X.media.press

Thomas Walter

# Kompendium der Web-Programmierung

## Dynamische Web-Sites

Mit 510 Abbildungen und 22 Tabellen

Springer

Prof. Dr. Thomas Walter
Programmierung und Betrieb von Web-Sites
Fachbereich Informatik und Mikrosystemtechnik
Fachhochschule Kaiserslautern
Amerikastraße 1
66482 Zweibrücken
thomas.walter@fh-kl.de
www.webkompendium.de

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.

ISSN 1439-3107
ISBN 978-3-540-33134-6 Springer Berlin Heidelberg New York

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland vom 9. September 1965 in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.

Springer ist ein Unternehmen von Springer Science+Business Media
springer.de

© Springer-Verlag Berlin Heidelberg 2008

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften. Text und Abbildungen wurden mit größter Sorgfalt erarbeitet. Verlag und Autor können jedoch für eventuell verbliebene fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Satz: Druckfertige Daten des Autors
Herstellung: LE-TEX, Jelonek, Schmidt & Vöckler GbR, Leipzig
Umschlaggestaltung: KünkelLopka Werbeagentur, Heidelberg
Gedruckt auf säurefreiem Papier 33/3180 YL – 5 4 3 2 1 0

# Vorwort

*If you want to know how to run a server,*
*or how to edit HTML,*
*check the W3C web or your local bookstore.*

*Tim Berners-Lee*
*(von seiner Web-Site beim W3C)*

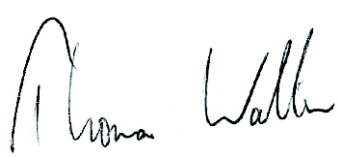
*Programmierung für das Internet* – kaum ein anderes Thema in der Informatik hat aktuell eine größere Bedeutung. Deshalb überrascht es nicht, wenn die Zahl der technischen Möglichkeiten in diesem Bereich besonders groß und der Fortschritt besonders rasch sind. Das „Kompendium der Web-Programmierung“ möchte einen Überblick über die wichtigsten, aktuellen Lösungsansätze geben. Dabei soll diese Darstellung jeweils so tief gehen, dass der Leser diese Techniken sinnvoll beurteilen und aktiv einsetzen kann.

Das „Kompendium der Web-Programmierung“ möchte den Leser somit in die Lage versetzen, an einer der größten Herausforderungen unserer Zeit teilzuhaben: die moderne Wissensgesellschaft aktiv mitzugestalten.

*Grau, teurer Freund, ist alle Theorie,*
*Und grün des Lebens goldner Baum.*

In diesem Sinne wünsche ich allen Lesern eine erfreuliche, fruchtbare und vor allem aktive Auseinandersetzung mit dem „Kompendium der Web-Programmierung“.

Zweibrücken, im Sommer 2007



# Inhaltsverzeichnis

**Hinweise zum Gebrauch des Buches** . . . . . XIII

**Teil I Grundlagen der Web-Programmierung**

**1 Entwicklung der Web-Programmierung** . . . . . 3
1.1 Der Weg zum World Wide Web . . . . . 3
1.2 Komponenten der frühen Technik . . . . . 5
1.3 Clientseitige Web-Programmierung . . . . . 6
1.4 Serverseitige Web-Programmierung . . . . . 7
1.5 Sprachen für die Web-Programmierung . . . . . 8
1.6 Technische Grundlage: die Internetprotokolle . . . . . 9
1.7 Sicherheit . . . . . 24

**2 Darstellung im Web – Auszeichnungssprachen** . . . . . 25
2.1 Auszeichnungssprachen und die Trennung von Inhalt und Formatierung . . . . . 25
2.2 SGML . . . . . 26
2.3 HTML . . . . . 26
2.4 XML . . . . . 32
2.5 Cascading Stylesheets – CSS: Format fürs Web . . . . . 38
2.6 Gestaltung barrierefreier Webseiten . . . . . 42

**3 Rechnersysteme für Webangebote** . . . . . 45
3.1 Die Hardware . . . . . 45
3.2 Betriebssysteme im Web . . . . . 46
3.3 Datenbankserver . . . . . 48
3.4 Alles aus einer Hand: XAMPP . . . . . 51

**4 Softwarearchitektur für das Internet** . . . . . 53
4.1 Projektmanagement für das Web . . . . . 53
4.2 Programmierparadigmen für das Web . . . . . 53
4.3 Das Entwurfsmuster Model-View-Controller . . . . . 57
4.4 Entwicklungsumgebungen . . . . . 59
4.5 Dokumentation . . . . . 59

**5 Der Webclient – Browser** . . . . . 63
5.1 Aufgaben und Arbeitsweise des Webclients . . . . . 63
5.2 Aktuelle Browser . . . . . 63
5.3 Browser-Tests . . . . . 68
5.4 Die Browser in diesem Buch . . . . . 70

**6 Der Webserver** . . . . . 71
6.1 Aufgaben und Arbeitsweise . . . . . 71
6.2 Ein einfacher Webserver in Java . . . . . 71
6.3 Der Apache Webserver . . . . . 81

**7 Das Beispiel** . . . . . 93
7.1 Die Literatur-Datenbanktabellen . . . . . 93
7.2 Reale Beispiele: Dublin Core . . . . . 97
7.3 Anwendungsfälle . . . . . 98

**8 Wichtige und nützliche Werkzeuge für die Web-Entwicklung** . . 101
8.1 Die Entwicklungsumgebung Eclipse . . . . . 101
8.2 Webeditoren . . . . . 104
8.3 Firebug . . . . . 107
8.4 Server-Logs . . . . . 108
8.5 Datenbank-Tools . . . . . 109

**Teil II Klassische Web-Programmierung: CGI, PHP und moderne Scriptsprachen**

**9 CGI: das Common Gateway Interface** . . . . . 113
9.1 Dynamik im Web: ein Prozess auf dem Webserver . . . . . 113
9.2 Der CGI-Mechanismus . . . . . 113
9.3 Kommunikation zwischen CGI und Webserver . . . . . 114
9.4 Beispiele . . . . . 115

**10 Perl** . . . . . 125
10.1 Die Scriptsprache Perl . . . . . 125
10.2 Quellen und Installation . . . . . 126
10.3 Grundlegende Syntax . . . . . 126
10.4 Einfachste CGIs mit Perl . . . . . 154
10.5 Perl erweitern: Module und mehr . . . . . 160
10.6 Das Perl-Modul CGI . . . . . 165
10.7 Das Perl-Modul DBI . . . . . 173
10.8 Das Perl-Modul LWP . . . . . 185
10.9 Zusammenfassung . . . . . 187

**11 PHP** . . . . . 189
11.1 Die Scriptsprache PHP . . . . . 189
11.2 Installation als Apache-Modul . . . . . 191
11.3 Grundlegende Syntax . . . . . 195
11.4 Mehr PHP . . . . . 220
11.5 Datenbankzugriff mit PHP . . . . . 229
11.6 Strukturierte Softwareentwicklung mit PHP . . . . . 243
11.7 Erweiterungen von PHP: PEAR und PECL . . . . . 255
11.8 Universeller Datenbankzugriff: PHP Data Objects PDO . . . . . 255

**12 Python** . . . . . 257
12.1 Die Scriptsprache Python . . . . . 257
12.2 Installation und Entwicklungsumgebungen . . . . . 258
12.3 Grundlegende Syntax der Scriptsprache Python . . . . . 260
12.4 Python im Web . . . . . 277
12.5 Python und Datenbanken . . . . . 288
12.6 Python und Java: Jython . . . . . 293
12.7 GUI-Programmierung mit Python . . . . . 295
12.8 Ausblick . . . . . 295

**13 Ruby** . . . . . 297
13.1 Die Scriptsprache Ruby . . . . . 297
13.2 Installation und Entwicklungsumgebung . . . . . 299
13.3 Grundlegende Syntax . . . . . 300
13.4 Objektorientierung mit Ruby . . . . . 312
13.5 Ruby im Web . . . . . 321
13.6 Datenbankzugriff mit Ruby . . . . . 330
13.7 Ein weiterer Ansatz: JRuby . . . . . 335

**14 Server Side Includes** . . . . . 337
14.1 Die einfache Alternative: SSI . . . . . 337
14.2 Beispiele: Was kann SSI . . . . . 337
14.3 Voraussetzungen für SSI und Konfiguration des Apache . . . . . 337
14.4 Syntax und Beispiele . . . . . 339
14.5 Beispiele . . . . . 341

