SSL/TLS für LDAP-Verbindungen verwenden |
| LDAPVerifyServerCert | s | | 9 | Überprüfung des Server-Zertfikats bei LDAP-Verbindungen |
| <Limit> | svdh | | 6 | Container zur Beschränkung von Direktiven auf bestimmte HTTP-Methoden |
| <LimitExcept> | svdh | | 6 | Container zur Beschränkung von Direktiven auf alle HTTP-Methoden außer den angegebenen |
| LimitInternalRecursion | sv | | 18 | maximale Anzahl interner Unterabfragen |
| LimitRequestBody | svdh | | 18 | maximale Größe des Anfrage-Bodys |
| LimitRequestFields | s | | 18 | maximale Anzahl der Anfrage-Header-Felder |
| LimitRequestFieldSize | s | | 18 | maximale Größe der Anfrage-Header-Felder |
| LimitRequestLine | s | | 18 | maximale Größe der HTTP-Anfragezeile |
| LimitXMLRequestBody | svdh | | 18 | maximale Größe eines XML-Anfrage-Bodys |
| Listen | s | | 6 | Lauschen an einem TCP-Port |
| ListenBackLog | s | | 6 | maximale Anzahl gepufferter Client-Verbindungsversuche |
| LoadFile | s | | 6 | eine zusätzliche Datei laden (Helfer für ein Modul) |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| LoadModule | s | | 6 | ein DSO-Modul laden |
| <Location> | sv | | 6 | Container für Direktiven, die einen bestimmten URL-Pfad betreffen |
| <LocationMatch> | sv | | 6 | Container für Direktiven, die bestimmten, durch RegExp angegebenen URL-Pfaden entsprechen |
| LockFile | s | 2.2 | 6 | Dateipfad für dateibasierte Sperren |
| LogFormat | sv | | 11 | Definition von Formaten für Log-Dateien |
| LogLevel | s | | 11 | Syslog-Level für die ErrorLog-Datei |
| LogMessage | d | 2.4 | 11 | Eine Debug-Log-Meldung ins ErrorLog schreiben |
| LuaCodeCache | svdh | 2.4 | 15 | Cache für vorkompilierten Lua-Code |
| LuaHookAccessChecker | svdh | 2.4 | 15 | Lua-Skript für den Hook access_checker |
| LuaHookAuthChecker | svdh | 2.4 | 15 | Lua-Skript für den Hook auth_checker |
| LuaHookCheckUserId | svdh | 2.4 | 15 | Lua-Skript für den Hook check_user_id |
| LuaHookFixups | svdh | 2.4 | 15 | Lua-Skript für den Hook fixups |
| LuaHookInsertFilter | svdh | 2.4 | 15 | Lua-Skript für den Hook insert_filter |
| LuaHookMapToStorage | svdh | 2.4 | 15 | Lua-Skript für den Hook map_to_storage |
| LuaHookTranslateName | svdh | 2.4 | 15 | Lua-Skript für den Hook translate_name |
| LuaHookTypeChecker | svdh | 2.4 | 15 | Lua-Skript für den Hook type_checker |
| LuaInherit | svdh | 2.4 | 15 | Umgang mit Lua-Hooks aus übergeordneten Kontexten |
| LuaMapHandler | svdh | 2.4 | 15 | Einem Pfad einen Lua-Handler zuordnen |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| `LuaPackageCPath` | svdh | 2.4 | DVD | Einen Pfad zu Luas *package.cpath* hinzufügen |
| `LuaPackagePath` | svdh | 2.4 | DVD | Einen Pfad zu Luas *package.path* hinzufügen |
| `LuaQuickHandler` | svdh | 2.4 | DVD | Lua-Skript für den Quick-Handler |
| `LuaRoot` | svdh | 2.4 | DVD | Basispfad für relative Pfadangaben in mod_lua-Direktiven |
| `LuaScope` | svdh | 2.4 | DVD | Wiederverwendbarkeit des Lua-Interpreters |
| `MaxClients` | s | | 6 | maximale Anzahl der Worker-Prozesse |
| `MaxConnectionsPerChild` | s | 2.4 | 6 | Neuer Name von `MaxRequestsPerChild` in 2.4 |
| `MaxKeepAliveRequests` | sv | | 6 | maximale Anzahl von HTTP-Anfragen innerhalb einer Verbindung |
| `MaxMemFree` | s | | 6 | maximale Speichermenge, die die Speicherverwaltung ohne `free()`-Aufruf verwalten darf |
| `MaxRangeOverlaps` | svd | 2.4 | DVD | Erlaubte Anzahl überlappender Ranges, bevor das gesamte Dokument gesendet wird |
| `MaxRangeReversals` | svd | 2.4 | DVD | Erlaubte Anzahl von Range-Richtungswechseln, bevor das gesamte Dokument gesendet wird |
| `MaxRequestsPerChild` | s | | 6 | maximale Anzahl von Anfragen, die ein Child-Prozess nacheinander abarbeitet |
| `MaxRequestsPerThread` | s | | 6 | maximale Anzahl von Anfragen, die ein Thread nacheinander abarbeitet |
| `MaxSpareServers` | s | | 6 | maximale Anzahl unbeschäftigter Worker-Prozesse |
| `MaxSpareThreads` | s | | 6 | maximale Anzahl von Reserve-Threads |
| `MaxThreads` | s | | 6 | maximale Anzahl von Worker-Threads (NetWare) |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| MCacheMaxObjectCount | s | 2.2 | 13 | maximale Anzahl der Elemente im Cache |
| MCacheMaxObjectSize | s | 2.2 | 13 | maximale Größe eines Dokuments im Cache |
| MCacheMaxStreaming-Buffer | s | 2.2 | 13 | maximale Puffergröße, bevor das Caching eines Dokuments abgebrochen wird |
| MCacheMinObjectSize | s | 2.2 | 13 | Mindestgröße für ein Objekt im Cache |
| MCacheRemovalAlgorithm | s | 2.2 | 13 | Algorithmus zur Auswahl der Dokumente, die aus dem Cache entfernt werden |
| MCacheSize | s | 2.2 | 13 | maximale Speichergröße für den Cache |
| MetaDir | svdh | | 7 | Name des Verzeichnisses mit CERN-Metadaten |
| MetaFiles | svdh | | 7 | CERN-Metainformationen ein-/ausschalten |
| MetaSuffix | svdh | | 7 | Dateiendung für Dateien mit CERN-Metainformationen |
| MimeMagicFile | sv | | 7 | MIME-Magic-Funktionalität auf der Basis der angegebenen Datei einschalten |
| MinSpareServers | s | | 6 | Mindestzahl von Reserve-Child-Prozessen |
| MinSpareThreads | s | | 6 | Mindestzahl von Reserve-Threads |
| MMapFile | s | | 12 | Liste von Dateien, die beim Start geladen werden sollen |
| ModemStandard | | 2.4 | DVD | |
| ModMimeUsePathInfo | d | | 7 | Bestimmt, dass mod_mime Path_Info-Daten beachten soll. |
| MultiviewsMatch | svdh | | 7 | Dateitypen, nach denen für MultiViews gesucht wird |
| Mutex | s | 2.4 | 6 | Allgemeine Methode für Serialisierungsaufgaben |
| NameVirtualHost | s | | 7 | IP-Adresse für namensbasierte virtuelle Hosts |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| NoProxy | sv | | 13 | Adressen, für die kein Proxy verwendet wird |
| NWSSLTrustedCerts | s | | 10 | Liste zusätzlicher Client-Zertifikate (NetWare) |
| NWSSLUpgradeable | s | | 10 | im laufenden Betrieb auf SSL umschalten (NetWare) |
| Options | svdh | | 6 | Server-Funktionen für ein bestimmtes Verzeichnis |
| Order | dh | | 6 | Reihenfolge, in der Allow und Deny beachtet werden |
| OutputSed | dh | 2.4 | DVD | sed-Befehl zur Filterung von Ausgabedaten |
| PassEnv | svdh | | 14 | Shell-Umgebungsvariablen an CGI-Skripte weitergeben |
| PidFile | s | | 6 | Datei für die PID des Apache-Hauptprozesses |
| ProtocolEcho | sv | | 17 | mod_echo ein-/ausschalten |
| <Proxy> | sv | | 13 | Einstellungen für Proxy-Ressourcen |
| ProxyAddHeaders | svd | 2.4 | 13 | Proxy-Daten zu X-Forwarded-For-Headern hinzufügen |
| ProxyBadHeader | sv | | 13 | Bestimmt den Umgang mit ungültigen Antwort-Headern. |
| ProxyBlock | sv | | 13 | Ressourcen, für die Apache nicht als Proxy fungieren soll |
| ProxyDomain | sv | | 13 | Domain-Name, der bei Proxy-Anfragen im Intranet automatisch angehängt wird |
| ProxyErrorOverride | sv | | 13 | Bestimmt, ob der Proxy Fehlerseiten überschreiben soll. |
| ProxyExpressDBMFile | sv | 2.4 | 13 | DBM-Datei mit Massen-Proxy-Einstellungen |
| ProxyExpressDBMType | sv | 2.4 | 13 | Typ der DBM-Datei für mod_proxy_express |
| ProxyExpressEnable | sv | 2.4 | 13 | ProxyExpress-Funktionalität aktivieren |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| ProxyFtpDirCharset | svd | | 13 | Zeichensatz für Verzeichnislistings des FTP-Proxys |
| ProxyFtpEscapeWildcards | svd | 2.4 | 13 | Escaping von Platzhaltern in Dateinamen, die an den FTP-Server gesendet werden |
| ProxyFtpListOnWildcard | svd | 2.4 | 13 | Eine Trefferliste anzeigen, wenn FTP-Dateinamen Platzhalter enthalten |
| ProxyHTMLBufSize | svd | 2.4 | 13 | Vergrößerung des Puffers für Inline-Skripte und –Stylesheets |
| ProxyHTMLCharsetOut | svd | 2.4 | 13 | Ausgabezeichensatz für mod_proxy_html |
| ProxyHTMLDocType | svd | 2.4 | 13 | Doctype-Deklaration für ausgegebene Dokumente |
| ProxyHTMLEnable | svd | 2.4 | 13 | Funktionalität von mod_proxy_html aktivieren |
| ProxyHTMLEvents | svd | 2.4 | 13 | Attribute, die als Skript-Events betrachtet werden |
| ProxyHTMLExtended | svd | 2.4 | 13 | Bestimmt, ob Links in Inline-Skripten, Stylesheets und Event-Attributen verarbeitet werden |
| ProxyHTMLFixups | svd | 2.4 | 13 | Berichtigung einfacher HTML-Fehler |
| ProxyHTMLInterp | svd | 2.4 | 13 | Interpolation von ProxyHTMLURLMap-Regeln pro Anfrage |
| ProxyHTMLLinks | svd | 2.4 | 13 | HTML-Elemente, deren Attribute durch mod_proxy_html verändert werden sollen |
| ProxyHTMLStripComments | svd | 2.4 | 13 | HTML-Kommentare entfernen |
| ProxyHTMLURLMap | svd | 2.4 | 13 | Regeln für das Umschreiben von HTML-Links |
| ProxyIOBufferSize | sv | | 13 | Größe des Proxy-Datenpuffers |
| <ProxyMatch> | sv | | 13 | Einstellungen für Gruppen von Proxy-Ressourcen, auf die ein RegExp zutrifft |
| ProxyMaxForwards | sv | | 13 | Höchstzahl von Proxys, durch die eine Anfrage weitergeleitet wird |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| ProxyPass | svd | | 13 | Bildet entfernte URLs in den lokalen Verzeichnisbaum ab |
| ProxyPassInterpolateEnv | svd | | 13 | Umgebungsvariablen in Proxy-Anfrage-URLs interpolieren |
| ProxyPassMatch | svd | | 13 | Bildet entfernte URLs per RegExp in den lokalen Verzeichnisbaum ab. |
| ProxyPassReverse | svd | | 13 | URL im HTTP-Antwort-Header eines Reverse-Proxys anpassen |
| ProxyPassReverse CookieDomain | svd | | 13 | URL in Cookies eines Reverse-Proxys anpassen |
| ProxyPassReverse CookiePath | svd | | 13 | Pfad in Cookies eines Reverse-Proxys anpassen |
| ProxyPreserveHost | sv | | 13 | Host-Header der Anfrage an Reverse-Proxy durchreichen |
| ProxyReceiveBufferSize | sv | | 13 | Netzwerkpuffergröße für Proxy-Verbindungen |
| ProxyRemote | sv | | 13 | bestimmte Anfragen an einen anderen Proxy weiterreichen |
| ProxyRemoteMatch | sv | | 13 | Gruppen von Anfragen weiterreichen, die einem RegExp entsprechen |
| ProxyRequests | sv | | 13 | Proxy-Anfragen zulassen/verbieten |
| ProxySet | d | | 13 | Optionen für Balancer und BalancerMember setzen |
| ProxySourceAddress | sv | 2.4 | 13 | Lokale IP-Adresse für ausgehende Proxy-Anfragen |
| ProxyStatus | sv | | 13 | Load-Balancer-Status mittels mod_status veröffentlichen |
| ProxyTimeout | sv | | 13 | Timeout für Proxy-Anfragen |
| ProxyVia | sv | | 13 | Konfiguration des Via-Headers für Proxy-Antworten |
| ReadmeName | svdh | | 8 | Datei, die unter dem Verzeichnisindex eingefügt wird |
| ReceiveBufferSize | s | | 6 | Größe des TCP-Empfangspuffers |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| Redirect | svdh | | 8 | Sendet dem Client eine Umleitungsmitteilung. |
| RedirectMatch | svdh | | 8 | Sendet eine Umleitungsmitteilung, wenn die URL einem RegeExp entspricht. |
| RedirectPermanent | svdh | | 8 | Sendet eine Mitteilung über eine permanente Umleitung. |
| RedirectTemp | svdh | | 8 | Sendet eine Mitteilung über eine vorübergehende Umleitung. |
| ReflectorHeader | svdh | 2.4 | DVD | Anfrage-Header als Antwort-Header übernehmen |
| RemoteIPHeader | sv | 2.4 | DVD | Header-Feld, in dem Client-IP-Adressen geparst werden |
| RemoteIPInternalProxy | sv | 2.4 | DVD | Intranet-IP-Adressen, deren RemoteIPHeader-Werte als gültige Client-IP-Adressen betrachtet werden |
| RemoteIPInternal-ProxyList | sv | 2.4 | DVD | Datei mit Intranet-IP-Adressen, deren RemoteIPHeader-Werte als gültige Client-IP-Adressen betrachtet werden |
| RemoteIPProxiesHeader | sv | 2.4 | DVD | Header-Feld, in dem IP-Adressen beteiligter Hosts gesammelt werden |
| RemoteIPTrustedProxy | sv | 2.4 | DVD | Intranet-IP-Adressen, deren RemoteIPHeader-Werte (außer private IPs) als gültige Client-IP-Adressen betrachtet werden |
| RemoteIPTrusted-ProxyList | sv | 2.4 | DVD | Datei mit Intranet-IP-Adressen, deren RemoteIPHeader-Werte (außer private IPs) als gültige Client-IP-Adressen betrachtet werden |
| RemoveCharset | vdh | | 7 | Zeichensatzzuordnung für Dateien mit bestimmten Endungen entfernen |
| RemoveEncoding | vdh | | 7 | Komprimierungszuordnung für Dateien mit bestimmten Endungen entfernen |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| RemoveHandler | vdh | | 7 | Handler-Zuordnung für Dateien mit bestimmten Endungen entfernen |
| RemoveInputFilter | vdh | | 7 | Filterzuordnung für Dateien mit bestimmten Endungen entfernen |
| RemoveLanguage | vdh | | 7 | Sprachzuordnung für Dateien mit bestimmten Endungen entfernen |
| RemoveOutputFilter | vdh | | 7 | Ausgabefilterzuordnung für Dateien mit bestimmten Endungen entfernen |
| RemoveType | vdh | | 7 | MIME-Type-Zuordnung für Dateien mit bestimmten Endungen entfernen |
| RequestHeader | vdh | | 7 | HTTP-Anfrage-Header manipulieren |
| Require | dh | | 9 | Benutzer/Gruppen, die auf eine Ressource zugreifen dürfen; seit 2.4 alle Arten der Zugriffsberechtigung |
| <RequireAll> | dh | 2.4 | 9 | Container für mehrere Require-Direktiven, die alle erfüllt sein müssen |
| <RequireAny> | dh | 2.4 | 9 | Container für mehrere Require-Direktiven, von denen mindestens eine erfüllt sein muss |
| <RequireNone> | dh | 2.4 | 9 | Container für mehrere Require-Direktiven, von denen keine erfüllt sein darf |
| RewriteBase | dh | | 8 | Basis-URL für Verzeichnis-Rewrites |
| RewriteCond | svdh | | 8 | Bedingung, unter der Rewrite durchgeführt wird |
| RewriteEngine | svdh | | 8 | Rewrite-Funktionalität ein-/ausschalten |
| RewriteLock | s | | 8 | Lock-Datei zur RewriteMap-Synchronisation |
| RewriteLog | sv | | 8 | Log-Datei für die Protokollierung von Rewrite-Vorgängen |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| RewriteLogLevel | sv | | 8 | Dringlichkeitsstufe, ab der Rewrite-Vorgänge in der Log-Datei erfasst werden |
| RewriteMap | sv | | 8 | Schlägt einen Rewrite-Schlüssel in einer Map-Datei nach. |
| RewriteOptions | svdh | | 8 | zusätzliche Rewrite-Optionen |
| RewriteRule | svdh | | 8 | Definiert Rewrite-Regeln. |
| RLimitCPU | svdh | | 18 | maximale CPU-Nutzung für Prozesse, die durch Apache-Child-Prozesse erzeugt wurden |
| RLimitMEM | svdh | | 18 | maximale Speichermenge für Prozesse, die durch Apache-Child-Prozesse erzeugt wurden |
| RLimitNPROC | svdh | | 18 | maximale Anzahl von »Enkel«-Prozessen, die durch Apache-Child-Prozesse erzeugt werden dürfen |
| Satisfy | dh | | 9 | Legt fest, ob Host/IP-Kontrolle und Benutzeranmeldung jeweils allein ausreichen oder ob beides erfüllt sein muss. |
| ScoreBoardFile | s | | 6 | Pfad der Datei zur Koordination von Child-Prozessen |
| Script | svd | | 14 | CGI-Skript für eine bestimmte HTTP-Methode aktivieren |
| ScriptAlias | sv | | 14 | Verzeichnis-Alias mit CGI-Ausführung erstellen |
| ScriptAliasMatch | sv | | 14 | CGI-Verzeichnisgruppen-Alias durch RegExp definieren |
| ScriptInterpreterSource | svdh | | 14 | Bestimmt, wie der CGI-Interpreter gefunden wird (nur Windows). |
| ScriptLog | sv | | 14 | Pfad der CGI-Log-Datei |
| ScriptLogBuffer | sv | | 14 | Höchstzahl von Bytes eines PUT- oder POST-Bodys im CGI-Log |
| ScriptLogLength | sv | | 14 | maximale Größe der CGI-Log-Datei |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| ScriptSock | sv | | 14 | Pfad des UNIX-Domain-Sockets zur CGI-Kommunikation |
| SecureListen | s | | 10 | SSL-Verschlüsselung für den angegebenen TCP-Port aktivieren (NetWare) |
| SendBufferSize | s | | 6 | Größe des TCP-Sendepuffers |
| ServerAdmin | sv | | 6 | E-Mail-Adresse des Webmasters für Fehlermeldungsseiten |
| ServerAlias | v | | 12 | alternativer Hostname für einen namensbasierten virtuellen Host |
| ServerLimit | s | | 6 | absolute Höchstgrenze für die Anzahl von Prozessen |
| ServerName | sv | | 6 | Hostname und Port des Servers |
| ServerPath | v | | 6 | alternativer URL-Pfad eines namensbasierten virtuellen Hosts für HTTP/1.0-Clients |
| ServerRoot | s | | 6 | Stammverzeichnis des Webservers; enthält u. a. logs/ und conf/ |
| ServerSignature | svdh | | 6 | Fußzeile servergenerierter Dokumente |
| ServerTokens | s | | 6 | Konfiguration der Versionsangabe für den Antwort-Header Server |
| Session | svdh | 2.4 | 14 | Session-Funktionalität aktivieren |
| SessionCookieName | svdh | 2.4 | 14 | Name des RFC-2109-Cookies für die Session-Daten |
| SessionCookieName2 | svdh | 2.4 | 14 | Name des RFC-2965-Cookies für die Session-Daten |
| SessionCookieRemove | svdh | 2.4 | 14 | Session-Cookies aus den Headern eingehender Anfragen entfernen |
| SessionCryptoCipher | svdh | 2.4 | 14 | Algorithmus für die Verschlüsselung der Session-Daten |
| SessionCryptoDriver | s | 2.4 | 14 | Treiber für die Verschlüsselung der Session-Daten |
| SessionCryptoPassphrase | svdh | 2.4 | 14 | Passphrase für die Verschlüsselung der Session-Daten |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| SessionDBDCookieName | svdh | 2.4 | 14 | RFC-2109-Cookie für die Session-ID |
| SessionDBDCookieName2 | svdh | 2.4 | 14 | RFC-2965-Cookie für die Session-ID |
| SessionDBDCookieRemove | svdh | 2.4 | 14 | Session-ID-Cookies aus den Headern eingehender Anfragen entfernen |
| SessionDBDDeleteLabel | svdh | 2.4 | 14 | SQL-Abfrage zum Löschen von Session-Daten aus der Datenbank |
| SessionDBDInsertLabel | svdh | 2.4 | 14 | SQL-Abfrage zum Einfügen von Session-Daten in die Datenbank |
| SessionDBDPerUser | svdh | 2.4 | 14 | Session-Keys anhand von User-Login-Namen erstellen |
| SessionDBDSelectLabel | svdh | 2.4 | 14 | SQL-Abfrage zum Lesen von Session-Daten aus der Datenbank |
| SessionDBDUpdateLabel | svdh | 2.4 | 14 | SQL-Abfrage zum Ändern von Session-Daten in der Datenbank |
| SessionEnv | svdh | 2.4 | 14 | Bestimmt, ob die Session-Daten in der Umgebungsvariablen HTTP_SESSION bereitgestellt werden |
| SessionExclude | svdh | 2.4 | 14 | URL-Präfixe, für die die Session ignoriert wird |
| SessionHeader | svdh | 2.4 | 14 | Daten des angegebenen HTTP-Headers als Session-Daten speichern |
| SessionInclude | svdh | 2.4 | 14 | URL-Präfixe, für die die Session verwendet wird |
| SessionMaxAge | svdh | 2.4 | 14 | Maximales Alter der Session |
| SetEnv | svdh | | 14 | Umgebungsvariablen setzen |
| SetEnvIf | svdh | | 14 | Umgebungsvariablen je nach Bedingungen in der Anfrage setzen |
| SetEnvIfExpr | svdh | 2.4 | DVD | Umgebungsvariablen aufgrund eines ap_expr-Ausdrucks setzen |
| SetEnvIfNoCase | svdh | | 14 | Umgebungsvariablen je nach Bedingungen (ohne Unterschied bei Groß-/ Kleinschreibung) setzen |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| SetHandler | svdh | | 7 | Erzwingt die Verarbeitung aller Dateien im Kontext durch den angegebenen Handler. |
| SetInputFilter | svdh | | 16 | Legt Eingabefilter fest. |
| SetOutputFilter | svdh | | 16 | Legt Ausgabefilter fest. |
| SSIAccessEnable | dh | | 16 | Bestimmt, ob der SSI-Zugriffstest -A möglich ist. |
| SSIEndTag | sv | | 16 | Endmarkierung eines Include-Elements |
| SSIErrorMsg | svdh | | 16 | SSI-Fehlermeldung |
| SSILegacyExprParser | dh | 2.4 | 16 | Das herkömmliche Format für SSI-Ausdrücke statt ap_expr verwenden |
| SSIStartTag | sv | | 16 | Startmarkierung eines Include-Elements |
| SSITimeFormat | svdh | | 16 | Format für Datum und Uhrzeit in Includes |
| SSIUndefinedEcho | svdh | | 16 | SSI-Anzeige bei Ausgabe einer undefinierten Variablen |
| SSLCACertificateFile | sv | | 10 | Datei mit CA-Zertifikaten zur Client-Authentifizierung |
| SSLCACertificatePath | sv | | 10 | Verzeichnispfad für CA-Zertifikate zur Client-Authentifizierung |
| SSLCADNRequestFile | sv | | 10 | Datei mit akzeptierten Zertifikaten zur Client-Authentifizierung |
| SSLCADNRequestPath | sv | | 10 | Verzeichnispfad mit akzeptierten Zertifikaten zur Client-Authentifizierung |
| SSLCARevocationCheck | sv | 2.4 | 10 | CRL-basierte Überprüfung von Client-Zertifikaten aktivieren |
| SSLCARevocationFile | sv | | 10 | Datei mit ungültigen CA-CRLs |
| SSLCARevocationPath | sv | | 10 | Verzeichnispfad für ungültige CA-CRLs |
| SSLCertificateChainFile | sv | | 10 | Datei mit Server-CA-Zertifikaten |
| SSLCertificateFile | sv | | 10 | Datei mit X.509-Zertifikaten |
| SSLCertificateKeyFile | sv | | 10 | Datei mit privatem Schlüssel |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| SSLCipherSuite | svdh | | 10 | verfügbare Verschlüsselungsalgorithmen für SSL-Handshake |
| SSLCryptoDevice | s | | 10 | Einsatz eines externen Verschlüsselungsgeräts |
| SSLEngine | sv | | 10 | SSL-Funktionalität ein-/ausschalten |
| SSLHonorCipherOrder | sv | | 10 | Server-Präferenzen für Verschlüsselungsalgorithmen bevorzugen |
| SSLMutex | s | 2.2 | 10 | Semaphore (Lock) zum gegenseitigen Sperren von SSL-Operationen |
| SSLOptions | svdh | | 10 | spezielle SSL-Optionen |
| SSLOCSPDefaultResponder | sv | 2.4 | 10 | Standard-URL der Gegenstelle für OCSP-Überprüfungen |
| SSLOCSPEnable | sv | 2.4 | 10 | OCSP-Validierung von Client-Zertifikaten aktivieren |
| SSLOCSPOverride-Responder | sv | 2.4 | 10 | Verwendung der Standard-Gegenstelle erzwingen |
| SSLOCSPResponderTimeout | sv | 2.4 | 10 | Timeout für OCSP-Anfragen |
| SSLOCSPResonseMaxAge | sv | 2.4 | 10 | Akzeptiertes Höchstalter von OCSP-Antworten |
| SSLOCSPResponseTimeSkew | sv | 2.4 | 10 | maximal akzeptierte Zeitverzögerung für OCSP-Antworten |
| SSLPassPhraseDialog | s | | 10 | Passphrasen-Dialog für private Schlüssel |
| SSLProtocol | sv | | 10 | bevorzugtes SSL-Protokoll |
| SSLProxyCACertificate-File | sv | | 10 | Datei mit CA-Zertifikaten zur Remote-Authentifizierung |
| SSLProxyCACertificate-Path | sv | | 10 | Verzeichnispfad für CA-Zertifikate zur Remote-Authentifizierung |
| SSLProxyCARevocationFile | sv | | 10 | Datei mit CA-CRLs zur Remote-Authentifizierung |
| SSLProxyCARevocation-Path | sv | | 10 | Verzeichnispfad für CA-CRLs zur Remote-Authentifizierung |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| SSLProxyCipherSuite | sv | | 10 | verfügbare Verschlüsselungsalgorithmen für SSL-Proxy-Handshake |
| SSLProxyEngine | sv | | 10 | SSL-Proxy ein-/ausschalten |
| SSLProxyMachine CertificateFile | sv | | 10 | Datei mit Client-Zertifikaten und Schlüsseln für den SSL-Proxy |
| SSLProxyMachine CertificatePath | sv | | 10 | Verzeichnispfad mit Client-Zertifikaten und Schlüsseln für den SSL-Proxy |
| SSLProxyProtocol | sv | | 10 | bevorzugtes Proxy-SSL-Protokoll |
| SSLProxyVerify | svdh | | 10 | Art der Überprüfung des Remote-Zertifikats |
| SSLProxyVerifyDepth | svdh | | 10 | maximale Tiefe der Überprüfung von Remote-Zertifikaten |
| SSLRandomSeed | s | | 10 | Quelle für die Basiszahl (Random-Seed) des SSL-Zufallsgenerators |
| SSLRequire | dh | | 10 | Zugriff von einer booleschen Bedingung abhängig machen |
| SSLRequireSSL | dh | | 10 | Zugriff ausschließlich über SSL zulassen |
| SSLSessionCache | s | | 10 | Art des SSL-Session-Caches |
| SSLSessionCacheTimeout | sv | | 10 | Dauer bis zum Timeout einer SSL-Session |
| SSLStaplingCache | sv | 2.4 | 10 | OCSP-Stapling-Cache konfigurieren |
| SSLStaplingErrorCache-Timeout | sv | 2.4 | 10 | Timeout für ungültige Antworten im OCSP-Stapling-Cache |
| SSLStaplingFakeTryLater | sv | 2.4 | 10 | »tryLater«-Antworten für fehlgeschlagene OCSP-Stapling-Anfragen senden |
| SSLStaplingForceURL | sv | 2.4 | 10 | OCSP-Responder-URI aus der AIA-Erweiterung des Zertifikats überschreiben |
| SSLStaplingResponder-Timeout | sv | 2.4 | 10 | Timeout für OCSP-Stapling-Anfragen |
| SSLStaplingResponse-MaxAge | sv | 2.4 | 10 | Höchstalter für OCSP-Stapling-Antworten |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| SSLStaplingResponse-TimeSkew | sv | 2.4 | 10 | Maximale Zeitverzögerung für OCSP-Stapling-Antworten |
| SSLStaplingReturn-ResponderErrors | sv | 2.4 | 10 | OCSP-Stapling-Fehlermeldungen an den Client weitergeben |
| SSLStaplingStandard-CacheTimeout | sv | 2.4 | 10 | Timeout für Antworten im OCSP-Stapling-Cache |
| SSLUserName | sdh | | 10 | Benutzername für SSL |
| SSLUseStapling | sv | 2.4 | 10 | OCSP-Stapling aktivieren |
| SSLVerifyClient | svdh | | 10 | Art der Überprüfung von Client-Zertifikaten |
| SSLVerifyDepth | svdh | | 10 | maximale Tiefe der Überprüfung von Client-Zertifikaten |
| StartServers | s | | 6 | Anzahl von Child-Prozessen, die beim Start erzeugt werden |
| StartThreads | s | | 6 | Anzahl von Threads, die beim Start erzeugt werden |
| Substitute | dh | | 16 | Inhalte im Body der Server-Antwort aufgrund regulärer Ausdrücke ersetzen |
| SuexecUserGroup | sv | | 6 | Benutzer- und Gruppen-ID für SuEXEC-CGI-Skripte |
| ThreadLimit | s | | 6 | absolute Höchstzahl von Threads |
| ThreadsPerChild | s | | 6 | Anzahl der Threads pro Child-Prozess |
| ThreadStackSize | s | | 6 | Größe des Stacks pro Thread (NetWare) |
| TimeOut | s | | 6 | Wartezeit bis zum Abbruch einer Anfrage |
| TraceEnable | s | | 6 | TRACE-Anfragen ein-/ausschalten |
| TransferLog | sv | | 11 | Pfad für der Standard-Log-Datei |
| TypesConfig | s | | 7 | Pfad der Datei mime.types |
| UnDefine | s | 2.4 | 6 | Eine mittels Define oder der Startoption –D gesetzte Variablendefinition entfernen |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

| Direktive | Kontext | Vers. | Kap. | Beschreibung |
|---|---|---|---|---|
| UnsetEnv | svdh | | 16 | Entfernt eine Umgebungsvariable. |
| UseCanonicalName | svd | | 6 | Legt fest, wie Apache seinen Hostnamen und Port ermittelt. |
| UseCanonicalPhysicalPort | svd | | 6 | Bestimmt, ob UseCanonicalName den physischen TCP-Port berücksichtigt. |
| User | s | | 6 | User-ID, unter der Apache Anfragen verarbeitet |
| UserDir | sv | | 8 | Verzeichnisname der User-Webverzeichnisse |
| VirtualDocumentRoot | sv | | 12 | Stammverzeichnisse für die Websites virtueller Hosts nach Hostnamen |
| VirtualDocumentRootIP | sv | | 12 | Stammverzeichnisse für die Websites virtueller Hosts nach IP-Adressen |
| <VirtualHost> | s | | 12 | Direktiven für einen virtuellen Host |
| VirtualScriptAlias | sv | | 12 | CGI-Verzeichnisse für die Websites virtueller Hosts nach Hostnamen |
| VirtualScriptAliasIP | sv | | 12 | CGI-Verzeichnisse für die Websites virtueller Hosts nach IP-Adressen |
| Win32DisableAcceptEx | s | 2.2 | 6 | AcceptEx() deaktivieren (Windows) |
| XBitHack | svdh | | 16 | SSI-Direktiven mit gesetztem Execute-Bit verarbeiten |
| xml2EncAlias | s | 2.4 | 16 | Aliasnamen für Encoding-Werte |
| xml2EncDefault | svdh | 2.4 | 16 | Standard-Encoding, wenn das Encoding anderweitig nicht ermittelt werden kann |
| xml2StartParse | svdh | 2.4 | 16 | Ungültige Daten am Anfang von XML/(X)HTML-Dokumenten ignorieren |

**Tabelle B.1** Kurzübersicht über alle Apache-Konfigurationsdirektiven (Forts.)

*»Jeden anständigen Menschen verlangt es nach innerer und äußerer Ordnung, aber andererseits verträgt man auch nicht zu viel von ihr, ja eine vollkommene Ordnung wäre sozusagen der Ruin allen Fortschritts und Vergnügens.«*
*– Robert Musil*

# C Sonstige Tabellen

## C.1 MIME-Types

Eine der vollständigsten verfügbaren Listen von MIME-Types wird mit Apache mitgeliefert: Es handelt sich um die von `mod_mime` verwendete Datei *conf/mime.types*. Sie bildet die Grundlage der leicht erweiterten Tabelle C.1.

| MIME-Types | Dateiendung(en) |
|---|---|
| application/access | mdf |
| application/activemessage | |
| application/andrew-inset | ez |
| application/applefile | |
| application/atomicmail | |
| application/batch-smtp | |
| application/beep+xml | |
| application/bleeper | bleep |
| application/cals-1840 | |
| application/cnrp+xml | |
| application/commonground | |
| application/cpl+xml | |
| application/cu-seeme | csm cu |
| application/cybercash | |
| application/dca-rft | |
| application/dec-dx | |
| application/dvcs | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen

| MIME-Types | Dateiendung(en) |
|---|---|
| application/edi-consent | |
| application/edi-x12 | |
| application/edifact | |
| application/eshop | |
| application/excel | xls xlt |
| application/font-tdpfr | pfr |
| application/futuresplash | spl |
| application/ghostview | |
| application/hep | hep |
| application/http | |
| application/hyperstudio | stk |
| application/iges | |
| application/imagemap | imagemap imap |
| application/index.cmd | |
| application/index.obj | |
| application/index.response | |
| application/index.vnd | |
| application/index | |
| application/iotp | |
| application/ipp | |
| application/isup | |
| application/lotus-123 | wks |
| application/mac-binhex40 | hqx |
| application/mac-compactpro | cpt |
| application/macwriteii | |
| application/marc | |
| application/mathcad | mcd |
| application/mathematica-old | |
| application/mathematica | |
| application/mathml+xml | mathml |
| application/msword | doc dot wrd |
| application/news-message-id | |
| application/news-transmission | |
| application/ocsp-request | |
| application/ocsp-response | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| application/octet-stream | bin dms lha lzh exe class ani pgp so dll |
| application/oda | oda |
| application/ogg | ogg |
| application/pagemaker | pm5 pt5 pm |
| application/parityfec | |
| application/pdf | pdf |
| application/pgp-encrypted | |
| application/pgp-keys | |
| application/pgp-signature | |
| application/pgp | |
| application/pkcs10 | |
| application/pkcs7-mime | |
| application/pkcs7-signature | |
| application/pkix-cert | |
| application/pkix-crl | |
| application/pkixcmp | |
| application/postscript | ai eps ps |
| application/postscript | ps-z |
| application/powerpoint | ppt pps pot |
| application/prs.alvestrand.titrax-sheet | |
| application/prs.cww | cw cww |
| application/prs.nprend | rnd rct |
| application/prs.plucker | |
| application/qsig | |
| application/quicktimeplayer | qtl |
| application/rdf+xml | rdf |
| application/reginfo+xml | |
| application/remote-printing | |
| application/riscos | |
| application/rtf | rtf |
| application/sdp | |
| application/set-payment-initiation | |
| application/set-payment | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| application/set-registration-initiation | |
| application/set-registration | |
| application/sgml-open-catalog | soc |
| application/sgml | |
| application/sieve | |
| application/slate | |
| application/smil | smi smil |
| application/srgs+xml | grxml |
| application/srgs | gram |
| application/timestamp-query | |
| application/timestamp-reply | |
| application/toolbook | tbk |
| application/tve-trigger | |
| application/vemmi | |
| application/VMSBACKUP | bck |
| application/vnd.3gpp.pic-bw-large | plb |
| application/vnd.3gpp.pic-bw-small | psb |
| application/vnd.3gpp.pic-bw-var | pvb |
| application/vnd.3gpp.sms | sms |
| application/vnd.3m.post-it-notes | |
| application/vnd.accpac.simply.aso | |
| application/vnd.accpac.simply.imp | |
| application/vnd.acucobol | |
| application/vnd.acucorp | atc acutc |
| application/vnd.adobe.xfdf | xfdf |
| application/vnd.aether.imp | |
| application/vnd.amiga.ami | |
| application/vnd.amiga.amu | ami |
| application/vnd.anser-web-certificate-issue-initiation | |
| application/vnd.anser-web-funds-transfer-initiation | |
| application/vnd.audiograph | |
| application/vnd.blueice.multipass | mpm |
| application/vnd.bmi | |
| application/vnd.businessobjects | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| application/vnd.canon-cpdl | |
| application/vnd.canon-lips | |
| application/vnd.cinderella | cdy |
| application/vnd.claymore | |
| application/vnd.commerce-battelle | |
| application/vnd.commonspace | |
| application/vnd.comsocaller | |
| application/vnd.contact.cmsg | |
| application/vnd.cosmocaller | |
| application/vnd.criticaltools.wbs+xml | |
| application/vnd.ctc-posml | |
| application/vnd.cups-postscript | |
| application/vnd.cups-raster | |
| application/vnd.cups-raw | |
| application/vnd.curl | curl |
| application/vnd.cybank | |
| application/vnd.data-vision.rdz | rdz |
| application/vnd.dna | |
| application/vnd.dpgraph | |
| application/vnd.dreamfactory | dfac |
| application/vnd.dxr | |
| application/vnd.ecdis-update | |
| application/vnd.ecowin.chart | |
| application/vnd.ecowin.filerequest | |
| application/vnd.ecowin.fileupdate | |
| application/vnd.ecowin.seriesrequest | |
| application/vnd.ecowin.seriesupdate | |
| application/vnd.ecowin.series | |
| application/vnd.enliven | |
| application/vnd.epson.esf | |
| application/vnd.epson.msf | |
| application/vnd.epson.quickanime | |
| application/vnd.epson.salt | |
| application/vnd.epson.ssf | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| application/vnd.ericsson.quickcall | |
| application/vnd.eudora.data | |
| application/vnd.fdf | |
| application/vnd.ffsns | |
| application/vnd.fints | |
| application/vnd.flographit | |
| application/vnd.framemaker | |
| application/vnd.fsc.weblauch | fsc |
| application/vnd.fujitsu.oasys2 | |
| application/vnd.fujitsu.oasys3 | |
| application/vnd.fujitsu.oasysgp | |
| application/vnd.fujitsu.oasysprs | |
| application/vnd.fujitsu.oasys | |
| application/vnd.fujixerox.ddd | |
| application/vnd.fujixerox.docuworks.binder | |
| application/vnd.fujixerox.docuworks | |
| application/vnd.fut-misnet | |
| application/vnd.grafeq | |
| application/vnd.groove-account | |
| application/vnd.groove-help | |
| application/vnd.groove-identity-message | |
| application/vnd.groove-injector | |
| application/vnd.groove-tool-message | |
| application/vnd.groove-tool-template | |
| application/vnd.groove-vcard | |
| application/vnd.hbci | hbci hbc kom upa pkd bpd |
| application/vnd.hhe.lesson-player | les |
| application/vnd.hp-hpgl | plt hpgl |
| application/vnd.hp-hpid | |
| application/vnd.hp-hps | |
| application/vnd.hp-pclxl | |
| application/vnd.hp-pcl | |
| application/vnd.httphone | |
| application/vnd.hzn-3d-crossword | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| application/vnd.ibm.afplinedata | |
| application/vnd.ibm.electronic-media | emm |
| application/vnd.ibm.minipay | |
| application/vnd.ibm.modcap | |
| application/vnd.ibm.rights-management | |
| application/vnd.ibm.secure-container | sc |
| application/vnd.informix-visionary | |
| application/vnd.intercon.formnet | |
| application/vnd.intertrust.digibox | |
| application/vnd.intertrust.nncp | |
| application/vnd.intu.qbo | |
| application/vnd.intu.qfx | |
| application/vnd.irepository.package+xml | irp |
| application/vnd.is-xpr | |
| application/vnd.japannet-directory-service | |
| application/vnd.japannet-jpnstore-wakeup | |
| application/vnd.japannet-payment-wakeup | |
| application/vnd.japannet-registration-wakeup | |
| application/vnd.japannet-registration | |
| application/vnd.japannet-setstore-wakeup | |
| application/vnd.japannet-verification-wakeup | |
| application/vnd.japannet-verification | |
| application/vnd.jisp | jisp |
| application/vnd.kde.karbon | karbon |
| application/vnd.kde.kchart | chrt |
| application/vnd.kde.kformula | kfo |
| application/vnd.kde.kivio | flw |
| application/vnd.kde.kontour | kon |
| application/vnd.kde.kpresenter | ksp |
| application/vnd.kde.kspread | ksp |
| application/vnd.kde.kword | kwd kwt |
| application/vnd.kenameaapp | |
| application/vnd.kenameapp | htke |
| application/vnd.koan | |
| application/vnd.liberty-request+xml | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| application/vnd.llamagraphics.life-balance.desktop | lbd |
| application/vnd.llamagraphics.life-balance.exchange+xml | |
| application/vnd.lotus-1-2-3 | |
| application/vnd.lotus-approach | |
| application/vnd.lotus-freelance | |
| application/vnd.lotus-notes | |
| application/vnd.lotus-organizer | |
| application/vnd.lotus-screencam | |
| application/vnd.lotus-wordpro | |
| application/vnd.mcd | |
| application/vnd.mediastation.cdkey | |
| application/vnd.meridian-slingshot | |
| application/vnd.micrografx.flo | flo |
| application/vnd.micrografx.igx | igx |
| application/vnd.mif | mif |
| application/vnd.minisoft-hp3000-save | |
| application/vnd.mitsubishi.misty-guard.trustweb | |
| application/vnd.mobius.daf | |
| application/vnd.mobius.dis | |
| application/vnd.mobius.mbk | |
| application/vnd.mobius.mqy | |
| application/vnd.mobius.msl | |
| application/vnd.mobius.plc | |
| application/vnd.mobius.txf | |
| application/vnd.mophun.application | mpn |
| application/vnd.mophun.certificate | mpc |
| application/vnd.motorola.flexsuite.adsi | |
| application/vnd.motorola.flexsuite.fis | |
| application/vnd.motorola.flexsuite.gotap | |
| application/vnd.motorola.flexsuite.kmr | |
| application/vnd.motorola.flexsuite.ttc | |
| application/vnd.motorola.flexsuite.wem | |
| application/vnd.motorola.flexsuite | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| application/vnd.mozilla.xul+xml | xul |
| application/vnd.ms-access | mda mdb mde |
| application/vnd.ms-artgalry | cil |
| application/vnd.ms-asf | asf |
| application/vnd.ms-excel | xls xlt |
| application/vnd.ms-lrm | lrm |
| application/vnd.ms-powerpoint | ppt pps pot |
| application/vnd.ms-project | mpp |
| application/vnd.ms-tnef | |
| application/vnd.ms-works | |
| application/vnd.ms-wpl | wpl |
| application/vnd.mseq | mseq |
| application/vnd.msign | |
| application/vnd.music-niff | |
| application/vnd.musician | |
| application/vnd.netfpx | |
| application/vnd.noblenet-directory | |
| application/vnd.noblenet-sealer | |
| application/vnd.noblenet-web | |
| application/vnd.novadigm.edm | |
| application/vnd.novadigm.edx | |
| application/vnd.novadigm.ext | |
| application/vnd.obn | |
| application/vnd.osa.netdeploy | |
| application/vnd.palm | prc pdb pqa oprc |
| application/vnd.pg.format | |
| application/vnd.pg.osasli | |
| application/vnd.powerbuilder6-s | |
| application/vnd.powerbuilder6 | |
| application/vnd.powerbuilder7-s | |
| application/vnd.powerbuilder75-s | |
| application/vnd.powerbuilder75 | |
| application/vnd.powerbuilder7 | |
| application/vnd.previewsystems.box | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| application/vnd.publishare-delta-tree | |
| application/vnd.pvi.ptid1 | pti ptid |
| application/vnd.pwg-multiplexed | |
| application/vnd.pwg-xhtml-print+xml | |
| application/vnd.quark.quarkxpress | qxd qxt qwd qwt qxl qxb |
| application/vnd.rapid | |
| application/vnd.s3sms | |
| application/vnd.sealed.net | |
| application/vnd.seemail | |
| application/vnd.shana.informed.formdata | |
| application/vnd.shana.informed.formtemplate | |
| application/vnd.shana.informed.interchange | |
| application/vnd.shana.informed.package | |
| application/vnd.smaf | mmf |
| application/vnd.sss-cod | |
| application/vnd.sss-dtf | |
| application/vnd.sss-ntf | |
| application/vnd.street-stream | |
| application/vnd.svd | |
| application/vnd.swiftview-ics | |
| application/vnd.triscape.mxs | |
| application/vnd.trueapp | |
| application/vnd.truedoc | |
| application/vnd.ufdl | |
| application/vnd.uplanet.alert-wbxml | |
| application/vnd.uplanet.alert | |
| application/vnd.uplanet.bearer-choice-wbxml | |
| application/vnd.uplanet.bearer-choice | |
| application/vnd.uplanet.cacheop-wbxml | |
| application/vnd.uplanet.cacheop | |
| application/vnd.uplanet.channel-wbxml | |
| application/vnd.uplanet.channel | |
| application/vnd.uplanet.list-wbxml | |
| application/vnd.uplanet.listcmd-wbxml | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| application/vnd.uplanet.listcmd | |
| application/vnd.uplanet.list | |
| application/vnd.uplanet.signal | |
| application/vnd.vcx | |
| application/vnd.vectorworks | |
| application/vnd.vidsoft.vidconference | vsc |
| application/vnd.visionary | vis |
| application/vnd.visio | vsd vst vsw vss |
| application/vnd.vividence.scriptfile | |
| application/vnd.vsf | |
| application/vnd.wap.sic | sic |
| application/vnd.wap.slc | slc |
| application/vnd.wap.wbxml | wbxml |
| application/vnd.wap.wmlc | wmlc |
| application/vnd.wap.wmlscriptc | wmlsc |
| application/vnd.webturbo | |
| application/vnd.wrq-hp3000-labelled | |
| application/vnd.wt.stf | |
| application/vnd.wv.csp+wbxml | |
| application/vnd.xara | |
| application/vnd.xfdl | |
| application/vnd.yamaha.hv-dic | |
| application/vnd.yamaha.hv-script | |
| application/vnd.yamaha.hv-voice | |
| application/vnd.yellowriver-custom-menu | |
| application/voicexml+xml | vxml |
| application/watcherinfo+xml | |
| application/whoispp-query | |
| application/whoispp-response | |
| application/wita | |
| application/wordperfect5.1 | wp5 |
| application/wordperfect6.1 | wp6 |
| application/wordperfectd | wpd |
| application/wordperfect | wp |
| application/x-123 | wk |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| application/x-bcpio | bcpio |
| application/x-bzip2 | bz2 |
| application/x-cdlink | vcd |
| application/x-chess-pgn | pgn |
| application/x-compress | z Z |
| application/x-cpio | cpio |
| application/x-csh | csh |
| application/x-debian-package | deb |
| application/x-director | dcr dir dxr |
| application/x-dvi | dvi |
| application/x-futuresplash | spl |
| application/x-gtar | gtar tgz tbz2 tbz |
| application/x-gunzip | |
| application/x-gzip | |
| application/x-gzip | gz |
| application/x-hdf | hdf |
| application/x-httpd-cgi | cgi |
| application/x-httpd-php | phtml pht php |
| application/x-ica | ica |
| application/x-javascript | js |
| application/x-koan | skp skd skt skm |
| application/x-latex | latex |
| application/x-maker | frm maker frame fm fb book fbdoc |
| application/x-mif | mif |
| application/x-msdos-program | com bat exe |
| application/x-msdownload | exe |
| application/x-netcdf | nc cdf |
| application/x-ns-proxy-autoconfig | pac |
| application/x-perl | pl pm |
| application/x-rar-compressed | rar |
| application/x-shar | shar |
| application/x-shockwave-flash | swf |
| application/x-sh | sh |
| application/x-spss | sav spp sbs sps spo |
| application/x-stuffit | sit |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| application/x-sv4cpio | sv4cpio |
| application/x-sv4crc | sv4crc |
| application/x-tar | tar |
| application/x-tcl | tcl |
| application/x-texinfo | texinfo texi |
| application/x-tex | tex |
| application/x-troff-man | man |
| application/x-troff-me | me |
| application/x-troff-ms | ms |
| application/x-troff | t tr roff |
| application/x-ustar | ustar |
| application/x-wais-source | src |
| application/x-Wingz | wz |
| application/x-x509-ca-cert | crt |
| application/x400-bp | |
| application/xhtml+xml | xhtml xht |
| application/xml-dtd | dtd |
| application/xml-external-parsed-entity | |
| application/xml | xml xsl |
| application/xslt+xml | xslt |
| application/zip | zip |
| audio/32kadpcm | |
| audio/amr-wb | awb |
| audio/amr | amr |
| audio/basic | au snd |
| audio/cn | |
| audio/dat12 | |
| audio/dsr-es201108 | |
| audio/dvi4 | |
| audio/evrc0 | |
| audio/evrc | |
| audio/g.722.1 | |
| audio/g722 | |
| audio/g723 | |
| audio/g726-16 | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| audio/g726-24 | |
| audio/g726-32 | |
| audio/g726-40 | |
| audio/g728 | |
| audio/g729D | |
| audio/g729E | |
| audio/g729 | |
| audio/gsm-efr | |
| audio/gsm | |
| audio/l16 | l16 |
| audio/l20 | |
| audio/l24 | |
| audio/l8 | |
| audio/lpc | |
| audio/midi | mid midi kar |
| audio/mp4a-latm | |
| audio/mpa-robust | |
| audio/mpa | |
| audio/mpeg | mpga mp2 mp3 |
| audio/parityfec | |
| audio/pcma | |
| audio/pcmu | |
| audio/prs.sid | sid psid |
| audio/qcelp | |
| audio/red | |
| audio/smv0 | |
| audio/smv | |
| audio/telephone-event | |
| audio/tone | |
| audio/vdvi | |
| audio/vnd.3gpp.iufp | |
| audio/vnd.cisco.nse | |
| audio/vnd.cns.anp1 | |
| audio/vnd.cns.inf1 | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| audio/vnd.digital-winds | eol |
| audio/vnd.everad.plj | plj |
| audio/vnd.lucent.voice | lvp |
| audio/vnd.nortel.vbk | vbk |
| audio/vnd.nuera.ecelp4800 | ecelp4800 |
| audio/vnd.nuera.ecelp7470 | ecelp7470 |
| audio/vnd.nuera.ecelp9600 | ecelp9600 |
| audio/vnd.octel.sbc | |
| audio/vnd.qcelp | qcp |
| audio/vnd.rhetorex.32kadpcm | |
| audio/vnd.vmx.cvsd | |
| audio/x-aiff | aif aiff aifc |
| audio/x-alaw-basic | |
| audio/x-mpegurl | m3u |
| audio/x-pn-realaudio-plugin | rpm |
| audio/x-pn-realaudio | rm ram |
| audio/x-realaudio | ra |
| audio/x-wav | wav |
| chemical/x-pdb | pdb |
| chemical/x-xyz | xyz |
| drawing/dwf | dwf |
| image/bmp | bmp |
| image/cgm | cgm |
| image/g3fax | |
| image/gif | gif |
| image/ief | ief |
| image/jpeg | jpeg jpg jpe |
| image/naplps | |
| image/png | png |
| image/prs.btif | |
| image/prs.pti | |
| image/svg+xml | svg |
| image/t38 | |
| image/tiff-fx | |
| image/tiff | tiff tif |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| image/vnd.cns.inf2 | |
| image/vnd.dgn | dgn |
| image/vnd.djvu | djvu djv |
| image/vnd.dwg | dwg |
| image/vnd.dxf | |
| image/vnd.fastbidsheet | |
| image/vnd.fpx | |
| image/vnd.fst | |
| image/vnd.fujixerox.edmics-mmr | |
| image/vnd.fujixerox.edmics-rlc | |
| image/vnd.glocalgraphics.pgb | pgb |
| image/vnd.mix | |
| image/vnd.ms-modi | |
| image/vnd.net-fpx | |
| image/vnd.svf | |
| image/vnd.wap.wbmp | wbmp |
| image/vnd.xiff | |
| image/x-cmu-raster | ras |
| image/x-icon | ico |
| image/x-portable-anymap | pnm |
| image/x-portable-bitmap | pbm |
| image/x-portable-graymap | pgm |
| image/x-portable-pixmap | ppm |
| image/x-rgb | rgb |
| image/x-xbitmap | xbm |
| image/x-xpixmap | xpm |
| image/x-xwindowdump | xwd |
| message/delivery-status | |
| message/disposition-notification | |
| message/external-body | |
| message/http | |
| message/news | |
| message/partial | |
| message/rfc822 | |
| message/s-http | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| message/sipfrag | |
| message/sip | |
| model/iges | igs iges |
| model/mesh | msh mesh silo |
| model/vnd.dwf | |
| model/vnd.flatland.3dml | |
| model/vnd.gdl | |
| model/vnd.gs-gdl | |
| model/vnd.gtw | |
| model/vnd.mts | |
| model/vnd.parasolid.transmit.binary | x_b xmt_bin |
| model/vnd.parasolid.transmit.text | x_t xmt_txt |
| model/vnd.vtu | |
| model/vrml | wrl vrml |
| multipart/alternative | |
| multipart/appledouble | |
| multipart/byteranges | |
| multipart/digest | |
| multipart/encrypted | |
| multipart/form-data | |
| multipart/header-set | |
| multipart/mixed | |
| multipart/parallel | |
| multipart/related | |
| multipart/report | |
| multipart/signed | |
| multipart/voice-message | |
| text/calendar | ics ifb |
| text/comma-separated-values | csv |
| text/css | css |
| text/directory | |
| text/enriched | |
| text/html | html htm htmlx shtml htx |
| text/parityfec | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| text/plain | txt asc c cc h hh cpp hpp dat hlp |
| text/prs.lines.tag | |
| text/rfc822-headers | |
| text/richtext | rtx |
| text/rtf | rtf |
| text/sgml | sgml sgm |
| text/t140 | |
| text/tab-separated-values | tsv |
| text/uri-list | |
| text/vnd.abc | |
| text/vnd.curl | |
| text/vnd.dmclientscript | |
| text/vnd.flatland.3dml | |
| text/vnd.fly | |
| text/vnd.fmi.flexstor | |
| text/vnd.in3d.3dml | |
| text/vnd.in3d.spot | |
| text/vnd.iptc.newsml | |
| text/vnd.iptc.nitf | |
| text/vnd.latex-z | |
| text/vnd.motorola.reflex | |
| text/vnd.ms-mediapackage | |
| text/vnd.net2phone.commcenter.command | ccc |
| text/vnd.sun.j2me.app-descriptor | jad |
| text/vnd.wap.si | si |
| text/vnd.wap.wmlscript | wmls |
| text/vnd.wap.wml | wml |
| text/x-setext | etx |
| text/x-sgml | sgml sgm |
| text/x-vCalendar | vcs |
| text/x-vCard | vcf |
| text/xml-external-parsed-entity | |
| text/xml | xml dtd |
| video/bmpeg | |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