**Teil III Clientseitige Programmierung**

**15 JavaScript** . . . . . 347
15.1 Dynamisches HTML: DHTML . . . . . 347
15.2 Dynamische Webseiten mit JavaScript . . . . . 347
15.3 Grundlegende Syntax von JavaScript . . . . . 356
15.4 Objektorientierung in JavaScript . . . . . 364
15.5 JavaScript und HTML: DOM . . . . . 368
15.6 Event-Behandlung mit JavaScript . . . . . 380
15.7 Übersicht über die Event-Handles . . . . . 382
15.8 Komplexere Strukturen: JSON . . . . . 383
15.9 Zum Einsatz von JavaScript für die Web-Programmierung . . . 384

**16 Ajax** . . . . . 387
16.1 Beispiele für Ajax . . . . . 388
16.2 Technische Grundlage für Ajax . . . . . 389
16.3 Entwicklungsumgebungen für Ajax . . . . . 389
16.4 Ablauf einer Ajax-Anfrage . . . . . 390
16.5 Beispiele für Ajax . . . . . 391
16.6 Das XMLHttpRequest-Objekt . . . . . 399
16.7 Zusammenfassung: Vorteile und Probleme von Ajax . . . . . 400

**17 Adobe Flash** . . . . . 401
17.1 Das Prinzip von Flash . . . . . 401
17.2 ActionScript . . . . . 406
17.3 Probleme von Flash . . . . . 406
17.4 Alternativen zu Flash . . . . . 407

**18 Gescheiterte Technik: das Applet** . . . . . 409
18.1 Idee des Applets . . . . . 409
18.2 Einbinden eines Applets . . . . . 411
18.3 Applet-Klassen in Java . . . . . 412
18.4 Probleme der Applets . . . . . 414

**Teil IV Fortgeschrittene Web-Programmierung**

**19 Von CGI zu fastCGI** . . . . . 417
19.1 Nachteile von CGI . . . . . 417
19.2 Die Ideen von fastCGI . . . . . 417
19.3 Das fastCGI-Protokoll . . . . . 419
19.4 fastCGI Developer's Kit . . . . . 419
19.5 Das fastCGI-Servermodul . . . . . 420
19.6 fastCGI-Anwendungen programmieren . . . . . 421
19.7 Leistungen, Grenzen und Ausblick . . . . . 424

**20 Das PHP-Framework PEAR** . . . . . 427
20.1 Struktur von PEAR . . . . . 427
20.2 Installation von PEAR . . . . . 427
20.3 Das Dienstprogramm PEAR . . . . . 428
20.4 Die PEAR-Pakete . . . . . 429
20.5 Das PEAR-Paket DB . . . . . 430

**21 Template-Engines: Smarty & Co** . . . . . 435
21.1 Templates . . . . . 435
21.2 Die Template-Engine Smarty . . . . . 436
21.3 Zusammenfassung: Template-Engines und Design Patterns . . . 446

**22 Das Python-Framework django** . . . . . 447
22.1 Komponenten und Betrieb . . . . . 447
22.2 Installation von django . . . . . 448
22.3 Ein Beispielprojekt mit django . . . . . 450
22.4 Das Python-Framework ZOPE . . . . . 461
22.5 Zusammenfassung . . . . . 461

**23 Das Ruby-Framework Ruby on Rails** . . . . . 463
23.1 Das Prinzip von Rails . . . . . 464
23.2 Scaffolding . . . . . 464
23.3 Webserver für Rails . . . . . 464
23.4 Unterstützte Datenbankmanagementsysteme . . . . . 465
23.5 Rails-Module und das MVC-Pattern . . . . . 465
23.6 Installation von Rails . . . . . 466
23.7 Entwicklungsumgebung für Rails . . . . . 468
23.8 Eine Beispielanwendung mit Rails . . . . . 470
23.9 Zusammenfassung . . . . . 491

**24 Serverseitiges Java** . . . . . 493
24.1 J2EE . . . . . 493
24.2 Java Servlets . . . . . 494
24.3 Datenbankanbindung mit Java . . . . . 506
24.4 JSP: JavaServer Pages . . . . . 511
24.5 Einige weitere J2EE-Begriffe . . . . . 515

## Teil V Ergänzungen zur Web-Programmierung

**25 Was sind Cookies, warum braucht man sie und warum sie keiner will** . . . . . 519
25.1 Was sind Cookies? . . . . . 519
25.2 Cookies im Browser kontrollieren . . . . . 519
25.3 Arbeitsweise von Cookies . . . . . 522
25.4 Die Datenstruktur der Cookies . . . . . 523
25.5 Cookies und Sicherheit . . . . . 523
25.6 Cookies in PHP . . . . . 524
25.7 Das Beispiel . . . . . 524
25.8 Cookies in den anderen Sprachen . . . . . 526

**26 Sessionmanagement** . . . . . 529
26.1 Vom Cookie zur Session . . . . . 529
26.2 Sessionmanagement in PHP . . . . . 530

**27 Media-Formate** . . . . . 543
27.1 Der MIME-Typ . . . . . 543
27.2 Die verschiedenen MIME-Typen . . . . . 543
27.3 Grafik-Formate: Bilddateien im Web . . . . . 544
27.4 Das pdf-Format . . . . . 548

**28 Content Management Systeme: TYPO3** . . . . . 551
28.1 Content Management Systeme . . . . . 551
28.2 Das CMS TYPO3 . . . . . 554

**29 Performance und Testverfahren für Web-Applikationen** . . . . . 569
29.1 Bedeutung der Testverfahren . . . . . 569
29.2 Performance mit JMeter . . . . . 569
29.3 Typisches Ergebnis und Performance-Optimierung . . . . . 574

**30 Sicherheit im Web** . . . . . 575
30.1 Die Netzwerkstruktur . . . . . 576
30.2 Die notwendige Konfiguration . . . . . 577
30.3 Die Apache-Kennung . . . . . 579
30.4 Der Highend-Angriff: DOS und DDOS . . . . . 580
30.5 Nicht zu viel verraten . . . . . 580
30.6 Selbstanalyse . . . . . 581
30.7 Sicherheit auf dem Client . . . . . 581
30.8 Lokale Firewalls . . . . . 582
30.9 Zusammenfassung Sicherheit . . . . . 584

**31 Quo vadis? Web 2.0 und die weitere Entwicklung** . . . . . 587
31.1 Die Bedeutung der einzelnen Techniken . . . . . 587
31.2 Web 2.0 . . . . . 588

**Persönliche Worte** . . . . . 591

**A Internetlinks** . . . . . 593
A.1 Zu Kapitel 1 . . . . . 593
A.2 Zu Kapitel 2 . . . . . 593
A.3 Zu Kapitel 3 . . . . . 594
A.4 Zu Kapitel 4 . . . . . 594
A.5 Zu Kapitel 5 . . . . . 594
A.6 Zu Kapitel 6 . . . . . 594
A.7 Zu Kapitel 8 . . . . . 594
A.8 Zu Kapitel 9 . . . . . 595
A.9 Zu Kapitel 10 . . . . . 595
A.10 Zu Kapitel 11 . . . . . 595
A.11 Zu Kapitel 12 . . . . . 595
A.12 Zu Kapitel 13 . . . . . 595
A.13 Zu Kapitel 15 . . . . . 596
A.14 Zu Kapitel 16 . . . . . 596
A.15 Zu Kapitel 17 . . . . . 596
A.16 Zu Kapitel 19 . . . . . 596
A.17 Zu Kapitel 20 . . . . . 596
A.18 Zu Kapitel 21 . . . . . 596
A.19 Zu Kapitel 22 . . . . . 596
A.20 Zu Kapitel 23 . . . . . 596
A.21 Zu Kapitel 24 . . . . . 597
A.22 Zu Kapitel 27 . . . . . 597
A.23 Zu Kapitel 28 . . . . . 597
A.24 Zu Kapitel 29 . . . . . 597
A.25 Zu Kapitel 30 . . . . . 597

**B Abkürzungen** . . . . . 599

**Literatur** . . . . . 601

**Personenverzeichnis** . . . . . 603

**Sachverzeichnis** . . . . . 605

# Hinweise zum Gebrauch des Buches

Dieses Buch gibt Ihnen einen Überblick über die Programmierung, welche heute für Web-Applikationen verwendet wird. Dabei soll dieser Überblick durchaus soweit gehen, dass die jeweils vorgestellten Techniken aktiv angewendet werden können.

Auf der anderen Seite kann zu fast jeder der hier behandelten Techniken ein Buch im Unfang des „Kompendium der Web-Programmierung“ geschrieben werden – die behandelten Techniken können also nur komprimiert vorgestellt werden.

Dieses Buch basiert auf einer langen Erfahrung in der Anwendung dieser Techniken. Deshalb sind an vielen Stellen wichtige Hinweise aus der Praxis integriert. Darüber hinaus wurde jeweils versucht, dem Leser auch eine praktische Hilfe mit nützlichen Hinweisen, einem effizienten Index und vielen Verweisen innerhalb des Buches zu weiterführender Literatur und zu Ressourcen im Internet zu geben. Im Anhang sind Internetlinks sowie eine Übersicht über alle relevanten Abkürzungen enthalten.