| MIME-Types | Dateiendung(en) |
|---|---|
| video/bt656 | |
| video/celb | |
| video/dl | dl |
| video/dv | |
| video/fli | fli |
| video/gl | gl |
| video/h261 | |
| video/h263-1998 | |
| video/h263-2000 | |
| video/h263 | |
| video/jpeg | |
| video/mp1s | |
| video/mp2p | |
| video/mp2t | |
| video/mp4v-es | |
| video/mpeg | mp2 mpe mpeg mpg |
| video/mpv | |
| video/nv | |
| video/parityfec | |
| video/pointer | |
| video/quicktime | qt mov |
| video/smpte292m | |
| video/vnd.fvt | fvt |
| video/vnd.motorola.videop | |
| video/vnd.motorola.video | |
| video/vnd.mpegurl | mxu m4u |
| video/vnd.nokia.interleaved-multimedia | nim |
| video/vnd.objectvideo | mp4 |
| video/vnd.vivo | |
| video/x-ms-asf | asf asx |
| video/x-msvideo | avi |
| video/x-sgi-movie | movie |
| x-conference/x-cooltalk | ice |
| x-world/x-vrml | wrl vrml |

**Tabelle C.1** Übersicht über die meisten zurzeit bekannten MIME-Types und ihre zugehörigen Dateiendungen (Forts.)

## C.2 Sprachcodes nach ISO

Die Sprachkürzel in Tabelle C.2 werden beispielsweise in HTTP-Headern wie `Accept-Language` und `Content-Language` verwendet, die bei der Content-Negotiation eine Rolle spielen.

| Kürzel | Sprache |
|---|---|
| aa | Afar |
| ab | Abchasisch |
| af | Afrikaans |
| am | Amharisch |
| ar | Arabisch |
| as | Assamesisch |
| ay | Aimara |
| az | Aserbaidschanisch |
| ba | Baschkirisch |
| be | Weißrussisch |
| bg | Bulgarisch |
| bh | Biharisch |
| bi | Bislama |
| bn | Bengalisch |
| bo | Tibetisch |
| br | Bretonisch |
| ca | Katalanisch |
| co | Korsisch |
| cs | Tschechisch |
| cy | Walisisch |
| da | Dänisch |
| de | Deutsch |
| dz | Bhutanisch |
| el | Griechisch |
| en | Englisch |
| eo | Esperanto |
| es | Spanisch |
| et | Estnisch |
| eu | Baskisch |
| fa | Persisch |

**Tabelle C.2** Sprachkürzel nach ISO 639-1

| Kürzel | Sprache |
|---|---|
| fi | Finnisch |
| fj | Fidschianisch |
| fo | Färöisch |
| fr | Französisch |
| fy | Friesisch |
| ga | Irisch |
| gd | Schottisches Gälisch |
| gl | Galizisch |
| gn | Guarani |
| gu | Gudschrati |
| ha | Haussa |
| he | Hebräisch |
| hi | Hindi |
| hr | Kroatisch |
| hu | Ungarisch |
| hy | Armenisch |
| ia | Interlingua |
| id | Indonesisch |
| ie | Interlingue |
| ik | Inupiak |
| is | Isländisch |
| it | Italienisch |
| iu | Inuktitut |
| ja | Japanisch |
| jw | Javanisch |
| ka | Georgisch |
| kk | Kasachisch |
| kl | Eskimoisch |
| km | Kambodschanisch |
| kn | Kannada |
| ko | Koreanisch |
| ks | Kashmiri |
| ku | Kurdisch |
| ky | Kirgisisch |
| la | Lateinisch |

**Tabelle C.2** Sprachkürzel nach ISO 639-1 (Forts.)

| Kürzel | Sprache |
|---|---|
| ln | Lingala |
| lo | Laotisch |
| lt | Litauisch |
| lv | Lettisch |
| mg | Madagassisch |
| mi | Maori |
| mk | Mazedonisch |
| ml | Malajalam |
| mn | Mongolisch |
| mo | Moldauisch |
| mr | Marati |
| ms | Malaiisch |
| mt | Maltesisch |
| my | Birmanisch |
| na | Nauruisch |
| ne | Nepalesisch |
| nl | Niederländisch |
| no | Norwegisch |
| oc | Okzitanisch |
| om | Oromo |
| or | Orija |
| pa | Pandschabisch |
| pl | Polnisch |
| ps | Paschtu |
| pt | Portugiesisch |
| qu | Quechua |
| rm | Rätoromanisch |
| rn | Kirundi |
| ro | Rumänisch |
| ru | Russisch |
| rw | Kinyarwanda |
| sa | Sanskrit |
| sd | Sindi |
| sg | Sango |
| sh | Serbokroatisch |

**Tabelle C.2** Sprachkürzel nach ISO 639-1 (Forts.)

| Kürzel | Sprache |
|---|---|
| si | Singhalesisch |
| sk | Slowakisch |
| sl | Slowenisch |
| sm | Samoanisch |
| sn | Shona |
| so | Somali |
| sq | Albanisch |
| sr | Serbisch |
| ss | Swazi |
| st | Sesotho |
| su | Sundanesisch |
| sv | Schwedisch |
| sw | Suaheli |
| ta | Tamilisch |
| te | Tegulu |
| tg | Tadschikisch |
| th | Thailändisch |
| ti | Tigrinja |
| tk | Turkmenisch |
| tl | Tagalog |
| tn | Tsuana |
| to | Tongaisch |
| tr | Türkisch |
| ts | Tsonga |
| tt | Tatarisch |
| tw | Twi |
| ug | Uigurisch |
| uk | Ukrainisch |
| ur | Urdu |
| uz | Usbekisch |
| vi | Vietnamesisch |
| vo | Volapuk |
| wo | Wolof |
| xh | Xhosa |
| yi | Jiddisch |

**Tabelle C.2** Sprachkürzel nach ISO 639-1 (Forts.)

| Kürzel | Sprache |
|---|---|
| yo | Joruba |
| za | Zhuang |
| zh | Chinesisch |
| zu | Zulu |

**Tabelle C.2** Sprachkürzel nach ISO 639-1 (Forts.)

## C.3 Zeichensätze

In Tabelle C.3 finden Sie eine Liste sämtlicher Zeichensätze. Sie können diese im Header `Content-Type` und einigen Apache-Konfigurationsdirektiven verwenden. Nähere Informationen finden Sie auch in den aufgeführten RFC-Dokumenten und unter *www.iana.org/assignments/character-sets* (alle diese Dokumente befinden sich auch im Verzeichnis `referenz` der beiliegenden DVD).

| Zeichensatz | Aliase | RFC |
|---|---|---|
| ANSI_X3.4-1968 | iso-ir-6, ANSI_X3.4-1986, ISO_646.irv:1991, ASCII, ISO646-US, US-ASCII, us, IBM367, cp367, csASCII | 1345 |
| ISO-10646-UTF-1 | csISO10646UTF1 | |
| ISO_646.basic:1983 | ref, csISO646basic1983 | 1345 |
| INVARIANT | csINVARIANT | 1345 |
| ISO_646.irv:1983 | iso-ir-2, irv, csISO2IntlRefVersion | 1345 |
| BS_4730 | iso-ir-4, ISO646-GB, gb, uk, csISO4UnitedKingdom | 1345 |
| NATS-SEFI | iso-ir-8-1, csNATSSEFI | 1345 |
| NATS-SEFI-ADD | iso-ir-8-2, csNATSSEFIADD | 1345 |
| NATS-DANO | iso-ir-9-1, csNATSDANO | 1345 |
| NATS-DANO-ADD | iso-ir-9-2, csNATSDANOADD | 1345 |
| SEN_850200_B | iso-ir-10, FI, ISO646-FI, ISO646-SE, se, csISO10Swedish | 1345 |
| SEN_850200_C | iso-ir-11, ISO646-SE2, se2, csISO11SwedishForNames | 1345 |
| KS_C_5601-1987 | iso-ir-149, KS_C_5601-1989, KSC_5601, korean, csKSC56011987 | 1345 |
| ISO-2022-KR | csISO2022KR | 1557 |
| EUC-KR | csEUCKR | 1557 |

**Tabelle C.3** Übersicht über sämtliche definierten Zeichensätze

| Zeichensatz | Aliase | RFC |
|---|---|---|
| ISO-2022-JP | csISO2022JP | 1468 |
| ISO-2022-JP-2 | csISO2022JP2 | 1554 |
| ISO-2022-CN | – | 1922 |
| ISO-2022-CN-EXT | – | 1922 |
| JIS_C6220-1969-jp | JIS_C6220-1969, iso-ir-13, katakana, x0201-7, csISO13JISC6220jp | 1345 |
| JIS_C6220-1969-ro | iso-ir-14, jp, ISO646-JP, csISO14JISC6220ro | 1345 |
| IT | iso-ir-15, ISO646-IT, csISO15Italian | 1345 |
| PT | iso-ir-16, ISO646-PT, csISO16Portuguese | 1345 |
| ES | iso-ir-17, ISO646-ES, csISO17Spanish | 1345 |
| greek7-old | iso-ir-18, csISO18Greek7Old | 1345 |
| latin-greek | iso-ir-19, csISO19LatinGreek | 1345 |
| DIN_66003 | iso-ir-21, de, ISO646-DE, csISO21German | 1345 |
| NF_Z_62-010_(1973) | iso-ir-25, ISO646-FR1, csISO25French | 1345 |
| Latin-greek-1 | iso-ir-27, csISO27LatinGreek1 | 1345 |
| ISO_5427 | iso-ir-37, csISO5427Cyrillic | 1345 |
| JIS_C6226-1978 | iso-ir-42, csISO42JISC62261978 | 1345 |
| BS_viewdata | iso-ir-47, csISO47BSViewdata | 1345 |
| INIS | iso-ir-49, csISO49INIS | 1345 |
| INIS-8 | iso-ir-50, csISO50INIS8 | 1345 |
| INIS-cyrillic | iso-ir-51, csISO51INISCyrillic | 1345 |
| ISO_5427:1981 | iso-ir-54, ISO5427Cyrillic1981 | 1345 |
| ISO_5428:1980 | iso-ir-55, csISO5428Greek | 1345 |
| GB_1988-80 | iso-ir-57, cn, ISO646-CN, csISO57GB1988 | 1345 |
| GB_2312-80 | iso-ir-58, chinese, csISO58GB231280 | 1345 |
| NS_4551-1 | iso-ir-60, ISO646-NO, no, csISO60DanishNorwegian, csISO60Norwegian1 | 1345 |
| NS_4551-2 | ISO646-NO2, iso-ir-61, no2, csISO61Norwegian2 | 1345 |
| NF_Z_62-010 | iso-ir-69, ISO646-FR, fr, csISO69French | 1345 |
| videotex-suppl | iso-ir-70, csISO70VideotexSupp1 | 1345 |
| PT2 | iso-ir-84, ISO646-PT2, csISO84Portuguese2 | 1345 |
| ES2 | iso-ir-85, ISO646-ES2, csISO85Spanish2 | 1345 |
| MSZ_7795.3 | iso-ir-86, ISO646-HU, hu, csISO86Hungarian | 1345 |

**Tabelle C.3** Übersicht über sämtliche definierten Zeichensätze (Forts.)

| Zeichensatz | Aliase | RFC |
|---|---|---|
| JIS_C6226-1983 | iso-ir-87, x0208, JIS_X0208-1983, csISO87JISX0208 | 1345 |
| greek7 | iso-ir-88, csISO88Greek7 | 1345 |
| ASMO_449 | ISO_9036, arabic7, iso-ir-89, csISO89ASMO449 | 1345 |
| iso-ir-90 | csISO90 | 1345 |
| JIS_C6229-1984-a | iso-ir-91, jp-ocr-a, csISO91JISC62291984a | 1345 |
| JIS_C6229-1984-b | iso-ir-92, ISO646-JP-OCR-B, jp-ocr-b, csISO92JISC62991984b | 1345 |
| JIS_C6229-1984-b-add | iso-ir-93, jp-ocr-b-add, csISO93JIS62291984badd | 1345 |
| JIS_C6229-1984-hand | iso-ir-94, jp-ocr-hand, csISO94JIS62291984hand | 1345 |
| JIS_C6229-1984-hand-add | iso-ir-95, jp-ocr-hand-add, csISO95JIS62291984handadd | 1345 |
| JIS_C6229-1984-kana | iso-ir-96, csISO96JISC62291984kana | 1345 |
| ISO_2033-1983 | iso-ir-98, e13b, csISO2033 | 1345 |
| ANSI_X3.110-1983 | iso-ir-99, CSA_T500-1983, NAPLPS, csISO99NAPLPS | 1345 |
| ISO_8859-1:1987 | iso-ir-100, ISO_8859-1, ISO-8859-1, latin1, l1, IBM819, CP819, csISOLatin1 | 1345 |
| ISO_8859-2:1987 | iso-ir-101, ISO_8859-2, ISO-8859-2, latin2, l2, csISOLatin2 | 1345 |
| T.61-7bit | iso-ir-102, csISO102T617bit | 1345 |
| T.61-8bit | T.61, iso-ir-103, csISO103T618bit | 1345 |
| ISO_8859-3:1988 | iso-ir-109, ISO_8859-3, ISO-8859-3, latin3, l3, csISOLatin3 | 1345 |
| ISO_8859-4:1988 | iso-ir-110, ISO_8859-4, ISO-8859-4, latin4, l4, csISOLatin4 | 1345 |
| ECMA-cyrillic | iso-ir-111, KOI8-E, csISO111ECMACyrillic | |
| CSA_Z243.4-1985-1 | iso-ir-121, ISO646-CA, csa7-1, ca, csISO121Canadian1 | 1345 |
| CSA_Z243.4-1985-2 | iso-ir-122, ISO646-CA2, csa7-2, csISO122Canadian2 | 1345 |
| CSA_Z243.4-1985-gr | iso-ir-123, csISO123CSAZ24341985gr | 1345 |
| ISO_8859-6:1987 | iso-ir-127, ISO_8859-6, ISO-8859-6, ECMA-114, ASMO-708, arabic, csISOLatinArabic | 1345 |

**Tabelle C.3** Übersicht über sämtliche definierten Zeichensätze (Forts.)

| Zeichensatz | Aliase | RFC |
|---|---|---|
| ISO_8859-6-E | csISO88596E, ISO-8859-6-E | 1556 |
| ISO_8859-6-I | csISO88596I, ISO-8859-6-I | 1556 |
| ISO_8859-7:1987 | iso-ir-126, ISO_8859-7, ISO-8859-7, ELOT_928, ECMA-118, greek, greek8, csISOLatinGreek | 1947, 1345 |
| T.101-G2 | iso-ir-128, csISO128T101G2 | 1345 |
| ISO_8859-8:1988 | iso-ir-138, ISO_8859-8, ISO-8859-8, hebrew, csISOLatinHebrew | 1345 |
| ISO_8859-8-E | csISO88598E, ISO-8859-8-E | 1556 |
| ISO_8859-8-I | csISO88598I, ISO-8859-8-I | 1556 |
| CSN_369103 | iso-ir-139, csISO139CSN369103 | 1345 |
| JUS_I.B1.002 | iso-ir-141, ISO646-YU, js, yu, csISO141JUSIB1002 | 1345 |
| ISO_6937-2-add | iso-ir-142, csISOTextComm | 1345 |
| IEC_P27-1 | iso-ir-143, csISO143IECP271 | 1345 |
| ISO_8859-5:1988 | iso-ir-144, ISO_8859-5, ISO-8859-5, cyrillic, csISOLatinCyrillic | 1345 |
| JUS_I.B1.003-serb | iso-ir-146, serbian, csISO146Serbian | 1345 |
| JUS_I.B1.003-mac | macedonian, iso-ir-147, csISO147Macedonian | 1345 |
| ISO_8859-9:1989 | iso-ir-148, ISO_8859-9, ISO-8859-9, latin5, l5, csISOLatin5 | 1345 |
| greek-ccitt | iso-ir-150, csISO150, csISO150GreekCCITT | 1345 |
| NC_NC00-10:81 | cuba, iso-ir-151, ISO646-CU, csISO151Cuba | 1345 |
| ISO_6937-2-25 | iso-ir-152, csISO6937Add | 1345 |
| GOST_19768-74 | ST_SEV_358-88, iso-ir-153, csISO153GOST1976874 | 1345 |
| ISO_8859-supp | iso-ir-154, latin1-2-5, csISO8859Supp | 1345 |
| ISO_10367-box | iso-ir-155, csISO10367Box | 1345 |
| ISO-8859-10 | iso-ir-157, l6, ISO_8859-10:1992, csISOLatin6, latin6 | 1345 |
| latin-lap | lap, iso-ir-158, csISO158Lap | 1345 |
| JIS_X0212-1990 | x0212, iso-ir-159, csISO159JISX02121990 | 1345 |
| DS_2089 | DS2089, ISO646-DK, dk, csISO646Danish | 1345 |
| us-dk | csUSDK | 1345 |
| dk-us | csDKUS | 1345 |

**Tabelle C.3** Übersicht über sämtliche definierten Zeichensätze (Forts.)

| Zeichensatz | Aliase | RFC |
|---|---|---|
| JIS_X0201 | X0201, csHalfWidthKatakana | 1345 |
| KSC5636 | ISO646-KR, csKSC5636 | 1345 |
| ISO-10646-UCS-2 | csUnicode | |
| ISO-10646-UCS-4 | csUCS4 | |
| DEC-MCS | dec, csDECMCS | 1345 |
| hp-roman8 | roman8, r8, csHPRoman8 | 1345 |
| macintosh | mac, csMacintosh | 1345 |
| IBM037 | cp037, ebcdic-cp-us, ebcdic-cp-ca, ebcdic-cp-wt, ebcdic-cp-nl, csIBM037 | 1345 |
| IBM038 | EBCDIC-INT, cp038, csIBM038 | 1345 |
| IBM273 | CP273, csIBM273 | 1345 |
| IBM274 | EBCDIC-BE, CP274, csIBM274 | 1345 |
| IBM275 | EBCDIC-BR, cp275, csIBM275 | 1345 |
| IBM277 | EBCDIC-CP-DK, EBCDIC-CP-NO, csIBM277 | 1345 |
| IBM278 | CP278, ebcdic-cp-fi, ebcdic-cp-se, csIBM278 | 1345 |
| IBM280 | CP280, ebcdic-cp-it, csIBM280 | 1345 |
| IBM281 | EBCDIC-JP-E, cp281, csIBM281 | 1345 |
| IBM284 | CP284, ebcdic-cp-es, csIBM284 | 1345 |
| IBM285 | CP285, ebcdic-cp-gb, csIBM285 | 1345 |
| IBM290 | cp290, EBCDIC-JP-kana, csIBM290 | 1345 |
| IBM297 | cp297, ebcdic-cp-fr, csIBM297 | 1345 |
| IBM420 | cp420, ebcdic-cp-ar1, csIBM420 | 1345 |
| IBM423 | cp423, ebcdic-cp-gr, csIBM423 | 1345 |
| IBM424 | cp424, ebcdic-cp-he, csIBM424 | 1345 |
| IBM437 | cp437, 437, csPC8CodePage437 | 1345 |
| IBM500 | CP500, ebcdic-cp-be, ebcdic-cp-ch, csIBM500 | 1345 |
| IBM775 | cp775, csPC775Baltic | |
| IBM850 | cp850, 850, csPC850Multilingual | 1345 |
| IBM851 | cp851, 851, csIBM851 | 1345 |
| IBM852 | cp852, 852, csPCp852 | 1345 |
| IBM855 | cp855, 855, csIBM855 | 1345 |
| IBM857 | cp857, 857, csIBM857 | 1345 |
| IBM860 | cp860, 860, csIBM860 | 1345 |

**Tabelle C.3** Übersicht über sämtliche definierten Zeichensätze (Forts.)

| Zeichensatz | Aliase | RFC |
|---|---|---|
| IBM861 | cp861, 861, cp-is, csIBM861 | 1345 |
| IBM862 | cp862, 862, csPC862LatinHebrew | 1345 |
| IBM863 | cp863, 863, csIBM863 | 1345 |
| IBM864 | cp864, csIBM864 | 1345 |
| IBM865 | cp865, 865, csIBM865 | 1345 |
| IBM866 | cp866, 866, csIBM866 | |
| IBM868 | CP868, cp-ar, csIBM868 | 1345 |
| IBM869 | cp869, 869, cp-gr, csIBM869 | 1345 |
| IBM870 | CP870, ebcdic-cp-roece, ebcdic-cp-yu, csIBM870 | 1345 |
| IBM871 | CP871, ebcdic-cp-is, csIBM871 | 1345 |
| IBM880 | cp880, EBCDIC-Cyrillic, csIBM880 | 1345 |
| IBM891 | cp891, csIBM891 | 1345 |
| IBM903 | cp903, csIBM903 | 1345 |
| IBM904 | cp904, 904, csIBBM904 | 1345 |
| IBM905 | CP905, ebcdic-cp-tr, csIBM905 | 1345 |
| IBM918 | CP918, ebcdic-cp-ar2, csIBM918 | 1345 |
| IBM1026 | CP1026, csIBM1026 | 1345 |
| EBCDIC-AT-DE | csIBMEBCDICATDE | 1345 |
| EBCDIC-AT-DE-A | csEBCDICATDEA | 1345 |
| EBCDIC-CA-FR | csEBCDICCAFR | 1345 |
| EBCDIC-DK-NO | csEBCDICDKNO | 1345 |
| EBCDIC-DK-NO-A | csEBCDICDKNOA | 1345 |
| EBCDIC-FI-SE | csEBCDICFISE | 1345 |
| EBCDIC-FI-SE-A | csEBCDICFISEA | 1345 |
| EBCDIC-FR | csEBCDICFR | 1345 |
| EBCDIC-IT | csEBCDICIT | 1345 |
| EBCDIC-PT | csEBCDICPT | 1345 |
| EBCDIC-ES | csEBCDICES | 1345 |
| EBCDIC-ES-A | csEBCDICESA | 1345 |
| EBCDIC-ES-S | csEBCDICESS | 1345 |
| EBCDIC-UK | csEBCDICUK | 1345 |
| EBCDIC-US | csEBCDICUS | 1345 |
| UNKNOWN-8BIT | csUnknown8BiT | 1428 |
| MNEMONIC | csMnemonic | 1345 |

**Tabelle C.3** Übersicht über sämtliche definierten Zeichensätze (Forts.)

| Zeichensatz | Aliase | RFC |
|---|---|---|
| MNEM | csMnem | 1345 |
| VISCII | csVISCII | 1456 |
| VIQR | csVIQR | 1456 |
| KOI8-R | csKOI8R | 1489 |
| KOI8-U | | 2319 |
| IBM00858 | CCSID00858, CP00858, PC-Multilingual-850+euro | |
| IBM00924 | CCSID00924, CP00924, ebcdic-Latin9-euro | |
| IBM01140 | CCSID01140, CP01140, ebcdic-us-37+euro | |
| IBM01141 | CCSID01141, CP01141, ebcdic-de-273+euro | |
| IBM01142 | CCSID01142, CP01142, ebcdic-dk-277 +euro, ebcdic-no-277+euro | |
| IBM01143 | CCSID01143, CP01143, ebcdic-fi-278+euro, ebcdic-se-278+euro | |
| IBM01144 | CCSID01144, CP01144, ebcdic-it-280+euro | |
| IBM01145 | CCSID01145, CP01145, ebcdic-es-284+euro | |
| IBM01146 | CCSID01146, CP01146, ebcdic-gb-285+euro | |
| IBM01147 | CCSID01147, CP01147, ebcdic-fr-297+euro | |
| IBM01148 | CCSID01148, CP01148, ebcdic-international-500+euro | |
| IBM01149 | CCSID01149, CP01149, ebcdic-is-871+euro | |
| Big5-HKSCS | – | |
| IBM1047 | IBM-1047 | |
| PTCP154 | csPTCP154, PT154, CP154, Cyrillic-Asian | |
| Amiga-1251 | Ami1251, Amiga1251, Ami-1251 | |
| KOI7-switched | – | |
| UNICODE-1-1 | csUnicode11 | 1641 |
| SCSU | – | |
| UTF-7 | – | 2152 |
| UTF-16BE | – | 2781 |
| UTF-16LE | – | 2781 |
| UTF-16 | – | 2781 |
| CESU-8 | csCESU-8 | |
| UTF-32 | – | |

**Tabelle C.3** Übersicht über sämtliche definierten Zeichensätze (Forts.)

| Zeichensatz | Aliase | RFC |
|---|---|---|
| UTF-32BE | – | |
| UTF-32LE | – | |
| BOCU-1 | csBOCU-1 | |
| UNICODE-1-1-UTF-7 | csUnicode11UTF7 | 1642 |
| UTF-8 | – | 3629 |
| ISO-8859-13 | – | |
| ISO-8859-14 | iso-ir-199, ISO_8859-14:1998, ISO_8859-14, latin8, iso-celtic, l8 | |
| ISO-8859-15 | ISO_8859-15, Latin-9 | |
| ISO-8859-16 | iso-ir-226, ISO_8859-16:2001, ISO_8859-16, latin10, l10 | |
| GBK | CP936, MS936, windows-936 | |
| GB18030 | – | |
| OSD_EBCDIC_DF04_15 | – | |
| OSD_EBCDIC_DF03_IRV | – | |
| OSD_EBCDIC_DF04_1 | – | |
| JIS_Encoding | csJISEncoding | |
| Shift_JIS | MS_Kanji, csShiftJIS | |
| Extended_UNIX_Code_Packed_Format_for_Japanese | csEUCPkdFmtJapanese, EUC-JP | |
| Extended_UNIX_Code_Fixed_Width_for_Japanese | csEUCFixWidJapanese | |
| ISO-10646-UCS-Basic | csUnicodeASCII | |
| ISO-10646-Unicode-Latin1 | csUnicodeLatin1, ISO-10646 | |
| ISO-10646-J-1 | – | |
| ISO-Unicode-IBM-1261 | csUnicodeIBM1261 | |
| ISO-Unicode-IBM-1268 | csUnicodeIBM1268 | |
| ISO-Unicode-IBM-1276 | csUnicodeIBM1276 | |
| ISO-Unicode-IBM-1264 | csUnicodeIBM1264 | |
| ISO-Unicode-IBM-1265 | csUnicodeIBM1265 | |
| ISO-8859-1-Windows-3.0-Latin-1 | csWindows30Latin1 | |
| ISO-8859-1-Windows-3.1-Latin-1 | csWindows31Latin1 | |
| ISO-8859-2-Windows-Latin-2 | csWindows31Latin2 | |
| ISO-8859-9-Windows-Latin-5 | csWindows31Latin5 | |

**Tabelle C.3** Übersicht über sämtliche definierten Zeichensätze (Forts.)

| Zeichensatz | Aliase | RFC |
|---|---|---|
| Adobe-Standard-Encoding | csAdobeStandardEncoding | |
| Ventura-US | csVenturaUS | |
| Ventura-International | csVenturaInternational | |
| PC8-Danish-Norwegian | csPC8DanishNorwegian | |
| PC8-Turkish | csPC8Turkish | |
| IBM-Symbols | csIBMSymbols | |
| IBM-Thai | csIBMThai | |
| HP-Legal | csHPLegal | |
| HP-Pi-font | csHPPiFont | |
| HP-Math8 | csHPMath8 | |
| Adobe-Symbol-Encoding | csHPPSMath | |
| HP-DeskTop | csHPDesktop | |
| Ventura-Math | csVenturaMath | |
| Microsoft-Publishing | csMicrosoftPublishing | |
| Windows-31J | csWindows31J | |
| GB2312 | csGB2312 | |
| Big5 | csBig5 | |
| windows-1250 | – | |
| windows-1251 | – | |
| windows-1252 | – | |
| windows-1253 | – | |
| windows-1254 | – | |
| windows-1255 | – | |
| windows-1256 | – | |
| windows-1257 | – | |
| windows-1258 | – | |
| TIS-620 | – | |
| HZ-GB-2312 | – | 1842, 1843 |

**Tabelle C.3** Übersicht über sämtliche definierten Zeichensätze (Forts.)

## C.4 Top-Level-Domains

Für die Analyse von Log-Dateien und andere Webmaster-Tätigkeiten ist eine Kenntnis der verschiedenen Top-Level-Domains wichtig. In diesem Abschnitt finden Sie deshalb zwei Tabellen mit sämtlichen Top-Level-Domains.

### C.4.1 Generische Top-Level-Domains

Die generischen TLDs (kurz gTLDs) in Tabelle C.4 sind für bestimmte Arten von Institutionen, Organisationen und Personen gedacht. Einige von ihnen können aber von jedem Interessierten reserviert werden; diese sind mit einem Sternchen (*) gekennzeichnet.

Seit einigen Jahren verlangen Politiker und Firmen, beliebig viele neue generische TLDs einzuführen, zum Beispiel Orts- oder Firmennamen; das am häufigsten zitierte Beispiel in Deutschland ist *.berlin*. Es sieht nun so aus, als würden die neuen gTLDs im Jahr 2012 endgültig eingeführt werden. Ob es sonderlich benutzerfreundlich ist, beispielsweise *http://info.sony* anstelle von *http://sony.com* einzugeben (zumal man gar nicht wissen kann, ob die Second-Level-Domain »info« lautet), sei dahingestellt; den meisten technisch versierten Beobachtern erscheint die verbindliche und flächendeckende Einführung von IPv6 erheblich wichtiger als die eher in Marketing-Kreisen erwünschte Vervielfachung der TLDs.

| Top-Level-Domain | Wortbedeutung | Erläuterung |
|---|---|---|
| .aero | Aerospace | Luft- und Raumfahrtunternehmen |
| .asia | Asien | Firmen, Organisationen und Personen in der Region Asien-Pazifik |
| .biz | Business | Unternehmen (neu) |
| .cat | Catalan | katalanische Sprache/Kultur |
| .com * | Commercial | Unternehmen (klassisch) |
| .coop | Cooperative Associations | Genossenschaften |
| .info * | Information | allgemeine Informationen |
| .jobs | Jobs | Jobangebote und -gesuche |
| .mobi | Mobile Devices | spezielle Angebote für Mobilgeräte |
| .museum | Museums | Museen |
| .name * | Persons' Name | natürliche Personen |
| .net * | Networking | Netzwerkbetreiber |
| .org * | (Non-Profit) Organization | Vereine usw. |
| .pro | Professionals | Freiberufler |
| .tel | Telephone | Dienste, die Telefonnetz und Internet verknüpfen |
| .travel | Travel | Reisebüros, Hotels, Tourismus usw. |
| .gov | Governmental | öffentlicher Dienst |
| .edu | Educational | Schulen, Universitäten |

**Tabelle C.4** Generische Top-Level-Domains

| Top-Level-Domain | Wortbedeutung | Erläuterung |
|---|---|---|
| .mil | Military | Militär |
| .int | International | internationale Organisationen |
| .xxx | XXX | digitales Rotlichtmilieu |

**Tabelle C.4** Generische Top-Level-Domains (Forts.)

### C.4.2 Länder-Top-Level-Domains

In Tabelle C.5 finden Sie die Tabelle mit den TLDs aller Länder der Welt und einiger Gebiete mit besonderem Status, die ebenfalls eine eigene Domain haben.

| Top-Level-Domain | Land |
|---|---|
| .ac | Ascension |
| .ad | Andorra |
| .ae | Vereinigte Arabische Emirate |
| .af | Afghanistan |
| .ag | Antigua und Barbuda |
| .ai | Anguilla |
| .al | Albanien |
| .am | Armenien |
| .an | Niederländische Antillen |
| .ao | Angola |
| .aq | Antarktis |
| .ar | Argentinien |
| .as | Amerikanisch-Samoa |
| .at | Österreich |
| .au | Australien |
| .aw | Aruba |
| .az | Aserbaidschan |
| .ba | Bosnien-Herzegowina |
| .bb | Barbados |
| .bd | Bangladesch |
| .be | Belgien |
| .bf | Burkina Faso |
| .bg | Bulgarien |
| .bh | Bahrain |
| .bi | Burundi |

**Tabelle C.5** Länder-Top-Level-Domains

| Top-Level-Domain | Land |
|---|---|
| .bj | Benin |
| .bm | Bermuda |
| .bn | Brunei |
| .bo | Bolivien |
| .br | Brasilien |
| .bs | Bahamas |
| .bt | Bhutan |
| .bv | Bouvetinsel |
| .bw | Botswana |
| .by | Weißrussland |
| .bz | Belize |
| .ca | Kanada |
| .cc | Kokosinseln |
| .cd | Demokratische Republik Kongo |
| .cf | Zentralafrikanische Republik |
| .cg | Republik Kongo |
| .ch | Schweiz |
| .ci | Elfenbeinküste |
| .ck | Cook-Inseln |
| .cl | Chile |
| .cm | Kamerun |
| .cn | China |
| .co | Kolumbien |
| .cr | Costa Rica |
| .cs | Serbien und Montenegro (wird abgeschafft) |
| .cu | Kuba |
| .cv | Kap Verde |
| .cx | Weihnachtsinsel |
| .cy | Zypern |
| .cz | Tschechien |
| .de | Deutschland |
| .dj | Djibouti |
| .dk | Dänemark |
| .dm | Dominica |
| .do | Dominikanische Republik |

**Tabelle C.5** Länder-Top-Level-Domains (Forts.)

| Top-Level-Domain | Land |
|---|---|
| .dz | Algerien |
| .ec | Ecuador |
| .ee | Estland |
| .eg | Ägypten |
| .eh | West-Sahara |
| .er | Eritrea |
| .es | Spanien |
| .et | Äthiopien |
| .fi | Finnland |
| .fj | Fiji |
| .fk | Falklandinseln |
| .fm | Mikronesien |
| .fo | Färöer |
| .fr | Frankreich |
| .ga | Gabun |
| .gb | *Großbritannien (ungebräuchlich – bevorzugt .uk)* |
| .gd | Grenada |
| .ge | Georgien |
| .gf | Französisch-Guayana |
| .gg | Guernsey |
| .gh | Ghana |
| .gi | Gibraltar |
| .gl | Grönland |
| .gm | Gambia |
| .gn | Guinea |
| .gp | Guadeloupe |
| .gq | Äquatorial-Guinea |
| .gr | Griechenland |
| .gs | Südgeorgien und die Süd-Sandwichinseln |
| .gt | Guatemala |
| .gu | Guam |
| .gw | Guinea-Bissau |
| .gy | Guyana |
| .hk | Hongkong |
| .hm | Heard- und McDonald-Inseln |

**Tabelle C.5** Länder-Top-Level-Domains (Forts.)

| Top-Level-Domain | Land |
|---|---|
| .hn | Honduras |
| .hr | Kroatien |
| .ht | Haiti |
| .hu | Ungarn |
| .id | Indonesien |
| .ie | Irland |
| .il | Israel |
| .im | Insel Man |
| .in | Indien |
| .io | Britisches Territorium im Indischen Ozean |
| .iq | Irak |
| .ir | Iran |
| .is | Island |
| .it | Italien |
| .je | Jersey |
| .jm | Jamaica |
| .jo | Jordanien |
| .jp | Japan |
| .ke | Kenia |
| .kg | Kirgisien |
| .kh | Kambodscha |
| .ki | Kiribati |
| .km | Komoren |
| .kn | St. Kitts and Nevis |
| .kp | Nordkorea |
| .kr | Südkorea |
| .kw | Kuwait |
| .ky | Kaimaninseln |
| .kz | Kasachstan |
| .la | Laos |
| .lb | Libanon |
| .lc | Santa Lucia |
| .li | Liechtenstein |
| .lk | Sri Lanka |
| .lr | Liberia |

**Tabelle C.5** Länder-Top-Level-Domains (Forts.)

| Top-Level-Domain | Land |
|---|---|
| .ls | Lesotho |
| .lt | Litauen |
| .lu | Luxemburg |
| .lv | Lettland |
| .ly | Libyen |
| .ma | Marokko |
| .mc | Monaco |
| .md | Moldawien |
| .me | Montenegro |
| .mg | Madagaskar |
| .mh | Marshallinseln |
| .mk | Mazedonien |
| .ml | Mali |
| .mm | Myanmar |
| .mn | Mongolei |
| .mo | Macao |
| .mp | Nördliche Marianen |
| .mq | Martinique |
| .mr | Mauretanien |
| .ms | Montserrat |
| .mt | Malta |
| .mu | Mauritius |
| .mv | Malediven |
| .mw | Malawi |
| .mx | Mexiko |
| .my | Malaysia |
| .mz | Mosambik |
| .na | Namibia |
| .nc | Neukaledonien |
| .ne | Niger |
| .nf | Norfolkinsel |
| .ng | Nigeria |
| .ni | Nicaragua |
| .nl | Niederlande |
| .no | Norwegen |

**Tabelle C.5** Länder-Top-Level-Domains (Forts.)

| Top-Level-Domain | Land |
|---|---|
| .np | Nepal |
| .nr | Nauru |
| .nt | *reserviert für neutrale Zonen* |
| .nu | Niue |
| .nz | Neuseeland |
| .om | Oman |
| .pa | Panama |
| .pe | Peru |
| .pf | Französisch-Polynesien |
| .pg | Papua-Neuguinea |
| .ph | Philippinen |
| .pk | Pakistan |
| .pl | Polen |
| .pm | St. Pierre und Miquelon |
| .pn | Pitcairninseln |
| .pr | Puerto Rico |
| .ps | Palästina |
| .pt | Portugal |
| .pw | Palau |
| .py | Paraguay |
| .qa | Katar |
| .re | Réunion |
| .ro | Rumänien |
| .rs | Serbien |
| .ru | Russland |
| .rw | Ruanda |
| .sa | Saudi-Arabien |
| .sb | Salomonen |
| .sc | Seychellen |
| .sd | Sudan |
| .se | Schweden |
| .sg | Singapur |
| .sh | St. Helena |
| .si | Slowenien |
| .sj | Svalbard und Jan Mayen |

**Tabelle C.5** Länder-Top-Level-Domains (Forts.)

| Top-Level-Domain | Land |
|---|---|
| .sk | Slowakei |
| .sl | Sierra Leone |
| .sm | San Marino |
| .sn | Senegal |
| .so | Somalia |
| .sr | Surinam |
| .ss | Südsudan |
| .st | São Tomé und Príncipe |
| .su | *ehemalige Sowjetunion* |
| .sv | El Salvador |
| .sy | Syrien |
| .sz | Swasiland |
| .tc | Turks- und Caicosinseln |
| .td | Tschad |
| .tf | Französische Süd- und Antarktisgebiete |
| .tg | Togo |
| .th | Thailand |
| .tj | Tadschikistan |
| .tk | Tokelau |
| .tm | Turkmenistan |
| .tn | Tunesien |
| .to | Tonga |
| .tp | Ost-Timor |
| .tr | Türkei |
| .tt | Trinidad und Tobago |
| .tv | Tuvalu |
| .tw | Taiwan |
| .tz | Tansania |
| .ua | Ukraine |
| .ug | Uganda |
| .uk | Großbritannien |
| .um | Amerikanisch-Ozeanien |
| .us | USA |
| .uy | Uruguay |
| .uz | Usbekistan |

**Tabelle C.5** Länder-Top-Level-Domains (Forts.)

| Top-Level-Domain | Land |
|---|---|
| .va | Vatikanstadt |
| .vc | Saint Vincent und die Grenadinen |
| .ve | Venezuela |
| .vg | Britische Jungferninseln |
| .vi | Amerikanische Jungferninseln |
| .vn | Vietnam |
| .vu | Vanuatu |
| .wf | Wallis- und Futuna-Inseln |
| .wg | *reserviert für Westjordanland und Gazastreifen (Palästina)* |
| .ws | West-Samoa |
| .ye | Jemen |
| .yt | Mayotte |
| .yu | *ehemaliges Jugoslawien (am 30.09.2009 ausgelaufen)* |
| .za | Südafrika |
| .zm | Sambia |
| .zw | Simbabwe |

**Tabelle C.5** Länder-Top-Level-Domains (Forts.)

*»Am allererdrückendsten sind doch die Leute, die Gesetze erlassen und ständig erneuern, stets im Glauben, den Betrügereien im Geschäftsleben Schranken setzen zu können, ohne zu ahnen, dass sie in Wirklichkeit einer Hydra Köpfe abschneiden.«*
*– Platon*

# D Die Apache-Lizenz 2.0

Bereits in Kapitel 3, »Apache 2 im Überblick«, wurde die allgemeine Bedeutung der Apache-Softwarelizenz erläutert. Seit Januar 2004 werden alle Projekte der Apache Software Foundation unter einer erweiterten Fassung dieser Lizenz verbreitet, der Version 2.0. Sie präzisiert einige Angaben der alten Version und reagiert auf einige neue Entwicklungen, beispielsweise auf die Bedrohung freier Software durch Softwarepatente. In diesem Anhang finden Sie den vollständigen Wortlaut der Lizenz.

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   »License« shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

   »Licensor« shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

   »Legal Entity« shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, »control« means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

   »You« (or »Your«) shall mean an individual or Legal Entity exercising permissions granted by this License.

»Source« form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

»Object« form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

»Work« shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

»Derivative Works« shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

»Contribution« shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, »submitted« means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as »Not a Contribution.«

»Contributor« shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent

license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a »NOTICE« text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an »AS IS« BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

> To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets »[]« replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same »printed page« as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the »License«); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an »AS IS« BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

*»(bb)|([^b]{2})«*
*– William Shakespeare –*
*(RegExp-Fassung von James Duncan Davidson)*

# E Reguläre Ausdrücke

In vielen Bereichen der Apache-Konfiguration sowie in der Programmierung von Webanwendungen werden *Reguläre Ausdrücke* (Regular Expressions) verwendet. Der Begriff wurde unter anderem von dem Linguisten Noam Chomsky definiert; ursprünglich dienten reguläre Ausdrücke der Beschreibung natürlicher Sprachen. In vielen Programmiersprachen, Konsolenprogrammen und Anwendungen bilden sie inzwischen die Basis für Mustervergleiche. Insbesondere die Implementierung in der Sprache Perl sowie die Festschreibung im POSIX-Standard haben für eine enorme Verbreitung gesorgt.

In diesem Anhang erhalten Sie nur eine kurze Übersicht der wichtigsten Konstrukte anhand von Beispielen. Eine sehr ausführliche Beschreibung finden Sie in [FRIED 2007]; ein praktischer Begleiter für den Alltag ist dagegen [STUB 2007]. Auch die mit Perl gelieferte Dokumentation enthält ein recht umfangreiches Dokument; Sie finden es, wenn Sie `perldoc perlre` eingeben.

| RegExp-Konstrukt | Beschreibung | Beispiele |
|---|---|---|
| `abc` | die Zeichenfolge `abc` | `öl` passt auf `Köln`, aber nicht auf `schön` oder `Schnösel`. |
| `[abc]` | eines der Zeichen `a`, `b` oder `c` | `B[ai]ll` passt auf `Ball` oder `Bill`, aber nicht auf `Bell`. |
| `[a-m]` | eines der Zeichen von `a` bis `m` | `10[1-5]` passt auf `102` und `104`, aber nicht auf `107`. |
| `[^abc]` | jedes beliebige Zeichen *außer* `a`, `b` und `c` | `B[^o]hn` passt auf `Bahn`, aber nicht auf `Bohne`. |
| `.` | ein beliebiges Zeichen | `R.in` passt auf `Ruine` und `Rein`, aber nicht auf `Rind`. |

**Tabelle E.1** Die wichtigsten Konstrukte für reguläre Ausdrücke

| RegExp-Konstrukt | Beschreibung | Beispiele |
|---|---|---|
| `\` | Escaping von Zeichen, die in RegExp eine besondere Bedeutung haben | `\.`, `\(` oder `\$`: `i[a-z]+\.html?` passt auf `info.htm`, `index.html` und `illustration.html`, aber nicht auf `inhtml` oder `info9.htm`. |
| `?` | Das links stehende Element kann einmal vorkommen oder nicht. | `Ei?s` passt auf `Eis` und `Es`, aber nicht auf `Eins`. |
| `+` | Das links stehende Element muss mindestens einmal vorkommen. | `25+` passt auf `25` und `25555`, aber nicht auf `2` oder `23`. |
| `*` | Das links stehende Element kann beliebig oft vorkommen. | `10*` passt auf `1`, `10`, `1000000` und sogar auf `12` usw. |
| `{n}` | Das links stehende Element muss genau n-mal vorkommen. | `test[0-9]{2}` passt auf `test03` und `test56`, aber nicht auf `test2`. |
| `{m,n}` | Das links stehende Element muss mindestens m- und höchstens n-mal vorkommen. | `test[0-9]{2,4}` passt auf `test082` oder `test25`, aber nicht auf `test2a` oder `test6354`. |
| `^` | Anfang des Ausdrucks | `^öl passt auf ölig`, aber nicht auf `Köln`. |
| `$` | Ende des Ausdrucks | `öl$` passt auf `Rhizinusöl`, aber nicht auf `Köln`. |
| `(...)` | Gruppierung; dient der Anwendung von `+`, `*` u. Ä. auf Textblöcke sowie der späteren Verfügbarkeit in `$1` bis `$9`. | `T(ischt)?elefon` passt auf `Telefon` und `Tischtelefon`. |
| `\|` | Ermöglicht Oder-Verknüpfung von `(...)`-Gruppen. | `Daten(bank\|verarbeitung)` passt auf `Datenbank` und `Datenverarbeitung`, aber nicht auf `Datenschutz`. |

**Tabelle E.1** Die wichtigsten Konstrukte für reguläre Ausdrücke (Forts.)

*»Insofern sich die Sätze der Mathematik auf die Wirklichkeit beziehen, sind sie nicht sicher, und insofern sie sicher sind, beziehen sie sich nicht auf die Wirklichkeit.«*
*– Albert Einstein –*

# F VMware Workstation

Ein sehr nützliches Hilfsmittel für die Einrichtung von Netzwerken und Servern ist der Einsatz *virtueller Maschinen*, das heißt in diesem Fall die Emulation eines vollständigen Computers als Softwareprogramm. Dies ermöglicht die gleichzeitige Verwendung unterschiedlicher Betriebssysteme und die Einrichtung eines virtuellen Netzwerks zwischen dem tatsächlichen Computer und einer oder mehreren virtuellen Maschinen. Zurzeit sind für x86-Hardware diverse gängige Lösungen verfügbar:

- VMware Workstation
- Microsoft Virtual PC (nach Aufkauf der gleichnamigen Firma durch Microsoft vom Macintosh auch auf Windows portiert)
- Open-Source-Produkte wie XEN und VirtualBox

In früheren Auflagen dieses Buches habe ich mit VMware Workstation unter Windows gearbeitet, um darauf die zahlreichen UNIX-Varianten (außer natürlich Mac OS X) zu testen, unter denen ich Apache installiert, konfiguriert und getestet habe. Für die vorliegende Auflage kam vornehmlich ein MacBook mit VirtualBox zum Einsatz.

Dieser Anhang bietet einen kurzen Überblick über die Installation und Nutzung von VMware Workstation. Hier wird exemplarisch die Windows-Version beschrieben; für Linux ist VMware ebenfalls verfügbar und funktioniert – soweit Plattformunterschiede dies zulassen – genauso. Eine Trial-Version und Verkaufsinformationen erhalten Sie unter *www.vmware.com*.

## F.1 Einrichtung einer virtuellen Maschine

Die Installation des Programms selbst funktioniert wie üblich. Doppelklicken Sie das Icon des Installer-Programms, und folgen Sie den Anweisungen.