## Gliederung des Inhalts

Das „Kompendium der Web-Programmierung“ gliedert sich strukturell in fünf Teile:

- Teil I besteht aus den Grundlagen der Programmierung für das Internet; hierzu gehört neben der Entwicklung auch die Darstellung der Internet-Protokollfamilie, die Standards HTML, XML und CSS, Betriebssysteme für Webserver, Erläuterung zu den verschiedenen Browsern und eine genauere Betrachtung des Webservers selbst.
- Der Teil II widmet sich der klassischen serverseitigen Web-Programmierung; hier werden die CGI-Programmierung sowie PHP behandelt. Für CGI werden die Sprachen Perl, Python und Ruby vorgestellt.
- Teil III ist der clientseitigen Programmierung gewidmet. Hierzu zählen insbesondere JavaScript und das darauf aufbauende Ajax; weitere Themen sind die Konzepte von Flash und Applets.
- Im Teil IV, der fortgeschrittenen Web-Programmierung, wird einleitend der Übergang von CGI zu fastCGI erklärt. Wichtige Frameworks wie Ruby on Rails und Template-Engines wie Smarty gehören ebenso dazu wie der serverseitige Einsatz von Java im J2EE-Framework insbesondere mit Servlets.

- Teil V sind die Ergänzungen zur Web-Programmierung; hierzu zählen insbesondere wichtige Bemerkungen zur Sicherheit im Web, die Begriffe Cookie und Session, Content Management Systeme und Testverfahren für Web-Applikationen.

Alle Bereiche verwenden konsequent das gleiche Beispiel, eine kleine Literatur-Datenbank, die in Kapitel 7 vorgestellt wird.

## Empfohlene Literatur

Das „Kompendium der Web-Programmierung“ spricht viele aktuelle Themen an, kann aber natürlich nicht alle vollständig behandeln, weshalb in den jeweiligen Abschnitten Hinweise zu weiterführender Literatur enthalten sind. Einige Werke sind besonders hervorzuheben.

Grundlegendes zur Thematik findet sich in [MS04], Einleitendes zur Programmierung mit Scriptsprachen in [Deh01]; allgemeine Betrachtungen zur Web-Performance sind in [Kil02] zu finden.

## Tipps und wichtige Hinweise im Buch

Nützliche – und natürlich „(be-)merkenswerte“ – Hinweise sind in diesem Buch zusätzlich gekennzeichnet:

**Diese Icons sollen Ihnen das Lesen erleichtern.**

Neben diesem allgemeinen Hinweis gibt es auch noch besondere:

Die Softwareentwicklung wird in den letzten Jahren besonders durch die Programmiersprache Java geprägt, wofür es viele Gründe gibt. Hier kann Java nicht ausführlich behandelt werden, viele Leser werden aber über gute Java-Kenntnisse verfügen. Besondere Parallelen der hier vorgestellten Sprachen verdienen deshalb eine Hervorhebung:

**So wird auf eine Parallele zu Java hingewiesen.**

Gefahren für den Entwickler werden als „Warn-Hinweis“ kenntlich gemacht.

**So sind besondere „Warn-Hinweise“ für den Entwickler gekennzeichnet.**

Darüber hinaus gibt es Hinweise, welche sich auf spezielle Betriebssysteme beziehen:

**So sehen Hinweise speziell für Unix und Linux aus.**

**Und dies sind Hinweise für Microsft Windows, teilweise noch spezialisiert für XP und Vista.**

## Das Webangebot zum Buch

Zum „Kompendium der Web-Programmierung“ gibt es natürlich auch eine Webseite. Diese ist unter der Adresse

`http://www.webkompendium.de`

zu finden und bietet den Lesern fortlaufend ergänzte und aktuelle Informationen. Dazu gehören unter anderem folgende Angebote:

- die vollständigen und getesteten Sourcecodes zu allen Beispielen des Buches; die Scripte sind dabei unter dem im Buch angegebenen Namen, gruppiert nach der jeweiligen Technik, zu finden;
- eine erweiterte, aktuelle und verlinkte Sammlung von Internetlinks in Ergänzung zu Anhang A;
- ein ausführlicheres Inhaltsverzeichnis zum Buch als pdf-Download;
- eine Übersicht über sich sicherlich einstellende Errata;
- weitere Hinweise und Ergänzungen.

**Nur durch eine aktive Auseinandersetzung mit der Materie wird ein richtiges Verständnis möglich. Das Webangebot möchte Sie dazu besonders anregen!**

Der Hinweis web bedeutet, dass im Buchanhang (Anhang A) oder direkt auf der Webseite `http://www.webkompendium.de` weitere Links zu Informationen im Internet zum jeweiligen Thema aufgeführt sind.

## Hinweise zur verwendeten Nomenklatur

Sourcecode ist typografisch jeweils abgesetzt, im Text in der Art `public static void main(String[] args)`, größere Codefragmente sind optisch abgesetzt:

```
/**
 *  Beispiel
 */
public class HelloWebKompendium{
   public static void main(String[] args) {
      System.out.println("Hello \"Kompendium der Web-Programmierung\"");
   }
}  // class
```

Die sehr wichtigen Kommentare (mehr dazu in 4.5) sind in der gedruckten Fassung teilweise reduziert, um den Umfang des Buches nicht unnötig groß werden zu lassen; die im Web bereitgehaltenen Sourcecodes sind ausführlicher.

Softwarebezogene Buchteile beziehen sich häufig auf die entsprechenden Menüs dieser Software; dies wird durch folgende Nomenklatur beschrieben: Datei|Speichern speichert unter dem Menüpunkt Datei und dort unter dem Unterpunkt Speichern eine Datei.

## Hinweise zum praktischen Gebrauch

Natürlich wünscht sich der Autor, dass der Leser dieses Buch vollständig liest – dafür wurde es ja geschrieben. Dennoch werden für viele nur einzelne Kapitel und Abschnitte von besonderem Interesse sein. Deshalb wurde versucht, die Materie so zu gestalten, dass alle Teile einzeln verständlich sind. Dafür wurden viele Verweise auf andere Abschnitte integriert, in denen zu Unterpunkten dann Genaueres zu finden ist.

Dem Autor war es dabei ein besonderes Anliegen, mit dem „Kompendium der Web-Programmierung“ auch eine praktische Hilfe für den Entwickler zu geben. Deshalb sind wie erwähnt viele nützliche Internetlinks im Buch sowie auf der Webseite angegeben. Wichtig ist hier auch ein effizienter Index zum Buch. Im „Kompendium der Web-Programmierung“ sind zahlreiche für die Praxis nützliche Tabellen enthalten, die auch über den Index schnell zu finden sind. In diesem technisch sich so rasch entwickelnden Bereich haben sich zunehmend mehr Abkürzungen etabliert; um hier eine Praxishilfe zu bieten, verfügt der Anhang B über ein Abkürzungsverzeichnis mit Bezug zu den relevanten Textpassagen im Buch.

Eine interessante inhaltliche Ergänzung zur semantischen Aufbereitung professioneller Webangebote findet der Leser insbesondere in [Arn06].

# Teil I

## Grundlagen der Web-Programmierung

# 1

# Entwicklung der Web-Programmierung

## 1.1 Der Weg zum World Wide Web

Die Anfänge „des Internets“ werden üblicherweise im Jahr 1969 angesiedelt und sind damit ziemlich genau 20 Jahre älter als die *Entstehung des World Wide Web*.

Theoretische Vorarbeiten für das erste Netz lieferten Paul Baran und Donald Watts Davies anfangs der sechziger Jahre des letzten Jahrhunderts mit ihrer Idee eines paketbasierten, selbstorganisierten Netzwerkes; Hintergrund hierfür scheint die Bestrebung nach ausfallsicheren Kommunikationsnetzen zum Höhepunkt des Kalten Krieges gewesen zu sein. 1969 wurden in den USA vier Universitäten durch ein Weitverkehrsnetz (Wide Area Network, WAN) verbunden. 1972 bildeten bereits 40 Universitäten das ARPANET, 1984 wurde in Deutschland der „Verein zur Förderung eines Deutschen Forschungsnetzes e. V.“ (DFN) gegründet [web], der bis heute die deutschen Hochschulen und Forschungseinrichtungen mit einem Hochleistungsnetz versorgt (aktuell das X-WiN mit einer Anschlusskapazität bis zu 10 Gigabit/s im Backbone).

Das Internet beherbergt viele Dienste wie telnet oder FTP; der heute weitaus wichtigste aber ist „das Web“ oder auch „das WWW“. Seine zentralen Vorteile sind

- die Möglichkeit, durch Hyperlinks auf andere Dokumente zu verweisen;
- die Möglichkeit des Einsatzes multimedialer Komponenten, angefangen von frei zu formatierenden Texten über Grafiken bis hin zu Ton- und Bewegtbilddaten.