In der Dokumentation und Online-Hilfe zu VMware sowie im nachfolgenden Text werden die beiden folgenden Begriffe verwendet:

- *Wirtsbetriebssystem* (Host Operation System) bezeichnet das Originalbetriebssystem des Rechners, auf dem das Programm VMware ausgeführt wird.
- *Gastbetriebssystem* (Guest Operating System) ist das in einer virtuellen Maschine installierte System.
- Um eine virtuelle Maschine einzurichten, können Sie zunächst FILE • NEW VIRTUAL MACHINE wählen; anschließend sind die folgenden dialoggesteuerten Arbeitsschritte erforderlich:
- Sie gelangen zunächst auf den Begrüßungsbildschirm WELCOME TO THE NEW VIRTUAL MACHINE WIZARD. Klicken Sie auf WEITER (dies muss auch bei allen nachfolgenden Schritten geschehen).
- Auf dem Bildschirm SELECT THE APPROPRIATE CONFIGURATION können Sie den Installationstyp festlegen. TYPICAL verwendet vernünftige Standardeinstellungen (die später geändert werden können); bei CUSTOM können Sie die meisten Optionen bereits bei der Installation selbst wählen.
- Nun gelangen Sie zu dem Dialog SELECT A GUEST OPERATING SYSTEM. Wählen Sie aus dem Popup-Menü das Betriebssystem aus, das Sie in der virtuellen Maschine installieren möchten. Beachten Sie, dass das jeweilige System natürlich nicht mitgeliefert wird; nach der Einrichtung der virtuellen Maschine müssen Sie es genauso darin installieren wie auf einem normalen Rechner. Wenn das gewünschte System nicht in der Liste vorkommt, können Sie OTHER wählen. In diesem Fall besteht keine Funktionsgarantie; zumindest bietet VMware für solche Systeme keine spezifische Unterstützung durch Treiber oder Grafikfähigkeiten.
- Auf der Seite NAME THE VIRTUAL MACHINE werden der Name und das Installationsverzeichnis für die virtuelle Maschine eingestellt.
- Falls Sie den Installationstyp CUSTOM ausgewählt haben, folgt nun der Dialog MEMORY FOR THE VIRTUAL MACHINE; hier können Sie einstellen, wie viel Arbeitsspeicher Sie dem Gastbetriebssystem zuweisen möchten.
- Unter NETWORK TYPE wird die Netzwerkunterstützung für die virtuelle Maschine eingestellt. Es stehen vier Optionen zur Auswahl:

- Use brigded networking: Das Gastbetriebssystem erhält direkten Zugriff auf die Netzwerkschnittstellen des Wirtsrechners; die Zuweisung einer eigenen IP-Adresse ist möglich.
- Use network address translation (NAT): Die interne IP-Adresse der virtuellen Maschine wird automatisch in die externe Adresse des Wirtsrechners umgesetzt; dies bietet Internetzugriff über Einwahlverbindungen.
- Use host-only networking: Es wird ausschließlich eine virtuelle Netzwerkverbindung zum Wirtsbetriebssystem hergestellt.
- Do not use a network connection: kein Netzwerkzugriff

- Auch die letzte Option, Select a Disk, steht nur zur Verfügung, wenn Custom gewählt wurde. Hier haben Sie folgende Punkte zur Auswahl:
  - Create a new virtual disk: Es wird eine neue Datei erstellt, die die virtuelle Maschine als Festplatte betrachtet.
  - Use an existing virtual disk: Die bereits angelegte virtuelle Festplatte einer anderen virtuellen Maschine wird mitbenutzt.
  - Use a physical disk: Der virtuellen Maschine wird Zugriff auf die physikalischen Festplatten gewährt. Sie sollten diesen Punkt nur mit äußerster Vorsicht wählen.

  Unter Specify Disk Capacity können Sie die Größe der virtuellen Festplatte wählen. Normalerweise wächst die virtuelle Platte mit zunehmender Nutzung bis zur gewählten Maximalgröße. Wenn Ihre echte Festplatte groß genug ist, können Sie Allocate all disk space now ankreuzen, um die Datei von Anfang an mit der Gesamtgröße zu erstellen; dies beschleunigt die virtuelle Maschine ein wenig. Die Option Split disk into 2 GB files unterteilt die Festplattendatei in virtuelle Partitionen.

  Schließlich können Sie noch Advanced anklicken, um zu wählen, ob die virtuelle Platte als SCSI- oder IDE-Emulation eingebunden werden soll; manche Gastbetriebssysteme bevorzugen eine dieser Varianten.

## F.2 Die virtuelle Maschine im Betrieb

Hier eine Kurzübersicht über die wichtigsten Befehle zur Steuerung einer virtuellen Maschine:

- Das Einschalten erfolgt über Power • Power On oder über die Play-Schaltfläche in der Symbolleiste. Um den Boot-Datenträger zu wählen, können Sie mit der Taste F2 ins BIOS-Setup der virtuellen Maschine gelangen.

- Um mit der virtuellen Maschine zu arbeiten, wählen Sie Power • Grab Input, oder klicken einfach in Ihren »Bildschirm« hinein.
- In den Vollbildmodus gelangen Sie über View • Full Screen oder mit dem passenden Symbolleisten-Button.
- Wenn Sie zum Wirtsbetriebssystem zurückkehren möchten, geschieht dies über Strg + Alt; die Tastenkombination können Sie mittels Edit • Preferences auf der Registerkarte Hot keys ändern.
- Das Herunterfahren des Gastbetriebssystems schaltet in der Regel auch die virtuelle Maschine aus; ansonsten können Sie den Stop-Button oder Power • Power Off verwenden.
- Alternativ besteht die Möglichkeit, die virtuelle Maschine anzuhalten und dabei den Systemzustand – sogar über das Ausschalten des Wirtscomputers hinaus – zu konservieren. Wählen Sie dazu Power • Suspend, oder klicken Sie auf den Pause-Button.
- Zur Wiederaufnahme nach der Unterbrechung können Sie Power • Resume wählen oder erneut die Pause-Schaltfläche anklicken.
- Wenn Sie mit dem Gastbetriebssystem experimentieren möchten, bietet sich ein Snapshot an, zu dem Sie später zurückkehren können. Wählen Sie dazu Snapshot • Save Snapshot, oder klicken Sie die Disketten-Schaltfläche an.
- Um den Snapshot später wieder zu laden, wählen Sie Snapshot • Revert to Snapshot oder die Diskette/Rücklauf-Schaltfläche.

## F.3 Einstellungen der virtuellen Maschine ändern

Wenn Sie nachträglich Optionen der virtuellen Maschine ändern möchten, können Sie Edit virtual machine settings wählen. Der Dialog besitzt zwei Registerkarten:

- Hardware: Hier können Sie die Einstellungen für Speicher, Festplatte, Netzwerk usw. ändern, die bereits bei der Installation beschrieben wurden. Eine besonders praktische Option finden Sie unter DVD-ROM: Anstelle eines physikalischen CD-ROM- oder DVD-Laufwerks können Sie hier ein ISO-Image mounten; dies ist vor allem deshalb praktisch, weil die meisten Linux-Distributionen in diesem Format heruntergeladen werden können (z.B. unter *www.linuxiso.org*).
- Options: Auf dieser Registerkarte finden Sie allerlei zusätzliche Einstellungsmöglichkeiten; folgende Kategorien stehen zur Verfügung:
  - General: Hier können Sie den Namen der virtuellen Maschine ändern und ein anderes Gastbetriebssystem einstellen.

- Power: Dieser Abschnitt steuert das Verhalten beim Ein- und Ausschalten der virtuellen Maschine, z.B. für Vollbild und die Ausführung der VMware Tools.
- Shared Folders: Sie können Verzeichnisse des Wirtsbetriebssystems festlegen, die in den Verzeichnisbaum des Gastbetriebssystems abgebildet werden, um den gemeinsamen Zugriff zu ermöglichen. Bei Linux-Gastbetriebssystemen stehen diese Verzeichnisse beispielsweise unter */mnt/hgfs/<Freigabename>* zur Verfügung.
- Snapshot: Einstellungen für Snapshots
- Guest isolation: Hier kann der Drag & Drop-Datenaustausch mit der virtuellen Maschine deaktiviert werden.
- Advanced: Diese Option ermöglicht Einstellungen für die Priorität, mit der Prozesse der virtuellen Maschine im Vergleich zu anderen Programmen auf dem Wirtsbetriebssystem ausgeführt werden.

## F.4 VMware Tools installieren

Die VMware Tools ermöglichen einen besseren Datenaustausch zwischen Wirts- und Gastbetriebssystem und stellen verbesserte Treiber bereit. Sie sind nur für Systeme in der offiziellen Liste verfügbar; nicht für die Auswahl Others. Für das Beispiel eines Linux-Gastbetriebssystems funktioniert ihre Installation folgendermaßen:

- Das Gastbetriebssystem muss aktiv sein; Sie sollten sich auf der Konsole befinden, nicht etwa in einem X11-Terminalfenster.
- Wählen Sie File • Install VMware Tools aus dem VMware-Menü. Unter der Gerätedatei des ersten CD-ROM-Laufwerks (hier */dev/cdrom*) wird daraufhin das Verzeichnis mit dem Installationsarchiv eingebunden.
- Führen Sie nun als root nacheinander die folgenden Befehle aus (*verzeichnis* und *festplattenverzeichnis* sind Platzhalter für beliebige Verzeichnisse):

```
# mount /dev/cdrom /verzeichnis
# cd /verzeichnis
# cp vmware-linux-tools.tar.gz /festplattenverzeichnis
# cd /festplattenverzeichnis
# tar -xzvf vmware-linux-tools.tar.gz
# cd vmware-tools-distrib
# ./vmware-install.pl
```

Nun erfolgt die eigentliche Installation; Sie müssen diverse Fragen über Installationsverzeichnisse und Einstellungen beantworten.

*»Erfahrene Juristen bezeugen, dass es vor Gericht von Vorteil sein kann, wenn man im Recht ist.«*
*– Graham Chapman (Monty Python)*

# G Rechtliche Aspekte

Sowohl der Betrieb einer Website als auch die Webhosting-Dienstleistung für Kunden sind verschiedenen gesetzlichen Bestimmungen unterworfen. Dieser Anhang ist eine kurz gefasste Zusammenfassung der relevanten Regelungen.

Die beiden wichtigsten Gesetze für Internetangebote sind folgende:

- **Informations- und Kommunikationsdienstegesetz (IuKDG)** – Besteht aus den drei Teilen Teledienstegesetz (TDG), Teledienste-Datenschutzgesetz (TDDSG) und Signaturgesetz (SigG). Der relevanteste Teil für Hoster und (besonders kommerzielle) Site-Betreiber ist das TDG.
- **Mediendienstestaatsvertrag (MDStV)**
  Der Mediendienstestaatsvertrag gilt ursprünglich für Rundfunkmedien; deshalb wird er auf Websites angewendet, die ein allgemeines, unbestimmtes Publikum ansprechen. Das TDG/IuKDG gilt dagegen für Sites, deren Zweck eher der Individualkommunikation ähnelt. Hier ist aber nicht unbedingt das persönliche Gespräch gemeint. Die folgenden Punkte sind der Inhalt von §2 Abs. 2 TDG; sie bestimmen die Fälle, für die das Gesetz gilt:
  1. Angebote im Bereich der Individualkommunikation (z.B. Telebanking, Datenaustausch)
  2. Angebote zur Information oder Kommunikation, soweit nicht die redaktionelle Gestaltung zur Meinungsbildung für die Allgemeinheit im Vordergrund steht (Datendienste, z.B. Verkehrs-, Wetter-, Umwelt- und Börsendaten, Verbreitung von Informationen über Waren und Dienstleistungsangebote)
  3. Angebote zur Nutzung des Internets oder weiterer Netze
  4. Angebote zur Nutzung von Telespielen
  5. Angebote von Waren und Dienstleistungen in elektronisch abrufbaren Datenbanken mit interaktivem Zugriff und unmittelbarer Bestellmöglichkeit

Wie Sie sehen, lässt sich die Mehrheit aller existierenden Websites in irgendeiner Weise hier wiederfinden.

Die folgenden Regelungen des Teledienstegesetzes sind besonders wichtig:

- **Impressumspflicht**
  Laut §6 TDG müssen geschäftsmäßige Webangebote mit einem Impressum versehen sein, das »leicht erkennbar, unmittelbar erreichbar und ständig verfügbar« zu sein hat. Ein entsprechend bezeichneter Hyperlink auf jeder Seite genügt normalerweise. Das Impressum muss folgende Informationen enthalten:
  - Name und Anschrift; bei juristischen Personen muss auch der Name eines Vertretungsberechtigten (z. B. Geschäftsführer) genannt werden.
  - Angaben zur elektronischen Erreichbarkeit; vor allem eine allgemeine E-Mail-Adresse
  - Bei zulassungspflichtigen Tätigkeiten: Name der Aufsichtsbehörde
  - Registereinträge (Handelsregister, Vereinsregister usw.)
  - Erweiterte Angaben zum Erwerb der Befähigung, falls die Dienstleistung bestimmten akademischen Berufen entspricht (gemäß Richtlinie 89/48/EWG des Europarates)
  - Umsatzsteuer-Identifikationsnummer, falls diese nach §27a Umsatzsteuergesetz erforderlich ist

  Diese Vorschrift ist sehr wichtig, denn §12 TDG bestimmt, dass fehlende, unvollständige oder falsche Informationen mit einem Bußgeld von bis zu 50.000 Euro geahndet werden können.
- **Kommerzielle Kommunikation**
  §7 TDG fordert, dass kommerzielle Angebote noch einmal zusätzlich gekennzeichnet werden müssen; dies gilt besonders für verkaufsfördernde Angebote wie Rabatte oder Zugaben, aber auch für Preisausschreiben und Gewinnspiele mit Werbecharakter. Im Unterschied zu einem »geschäftsmäßigen« Angebot im Sinne von §6 ist mit einem kommerziellen Angebot hier ein Verkaufsangebot (E-Commerce) gemeint.
- **Verantwortlichkeit**
  Unter diesem Titel stehen die §§8 bis 11 des TDG. Sie bestimmen, dass ein Vermittler (Provider oder Hoster) zunächst einmal nicht für die von seinen Kunden übermittelten oder angebotenen Informationen verantwortlich ist. Dies gilt allerdings nur, sofern der Vermittlungsdienstleister sich absolut passiv verhält, das heißt, wenn er selbst nicht in die Entstehung der entsprechenden Daten involviert ist – beispielsweise darf der Anbieter der Inhalte nicht wirtschaftlich abhängig vom Vermittler sein.

Insbesondere sind die drei folgenden Aspekte von diesen Regelungen betroffen:

- Zugangsvermittlung (§9, »Durchleitung von Informationen«) – die klassische Dienstleistung eines Internet-Service-Providers
- Caching (§10, »Zwischenspeicherung zur beschleunigten Übermittlung von Informationen«)
- Hosting (§11, »Speicherung von Informationen«)

Das Teledienste-Datenschutzgesetz enthält Bestimmungen darüber, wie Anbieter mit Daten ihrer Kunden umzugehen haben. Dazu gehören vor allem folgende Regelungen:

- Gemäß §3 TDDSG dürfen Daten nur erhoben und verwendet werden, sofern dieses Gesetz es erlaubt oder wenn die Benutzer ausdrücklich zustimmen.
- §4 legt fest, dass die Nutzer über Inhalt und Zweck der Datenspeicherung zu informieren sind. Wenn Sie also in irgendeiner Form Daten Ihrer Kunden oder Besucher erheben, sollten Sie auf jeder Seite einen gut sichtbaren Link zu Ihrer Datenschutzerklärung platzieren.
- Weiterhin wird in §4 bestimmt, dass eine eventuelle Einwilligungserklärung durch die Nutzer ausdrücklich zu erfolgen hat. Implementieren können Sie diese Forderung beispielsweise als Formularfeld, das ausdrücklich angekreuzt werden muss.
- Die wichtigsten Bestimmungen sind in §4 Abs. 4 enthalten, der hier aus diesem Grund vollständig abgedruckt wird:

  »Der Diensteanbieter hat durch technische und organisatorische Vorkehrungen sicherzustellen, dass

  1. der Nutzer seine Verbindung mit dem Diensteanbieter jederzeit abbrechen kann,
  2. die anfallenden personenbezogenen Daten über den Ablauf des Zugriffs oder der sonstigen Nutzung unmittelbar nach deren Beendigung gelöscht oder gesperrt werden können,
  3. der Nutzer Teledienste gegen Kenntnisnahme Dritter geschützt in Anspruch nehmen kann,
  4. die personenbezogenen Daten über die Inanspruchnahme verschiedener Teledienste durch einen Nutzer getrennt verarbeitet werden können,
  5. Daten nach §6 Abs. 2 nur für Abrechnungszwecke und
  6. Nutzerprofile nach §6 Abs. 3 nicht mit Daten über den Träger des Pseudonyms zusammengeführt werden können.

An die Stelle der Löschung nach Nummer 2 tritt eine Sperrung, soweit einer Löschung gesetzliche, satzungsmäßige oder vertragliche Aufbewahrungsfristen entgegenstehen.«

- Der Datenschutz findet seine Grenzen in der Mitwirkungspflicht des Anbieters bei der Strafverfolgung (geregelt jeweils in §5, »Bestandsdaten«, und §6, »Nutzungsdaten«).

Der Mediendienstestaatsvertrag (MDStV) enthält im Wesentlichen ähnliche Bestimmungen für allgemein informative Websites. Im Gegensatz zu den Gesetzen des IuKDG gilt er auch für Websites von Privatpersonen. Diese sind übrigens von der Impressumspflicht gemäß §6 TDG ausgenommen.

*»Der Leser hat's gut: Er kann sich seine Schriftsteller aussuchen.«*
*– Kurt Tucholsky*

# H Literatur


- [ALLE 2003] Robbie Allen, Alistair G. Lowe-Norris: »Active Directory«. 2. Auflage. Köln: O'Reilly Verlag 2003.
- [BANN 2001] Jens Banning: »LDAP unter Linux«. München: Addison-Wesley 2003.
- [BRIT 2003] Jason Brittain, Ian F. Darwin: »Tomcat. The Definitive Guide«. Sebastopol: O'Reilly Media 2003.
- [FRIED 2007] Jeffrey E. F. Friedl: »Reguläre Ausdrücke«. 3. Auflage. Köln: O'Reilly Verlag 2007.
- [HEIN 2003] Eduard Heindl: »Logfiles richtig nutzen«. Bonn: Galileo Press 2003.
- [HUNT 2003] Craig Hunt: »TCP/IP-Netzwerkadministration«. 3. Auflage. Köln: O'Reilly Verlag 2003.
- [LERD 2006] Rasmus Lerdorf, Kevin Tatroe: »Programmieren mit PHP«. Köln: O'Reilly Verlag 2006.
- [LIU 2003] Cricket Liu: »DNS und BIND Kochbuch«. Köln: O'Reilly Verlag 2003.
- [MITN 2003] Kevin Mitnick, William Simon: »Die Kunst der Täuschung«. Bonn: MITP-Verlag 2003.
- [SCHWA 2005] Randal L. Schwartz, Tom Phoenix, Brian D. Foy: »Einführung in Perl«. 4. Auflage. Köln: O'Reilly Verlag 2005.
- [Stein 2000] Lincoln D. Stein: »Network Programming with Perl«. Reading: Addison-Wesley 2000.
- [STUBB 2007] Tony Stubblebine: »Reguläre Ausdrücke kurz&gut«. 2. Auflage. Köln: O'Reilly Verlag 2007.
- [ULLE 2008] Christian Ullenboom: »Java ist auch eine Insel«. 5. Auflage. Bonn: Galileo Press 2008.
- [RIST 2005] Ivan Ristic: »Apache Security«. Sebastopol: O'Reilly Media 2005.
- [WILL 2008] Arnold Willemer: »Das UNIX-Handbuch«. Bonn: Galileo Computing 2008.

# Index


$/, Perl-Spezialvariable 89
+-Zeichen
  *bei Installationslayouts* 158
/etc/hosts, Datei 32
:lange Zeilen in httpd.conf 229


## A


A Patchy Web Server 106
AAA 400
ab, Hilfsprogramm 215, 568
Accept, HTTP-Header 70
Accept-Charset, HTTP-Header 71
Accept-Encoding, HTTP-Header 71
AcceptFilter, Direktive 263
Accept-Language, HTTP-Header 72
AcceptMutex, Direktive 270
AcceptPathInfo, Direktive 652
Accept-Ranges, HTTP-Header 72
access control 400
access.conf 225
AccessFileName, Direktive 287
Action, Direktive 643
ActivePerl 724
AddAlt, Direktive 390
AddAltByEncoding, Direktive 391
AddAltByType, Direktive 391
AddCharset, Direktive 324
AddDefaultCharset, Direktive 324
AddDescription, Direktive 392
AddEncoding, Direktive 326
AddHandler, Direktive 331
AddIcon, Direktive 388
AddIconByEncoding, Direktive 389
AddIconByType, Direktive 390
AddInputFilter, Direktive 769
AddLanguage, Direktive 328
AddModuleInfo, Direktive 379
AddOutputFilter, Direktive 768
AddOutputFilterByType, Direktive 767
AddType, Direktive 318
Age, HTTP-Header 72
AJP-Proxy
  *Tomcat* 735
Alexandria 109
Alias 345
Alias, Direktive 346
AliasMatch, Direktive 347
Allow, Direktive 292
Allow, HTTP-Header 72
AllowCONNECT, Direktive 602
AllowEncodedSlashes, Direktive 653
AllowMethods, Direktive 296
AllowOverride, Direktive 290
Alternates, HTTP-Header 340
Anmeldung → Authentifizierung 399
Anonymous, Direktive 454
Anonymous_Authoritative, Direktive 456
Anonymous_LogEmail, Direktive 456
Anonymous_NoUserID, Direktive 455
Anonymous_VerifyEmail, Direktive 455
Ant 108
ap_expr 240
Apache
  *ab, Hilfsprogramm* 215
  *als Dienst (Windows)* 209
  *Apache Portable Runtime* 127
  *Apache-Monitor (Windows)* 213
  *apxs, Hilfsprogramm* 215
  *Automatische Indizes* 381
  *Autostart (UNIX)* 203
  *beenden (UNIX)* 202
  *beenden und neu starten* 200
  *CGI-Konfiguration* 633
  *Dateiendungen* 316
  *dbmmanage, Hilfsprogramm* 215
  *Demo-Website* 216
  *Dienst starten (Windows)* 209
  *dynamische Inhalte* 123
  *fcgistarter, Hilfsprogramm* 216
  *Fehlerbehandlung* 375
  *Filter* 122
  *Funktionen* 116
  *Geschichte* 106
  *Header-Manipulation* 123
  *Hilfsprogramme* 215
  *htcacheclean, Hilfsprogramm* 216
  *htdbm, Hilfsprogramm* 216
  *htdigest, Hilfsprogramm* 216
  *htpasswd, Hilfsprogramm* 216

*httpd, Binary* 196
*httpd.conf, Konfigurationsdatei* 225
*httpd.conf-Notwendigkeiten* 218
*httpd.exe (Windows-Binary)* 206
*httxt2dbm, Hilfsprogramm* 216, 364
*Image-Maps* 393
*Index, automatischer* 381
*Installation, Mac OS X* 152
*Installationsarten* 149
*Installationsverzeichnisse* 155
*Konfigurationsinformationen* 378
*Laufzeitmodelle* 129
*log_server_status, Hilfsprogramm* 216
*logresolve, Hilfsprogramm* 216
*manuell beenden (Windows)* 207
*manuell neu starten (Windows)* 207
*manuell starten (Windows)* 207
*MIME-Konfiguration* 315
*MIME-Type-Ermittlung* 122
*MIME-Types* 316
*mit apachectl steuern* 201
*Module* 133
*neu starten (UNIX)* 202
*Performance* 118
*rotatelogs, Hilfsprogramm* 216
*Sicherheit* 119
*Skalierbarkeit* 119
*Softwarelizenz* 111
*split_logfile, Hilfsprogramm* 216
*SSL* 123
*Stabilität* 119
*starten (UNIX)* 202
*Statusinformationen* 378
*steuern (UNIX)* 195
*steuern (Windows)* 206
*steuern mit kill* 200
*suexec, Hilfsprogramm* 216
*technische Details* 118
*Test mit einfacher Site* 217
*testen* 216
*testen mit ps* 195
*unterstützte Betriebssysteme* 116
*URL-Manipulation* 122
*User- und Group-ID* 267
*Verbreitung* 107, 118
*Versionierung* 120
*Versionsgeschichte* 107
*virtuelle Hosts* 123
*XAMPP, Komplettpaket* 192
*Zeichensatzeinstellungen* 323
*Zugriffsbeschränkung* 122
Apache 2.2, Neuerungen 125
Apache 2.4, Neuerungen 124
Apache Benchmark → ab, Hilfsprogramm 568
Apache Group
*Mitwirkende* 106
Apache Portable Runtime (APR) 108
*Einsatzgebiete (außer HTTPD)* 128
*Funktionsumfang* 127
Apache Software Foundation (ASF) 108
Apache.exe, Windows-Binary 206
apachectl, Steuerskript 201
Apache-Monitor (Windows) 213
*automatisch starten* 214
Apache-Negotiation-Algorithmus 337
Apache-Softwarelizenz
*Version 2.0* 113
apxs, Hilfsprogramm 193, 215
A-Record (DNS) 40
ARPANet 21
ASP.NET 749
ASP.NET einbinden 750
AssignUserID, Direktive 899
Asymmetrische Verschlüsselung 476
Ausdrücke, komplexe (ap_expr) 240
AuthAuthoritative, Direktive 894
AuthBasicAuthoritative, Direktive 418
AuthBasicProvider, Direktive 416
AuthDBDUserPWQuery, Direktive 461
AuthDBDUserRealmQuery, Direktive 462
AuthDBMAuthoritative, Direktive 897
AuthDBMGroupFile, Direktive 432
AuthDBMType, Direktive 432
AuthDBMUserFile, Direktive 431
AuthDefaultAuthoritative, Direktive 472
AuthDigestAlgorithm, Direktive 423
AuthDigestDomain, Direktive 423
AuthDigestEnableQueryStringHack, Variable 420
AuthDigestFile, Direktive 895
AuthDigestGroupFile, Direktive 896
AuthDigestNcCheck, Direktive 426
AuthDigestNonceFormat, Direktive 426
AuthDigestNonceLifetime, Direktive 424
AuthDigestProvider, Direktive 422
AuthDigestQop, Direktive 425
AuthDigestShmemSize, Direktive 425

Authentifizierung
*anonyme* 454
*Apache 2.0* 892
*auf Benutzerseite* 404
*Autorisierung* 400
*Basic-Authentifizierung* 414
*Core-Direktiven* 405
*DBM-Dateien* 426
*Digest-Authentifizierung* 419
*Grundbegriffe* 399
*htpasswd* 414
*im engeren Sinne* 400
*in Apache 2.0* 401
*in Apache 2.2* 401
*Konfigurationsbeispiel* 403
*mod_auth* 414
*mod_auth_digest* 419
*mod_auth_ldap* 434
*mod_authn_anon* 454
*Reihenfolgeproblem* 895
*SQL-Datenbank* 461
*Vergleich zur Adresskontrolle* 399
*vollständige Konfigurationsdatei* 879
*Zugriffskontrolle* 400
AuthFormAuthoritative, Direktive 468
AuthFormBody, Direktive 463
AuthFormDisableNoStore, Direktive 464
AuthFormFakeBasicAuth, Direktive 464
AuthFormLocation, Direktive 465
AuthFormLoginRequiredLocation, Direktive 465
AuthFormLoginSuccessLocation, Direktive 465
AuthFormLogoutLocation, Direktive 466
AuthFormMethod, Direktive 466
AuthFormMimetype, Direktive 466
AuthFormPassword, Direktive 467
AuthFormProvider, Direktive 467
AuthFormSitePassphrase, Direktive 467
AuthFormSize, Direktive 468
AuthFormUsername, Direktive 468
AuthGroupFile, Direktive 417
AuthLDAPAuthoritative, Direktive 448, 897
AuthLDAPBindDN, Direktive 444
AuthLDAPBindPassword, Direktive 445
AuthLDAPCharsetConfig, Direktive 445
AuthLDAPCompareDNOnServer, Direktive 445
AuthLDAPDereferenceAliases, Direktive 446
AuthLDAPEnabled, Direktive 897
AuthLDAPFrontPageHack, Direktive 898
AuthLDAPGroupAttribute, Direktive 446
AuthLDAPGroupAttributeIsDN, Direktive 447
AuthLDAPRemoteUserAttribute, Direktive 448
AuthLDAPRemoteUserIsDN, Direktive 447
AuthLDAPUrl, Direktive 442
AuthMerging, Direktive 412
AuthName, Direktive 406
<AuthnProviderAlias>, Container 469
Authorization, HTTP-Header 72
AuthType, Direktive 406
*form* 462
AuthUserFile, Direktive 417
AuthzDBMAuthoritative, Direktive 433
AuthzDBMType, Direktive 433
AuthzDefaultAuthoritative, Direktive 472
AuthzGroupFileAuthoritative, Direktive 419
AuthzOwnerAuthoritative, Direktive 471
<AuthzProviderAlias>, Container 470
AuthzSendForbiddenOnFailure, Direktive 413
AuthzUserAuthoritative, Direktive 419
automatisch starten
*chkconfig* 204
*System V Init* 203
*UNIX* 203
Autorisierung 400
awk 183

## B

BalancerGrowth, Direktive 600
BalancerMember, Direktive 599
Banana Ware 120
Benutzerverzeichnisse
*veröffentlichen* 372
Berners-Lee, Tim 106
BIND-Nameserver
*CNAME-Record* 41
*Installation* 36
*Konfiguration* 37
*MX-Record* 42
*NS-Record* 41

*Reverse-Lookup-Zone* 38
*Zonendaten-Dateien* 39
*Zonendefinition* 38
Booten
*BSD-Startskripte* 205
*System V Init* 203
BrowserMatch, Direktive 651
BrowserMatchNoCase, Direktive 652
BS2000Account, Direktive 900
BSD-Softwarelizenz 112
BSD-Startskripte 205
BufferedLogs, Direktive 527

## C

Cache 611
*aufräumen* 625
*htcacheclean, Hilfsprogramm* 625
*Konfigurationsbeispiele* 612
*squid* 567
*varnish* 567
Cache-Control, HTTP-Header 73
CacheDefaultExpire, Direktive 615
CacheDeteailHeader, Direktive 616
CacheDirLength, Direktive 622
CacheDirLevels, Direktive 622
CacheDisable, Direktive 614
CacheEnable, Direktive 614
CacheFile, Direktive 573
CacheHeader, Direktive 615
CacheIgnoreCacheControl, Direktive 616
CacheIgnoreNoLastMod, Direktive 616
CacheIgnoreQueryString, Direktive 617
CacheKeyBaseURL, Direktive 617
CacheLastModifiedFactor, Direktive 618
CacheMaxExpire, Direktive 618
CacheMaxFileSize, Direktive 621
CacheMinExpire, Direktive 619
CacheMinFileSize, Direktive 622
CacheNegotiatedDocs, Direktive 342
CacheQuickHandler, Direktive 619
CacheRoot, Direktive 621
CacheSize, Direktive 900
CacheStaleOnError, Direktive 619
CacheStoreExpired, Direktive 620
CacheStoreNoStore, Direktive 620
CacheStorePrivate, Direktive 621
ccTLDs, Tabelle → Länder-Top-Level-Domains 968
CERN-Meta-Dateien 313
CGI → Common Gateway Interface (CGI)
*Non-pa* 313
*SuEXEC* 834
CGI.pm, Perl-Modul 668
*CGI-Methoden* 681
*checkbox_group(), Methode* 687
*cookie(), Methode* 672, 682
*end_form(), Methode* 685
*end_html(), Methode* 673, 684
*filefield(), Methode* 685
*Formularmethoden* 684
*Formularpraxis* 677
*Header einstellen* 672
*header(), Methode* 681
*HTML-Erzeugung* 672
*Import-Referenz* 680
*param(), Methode* 670
*password_field(), Methode* 685
*popup_menu(), Methode* 686
*Pragmata* 680
*prozedural* 669
*radio_group(), Methode* 686
*redirect(), Methode* 682
*reset(), Methode* 687
*self_url(), Methode* 684
*start_form(), Methode* 685
*start_html(), Methode* 673, 683
*submit(), Methode* 687
*textfield(), Methode* 685
*url(), Methode* 684
CGI::Carp, Perl-Modul 669
CGI::Pretty, Perl-Modul 670
cgi-bin, Verzeichnis 633
CGIMapExtension, Direktive 643
cgi-script, Handler 329, 638
CGI-Skripte
*Erzwingen bei GET* 56
*vs. CGI-Programme* 630
CharsetDefault, Direktive 779
CharsetOptions, Direktive 779
CharsetSourceEnc, Direktive 778
check_config, Hook 808
checkbox_group(), CGI.pm-Methode 687
CheckCaseOnly, Direktive 378
CheckSpelling, Direktive 377
child_init, Hook 808
chkconfig (für Apache-Autostart) 204
chmod
*CGI ausführbar machen* 634

ChrootDir, Direktive 829
chroot-Umgebung 826, 829
CLF → Common Log Format 515
Client-Anfrage (HTTP) 51
Cloud Computing 575
Cocoon 108
Combined Log Format 515
Common Gateway Interface (CGI) 629
  *CGI.pm, Perl-Modul* 668
  *CGI-Skript/-Programm* 630
  *Entwicklung* 629
  *Formulardaten manuell einlesen* 667
  *Grundlagen* 630
  *in DocumentRoot-Verzeichnissen* 636
  *Konfiguration in Apache* 633
  *objektorientiert* 669
  *plattformspezifische Einstellungen* 641
  *Programmierung* 665
  *Shebang* 634
  *Skripte ausführbar machen* 634
  *Umgebungsvariablen* 645
  *Verzeichnisse für* 633
  *Zuordnung unter Windows* 634
Common Log Format 515
config, SSI-Element 755
config.layout 166
configure
  *PREFIX* 156
configure, Build-Einstellungen
  *Optionen* 154
configure, Build-Optionen
  *Umgebungsvariablen* 181
CONNECT, HTTP-Methode 61
Connection, HTTP-Header 74
Container
  *<AuthnProviderAlias>* 469
  *<AuthzProviderAlias>* 470
  *<Directory>* 236
  *<DirectoryMatch>* 237
  *<Else>* 240
  *<ElseIf>* 240
  *<Files>* 239
  *<FilesMatch>* 240
  *<If>* 240
  *<IfDefine>* 247
  *<IfModule>* 249
  *<IfVersion>* 251
  *<Limit>* 250
  *<LimitExcept>* 251
  *<Location>* 238
  *<LocationMatch>* 239
  *<Perl>* 727
  *<Proxy>* 584
  *<ProxyMatch>* 585
  *<RequireAll>* 410
  *<RequireAny>* 411
  *<RequireNone>* 412
  *<VirtualHost>* 234
  *für Direktiven* 234
  *Verschachtelung* 253
Content Negotiation 122
ContentDigest, Direktive 301
Content-Encoding, HTTP-Header 74
Content-Language, HTTP-Header 75
Content-Length, HTTP-Header 75
Content-Location, HTTP-Header 75
Content-MD5, HTTP-Header 75
  *mit Apache setzen* 301
Content-Negotiation 332
  *Apache-Negotiation-Algorithmus* 337
  *Direktiven* 341
  *MultiViews* 336
  *servergesteuerte* 333
  *transparente* 339
  *Type-Maps* 333
Content-Range, HTTP-Header 75
Content-Type, HTTP-Header 76
cookie(), CGI.pm-Methode 672, 682
Cookie, HTTP-Header 76
CookieDomain, Direktive 535
CookieExpires, Direktive 535
CookieLog, Direktive 528
CookieName, Direktive 536
Cookies
  *in PHP* 713
  *mit CGI.pm* 682
CookieStyle, Direktive 536
CookieTracking, Direktive 536
CoreDumpDirectory, Direktive 270
Crackertools 826
cronolog 552
CustomLog, Direktive 528

## D

Date, HTTP-Header 76
Datei-Container 235
Dateien
  *.htaccess* 253

Dateiendungen 316
Datenbankverbindungen (Apache 2.2) 457
Datum und Uhrzeit
  *strftime( )* 526
DAV → WebDAV 795
Dav, Direktive 796
DavDepthInfinity, Direktive 797
DavGenericLockDB, Direktive 798
DavLockDB, Direktive 798
DavMinTimeout, Direktive 797
DB 109
DBDExpTime, Direktive 460
DBDKeep, Direktive 460
DBDMax, Direktive 460
DBDMin, Direktive 459
DBDPersist, Direktive 458
DBDPrepareSQL, Direktive 459
DBDriver, Direktive 457
DBI, Perl-Modul 726
DBM-Dateien
  *für RewriteMaps* 363
  *zur Authentifizierung* 426
dbmmanage, Hilfsprogramm 215, 427
DBParams, Direktive 458
DDN-Schichtenmodell 23
default-handler 329
DefaultIcon, Direktive 390
DefaultLanguage, Direktive 327
DefaultType, Direktive 317
Define, Direktive 248
DEFLATE, Filter 775
DeflateBufferSize, Direktive 775
DeflateCompressionLevel, Direktive 776
DeflateFilterNote, Direktive 776
DeflateMemLevel, Direktive 777
DeflateWindowSize, Direktive 777
DELETE, HTTP-Methode 60
Deny, Direktive 295
Dienst
  *Apache als* 209
  *deinstallieren* 209
  *installieren* 209
  *starten* 209
dig, Dienstprogramm 45
Digest-Authentifizierung 419
  *Internet Explorer* 420
<Directory>, Container 236
DirectoryIndex, Direktive 285
<DirectoryMatch>, Container 237
DirectorySlash, Direktive 287
Direktiven
  *<AuthnProviderAlias>, Container* 469
  *<AuthzProviderAlias>, Container* 470
  *<Directory>, Container* 236
  *<DirectoryMatch>, Container* 237
  *<Else>, Container* 240
  *<ElseIf>, Container* 240
  *<Files>, Container* 239
  *<FilesMatch>, Container* 240
  *<If>, Container* 240
  *<IfDefine>, Container* 247
  *<IfModule>, Container* 249
  *<IfVersion>, Container* 251
  *<Limit>, Container* 250
  *<LimitExcept>, Container* 251
  *<Location>, Container* 238
  *<LocationMatch>, Container* 239
  *<Perl>, Container* 727
  *<Proxy>, Container* 584
  *<ProxyMatch>, Container* 585
  *<RequireAll>, Container* 410
  *<RequireAny>, Container* 411
  *<RequireNone>, Container* 412
  *<VirtualHost>, Container* 234
  *AcceptFilter* 263
  *AcceptMutex* 270
  *AcceptPathInfo* 652
  *AccessFileName* 287
  *Action* 643
  *AddAlt* 390
  *AddAltByEncoding* 391
  *AddAltByType* 391
  *AddCharset* 324
  *AddDefaultCharset* 324
  *AddDescription* 392
  *AddEncoding* 326
  *AddHandler* 331
  *AddIcon* 388
  *AddIconByEncoding* 389
  *AddIconByType* 390
  *AddInputFilter* 769
  *AddLanguage* 328
  *AddModuleInfo* 379
  *AddOutputFilter* 768
  *AddOutputFilterByType* 767
  *AddType* 318
  *Alias* 346
  *AliasMatch* 347

*AllowCONNECT* 602
*AllowEncodedSlashes* 653
*AllowMethods* 296
*AllowOverride* 290
*Anonymous* 454
*Anonymous_Authoritative* 456
*Anonymous_LogEmail* 456
*Anonymous_MustGiveEmail* 455
*Anonymous_NoUserID* 455
*Anonymous_VerifyEmail* 455
*AssignUserID* 899
*AuthAuthoritative* 894
*AuthBasicAuthoritative* 418
*AuthBasicProvider* 416
*AuthDBDUserPWQuery* 461
*AuthDBDUserRealmQuery* 462
*AuthDBMAuthoritative* 897
*AuthDBMGroupFile* 432
*AuthDBMType* 432
*AuthDBMUserFile* 431
*AuthDefaultAuthoritative* 472
*AuthDigestAlgorithm* 423
*AuthDigestDomain* 423
*AuthDigestFile* 895
*AuthDigestGroupFile* 896
*AuthDigestNcCheck* 426
*AuthDigestNonceFormat* 426
*AuthDigestNonceLifetime* 424
*AuthDigestProvider* 422
*AuthDigestQop* 425
*AuthDigestShmemSize* 425
*AuthFormAuthoritative* 468
*AuthFormBody* 463
*AuthFormDisableNoStore* 464
*AuthFormFakeBasicAuth* 464
*AuthFormLocation* 465
*AuthFormLoginRequiredLocation* 465
*AuthFormLoginSuccessLocation* 465
*AuthFormLogoutLocation* 466
*AuthFormMethod* 466
*AuthFormMimetype* 466
*AuthFormPassword* 467
*AuthFormProvider* 467
*AuthFormSitePassphrase* 467
*AuthFormSize* 468
*AuthFormUsername* 468
*AuthGroupFile* 417
*AuthLDAPAuthoritative* 448, 897
*AuthLDAPBindDN* 444
*AuthLDAPBindPassword* 445
*AuthLDAPCharsetConfig* 445
*AuthLDAPCompareDNOnServer* 445
*AuthLDAPDereferenceAliases* 446
*AuthLDAPEnabled* 897
*AuthLDAPFrontPageHack* 898
*AuthLDAPGroupAttribute* 446
*AuthLDAPGroupAttributeIsDN* 447
*AuthLDAPRemoteUserAttribute* 448
*AuthLDAPRemoteUserIsDN* 447
*AuthLDAPUrl* 442
*AuthMerging* 412
*AuthName* 406
*AuthType* 406
*AuthUserFile* 417
*AuthzDBMAuthoritative* 433
*AuthzDBMType* 433
*AuthzDefaultAuthoritative* 472
*AuthzGroupFileAuthoritative* 419
*AuthzOwnerAuthoritative* 471
*AuthzSendForbiddenOnFailure* 413
*AuthzUserAuthoritative* 419
*BalancerGrowth* 600
*BalancerMember* 599
*BrowserMatch* 651
*BrowserMatchNoCase* 652
*BS2000Account* 900
*BufferedLogs* 527
*CacheDefaultExpire* 615
*CacheDetailHeader* 616
*CacheDirLength* 622
*CacheDirLevels* 622
*CacheDisable* 614
*CacheEnable* 614
*CacheFile* 573
*CacheHeader* 615
*CacheIgnoreCacheControl* 616
*CacheIgnoreNoLastMod* 616
*CacheIgnoreQueryString* 617
*CacheKeyBaseURL* 617
*CacheLastModifiedFactor* 618
*CacheMaxExpire* 618
*CacheMaxFileSize* 621
*CacheMinExpire* 619
*CacheMinFileSize* 622
*CacheNegotiatedDocs* 342
*CacheQuickHandler* 619
*CacheRoot* 621
*CacheSize* 900

*CacheStaleOnError* 619
*CacheStoreExpired* 620
*CacheStoreNoStore* 620
*CacheStorePrivate* 621
*CGIMapExtension* 643
*CharsetDefault* 779
*CharsetOptions* 779
*CharsetSourceEnc* 778
*CheckCaseOnly* 378
*CheckSpelling* 377
*ChrootDir* 829
*Container* 234
*ContentDigest* 301
*CookieDomain* 535
*CookieExpires* 535
*CookieLog* 528
*CookieName* 536
*CookieStyle* 536
*CookieTracking* 536
*CoreDumpDirectory* 270
*CustomLog* 528
*Dateiendungen als Wert* 231
*Dav* 796
*DavDepthInfinity* 797
*DavGenericLockDB* 798
*DavLockDB* 798
*DavMinTimeout* 797
*DBDExpTime* 460
*DBDKeep* 460
*DBDMax* 460
*DBDMin* 459
*DBDPersist* 458
*DBDPrepareSQL* 459
*DBDriver* 457
*DBParams* 458
*DefaultIcon* 390
*DefaultLanguage* 327
*DefaultType* 317
*Define* 248
*DeflateBufferSize* 775
*DeflateCompressionLevel* 776
*DeflateFilterNote* 776
*DeflateMemLevel* 777
*DeflateWindowSize* 777
*Deny* 295
*DirectoryIndex* 285
*DirectorySlash* 287
*DocumentRoot* 284
*DumpIOInput* 533
*DumpIOLogLevel* 534
*DumpIOOutput* 533
*EnableExceptionHook* 833
*EnableMMAP* 571
*EnableSendfile* 571
*Error* 250
*ErrorDocument* 375
*ErrorLog* 517
*ErrorLogFormat* 518
*Example* 806
*ExpiresActive* 309
*ExpiresByType* 312
*ExpiresDefault* 310
*ExtendedStatus* 380
*ExtFilterDefine* 786
*ExtFilterOptions* 789
*feste Werte* 231
*FileETag* 302
*FilterChain* 774
*FilterDeclare* 771
*FilterProtocol* 773
*FilterProvider* 771
*FilterTrace* 774
*ForceLanguagePriority* 343
*ForceType* 318
*ForensicLog* 532
*für Content-Negotiation* 341
*für virtuelle Hosts* 234
*GracefulShutdownTimeout* 266
*Group* 268
*Header* 303
*HeaderName* 392
*HolidayMonth* 808
*HolidayURL* 808
*HostnameLookups* 522
*IASPIFakeAsync* 748
*IdentityCheck* 523
*ImapBase* 396
*ImapDefault* 397
*ImapMenu* 397
*Include* 255
*IndexIgnore* 386
*IndexIgnoreReset* 387
*IndexOptions* 383
*IndexOrderDefault* 387
*IndexStyleSheet* 385
*ISAPIAppendLogToErrors* 747
*ISAPIAppendLogToQuery* 747
*ISAPICacheFile* 748

*ISAPILogNotSupported* 748
*ISAPIReadAheadBuffer* 749
*KeepAlive* 258
*KeepAliveTimeout* 259
*komplexe Ausdrücke* 232
*Kontextangabe* 233
*Kontexte* 234
*LanguagePriority* 343
*LDAPCacheEntries* 449
*LDAPCacheTTL* 450
*LDAPOpCacheEntries* 450
*LDAPOpCacheTTL* 451
*LDAPSharedCacheFile* 451
*LDAPSharedCacheSize* 452
*LDAPTrustedCA* 898
*LDAPTrustedCAType* 899
*LDAPTrustedClientCert* 452
*LDAPTrustedGlobalCert* 453
*LDAPTrustedMode* 453
*LimitInternalRecursion* 830
*LimitRequestBody* 830
*LimitRequestFields* 830
*LimitRequestFieldSize* 831
*LimitRequestLine* 831
*LimitXMLRequestBody* 831
*Listen* 260
*ListenBackLog* 269
*LoadFile* 265
*LoadModule* 264
*LockFile* 271
*LogFormat* 530
*LogLevel* 520
*LogMessage* 538
*LuaCodeCache* 742
*LuaHook** 743
*LuaInherit* 745
*LuaMapHandler* 745
*MaxClients* 271
*MaxKeepAliveRequests* 258
*MaxMemFree* 272
*MaxRequestsPerChild* 272
*MaxRequestsPerThread* 278
*MaxSpareServers* 277
*MaxSpareThreads* 272
*MaxThreads* 278
*MCacheMaxObjectCount* 623
*MCacheMaxObjectSize* 623
*MCacheMaxStreamingBuffer* 624
*MCacheMinObjectSize* 624
*MCacheRemovalAlgorithm* 625
*MCacheSize* 623
*MetaDir* 314
*MetaFiles* 314
*MetaSuffix* 314
*MimeMagicFile* 321
*MinSpareServers* 277
*MinSpareThreads* 273
*MMapFile* 572
*ModMimeUsePathInfo* 320
*Modulangabe* 233
*mögliche Werte* 230
*MultiViewsMatch* 341
*Mutex* 261
*NameVirtualHost* 561
*NoProxy* 586
*numerische Werte* 230
*NWSSLTrustedCerts* 513
*ohne Wert* 232
*On|Off* 230
*Options* 288
*Order* 291
*PerlModule* 722
*Pfadangaben* 230
*PidFile* 267
*plattformspezifische* 266
*Protocol* 261
*ProtocolDaytime* 820
*ProtocolEcho* 800
*ProxyAddHeaders* 589
*ProxyBadHeader* 587
*ProxyBlock* 588
*ProxyDomain* 588
*ProxyErrorOverride* 589
*ProxyExpressDBMFile* 610
*ProxyExpressDBMType* 611
*ProxyExpressEnable* 610
*ProxyFtpDirCharset* 590
*ProxyFtpEscapeWildcards* 590
*ProxyFtpListOnWildcard* 591
*ProxyIOBufferSize* 591
*ProxyMaxForwards* 591
*ProxyPass* 592
*ProxyPassInterpolateEnv* 597
*ProxyPassMatch* 596
*ProxyPassReverse* 597
*ProxyPassReverseCookieDomain* 598
*ProxyPassReverseCookiePath* 599
*ProxyPreserveHost* 600

*ProxyReceiveBufferSize* 600
*ProxyRemote* 586
*ProxyRemoteMatch* 586
*ProxyRequests* 585
*ProxySet* 596
*ProxySourceAddress* 589
*ProxyStatus* 602
*ProxyTimeout* 601
*ProxyVia* 601
*ReadmeName* 393
*ReceiveBufferSize* 274
*Redirect* 348
*RedirectMatch* 350
*RedirectPermanent* 352
*RedirectTemp* 352
*reguläre Ausdrücke* 231
*RemoveCharset* 325
*RemoveEncoding* 327
*RemoveHandler* 332
*RemoveInputFilter* 770
*RemoveLanguage* 328
*RemoveOutputFilter* 769
*RemoveType* 319
*RequestHeader* 308
*RequestReadTimeout* 259
*Require* 407
*RewriteBase* 360
*RewriteCond* 358
*RewriteEngine* 353
*RewriteLock* 367
*RewriteLog* 537
*RewriteLogLevel* 538
*RewriteMap* 360
*RewriteOptions* 368
*RewriteRule* 354
*RLimitCPU* 832
*RLimitMEM* 832
*RLimitNPROC* 833
*Satisfy* 409
*ScoreBoardFile* 273
*Script* 644
*ScriptAlias* 633
*ScriptAliasMatch* 635
*ScriptInterpreterSource* 641
*ScriptLog* 639
*ScriptLogBuffer* 639
*ScriptLogSize* 640
*ScriptSockSize* 640
*SecureListen* 514
*SeeRequestTail* 380
*SendBufferSize* 274
*ServerAdmin* 280
*ServerAlias* 562
*Server-Kontext* 234
*ServerLimit* 274
*ServerName* 280
*ServerPath* 562
*ServerRoot* 257
*ServerSignature* 284
*ServerTokens* 282
*Session* 656
*SessionCookieName* 659
*SessionCookieName2* 659
*SessionCookieRemove* 660
*SessionCryptoCipher* 665
*SessionCryptoDriver* 664
*SessionCryptoPassphrase* 664
*SessionDBDCookieName* 660
*SessionDBDCookieName2* 660
*SessionDBDCookieRemove* 661
*SessionDBDDeleteLabel* 661, 662
*SessionDBDInsertLabel* 662
*SessionDBDPerUser* 663
*SessionDBDSelectLabel* 663
*SessionDBDUpdateLabel* 663
*SessionEnv* 656
*SessionExclude* 656
*SessionHeader* 657
*SessionInclude* 658
*SessionMaxAge* 658
*SetEnvIf* 649
*SetEnvIfNoCase* 650
*SetHandler* 330
*SetInputFilter* 768
*SetOutputFilter* 766
*SSIAccessEnable* 764
*SSIEndTag* 762
*SSIErrorMsg* 762
*SSILegacyExprParser* 764
*SSIStartTag* 761
*SSITimeFormat* 763
*SSIUndefinedEcho* 763
*SSLCACertificateChainFile* 489
*SSLCACertificateFile* 486
*SSLCACertificatePath* 486
*SSLCADNRequestFile* 487