**Abbildung 1.1:** Verein zur Förderung eines Deutschen Forschungsnetzes (DFN)

Die Idee für das WWW geht auf den 1955 in London geborenen englischen Physiker Sir Timothy J. Berners-Lee zurück, der 1989 an der europäischen Großforschungseinrichtung CERN den damaligen Informationsdienst Gopher ablösen wollte; die erste „Webadresse“ überhaupt war `http://info.cern.ch`. Sein Konzept basiert auf dem Zusammenspiel mehrerer Komponenten:

- ein Webserver;
- ein Webclient;
- das Vermittlungsprotokoll HTTP;
- eine Formatierungssprache HTML.

Im Detail führt er aus:

*Abstract:*

*HyperText is a way to link and access information of various kinds as a web of nodes in which the user can browse at will. Potentially, HyperText provides a single user-interface to many large classes of stored information such as reports, notes, data-bases, computer documentation and on-line systems help. We propose the implementation of a simple scheme to incorporate several different servers of machine-stored information already available at CERN, including an analysis of the requirements for information access needs by experiments.*

*(Tim Berners-Lee, 12. November 1990*

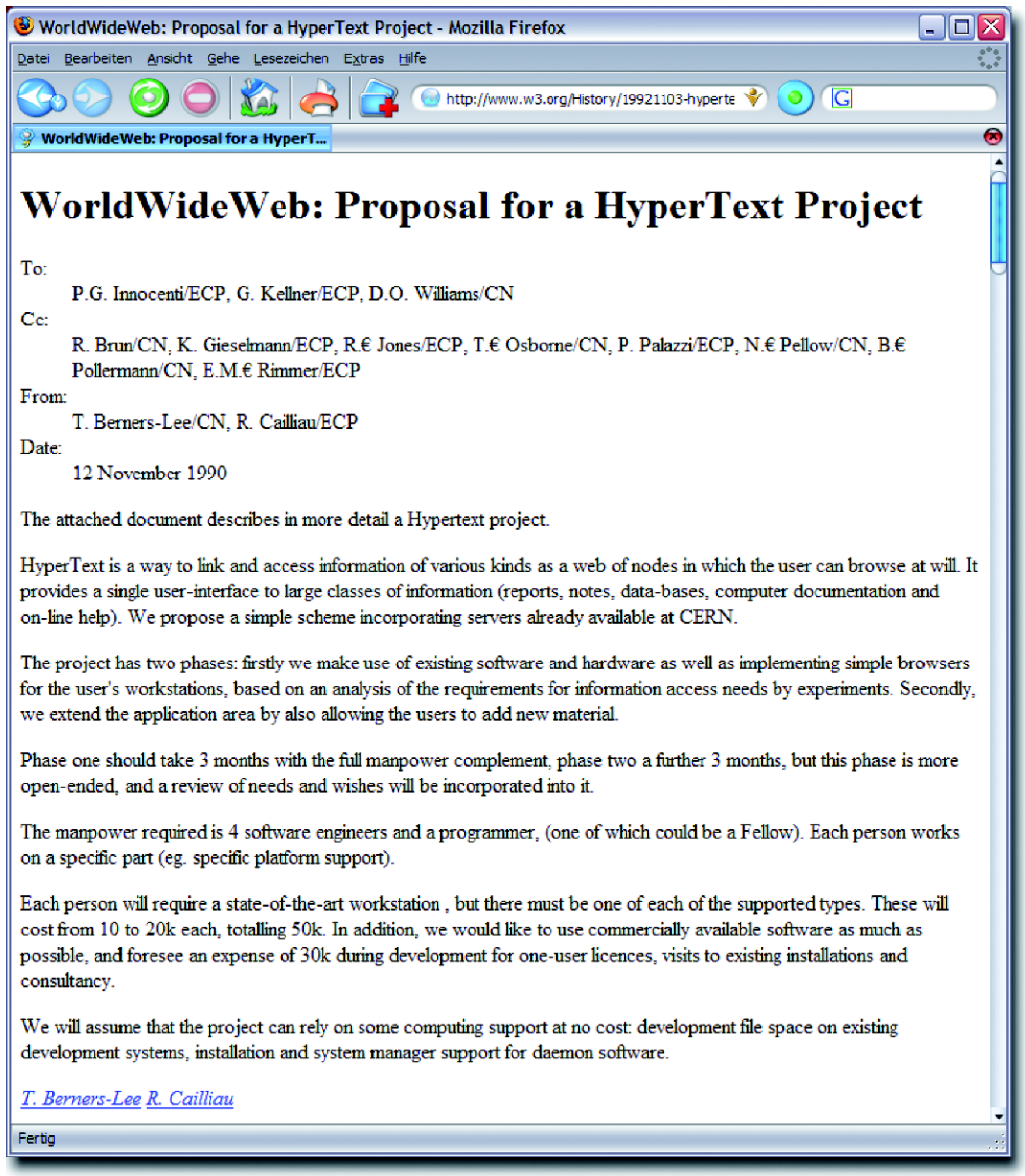



**Abbildung 1.2:** Proposal von Berners-Lee, November 1990

Seit dieser Initialpostulierung des WWW ist sein Wachstum ungebrochen; Abbildung 1.3 zeigt die Entwicklung seit August 1995.

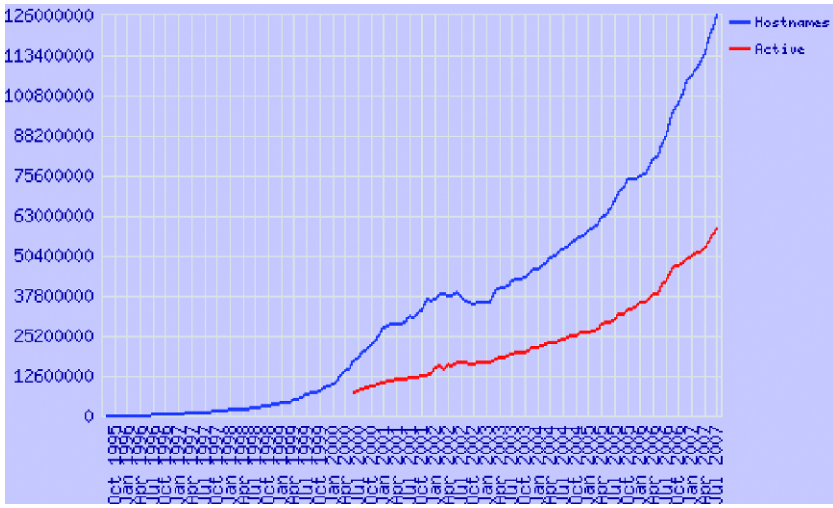



**Abbildung 1.3:**
Wachstum des WWW von August 1995 bis Juli 2007 (Quelle: Netcraft)

## 1.2 Komponenten der frühen Technik

Die ursprüngliche, auf Berners-Lee zurückgehende „WWW-Technik“ kommt mit wenigen fundamentalen Komponenten aus: ein Serverprogramm, welches angeforderte Seiten ausliefert, ein Clientprogramm, welches vom Server diese Informationen anfordert und die Rückantwort zur Anzeige bringt, eine Protokollfamilie, welche die Kommunikation zwischen Server und Client festlegt, sowie eine Auszeichnungssprache für die Codierung der anzuzeigenden Information.

Mit diesen Techniken kann zunächst nur statische Information zur Anzeige gebracht werden, etwa eine Ausgabe „Die Uhrzeit des Servers ist 12h 13min 14sek“ (mit echten, „dynamischen“ Daten) ist ursprünglich nicht möglich.

Diese grundlegenden Bestandteile der Kerntechnik sollen nun vorgestellt werden.

### 1.2.1 Der Webserver

Tim Berner-Lees zentraler Beitrag für das „erste WWW“ war die Bereitstellung eines *Webservers:*

**Ein Webserver ist eine Serverapplikation, welche auf Anfragen im HTTP-Protokoll reagiert und die entsprechenden Antworten generiert.**

Der erste Webserver war der CERN-Webserver, der über den NSCA-Webserver zum Apache weiterentwickelt wurde. Genaueres hierzu in Kapitel 6.

### 1.2.2 Der Webclient

Der *Webclient* hat zwei zentrale Aufgaben: Er stellt die korrekt formulierte Anfrage an den Webserver und bringt dessen Antwort (meist) grafisch formatiert zur Anzeige.

Der erste relevante Webclient war der Mosaic-Browser, welcher von Marc Andreessen entwickelt wurde; Andreessen hat damit die Firma Netscape gegründet, der Mosaic-Browser ging dann im Netscape-Browser auf. Mehr zu den aktuellen Browsern ist in Kapitel 1.2.2 zu finden.

### 1.2.3 Die Protokollfamilie

Für die Kommunikation zwischen Webclient und -server ist die Sprachebene festzulegen. Dies beginnt bei technischen Netzwerkbestandteilen bis hin zu den zulässigen Anweisungen wie `GET` für die Anforderungen eines Dokumentes und den möglichen Antwortarten. Die netzwerktechnische Ebene wird durch tcp/ip (siehe 1.6.2) festgelegt, die eigentliche Kommunikationsebene bestimmt das HTTP-Protokoll nach 1.6.5.

Hinzu kommt noch das Adressierungsschema der URL bzw. URI, welches in 1.6.9 dargestellt wird.

### 1.2.4 Die Auszeichnungssprache HTML

Neben den technischen Notwendigkeiten der Kommunikation zwischen Server und Client ist die Möglichkeit der Formatierung – etwa mit Grafiken und Hyperlinks – ein zentraler Grund für den unfassbaren Erfolg des World Wide Web. Diese Formatierung wird ursprünglich durch die Auszeichnungssprache HTML bereitgestellt, welche in 2.3 im Kern erläutert wird.

### 1.2.5 ...und die Dynamik?

Mit den bisher genannten Technikkomponenten lassen sich statische Inhalte anzeigen – aber keine dynamischen. Einfachstes Beispiel für eine Dynamik ist die Ausgabe der Uhrzeit auf dem Webserver, schon hier endet unser momentanes Wissen.

Um die zu erweitern, ist es notwendig, eine Programmierung bereitzustellen: Das ursprüngliche HTML muss um die Möglichkeit erweitert werden, dynamische Anteile zu implementieren. Hierfür wurde auch das umstrittene Schlagwort „Dynamisches HTML“ – DHTML – eingeführt, auf welches hier bewusst verzichtet wird.

Es gibt inzwischen zahllose Möglichkeiten, Dynamik im Web anzubieten. Die momentan wichtigsten werden in diesem Buch vorgestellt.

## 1.3 Clientseitige Web-Programmierung

Die Programmierung für das Internet kann sowohl client- als auch serverseitig erfolgen – und in vielen Fällen ist sogar die Verbindung beider Ansätze notwendig. Dabei verfolgen beide Ansätze unterschiedliche Ziele.

**Abbildung 1.4:** Der Client in diesem Buch

Domäne der *clientseitigen Web-Programmierung* ist die direkte Interaktion mit dem Benutzer, die Integration von aktiven Programmen in den Browser. Hierzu zählt die Reaktion auf das Mausverhalten und die direkte Überprüfung von Eingabedaten vor Übertragung zu einem Server. Fortgeschrittenere grafische Effekte wie eine schematische Filmsequenz gehören dazu und inzwischen auch eine asynchrone Kommunikation mit Serverdiensten, um ein ähnliches Verhalten wie bei einer Desktop-Applikation auch im Browser anbieten zu können.

Einige der Kern-Techniken in diesem Bereich sind:

- Darstellung mit (X)HTML und CSS;
- JavaScript als die grundlegende Client-Programmiersprache;
- Flash-Techniken etwa für Filmsequenzen mit ActionScript;
- Ajax für ein desktopähnliches Verhalten;

- Java Applets für vollständige Client-Applikationen, welche in einem Browser ablaufen.

Mit diesen Techniken ist auch eine direkte Kommunikation mit einem Datenbanksystem denkbar – insbesondere mit dem Applet –, wird aber aus Gründen der Sicherheit heute nicht mehr angewendet.

Eine typische Client-Programmierung ist der Farbwähler von Selfhtml unter `http://de.selfhtml.org/helferlein/farben.htm`: In einem Web-Formular sind Farbwerte auszuwählen, durch Druck auf den Button „Farben anzeigen“ werden die zugehörigen Hex-Farbwerte berechnet (unten links) und das rechte Fenster wird in den gewählten Farben neu gezeichnet. Alles geschieht direkt auf dem Client.

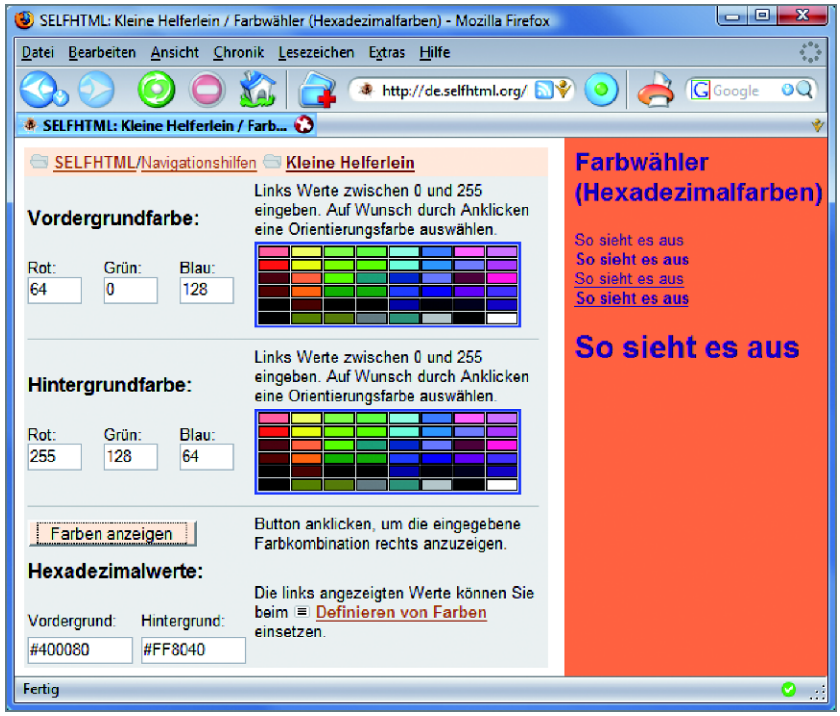



**Abbildung 1.5:**
Exemplarische clientbasierte Web-Programmierung

## 1.4 Serverseitige Web-Programmierung

Wesentlich komplexere Programmierungen finden sich serverseitig, insbesondere, weil hier Datenbankanbindungen genutzt werden können. Typische Beispiele für solche serverseitigen Web-Programmierungen sind Google, Ebay und Amazon.

Hier ist das Spektrum der eingesetzten Techniken groß:

- CGI-Programmierungen mit ausführbaren Programmen wie compiliertes C oder mit Scriptsprachen wie Perl, Python oder Ruby;
- PHP;
- Server Side Includes (SSI);
- fastCGI-Anwendungen;
- spezielle Module für den Webserver;
- Frameworks wie Ruby on Rails;
- Java-Entwicklungen auf dem J2EE-Framework wie Servlets und Java Server Pages;

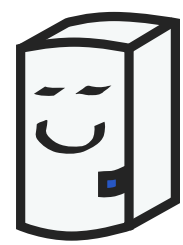


**Abbildung 1.6:**
Der Server in diesem Buch

- Content-Management-Systeme wie TYPO3 mit TypoScript.

## 1.5 Sprachen für die Web-Programmierung

In 1.3 und 1.4 wurden die gängigen Sprachen für den Einsatz in der Web-Programmierung genannt. Einige davon, etwa Perl und Java, sind Sprachen, welche auch unabhängig von der Web-Programmierung eingesetzt werden; andere wie PHP und ActionScript werden faktisch ausschließlich in diesem Bereich verwendet.

Viele dieser Sprachen sind Scriptsprachen, weshalb diese genauer betrachtet werden.

### 1.5.1 Scriptsprachen

*Scriptsprachen* sind einfache Programmiersprachen, die sich meistens durch folgende Eigenschaften auszeichnen:

- sie werden interpretiert und nicht compiliert;
- Variablen in Scriptsprachen sind nicht typisiert: Eine Variable wird nicht deklariert und hat keinen festgelegten Typ wie „Zeichenkette“ oder „Gleitkommazahl“, sondern nimmt beliebige Werte auf und wandelt diese je nach Bedarf in einen anderen Typ um.

Die meisten der in diesem Buch behandelten Sprachen gehören zu der Gruppe der Scriptsprachen, etwa Perl, PHP, JavaScript und Ruby.

### 1.5.2 Java

**Abbildung 1.7:** Java-Logo

Die populäre Sprache *Java* ist keine Scriptsprache, sondern eine moderne Hochsprache, welche typisch für die aktuelle Programmierung ist.

Java wurde um 1995 von James Gosling bei Sun Microsystems entwickelt und hat seither einen beispiellosen Erfolg.