*SSLCADNRequestPath* 487
*SSLCARevocationCheck* 488
*SSLCARevocationFile* 488
*SSLCARevocationPath* 489
*SSLCertificateFile* 489
*SSLCertificateKeyFile* 490
*SSLCipherSuite* 490
*SSLCryptoDevice* 493
*SSLEngine* 494
*SSLFIPS* 495
*SSLHonorCipherOrder* 495
*SSLMutex* 495
*SSLOCSPDefaultResponder* 497
*SSLOCSPEnable* 497
*SSLOCSPOverrideResponder* 497
*SSLOCSPResponderTimeout* 498
*SSLOCSPResponseMaxAge* 498
*SSLOCSPResponseTimeSkew* 498
*SSLOptions* 502
*SSLPassPhraseDialog* 503
*SSLProtocol* 504
*SSLProxyCACertificateFile* 510
*SSLProxyCACertificatePath* 510
*SSLProxyCARevocationFile* 511
*SSLProxyCARevocationPath* 511
*SSLProxyCipherSuite* 511
*SSLProxyEngine* 512
*SSLProxyMachineCertificateFile* 512
*SSLProxyMachineCertificatePath* 512
*SSLProxyProtocol* 512
*SSLProxyVerify* 513
*SSLProxyVerifyDepth* 513
*SSLRandomSeed* 505
*SSLRequire* 506
*SSLRequireSSL* 507
*SSLSessionCache* 508
*SSLSessionCacheTimeout* 508
*SSLStaplingCache* 499
*SSLStaplingErrorCacheTimeout* 499
*SSLStaplingFakeTryLater* 500
*SSLStaplingForceURL* 500
*SSLStaplingResponderTimeout* 500
*SSLStaplingResponseMaxAge* 501
*SSLStaplingResponseTimeSkew* 501
*SSLStaplingReturnResponderErrors* 501
*SSLStaplingStandardCacheTimeout* 502
*SSLUserName* 508
*SSLUseStapling* 499
*SSLVerifyClient* 509
*SSLVerifyOpen* 510
*Standardwertangabe* 234
*StartServers* 275
*StartThreads* 275
*String-Werte* 231
*Substitute* 782
*Suexec* 836
*SuexecUserGroup* 837
*Syntaxangabe* 233
*Syntaxschema (in diesem Buch)* 232
*ThreadLimit* 276
*ThreadsPerChild* 276
*ThreadStackSize* 278
*TraceEnable* 265
*TransferLog* 531
*TypesConfig* 320
*UnDefine* 249
*UnsetEnv* 649
*UseCanonicalName* 281
*UseCanonicalPhysicalPort* 282
*User* 267
*UserDir* 373
*Versionsangabe* 233
*Verzeichniscontainer* 235
*VirtualDocumentRoot* 564
*VirtualDocumentRootIP* 565
*VirtualScriptAlias* 565
*VirtualScriptAliasIP* 565
*wichtigste im Überblick* 221
*Win32DisableAcceptEx* 279
*XBitHack* 763
*xml2EncAlias* 780
*xml2EncDefault* 781
*xml2StartParse* 781
*zur Grundkonfiguration* 256
Direktiven, Allow 292
Direktiven, TimeOut 257
--disable-mods-shared, configure-Option 171
--disable-modules, configure-Option 171
DocumentRoot, Direktive 284
Domain Name System (DNS) 33
*Funktionsweise* 33
*Nameserver* 35
*Sicherheitsproblem* 36
Doxygen 809
DumpIOInput, Direktive 533
DumpIOLogLevel, Direktive 534
DumpIOOutput, Direktive 533

## E


echo, SSI-Element 756
Eddie, Load-Balancer 577
Eigener Webserver (Perl)
  *Accept-Schleife* 90
  *Benutzerdokumentation* 102
  *Client-Anfrage* 91
  *Dateigröße ermitteln* 95
  *Datumsformate* 92
  *Implementierungsdetails* 87
  *Kommandozeilenparameter* 90
  *Logging* 95
  *MIME-Type ermitteln* 94
  *MIME-Types* 89
  *Projektanforderungen* 87
  *Quellcode* 97
  *Server-Antwort* 93
  *Socket-Erzeugung* 90
  *Startseite* 92
eigener Webserver (Perl) 86
Einwegverschlüsselung 476
<Else>, Container 240
<ElseIf>, Container 240
EnableExceptionHook, Direktive 833
EnableMMAP, Direktive 571
--enable-mods-shared, configure-Option 171
--enable-modules, configure-Option 170
EnableSendfile, Direktive 571
--enable-so, configure-Option 170
Encoding → MIME-Codierung 326
end_form(), CGI.pm-Methode 685
end_html(), CGI.pm-Methode 673, 684
Engelschall, Ralf S. 352, 475
Error, Direktive 250
ErrorDocument, Direktive 375
ErrorLog, Direktive 517
ErrorLogFormat, Direktive 518
ETag, HTTP-Header 77
  *mit Apache setzen* 302
Example, Direktive 806
example-handler, Handler 806
exec, SSI-Element 757
Expect, HTTP-Header 77
Expires, HTTP-Header 78
  *mit Apache setzen* 309
ExpiresActive, Direktive 309
ExpiresByType, Direktive 312
ExpiresDefault, Direktive 310
ExtendedStatus, Direktive 380
ExtFilterDefine, Direktive 786
ExtFilterOptions, Direktive 789


## F


Fancy-Index 382
FastCGI 632
fcgistarter, Hilfsprogramm 216, 632
Fehlerbehandlung 375
Fielding, Roy T. 106
FileETag, Direktive 302
filefield(), CGI.pm-Methode 685
<Files>, Container 239
<FilesMatch>, Container 240
Filter 765
  *colors, eigenes Beispiel* 793
  *DEFLATE* 775
  *Direktiven für* 765
  *externe* 786
  *INCLUDES* 754
  *Perl-Beispiele* 789
  *sourceview, eigenes Beispiel* 792
  *txt2html, eigenes Beispiel* 791
  *x4u, eigenes Beispiel* 790
Filter Chain 766, 770
FilterChain, Direktive 774
FilterDeclare, Direktive 771
FilterProtocol, Direktive 773
FilterProvider, Direktive 771
FilterTrace, Direktive 774
Firewalls 825
fixups, Hook 808
flastmod, SSI-Element 758
ForceLanguagePriority, Direktive 343
ForceType, Direktive 318
Ford, Andrew 552
ForensicLog, Direktive 532
forensische Log-Dateien 532
Forking-Server 129
Forward-Proxy 579
  *Konfigurationsbeispiele* 582
  *mit Cache* 613
  *vollständige Konfigurationsdatei* 881
freie Software 105
  *Versionierung* 120
From, HTTP-Header 78
fsize, SSI-Element 758

## G

Generische Top-Level-Domains, Tabelle 967
GET, HTTP-Methode 52
*CGI-Ausführung erzwingen* 56
*Formularversand* 54
GracefulShutdownTimeout, Direktive 266
Group, Direktive 268
gTLDs, Tabelle → generische Top-Level-Domain 967
Gültigkeitsdauer (Dokumente) 309

## H

Hagberg, Eric 106
Handler 329
*cgi-script* 329, 638
*default-handler* 329
*example-handler* 806
*imap-file* 329, 394
*ldap-status* 449
*send-as-is* 312, 329
*server-info* 329, 378
*server-parsed* 329
*server-status* 329, 379
*type-map* 330, 334
handler, Hook 808
Hartill, Rob 106
HEAD, HTTP-Methode 55
header(), CGI.pm-Methode 681
Header, Direktive 303
header_parser, Hook 808
HeaderName, Direktive 392
Hilfsprogramme
*dbmmanage* 427
*fcgistarter* 632
*htcacheclean* 625
*htdbm* 430
*htdigest* 421
*htpasswd* 414
Hilfsprogramme (mit Apache geliefert) 215
HolidayMonth, Direktive 808
HolidayURL, Direktive 808
Home-Verzeichnisse → Benutzerverzeichnisse 372
Hooks 807
*check_config* 808
*child_init* 808
*fixups* 808
*handler* 808
*header_parser* 808
*insert_filter* 808
*map_to_storage* 808
*open_logs* 808
*post_config* 808
*pre_config* 808
*translate_name* 808
*type_checker* 808
Host, HTTP-Header 78
HostnameLookups, Direktive 522
.htaccess
*Benutzerverzeichnisse* 372
*DocumentRoot-Einstellungen* 297
*Verzeichnisvoreinstellung* 297
.htaccess-Dateien 253
*erlaubte Direktiven festlegen* 290
*Namen ändern* 287
*sinnvoller Einsatz* 255
htcacheclean, Hilfsprogramm 216, 625
htdbm, Hilfsprogramm 216, 430
htdigest, Hilfsprogramm 216, 421
htpasswd, Hilfsprogramm 216, 414
HTTP-Anfrage 51
httpd, Binary 196
*Kommandozeilenoptionen* 197
httpd.conf
*Abschnitte* 229
*Aufbau* 225
*bedingte Anweisungen* 240
*CGI-Einstellungen* 633
*Container* 234
*Container für virtuelle Hosts* 234
*Container-Verschachtelung* 253
*Direktiven-Syntaxschema* 232
*Direktivenwerte* 230
*Entwicklung* 225
*erstellen* 226
*Handler* 329
*Konfigurationsdatei-Import* 255
*Kontexte* 234
*lange Zeilen trennen* 229
*Notwendiges im Überblick* 218
*plattformspezifische Einstellungen* 266
*Server-Kontext* 234
*Syntax* 229
*Verzeichniscontainer* 235
*Verzeichnisoptionen* 288
*wichtigste Direktiven* 221

httpd.conf, Volltext 842
httpd-autoindex.conf, Volltext 858, 865
httpd-dav.conf, Volltext 869
httpd-default.conf, Volltext 870
httpd-languages.conf, Volltext 861
httpd-manual.conf, Volltext 868
httpd-mpm.conf, Volltext 854
httpd-multilang-errordoc.conf, Volltext 857
httpd-ssl.conf, Volltext 872
httpd-userdir.conf, Volltext 864
httpd-vhosts.conf, Volltext 867
HTTP-Header
*Accept* 70
*Accept-Charset* 71
*Accept-Encoding* 71
*Accept-Language* 72
*Accept-Ranges* 72
*Age* 72
*Allow* 72
*Alternates* 340
*Authorization* 72
*Cache-Control* 73
*Connection* 74
*Content-Encoding* 74
*Content-Language* 75
*Content-Length* 75
*Content-Location* 75
*Content-MD5* 75, 301
*Content-Range* 75
*Content-Type* 76
*Cookie* 76
*Date* 76
*ETag* 77, 302
*Expect* 77
*Expires* 78, 309
*From* 78
*Host* 78
*If-Match* 78
*If-Modified-Since* 79
*If-None-Match* 79
*If-Range* 79
*If-Unmodified-Since* 80
*Last-Modified* 80
*Location* 80
*Manipulation mit Apache* 301
*Max-Forwards* 80
*mit Apache modifizieren* 303
*mit CGI.pm setzen* 672
*Negotiate* 81, 339
*Pragma* 81
*Proxy-Authenticate* 81
*Proxy-Authorization* 82
*Range* 82
*Referer* 82
*Retry-After* 82
*Server* 83
*Set-Cookie* 83
*TCN* 340
*TE* 84
*Trailer* 84
*Transfer-Encoding* 84
*Übersicht* 68
*Upgrade* 84
*User-Agent* 84
*Vary* 85
*Via* 85
*Warning* 85
*WWW-Authenticate* 86
HTTP-Methoden
*CONNECT* 61
*DELETE* 60
*GET* 52
*HEAD* 55
*Idempotenz* 56
*OPTIONS* 62
*POST* 56
*PUT* 58
*TRACE* 60
HTTP-Protokoll → Hypertext Transfer Protocol 49
HTTPS-Verbindungen 477
httxt2dbm, Hilfsprogramm 216, 364
Hypertext Transfer Protocol (HTTP) 49
*Anfrage* 51
*CONNECT-Methode* 61
*DELETE-Methode* 60
*GET-Formularversand* 54
*GET-Methode* 52
*Header, Übersicht* 68
*Header-Manipulation mit Apa* 301
*HEAD-Methode* 55
*idempotente Methoden* 56
*Kommunikationsablauf* 49
*OPTIONS-Methode* 62
*POST-Formularversand* 56
*POST-Methode* 56
*PROFIND-Methode* 800

*PUT-Methode* 58
*Query-String* 55
*Statuscodes* 62
*TRACE-Methode* 60

## I

IANA (Internet Assigned Numbers Authority) 27
IASPIFakeAsync, Direktive 748
Idempotenz (HTTP) 56
IdentityCheck, Direktive 523
<If>, Container 240
if/elif/else/endif, SSI-Elemente 759
<IfDefine>, Container 247
If-Match, HTTP-Header 78
If-Modified-Since, HTTP-Header 79
<IfModule>, Container 249
If-None-Match, HTTP-Header 79
If-Range, HTTP-Header 79
If-Unmodified-Since, HTTP-Header 80
<IfVersion>, Container 251
Image-Maps 393
*Syntax* 394
ImapBase, Direktive 396
ImapDefault, Direktive 397
imap-file, Handler 329, 394
ImapMenu, Direktive 397
Include, Direktive 255
include, SSI-Element 758
INCLUDES, Filter 754
inconv, Programm 778
Incubator 109
Index
*aktivieren* 382
*automatisch generierter* 381
*Fancy-Index* 382
IndexIgnore, Direktive 386
IndexIgnoreReset, Direktive 387
IndexOptions, Direktive 383
IndexOrderDefault, Direktive 387
IndexStyleSheet, Direktive 385
insert_filter, Hook 808
Installation
*Windows* 187
Installationslayouts 158
*Apache* 158
*BeOS* 162
*BSD/OS* 163
*Darwin* 160
*Debian* 165
*eigene* 166
*FreeBSD* 164
*GNU* 159
*Mac OS X* 160
*OpenBSD* 164
*opt* 161
*RedHat* 161
*Solaris* 163
*SuSE* 162
Internet Information Server
*Active Server Pages (ASP)* 113
Internet Protocol (IP) 24
*IP-Adressen* 25
*IPv4 und IPv6* 24
*Routing* 29
*TTL* 30
Intrusion Detection Systems 825
IP-Adressen
*CIDR* 26
*IP-basierte virtuelle Hosts* 557
*IPv6* 28
*Klassen* 26
*private* 27
*spezielle* 28
*Subnet Mask* 26
ISAPI 114, 746
ISAPIAppendLogToErrors, Direktive 747
ISAPIAppendLogToQuery, Direktive 747
ISAPICacheFile, Direktive 748
ISAPILogNotSupported, Direktive 748
ISAPIReadAheadBuffer, Direktive 749
ISO-Sprachkürzel, Tabelle 954

## J

Jakarta 109
James (Java-Mailserver) 109
Java
*Tomcat-Server* 729
JavaServer Pages (JSP) 729
*Beispiel* 737
*MySQL-Zugriff* 737
JavaServlets 729
*Beispiel* 736
*MySQL-Zugriff* 737
JDBC-Schnittstelle 737
JSP → JavaServer Pages (JSP) 729

## K

Kaminsky, Dan 36
KeepAlive, Direktive 258
KeepAliveTimeout, Direktive 259
Kew, Nick 457
kill
  *Apache steuern mit* 200
Kompilieren
  *configure* 154
  *Einführung* 150
  *Module* 167
  *Quellcode herunterladen* 150
  *UNIX, Überblick* 152
  *weitere Optionen* 178
  *Windows* 182
  *Windows, IDE* 187
  *Windows, Kommandozeile* 184
Konfiguration
  *.htaccess* 253
  *Dateien importieren* 255
  *Notwendiges im Überblick* 218
Konfigurationsdatei → httpd.conf 225
Konfigurationsdateien
  *Apache 2.0* 226
  *Apache 2.2* 226
  *openSUSE Linux* 227
  *slapd.conf (OpenLDAP)* 438
  *Volltext* 841
Konfigurationsdirektiven → Direktiven 221
Konfigurationsinformationen 378
Kontexte
  *für Direktiven* 234
Kryptografie → Verschlüsselung 475

## L

Länder-Top-Level-Domains, Tabelle 968
LanguagePriority, Direktive 343
Last-Modified, HTTP-Header 80
Laurie, Ben 475, 532
lbnamed, DNS-Load-Balancer 577
LDAP
  *Attribute* 436
  *dc, Wurzeleintrag* 435
  *Grundlagen* 434
  *LDIF-Format* 434
  *Objektklassen* 435
  *OpenLDAP-Server* 437
  *Organisationseinheiten* 435
LDAP → Lightweight Directory Access Protocol 434
ldapadd, LDAP-Kommando 440
LDAPCacheEntries, Direktive 449
LDAPCacheTTL, Direktive 450
ldapmodify, LDAP-Kommando 440
LDAPOpCacheEntries, Direktive 450
LDAPOpCacheTTL, Direktive 451
ldapsearch, LDAP-Kommando 440
LDAPSharedCacheFile, Direktive 451
LDAPSharedCacheSize, Direktive 452
ldap-status, Handler 449
LDAPTrustedCA, Direktive 898
LDAPTrustedCAType, Direktive 899
LDAPTrustedClientCert, Direktive 452
LDAPTrustedGlobalCert, Direktive 453
LDAPTrustedMode, Direktive 453
LDIF 434
leader, MPM (veraltet) 891
Lenya 109
Lerdorf, Rasmus 690
lighttpd, Webserver 114
Lightweight Directory Access Protocol (LDAP) 434
  *Connection-Poolin* 448
<Limit>, Container 250
<LimitExcept>, Container 251
LimitInternalRecursion, Direktive 830
LimitRequestBody, Direktive 830
LimitRequestFields, Direktive 830
LimitRequestFieldSize, Direktive 831
LimitRequestLine, Direktive 831
LimitXMLRequestBody, Direktive 831
Linux
  *Apache steuern* 195
  *apachectl, Steuerskript* 201
  *Apache-Verwaltung (openSUSE)* 205
  *mod_perl-Installation* 720
  *MySQL-Installation* 691
  *Perl-Installation* 720
  *PHP-Installation* 698
  *Runlevel* 203
  *Runlevel-Editor (openSUSE)* 206
  *System V Init* 203
  *Tomcat-Installation* 730
Listen, Direktive 260
ListenBackLog, Direktive 269
Load-Balancing 574
  *mit mod_proxy* 583
  *mit mod_rewrite* 575

*Round-Robin-DNS* 575
*spezielle Lösungen* 577
LoadFile, Direktive 265
LoadModule, Direktive 264
<Location>, Container 238
Location, HTTP-Header 80
<LocationMatch>, Container 239
LockFile, Direktive 271
log_server_status, Hilfsprogramm 216, 541
log4j 109
Log-Dateien 515
*Combined Log Format* 515
*Common Log Format* 515
*cronolog* 552
*Datums- und Uhrzeitformate* 526
*forensische* 532
*Formatdefinitionen* 524
*logresolve, Hilfsprogramm* 541
*Mescalero* 553
*rotatelogs, Hilfsprogramm* 540
*split-logfile, Hilfsprogramm* 541
*Webalizer* 553
*Wusage* 553
LogFormat, Direktive 530
Logging 515
*Analyse-Tools* 552
*Apache-Direktiven* 516
*CGI-Skripte* 639
*eigene Perl-Skripte* 542
*formatierte Ausgabe (Perl)* 542
*Hilfsprogramme* 539
*in Datenbank schreiben (Perl)* 548
*Statistik (Perl)* 545
LogLevel, Direktive 520
LogMessage, Direktive 538
logresolve, Hilfsprogramm 216, 541
Lua, Programmiersprache 740
LuaCodeCache, Direktive 742
LuaHook*, Direktiven 743
LuaInherit, Direktive 745
LuaMapHandler, Direktive 745
Lucene 110

## M

Mac OS X
*Apache installieren* 152
*Apache-Verwaltung* 206
Makefile 154
map_to_storage, Hook 808
Matsumoto, Yukihiro 749
Matz → Matsumoto, Yukihiro 749
Maven 110
MaxClients, Direktive 271
Max-Forwards, HTTP-Header 80
MaxKeepAliveRequests, Direktive 258
MaxMemFree, Direktive 272
MaxRequestsDirektive 272
MaxRequestsPerThread, Direktive 278
MaxSpareServers, Direktive 277
MaxSpareThreads, Direktive 272
MaxThreads, Direktive 278
MCacheMaxObjectCount, Direktive 623
MCacheMaxObjectSize, Direktive 623
MCacheMaxStreamingBuffer, Direktive 624
MCacheMinObjectSize, Direktive 624
MCacheRemovalAlgorithm, Direktive 625
MCacheSize, Direktive 623
McCool, Rob 106
McEachern, Doug 719
Meritocracy 108
Mescalero 553
Meta-Dateien 313
MetaDir, Direktive 314
MetaFiles, Direktive 314
MetaSuffix, Direktive 314
Microsoft .NET 114, 749
MIME-Codierung 326
MIME-Konfiguration 315
MIME-Magic-Datei 321
MimeMagicFile, Direktive 321
MIME-Spracheinstellungen 327
MIME-Type
*Einstellung in Apache* 316
*für Webformulare* 56
MIME-Types, Tabelle 935
MinSpareServers, Direktive 277
MinSpareThreads, Direktive 273
MMapFile, Direktive 572
mod_access, Modul 291
mod_actions, Modul 643
mod_alias, Modul 346, 633
mod_allowmethods, Modul 296
mod_asis, Modul 312
mod_aspdotnet, Modul 750
mod_auth, Modul 414
mod_auth_basic, Modul 414

mod_auth_dbm, Modul 426
mod_auth_digest, Modul 419
*und Internet Explorer* 420
mod_auth_form, Modul 462
mod_auth_ldap, Modul 434
mod_authn_alias, Modul 469
mod_authn_anon, Modul 454
mod_authn_core, Modul 406
mod_authn_dbd, Modul 461
mod_authn_dbm, Modul 426
mod_authn_default, Modul 471
mod_authn_file, Modul 414
mod_authnz_ldap, Modul 434
mod_authz_core, Modul 407
mod_authz_dbm, Modul 426
mod_authz_default, Modul 471
mod_authz_groupfile 414
mod_authz_host, Modul 291
mod_authz_owner, Modul 471
mod_authz_user, Modul 414
mod_autoindex, Modul 382
mod_cache, Modul 614
mod_cache_disk, Modul 621
mod_cern_meta, Modul 313
mod_cgi, Modul 633
mod_cgid, Modul 633
mod_charset_lite, Modul 778
mod_dav, Modul 795
*Konfiguration* 796
mod_dav_fs, Modul 795
mod_dav_lock, Modul 798
mod_daytime, Modul (eigenes) 820
mod_dbd, Modul 457
mod_deflate, Modul 775
mod_dir, Modul 285
mod_disk_cache, Modul 621
mod_dumpio, Modul 533
mod_echo, Modul 799
mod_env, Modul 648
mod_example, Modul 806
mod_expires, Modul 309
mod_ext_filter, Modul 786
mod_fcgid, Modul 632
mod_file_cache, Modul 572
mod_filter, Modul 770
mod_ftpd, Modul 799
mod_headers, Modul 303
mod_holiday, Modul (eigenes) 808
mod_ident, Modul 523
mod_imagemap, Modul 393
mod_imap, Modul 393
mod_include, Modul 761
mod_info, Modul 378
mod_isapi, Modul 746
mod_jk, Modul+ 730, 734
mod_lbmethod_bybusiness, Modul 595
mod_lbmethod_byrequests, Modul 595
mod_lbmethod_bytraffic, Modul 595
mod_lbmethod_heartbeat, Modul 595
mod_ldap, Modul 448
mod_log_config, Modul 524
mod_log_debug, Modul 538
mod_log_forensic, Modul 532
mod_lua, Modul 740
mod_mem_cache, Modul 623
mod_mime, Modul 315
mod_mime_magic, Modul 321
mod_mono, Modul 749
mod_negotiation, Modul 332
mod_nw_ssl, Modul 513
mod_perl 110, 719
*Installation, UNIX* 720
*Installation, Windows* 724
*MySQL-Zugriff* 726
*PerlModule, Direktive* 722
*Startup-Datei* 723
mod_pop3, Modul 799
mod_proxy, Modul 580
mod_proxy_ajp, Modul 580
mod_proxy_balancer, Modul 580
mod_proxy_connect, Modul 580
mod_proxy_express, Modul 610
mod_proxy_fcgi, Modul 581, 632
mod_proxy_fdpass, Modul 581
mod_proxy_ftp, Modul 580
mod_proxy_html, Modul 603
mod_proxy_http, Modul 580
mod_proxy_scgi, Modul 581
mod_python, Modul 749
mod_reqtimeout, Modul 259
mod_rewrite, Modul 352
*Beispiele* 368
*für Load-Balancing* 575
*für Session-Tracking* 371
*Logging-Direktiven* 537
mod_ruby, Modul 749
mod_security, Modul 837
mod_session, Modul 654

mod_session_cookie, Modul 659
mod_session_crypto, Modul 664
mod_session_dbd, Modul 660
mod_setenvif, Modul 648
mod_speling, Modul 376
mod_ssl, Modul
  *Grundkonfiguration* 482
  *im Proxy-Betrieb* 510
  *Umgebungsvariablen* 484
mod_status, Modul 379
mod_substitute, Modul 781
mod_unique_id, Modul 532
mod_userdir, Modul 372
mod_usertrack, Modul 534
mod_vhost_alias, Modul 563
  *Formatkürzel* 563
mod_xml2enc, Modul 780
ModMimeUsePathInfo, Direktive 320
Module
  *Apache 2.0* 892
  *Arbeitsablauf* 807
  *Dynamic Shared Objects (DSO)* 169
  *externe* 146
  *Liste* 134
  *mod_access* 291
  *mod_actions* 643
  *mod_alias* 346, 633
  *mod_allowmethods* 296
  *mod_asis* 312
  *mod_aspdotnet* 750
  *mod_auth* 414
  *mod_auth_basic* 414
  *mod_auth_dbm* 426
  *mod_auth_digest* 419
  *mod_auth_form* 462
  *mod_auth_ldap* 434
  *mod_authn_alias* 469
  *mod_authn_anon* 454
  *mod_authn_core* 406
  *mod_authn_dbd* 461
  *mod_authn_dbm* 426
  *mod_authn_default* 471
  *mod_authn_file* 414
  *mod_authnz_ldap* 434
  *mod_authz_core* 407
  *mod_authz_dbm* 426
  *mod_authz_default* 471
  *mod_authz_groupfile* 414
  *mod_authz_host* 291
  *mod_authz_owner* 471
  *mod_authz_user* 414
  *mod_autoindex* 382
  *mod_cache* 614
  *mod_cache_disk* 621
  *mod_cern_meta* 313
  *mod_cgi* 633
  *mod_cgid* 633
  *mod_charset_lite* 778
  *mod_dav* 795
  *mod_dav_fs* 795
  *mod_dav_lock* 798
  *mod_daytime (eigenes)* 820
  *mod_dbd* 457
  *mod_deflate* 775
  *mod_dir* 285
  *mod_disk_cache* 621
  *mod_dumpio* 533
  *mod_echo* 799
  *mod_env* 648
  *mod_example* 806
  *mod_expires* 309
  *mod_ext_filter* 786
  *mod_fcgid* 632
  *mod_file_cache* 572
  *mod_filter* 770
  *mod_ftpd* 799
  *mod_headers* 303
  *mod_holiday (eigenes)* 808
  *mod_ident* 523
  *mod_imagemap* 393
  *mod_imap* 393
  *mod_include* 761
  *mod_info* 378
  *mod_isapi* 746
  *mod_jk* 730, 734
  *mod_lbmethod_bybusiness* 595
  *mod_lbmethod_byrequests* 595
  *mod_lbmethod_bytraffic* 595
  *mod_lbmethod_heartbeat* 595
  *mod_ldap* 448
  *mod_log_config* 524
  *mod_log_debug* 538
  *mod_log_forensic* 532
  *mod_lua* 740
  *mod_mem_cache* 623
  *mod_mime* 315
  *mod_mime_magic* 321
  *mod_mono* 749