Wesentlich für den Erfolg von Java ist das hier erstmals erfolgreich umgesetzte Konzept der plattformunabhängigen Programmierung: Java Sourcecode wird durch den Java-Compiler `javac` in eine plattformunabhängige class-Datei übersetzt; diese kann direkt auf verschiedenen Betriebssystemen durch die Java Virtual Machine (JVM) ausgeführt werden.

Die Plattformunabhängigkeit von Java hat die Verbreitung der Sprache wesentlich beschleunigt, da sie eine neue Technik in der Web-Programmierung ermöglicht: das Java Applet, ein im Web-Browser ausgeführtes Java-Programm. Dieser Weg hat sich allerdings nicht durchgesetzt, heute ist dafür serverseitiges Java wie Servlets und Java Server Pages eine wesentliche Technik.

Java ist heute in der Anwendung und genauso in der Lehre/Ausbildung eine vorherrschende Sprache. Aus diesem Grund wird in diesem Buch an vielen Stellen auf Java Bezug genommen.

**Dieser spezielle Hinweis zeigt Parallelen und Unterschiede der vorgestellten Sprachen zu Java auf.**

## 1.6 Technische Grundlage: die Internetprotokolle

In technischer Sicht ist „das Internet“ letztlich nichts anderes als die physikalische Infrastruktur mit einer ganzen Familie an *Protokollen,* die den Daten- und Kommunikationsaustausch über das physikalische Netz regeln. Die Gesamtzahl dieser Protokolle wird mit über 500 angegeben.

Welche Protokolle nun genau zu den Definitionen des Internets zählen, ist in letzter Konsequenz nicht zu sagen; die wichtigsten werden hier kurz vorgestellt. Eine umfassendere Darstellung der wichtigsten Protokolle ist in [MS04, S. 236 ff] enthalten.

### 1.6.1 Das Schichtmodell

Das „Netzwerk“ stellt letztlich eine verteilte Anwendung bereit, die zum Teil im Browser des Clientrechners und zum anderen Teil auf einem Webserver abläuft; dazwischen regelt das Netzwerk die Kommunikation zwischen den beteiligten Komponenten.

Um die Arbeitsweise des Netzwerks zu verstehen, ist es sinnvoll, das Netzwerk in verschiedene Hierarchieebenen mit klar definierten Schnittstellen zu zerlegen: das *Schichtmodell* (vgl. [MS04, S. 239 f]).

Das verbreitete ISO/OSI-Schichtmodell geht von sieben aufeinander aufbauenden Netzwerkschichten aus; die ersten vier davon zählen zum Transportsystem.

- **Schicht 1: Bitübertragungsschicht**
  Diese unterste, auch als physikalisch bezeichnete Schicht stellt die technische Bitübertragung bereit.
- **Schicht 2: Sicherungsschicht**
  Die Sicherungsschicht ist für eine fehlerfreie Übertragung verantwortlich und stellt die Verbindung zum Übertragungsmedium her.
- **Schicht 3: Vermittlungsschicht**
  Diese Schicht regelt die Adressierung innerhalb des Netzwerks, also die Zuordnung von Netzwerkadressen zu physikalischen Endpunkten des Netzwerkes.
- **Schicht 4: Transportschicht**
  Diese stellt einen sicheren und korrekten Datenausstausch sicher.

Die oberen drei Schichten bilden das Anwendungssystem; sie bestehen aus:

- **Schicht 5: Sitzungsschicht**
  Diese stellt Dienste für einen synchronisierten Datenaustausch und für die Prozesskommunikation bereit, um den Abbruch einer Sitzung zu verhindern.
- **Schicht 6: Darstellungsschicht**
  Hier werden systemabhängig codierte Daten in unabhängiger Form dargestellt.
- **Schicht 7: Anwendungsschicht**
  Diese oberste Schicht stellt den darüber liegenden Anwendungen die benötigten Funktionalitäten wie E-Mail und Renote Login zur Verfügung.

Die Internet-Programmierung betrifft mindestens ab der dritten Schicht alle vorhandenen.

### 1.6.2 tcp/ip

Bei dem *tcp/ip-Protokoll* handelt es sich um die wesentliche Grundlage des Internets schlechthin, und es sind zwei aufeinander aufbauende Protokolle. Sie betreffen die dritte und vierte Schicht des ISO/OSI-Modells nach 1.6.1.

### 1.6.3 Internet Protocol (ip)

Das *Internet Prococol* ip regelt die Kommunikation zwischen zwei Rechnern ohne vorherigen Verbindungsaufbau auf der Vermittlungsschicht (Schicht drei im ISO/OSI-Modell). Dabei sind keine Mechanismen der Fehlererkennung und -korrektur implementiert, dies bleibt der höheren tcp-Schicht überlassen.

Zwei Versionen des ip sind momentan verbreitet, ip v4 und ip v6.

#### 1.6.3.1 ip v4

Die *Version 4* des ip ist das verbreitetste Protokoll. Bekannt ist es insbesondere durch das Schema der ip-Adressen bestehend aus vier Gruppen zu je 8 bit für die Netzwerkadresse, etwa

```
143.93.17.94
```

für den Rechner mit dem Namen `ceres.informatik.fh-kl.de`. Die Beziehung zwischen Namen und Netzwerkadresse regelt der Domain Name Service nach 1.6.6.

Da für jede Zahl nur 8 bit zur Verfügung stehen, gibt es insgesamt

$$N = 24 \cdot 8 = 2^{32} = 4.294.967.296$$

Netzwerkadressen, wobei einige zusätzlich nur für besondere Zwecke bereitstehen. Dieser Adressraum erweist sich zunehmend als zu klein, weshalb eine neue Version des ip notwendig wird.

Lokale Netzwerke (LAN) bestehen aus einem beschränkten Adressraum. Typisch ist hier die Unterscheidung in Class-A-, Class-B- und Class-C-Netzwerke. Class-C-Netzwerke

- 123.123.123.abc

und damit aus $2^8 = 256$ vielen Adressen. Per Konvention bekommt der lokale Übergangspunkt in andere Netze, der Router, eine Adresse am Ende des Zahlenbereichs, also im Beispiel `123.123.123.255`. Class-B-Netze schränken die Adressen nur nach dem Schema

```
123.123.abc.def
```

ein und bestehen somit aus 65.536 Einzeladressen, Class-A-Netzwerke sind nochmals nach dem Schema

```
123.abc.def.ghi
```

größer und kommen somit auf 16.777.216 einzelne Netzwerkadressen.

#### 1.6.3.2 Sonderadressen in ip v4

Einigen ip-Adressen kommt eine besondere Bedeutung zu, die auch in der Web-Programmierung von großer Bedeutung sind.

Zunächst ist hier der Bezug zum eigenen Netzwerkinterface genannt, die Adresse `127.0.0.1`, die auch als `loopback` oder `localhost` bezeichnet wird. Wenn der Webserver auf dem lokalen Rechner zu testen ist, dann genügt die URL

```
http://127.0.0.1
```

zum Testen. Das gesamte Class-C-Netzwerk

```
127.0.0.abc
```

ist dem Bezug zum lokalen Rechnersystem vorbehalten und wird nicht über das Netzwerk geleitet.

Für Multicast-Dienste gibt es einen eigenen Adressraum von `224.0.0.0` bis `239.255.255.255`.

Wichtig für den Aufbau sicherer Netzwerke sind nichtroutbare Netzwerkadressen. Dieser Adressbereich ist in RFC 1918 definiert; Tabelle 1.1 gibt eine Übersicht über den privaten Netzwerkadressbereich. Diese Adressen können in dieser Form nur lokal verwendet werden.

#### 1.6.3.3 Network Address Translation (NAT)

Um mit nichtroutbaren internen Adressen nach außen arbeiten zu können – etwa um „zu surfen“ – bedient man sich der Adressumsetzung *Network Address Translation* (NAT); hier wird jede ausgehende Netzwerkverbindung auf eine offizielle, routbare Adresse umgesetzt. Der Vorteil davon ist, dass somit keinerlei Informationen über die tatsächlichen internen Adressen nach außen gelangt. Üblicherweise werden alle internen Adressen dabei auf die gleiche externe Adresse umgeleitet.

Vergleichbar zu NAT ist Port Adress Translation (PAT); hierbei werden die Netzwerk-Ports umgesetzt.

| Adressbereich | Anzahl Adressen |
|---|---|
| `10.0.0.0` – `10.255.255.255` | 16.777.214 |
| `172.16.0.0` – `172.31.255.255` | 1.048.544 |
| `192.168.0.0` – `192.168.255.255` | 65.536 |
| `169.254.0.0` – `169.254.255.255` | 65.536 |

**Tabelle 1.1:** Private Netzwerkadressen

#### 1.6.3.4 ip v6

Die neuere *Version 6* des ip beinhaltet wesentliche Fortschritte gegenüber der Version 4. Hierzu zählen insbesondere:

- Der Adressraum in IPv6 ist deutlich erhöht bei Beibehaltung der Kompatibilität zu den IPv4-Adressen.
- IPv6 beinhaltet die Sicherungsmaßnahmen, die als IPSec bezeichnet werden. Damit wird eine netzwerkseitig integrierte Verschlüsselung des Datentransfers ermöglicht.