*mod_negotiation* 332
*mod_nw_ssl* 513
*mod_perl* 720
*mod_pop3* 799
*mod_proxy* 580
*mod_proxy_ajp* 580
*mod_proxy_balancer* 580
*mod_proxy_connect* 580
*mod_proxy_express* 610
*mod_proxy_fcgi* 581, 632
*mod_proxy_fdpass* 581
*mod_proxy_ftp* 580
*mod_proxy_html* 603
*mod_proxy_http* 580
*mod_proxy_scgi* 581
*mod_python* 749
*mod_reqtimeout* 259
*mod_rewrite* 352
*mod_ruby* 749
*mod_security* 837
*mod_session* 654
*mod_session_cookie* 659
*mod_session_crypto* 664
*mod_session_dbd* 660
*mod_setenvif* 648
*mod_speling* 376
*mod_status* 379
*mod_substitute* 781
*mod_suexec* 834
*mod_unique_id* 532
*mod_userdir* 372
*mod_usertrack* 534
*mod_vhost_alias* 563
*mod_xml2enc* 780
*Modulstruktur* 810
*nachinstallieren* 192
*selbst programmieren* 806
*statisch* 169
*Typen* 133
*weitere von Drittanbietern* 800
*zum Kompilieren auswählen* 172
Module Magic Number 199
Modulprogrammierung 806
*Anfragen verarbeiten* 815
*Direktiven registrieren* 812
*Direktivenfunktionen* 813
*Grundkonfiguration erzeugen* 811
*Header-Dateien* 808
*Hooks* 807
*Hooks registrieren* 814
*Modulstruktur* 810
*Multiprotokoll-Unterstützung* 820
*request_rec, Struktur* 815
Modulstruktur 810
Mono 749
MPM Siehe Multiprocessing-Module
mpm_winnt 132
Multiprocessing-Module (MPM) 129
*auswählen* 177
*beos* 132
*Direktiven* 266
*dynamisch kompilieren (2.4)* 177
*event* 132
*leader (veraltet)* 891
*mpm_netware* 132
*mpmt_os2* 132
*perchild (veraltet)* 891
*prefork* 131
*threadpool (veraltet)* 891
*worker* 132
Multiprotokoll-Unterstützung 799
*Programmierbeispiel* 820
MultiViews 336
MultiViewsMatch, Direktive 341
Mutex, Direktive 261
MySQL
*Installation, UNIX* 691
*Installation, Windows* 693
*mysqli, PHP-Schnittstelle* 717
*phpMyAdmin* 706
*Testdatenbank* 695
*XAMPP, Komplettpaket* 192
*Zugriff über Java* 737
*Zugriff über Perl* 726
*Zugriff über PHP* 715
mysql-Client 693
mysqli-Schnittstelle 717

## N

namensbasierte virtuelle Hosts 558
Nameserver
*BIND* 36
NameVirtualHost, Direktive 561
NCSA HTTPd 106
Nebenläufigkeit 129
*select( )-Server* 131
Negotiate, HTTP-Header 81, 339

Nessus 826
.NET 749
NetWare
*SSL-Einsatz* 513
nginx, Webserver 115
Non-parsed Header (NPH) 313
NoProxy, Direktive 586
NPH → Non-parsed Header (NPH) 313
nslookup, Dienstprogramm 45
NWSSLTrustedCerts, Direktive 513

## O

ObjectRelationalBridge 109
OCSP 497
Online Certificate Status Protocol (OCSP) 497
open_logs, Hook 808
OpenLDAP
*einrichten* 437
*ldapadd, Kommando* 440
*ldapmodify, Kommando* 440
*ldapsearch, Kommando* 440
*phpLDAPadmin* 441
*slapd.conf* 438
*slappasswd* 438
OpenSSL 475
*Installation, Windows* 478
*Zertifikat erzeugen* 479
openSUSE
*Apache-Konfigurationsdateien* 227
*Apache-Verwaltung* 205
*Runlevel-Editor* 206
Options, Direktive 288
OPTIONS, HTTP-Methode 62
Order, Direktive 291
Organisationseinheiten (LDAP) 435
OSI-Referenzmodell 22

## P

papaya CMS, Apache-Konfiguration für 884
param(), CGI.pm-Methode 670
PassEnv, Umgebungsvariable 649
password_field(), CGI.pm-Methode 685
Passwörter 827
Paulsen, Brian 670
perchild, MPM (veraltet) 891
Performance-Tuning 566
*allgemeine Hinweise* 567
*Datei-Caching* 572
*Direktiven für* 571
<Perl>, Container 727
Perl
*$/, Variable* 89
*CGI.pm, Modul* 668
*CGI::Carp, Modul* 669
*CGI::Pretty, Modul* 670
*DBI, Modul* 726
*eigener Webserver* 86
*HTTP::Date* 89
*in httpd.conf* 727
*Input Record Separator* 89
*Installation, UNIX* 720
*Installation, Windows* 724
*IO::Socket* 88
*Log-Dateien auswerten* 542
*mod_perl* 719
*MySQL-Zugriff* 726
*POSIX* 89
*Zeilenumbrüche* 89
PerlModule, Direktive 722
Peters, Frank 106
Pfadangaben
*in Direktiven* 230
PHP 690
*Cookies* 713
*Datei-Uploads* 711
*Formulardaten auslesen* 710
*Installation, UNIX* 698
*Installation, Windows* 701
*mysqli-Schnittstelle* 717
*mysql-Schnittstelle* 715
*php.ini, Konfigurationsdatei* 703
*Programmiertipps* 708
*Sessions* 713
*vollständige Konfigurationsdatei* 883
*XAMPP, Komplettpaket* 192
*Zend Engine II* 690
php.ini, Konfigurationsdatei 703
phpLDAPadmin 441
phpMyAdmin 706
PidFile, Direktive 267
ping, Dienstprogramm 42
Pioch, Nicolas 106
popup_menu(), CGI.pm-Methode 686
Port-basierte virtuelle Hosts 559

POST, HTTP-Methode 56
*Formularversand* 56
post_config, Hook 808
Postel, Jon 77
Pound, Load-Balancer 577
Pragma, HTTP-Header 81
pre_config, Hook 808
Preforking-Server 130
Pretty Good Privacy (PGP)
*Quellcode-Integritätsprüfung* 151
printenv, SSI-Element 759
Programmierung von Modulen 806
PROPFIND, HTTP-Methode 797
Protocol, Direktive 261
ProtocolDaytime, Direktive 820
ProtocolEcho, Direktive 800
Protokolldateien → Log-Dateien 515
<Proxy>, Container 584
Proxy
*Tomcat einbinden* 734
*vollständige Konfigurationsdatei* 881
ProxyAddHeaders, Direktive 589
Proxy-Authenticate, HTTP-Header 81
Proxy-Authorization, HTTP-Header 82
ProxyBadHeader, Direktive 587
ProxyBlock, Direktive 588
ProxyDomain, Direktive 588
ProxyErrorOverride, Direktive 589
ProxyExpressDBMFile, Direktive 610
ProxyExpressDBMType, Direktive 611
ProxyExpressEnable, Direktive 610
ProxyFtpDirCharset, Direktive 590
ProxyFtpEscapeWildcards, Direktive 590
ProxyFtpListOnWildcard, Direktive 591
ProxyHTMLBufSize 605
ProxyHTMLBufSize, Direktive 605
ProxyHTMLCharsetOut 606
ProxyHTMLCharsetOut, Direktive 606
ProxyHTMLDocType 606
ProxyHTMLDocType, Direktive 606
ProxyHTMLEnable 603
ProxyHTMLEnable, Direktive 603
ProxyHTMLEvents 607
ProxyHTMLEvents, Direktive 607
ProxyHTMLExtended 607
ProxyHTMLExtended, Direktive 607
ProxyHTMLFixups 608
ProxyHTMLFixups, Direktive 608
ProxyHTMLInterp 608
ProxyHTMLInterp, Direktive 608
ProxyHTMLLinks 609
ProxyHTMLLinks, Direktive 609
ProxyHTMLStripComments 609
ProxyHTMLStripComments, Direktive 609
ProxyHTMLURLMap 604
ProxyHTMLURLMap, Direktive 604
ProxyIOBufferSize, Direktive 591
<ProxyMatch>, Container 585
ProxyMaxForwards, Direktive 591
ProxyPass, Direktive 592
ProxyPassInterpolateEnv, Direktive 597
ProxyPassMatch, Direktive 596
ProxyPassReverse, Direktive 597
ProxyPassReverseCookieDomain, Direktive 598
ProxyPassReverseCookiePath, Direktive 599
ProxyPreserveHost, Direktive 600
ProxyReceiveBufferSize, Direktive 600
ProxyRemote, Direktive 586
ProxyRemoteMatch, Direktive 586
ProxyRequests, Direktive 585
Proxy-Server 579
*Aufgaben* 580
*Forward-Proxy* 579
*Grundkonfiguration* 581
*mit Cache* 613
*Reverse-Proxy* 580
*umgehen* 586
ProxySet, Direktive 596
ProxySourceAddress, Direktive 589
ProxyStatus, Direktive 602
ProxyTimeout, Direktive 601
ProxyVia, Direktive 601
ps, Apache-Test mit 195
PTR-Record (DNS) 40
Public-Key-Verschlüsselung 476
PUT, HTTP-Methode 58
Python 749

## Q

Quellcode
*Integritätsprüfung* 151
*Windows* 183
Query-String 55

## R

radio_group(), CGI.pm-Methode 686
Range, HTTP-Header 82
ReadmeName, Direktive 393
ReceiveBufferSize, Direktive 274
Rechtschreibkorrektur
  *in URLs* 376
Redirect → Weiterleitung 345
redirect(), CGI.pm-Methode 682
Redirect, Direktive 348
  *häufige Fehlerquelle* 350
RedirectMatch, Direktive 350
RedirectPermanent, Direktive 352
RedirectTemp, Direktive 352
Referer, HTTP-Header 82
RegExp → Reguläre Ausdrücke (RegExp) 231
Reguläre Ausdrücke (RegExp)
  *als Direktivenwerte* 231
Remote Variant Selection Algorithm (RVSA) 339
RemoveCharset, Direktive 325
RemoveEncoding, Direktive 327
RemoveHandler, Direktive 332
RemoveInputFilter, Direktive 770
RemoveLanguage, Direktive 328
RemoveOutputFilter, Direktive 769
RemoveType, Direktive 319
request_rec, Datenstruktur 815
RequestHeader, Direktive 308
RequestReadTimeout, Direktive 259
Require, Direktive 407
  *all denied* 408
  *all granted* 408
  *env* 408
  *expr* 408
  *group* 407
  *host* 408
  *ip* 408
  *local* 408
  *method* 408
  *not* 408
  *user* 407
  *valid-user* 408
<RequireAll>, Container 410
<RequireAny>, Container 411
<RequireNone>, Container 412
reset(), CGI.pm-Methode 687
Retry-After, HTTP-Header 82
Reverse-Proxy 580
  *Konfigurationsbeispiele* 583
RewriteBase, Direktive 360
Rewrite-Beispiele 368
RewriteCond, Direktive 358
RewriteEngine, Direktive 353
RewriteLock, Direktive 367
RewriteLog, Direktive 537
RewriteLogLeve, Direktive 538
RewriteMap, Direktive 360
RewriteOptions, Direktive 368
RewriteRule, Direktive 354
RFC 25
  *1035 (DNS)* 33
  *1123 (Internet-Host-Anforderungen)* 77
  *1413 (identd)* 523
  *2295 (TCN)* 339
  *2296 (RVSA)* 339
  *2460 (IPv6)* 25
  *2518 (WebDAV)* 795
  *2616 (HTTP/1.1)* 49
  *2822 (Textnachricht, aktuell)* 51
  *768 (UDP)* 32
  *791 (IPv4)* 25
  *822 (Textnachricht)* 51
Ristic, Ivan 837
RLimitCPU, Direktive 832
RLimitMEM, Direktive 832
RLimitNPROC, Direktive 833
Robinson, David 106
Robustheitsprinzip → Robustness Principle 77
Robustness Principle 77
Rossum, Guido van 749
rotatelogs, Hilfsprogramm 216, 540
Round-Robin-DNS 41
Ruby 749
Runlevel 203
Runlevel-Editor (openSUSE) 206
RVSA → Remote Variant Selection Algorithm (RVSA) 339

## S

Satisfy, Direktive 409
Schichtenmodell 22
  *Alltagsbeispiel* 22
Schrauwen, Jorge 188

ScoreBoardFile, Direktive 273
Script, Direktive 644
ScriptAlias, Direktive 633
ScriptAliasMatch, Direktive 635
ScriptInterpreterSource, Direktive 641
ScriptLog, Direktive 639
ScriptLogBuffer, Direktive 639
ScriptLogSize, Direktive 640
ScriptSockSize, Direktive 640
Secure Sockets Layer (SSL) 475
  *Apache-Konfiguration für* 482
  *HTTPS-Verbindungen* 477
  *OpenSSL* 475
  *OpenSSL-Einrichtung* 478
  *Überblick* 476
  *Umgebungsvariablen* 484
  *Zertifikate* 479
  *Zertifizierungspfad* 481
SecureListen, Direktive 514
SeeRequestTail, Direktive 380
self_url(), CGI.pm-Methode 684
send-as-is, Handler 312, 329
SendBufferSize, Direktive 274
Server Name Indication (SNI) 483
Server Side Includes (SSI) 753
  *aktivieren* 754
  *config, Element* 755
  *echo, Element* 756
  *Elemente* 754
  *exec, Element* 757
  *flastmod, Element* 758
  *fsize, Element* 758
  *if/elif/else/endif, Elemente* 759
  *include, Element* 758
  *printenv, Element* 759
  *Programme ausführen* 757
  *set, Element* 759
  *Umgebungsvariablen* 756
  *Variablen definieren* 759
Server, HTTP-Header 83
ServerAdmin, Direktive 280
ServerAlias, Direktive 562
server-info, Handler 329, 378
Server-Kontext 234
ServerLimit, Direktive 274
ServerName, Direktive 280
server-parsed, Handler 329
ServerPath, Direktive 562
ServerRoot, Direktive 257
ServerSignature, Direktive 284
server-status, Handler 329, 379
ServerTokens, Direktive 282
Service → Dienst 209
Servlets → Java Servlets 729
Session, Direktive 656
SessionCookieName, Direktive 659
SessionCookieName2, Direktive 659
SessionCookieRemove, Direktive 660
SessionCryptoCipher, Direktive 665
SessionCryptoDriver, Direktive 664
SessionCryptoPassphrase, Direktive 664
SessionDBDCookieName, Direktive 660
SessionDBDCookieName2, Direktive 660
SessionDBDCookieRemove, Direktive 661
SessionDBDDeleteLabel, Direktive 661, 662
SessionDBDInsertLabel, Direktive 662
SessionDBDPerUser, Direktive 663
SessionDBDSelectLabel, Direktive 663
SessionDBDUpdateLabel, Direktive 663
SessionEnv, Direktive 656
SessionExclude, Direktive 656
SessionHeader, Direktive 657
SessionInclude, Direktive 658
SessionMaxAge, Direktive 658
Session-Tracking
  *mit mod_rewrite* 371
set, SSI-Element 759
Set-Cookie, HTTP-Header 83
SetEnv, Direktive 649
SetEnv, Umgebungsvariable 648
SetEnvIfNoCase, Direktive 650
SetHandler, Direktive 330
SetInputFilter, Direktive 768
SetOutputFilter, Direktive 766
Shebang 634
Sicherheit 825
  *chroot-Umgebung* 826, 829
  *Crackertools* 826
  *Direktiven für* 829
  *Firewalls* 825
  *Intrusion Detection Systems* 825
  *menschliches Versagen* 826
  *Passwörter* 827
  *SuEXEC* 834
  *Überblick* 827
Skolnick, David 106
slapd, OpenLDAP-Daemon 438

slapd.conf, Konfigurationsdatei 438
slappasswd, LDAP-Passwort-Tool 438
SOA-Record (DNS) 40
Softwarelizenz
*GPL* 112
SpamAssassin 110
split_logfile, Hilfsprogramm 216
split-logfile, Hilfsprogramm 541
Spracheinstellungen 327
Sprachkürzel, Tabelle 954
squid, Webcache 567
srm.conf 225
SSI → Server Side Includes (SSI) 753
SSIAccessEnable, Direktive 764
SSIEndTag, Direktive 762
SSIErrorMsg, Direktive 762
SSILegacyExprParser, Direktive 764
SSIStartTag, Direktive 761
SSITimeFormat, Direktive 763
SSIUndefinedEcho, Direktive 763
SSL → Secure Sockets Layer (SSL) 475
SSLCACertificateFile, Direktive 486
SSLCACertificatePath, Direktive 486
SSLCADNRequestFile, Direktive 487
SSLCADNRequestPath, Direktive 487
SSLCARevocationCheck, Direktive 488
SSLCARevocationFile, Direktive 488
SSLCARevocationPath, Direktive 489
SSLCertificateChainFile, Direktive 489
SSLCertificateFile, Direktive 489
SSLCertificateKeyFile, Direktive 490
SSLCipherSuite, Direktive 490
SSLCryptoDevice, Direktive 493
SSLEngine, Direktive 494
SSLFIPS, Direktive 495
SSLHonorCipherOrder, Direktive 495
SSLMutex, Direktive 495
SSLOCSPDefaultResponder, Direktive 497
SSLOCSPEnable, Direktive 497
SSLOCSPOverrideResponder, Direktive 497
SSLOCSPResponderTimeout, Direktive 498
SSLOCSPResponseMaxAge, Direktive 498
SSLOCSPResponseTimeSkew, Direktive 498
SSLOptions, Direktive 502
SSLPassPhraseDialog, Direktive 503
SSLProtocol, Direktive 504
SSLProxyCACertificateFile, Direktive 510
SSLProxyCACertificatePath, Direktive 510
SSLProxyCARevocationFile, Direktive 511
SSLProxyCARevocationPath, Direktive 511
SSLProxyCipherSuite, Direktive 511
SSLProxyEngine, Direktive 512
SSLProxyMachineCertificateFile, Direktive 512
SSLProxyMachineCertificatePath, Direktive 512
SSLProxyProtocol, Direktive 512
SSLProxyVerify, Direktive 513
SSLProxyVerifyDepth, Direktive 513
SSLRandomSeed, Direktive 505
SSLRequire, Direktive 506
SSLRequireSSL, Direktive 507
SSLSessionCache, Direktive 508
SSLSessionCacheTimeout, Direktive 508
SSLStaplingCache, Direktive 499
SSLStaplingErrorCacheTimeout, Direktive 499
SSLStaplingFakeTryLater, Direktive 500
SSLStaplingForceURL, Direktive 500
SSLStaplingResponderTimeout, Direktive 500
SSLStaplingResponseMaxAge, Direktive 501
SSLStaplingResponseTimeSkew, Direktive 501
SSLStaplingReturnResponderErrors, Direktive 501
SSLStaplingStandardCacheTimeout, Direktive 502
SSLUserName, Direktive 508
SSLUseStapling, Direktive 499
SSLVerifyClient, Direktive 509
SSLVerifyOpen, Direktive 510
start_form(), CGI.pm-Methode 685
start_html(), CGI.pm-Methode 673, 683
StartServers, Direktive 275
StartThreads, Direktive 275
Statuscodes (HTTP) 62
*100 Continue* 63
*101 Switching Protocols* 63
*1xx (Informationen)* 63
*200 OK* 64
*201 Created* 64
*2xx (Erfolg)* 63
*301 Moved Permanently* 65
*302 Found* 65
*303 See Other* 65

*304 Not Modified* 65
*307 Temporary Redirect* 65
*3xx (Umleitung)* 64
*401 Unauthorized* 67
*403 Forbidden* 67
*404 Not Found* 67
*4xx (Client-Fehler)* 65
*500 Internal Server Error* 68
*5xx (Server-Fehler)* 67
*Typen* 62
Statusinformationen 378
Stein, Lincoln D. 668
strftime( ) 526
Struts 109
submit(), CGI.pm-Methode 687
Substitute, Direktive 782
SuEXEC 834
*Compiler-Optionen für* 834
*Sicherheitsüberprüfungen* 835
Suexec, Direktive 836
suexec, Hilfsprogramm 216
suexec, Modul 834
SuexecUserGroup, Direktive 837
symmetrische Verschlüsselung 475
System V Init 203

## T

TCL, ASF-Projekt 110
TCN → Transparente Content-Negotiation 339
TCN, HTTP-Header 340
TCP/IP
*Diagnose und Fehlersuche* 42
*HTTP* 49
*Kommunikationsverfahren* 23
*netstat* 44
*traceroute* 43
*Transportprotokolle* 30
TE, HTTP-Header 84
telnet, Dienstprogramm 46
Terbush, Randy 106
textfield(), CGI.pm-Methode 685
Thau, Robert S. 106
Threading-Server 130
ThreadLimit, Direktive 276
threadpool, MPM (veraltet) 891
ThreadsDirektive 276
ThreadStackSize, Direktive 278
TimeOut, Direktive 257
TLS → Transport Layer Security (TLS) 475
Tomcat 109, 115, 729
*Installation, UNIX* 730
*Installation, Windows* 733
*Konfiguration* 732
*mod_jk* 730
*mod_jk2* 734
*Programmierbeispiele* 735
*Proxy-Anbindung* 734
Top-Level-Domains, Tabelle 966
TRACE, HTTP-Methode 60
TraceEnable, Direktive 265
Trailer, HTTP-Header 84
Transfer-Encoding, HTTP-Header 84
TransferLog, Direktive 531
translate_name, Hook 808
Transmission Control Protocol (TCP) 31
Transparente Content-Negotiation 339
Transport Layer Security (TLS) 475
tripwire 826
type_checker, Hook 808
type-map, Handler 330, 334
Type-Maps 333
*Syntax* 335
TypesConfig, Direktive 320

## U

Umgebungsvariablen 420, 645
*in verschiedenen Sprachen* 645
*mit Apache setzen* 648
*PassEnv* 649
*SetEnv* 648
*SSI* 756
*SSL* 484
uname( ) 89
UnDefine, Direktive 249
UNIX
*Apache beenden* 202
*Apache neu starten* 202
*Apache starten* 202
*Apache steuern* 195
*apachectl, Steuerskript* 201
*BSD-Startstkripte* 205
*mod_perl-Installation* 720
*MySQL-Installation* 691
*Perl-Installation* 720
*PHP-Installation* 698

*Runlevel* 203
*Tomcat-Installation* 730
*User- und Group-ID für Apache* 267
UNIX System V Init 203
UnsetEnv, Direktive 649
Upgrade, HTTP-Header 84
url(), CGI.pm-Methode 684
UseCanonicalName, Direktive 281
UseCanonicalPhysicalPort, Direktive 282
User Datagram Protocol (UDP) 32
User, Direktive 267
User-Agent, HTTP-Header 84
UserDir, Direktive 373
Usertracking → Session-Tracking 371

## V

van Rossum, Guido 749
Variablen
*AuthDigestEnableQueryStringHack* 420
varnish, Webcache 567
Vary, HTTP-Header 85
Verfallsdatum (Dokumente) 309
Verschlüsselung
*asymmetrische* 476
*Einwegverschlüsselung* 476
*Grundbegriffe* 475
*symmetrische* 475
Verzeichniscontainer 235
Verzeichnisdienste 434
Verzeichnisse
*cgi-bin* 633
*für CGI* 633
Via, HTTP-Header 85
VirtualDocumentRoot, Direktive 564
VirtualDocumentRootIP, Direktive 565
<VirtualHost>, Container 234
VirtualScriptAlias, Direktive 565
VirtualScriptAliasIP, Direktive 565
Virtuelle Hosts 555
*als httpd.conf-Kontext* 234
*Direktiven für* 560
*IP-basierte* 557
*Konfigurationsbeispiele* 556
*namensbasierte* 558
*Port-basierte* 559
VLSM 27

## W

Warning, HTTP-Header 85
Web Services
*ASF-Projekt* 110
Webalizer 553
Web-Cache 611
*Konfigurationsbeispiele* 612
Webcache 567
WebDAV 795
*Konfiguration* 796
*PROPFIND-Methode* 797
Web-Formulare
*GET-Versand* 54
Webformulare
*manuell in CGIs einlesen* 667
*MIME-Types* 56
*mit CGI.pm* 677
*mit CGI.pm, Referenz* 684
*mit PHP auslesen* 710
*POST-Versand* 56
Webserver
*Apache-Konkurrenz* 113
*CERN httpd* 106
*Microsoft IIS* 113
Website
*einfache zum Test* 217
Weiterleitung 345
Wilson, Andrew 106
Win32DisableAcceptEx, Direktive 279
Windows
*Apache manuell beenden* 207
*Apache manuell neu starten* 207
*Apache manuell starten* 207
*Apache steuern* 206
*Apache-Monitor* 213
*ASP.NET mit Apache* 750
*CGI-Zuordnung* 634
*Deinstallation* 192
*httpd.exe, Binary* 206
*ISAPI* 746
*mod_perl-Installation* 724
*MySQL-Installation* 693
*OpenSSL-Installation* 478
*Perl-Installation* 724
*Pfadangaben in httpd.conf* 230
*PHP-Installation* 701
*Tomcat-Installation* 733
Wusage 553
WWW-Authenticate, HTTP-Header 86

## X

XAMPP 192
XBitHack, Direktive 763
XML
*ASF-Projekt* 111
xml2EncAlias, Direktive 780
xml2EncDefault, Direktive 781
xml2StartParse, Direktive 781

## Y

YaST
*Apache-Verwaltung* 205

## Z

Zeichensätze
*Einstellung in Apache* 323
*Konvertierung* 778
Zeichensätze, Tabelle 958
Zeilenumbrüche
*in verschiedenen Systemen* 89
Zend Engine II 690
Zertifikat (OpenSSL)
*erstellen* 479
*Zertifizierungspfad* 481
Zeus, Webserver 115
Zugriffsbeschränkung
*nach IP-Adresse* 291
Zugriffskontrolle 400