#### 1.6.3.5 Transmission Control Protocol (tcp)

Das mit dem Internet Protocol 1982 eingeführte Transmission Control Protocol (tcp) ist ein verbindungsorientiertes Protokoll auf der ISO/OSI-Schicht vier (Transportschicht). Es setzt auf dem ip auf und sichert die wichtige Korrektheit der Datenübertragung.

Konkret bedeutet das tcp, dass die einzelnen Datenpakete, aus denen ein Datenstrom besteht, überprüft werden auf

- die Vollständigkeit;
- die richtige Reihenfolge des Empfangs.

Somit ist eine auf tcp/ip basierende Netzwerkverbindung die ideale geprüfte Verbindung für die Daten einer Web-Site. Wir werden vorwiegend bei der Web-Programmierung deshalb auf tcp/ip aufbauen, benötigen aber mitunter auch andere Protokolle, die hier noch kurz vorgestellt werden sollen.

### 1.6.4 User Datagram Protocol (udp)

Das *User Datagram Protocol* (udp) ist die Alternative zu tcp, wenn die Schnelligkeit der Übertragung wichtiger ist als die Korrektheit. Dies ist etwa im Bereich von Streaming-Diensten der Fall.

udp ist ein verbindungsloses Protokoll. Es verzichtet auf die Kontrolle der Korrektheit der Datenlieferung, die wesentlicher Bestandteil von tcp ist.

### 1.6.5 Hypertext Transfer Protocol (HTTP)

Das *Hypertext Transfer Protocol* (HTTP) nimmt für die Thematik dieses Buches eine ganz entscheidende Rolle ein, da es regelt, wie zwischen Webclient und Webserver Informationen ausgetauscht werden.

HTTP ist Kern der „Erfindung des Webs“ durch Tim Berners-Lee (vgl. 1.1). 1989 wurde Version 0.9 von HTTP vorgestellt, ihr folgten 1992 die Versionen 1.0 und die seit 1997 aufgrund von RFC 2068 verwendete Version 1.1.

HTTP ist ein sehr einfaches Protokoll. Die drei Versionen lassen sich folgendermaßen abgrenzen:

- HTTP Version 0.9 (Abbildung 1.8)
  - vollständig textbasiert (keine Multimedia-Daten)
  - der Server kann nur angefordertes Dokument senden, keine zusätzlichen Informationen
  - der Client kann keine Daten an den Server übertragen
  - HTTP 0.9 ist nicht mehr aktuell
- HTTP Version 1.0 (Abbldlung 1.9)
  - bietet neben der Methode `GET` auch die Methode `POST`
  - implementiert HEADer-Informationen mit Meta-Daten
  - stellt Internet Medientypen (MIME) zur Verfügung
  - unterstützt mit der Information last-modified Caching-Mechanismen
  - bietet Authentifizierungsmöglichkeiten.
- HTTP Version 1.1 (Abbildung 1.10)
  - Entitäts-Tag: eindeutige Kennung für jedes Dokument (erleichtert Caching auf mehreren Servern)
  - feste Verbindungen (Zeitbegrenzung durch keep-alive), dadurch verbessertes Netzwerkmanagement
  - verbesserte Authentifizierung (DIGEST)
  - Multihoming: ein Server verwaltet mehrere WWW-Domänen mit unterschiedlichem Dokumentenstamm

Die Bedeutung der wichtigsten HTTP-Sprachbestandteile ist in Tabelle 1.2 aufgeführt.

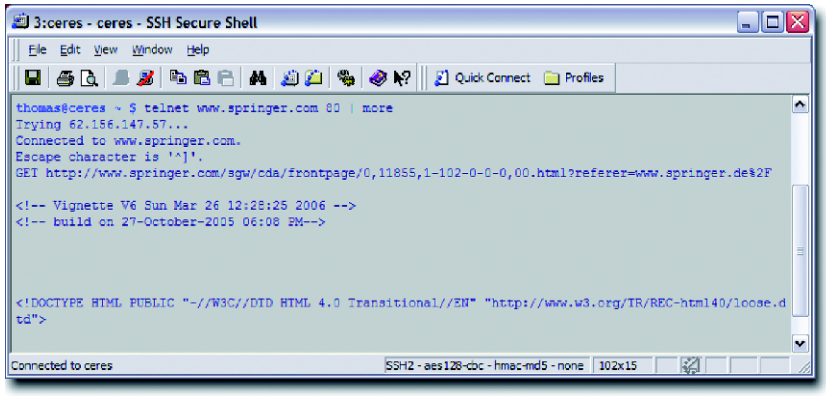



**Abbildung 1.8:**
GET-Request zur Springer-Homepage in HTTP 0.9

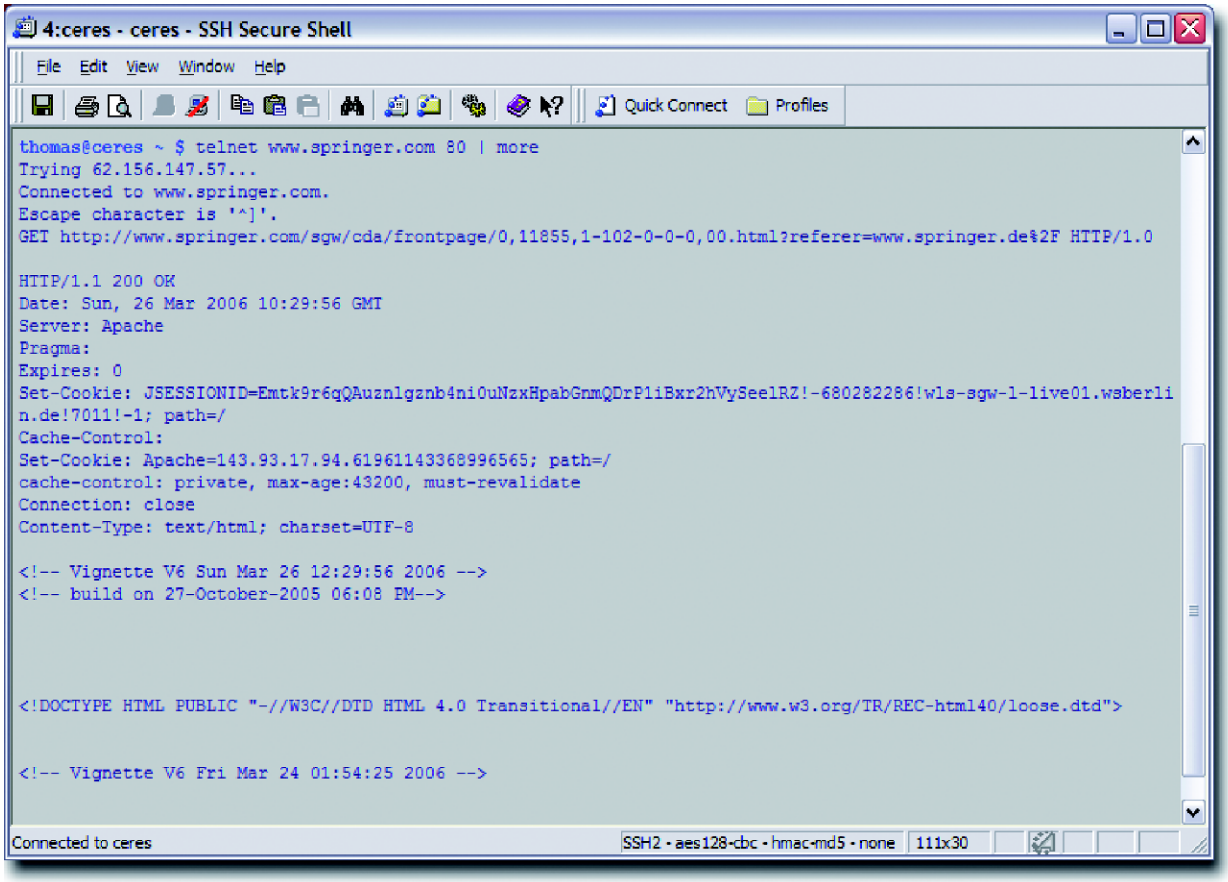



**Abbildung 1.9:**
GET-Request zur Springer-Homepage in HTTP 1.0

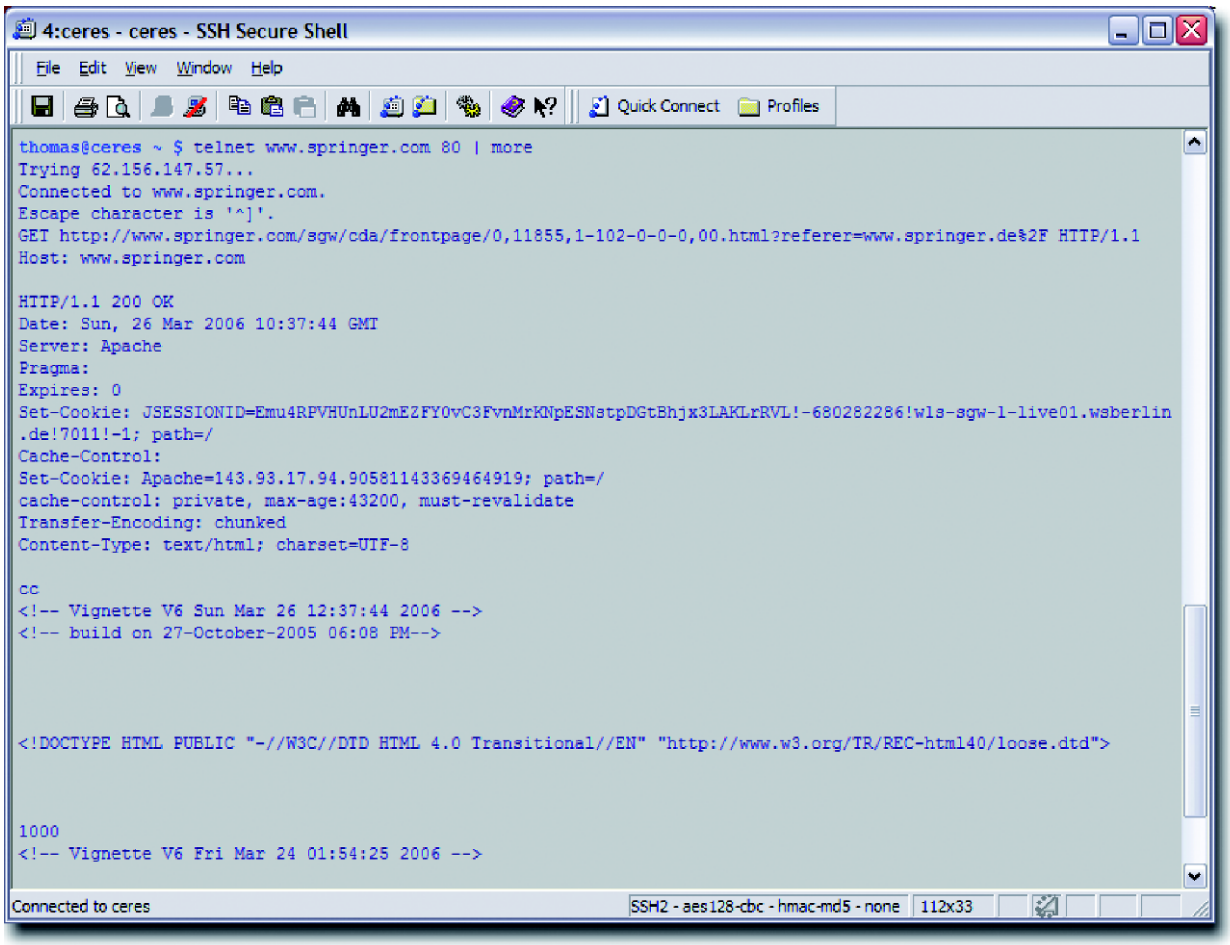



**Abbildung 1.10:** `GET`-Request zur Springer-Homepage in HTTP 1.1

**Tabelle 1.2:** HTTP-Syntax (Version 1.1)

| Anweisung | Bedeutung |
|---|---|
| `get` | fordert Dokument an |
| `post` | wie `get`, überträgt Daten zu Server über separate IO-Verbindung |
| `head` | fordert nur die HEADER-Informationen an |
| `put` | Anweisung für Upload |
| `trace` | Ausweisung aller Proxy-Server |
| `delete` | Entfernen eines Dokuments |
| `options` | Auflistung der möglichen HTTP-Anweisungen |
| `connect` | Proxy-Funktionalität |

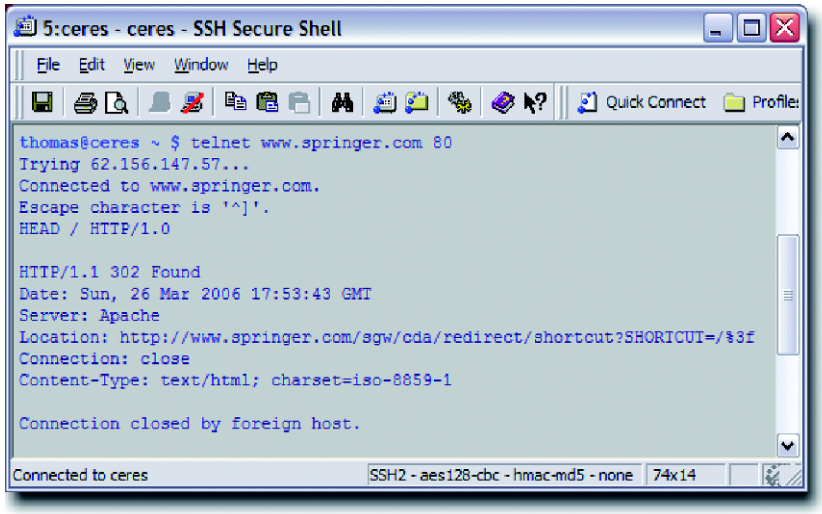



**Abbildung 1.11:**
`HEAD` der Springer-Homepage

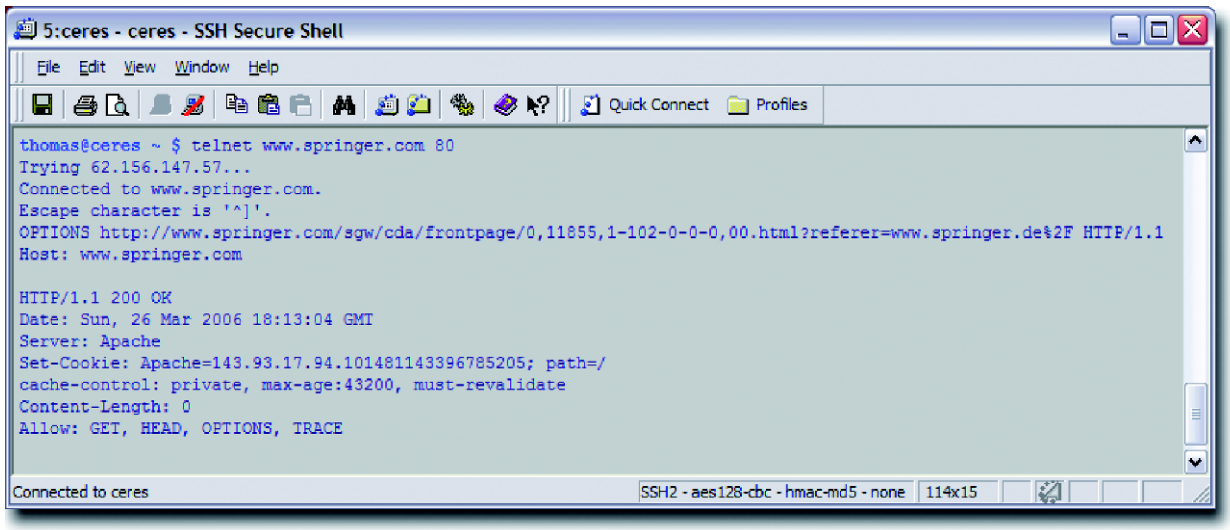



**Abbildung 1.12:**
`OPTIONS` der Springer-Homepage: Möglich sind die HTTP-Anweisungen `GET`, `HEAD`, `OPTIONS`, `TRACE`

Das HTTP-Protokoll beinhaltet auch standardisierte numerische Antwortcodes, die in fünf Gruppen unterteilt werden:

- informative Codes (ab HTTP 1.1) im Bereich 100 – 199;
- Kennzeichnung des erfolgreichen Requests: 200 – 299;
- jede Art von Umleitung: 300 – 399;
- Fehler aufgrund unkorrekter Client-Anfrage (unvollständig, Anfrage auf nichtexistierende Ressource): 400 – 499;
- Serverfehler: 500 – 599.

Der „gutartige“ Fall ist stets der Antwortcode 200 (ok). Jeder, der mehr Erfahrung mit der Web-Programmierung hat, kennt insbesondere den HTTP-Fehlercode 500 (Abbildung 1.13).

Tabelle 1.3 gibt eine Übersicht über die wichtigsten HTTP-Codes.

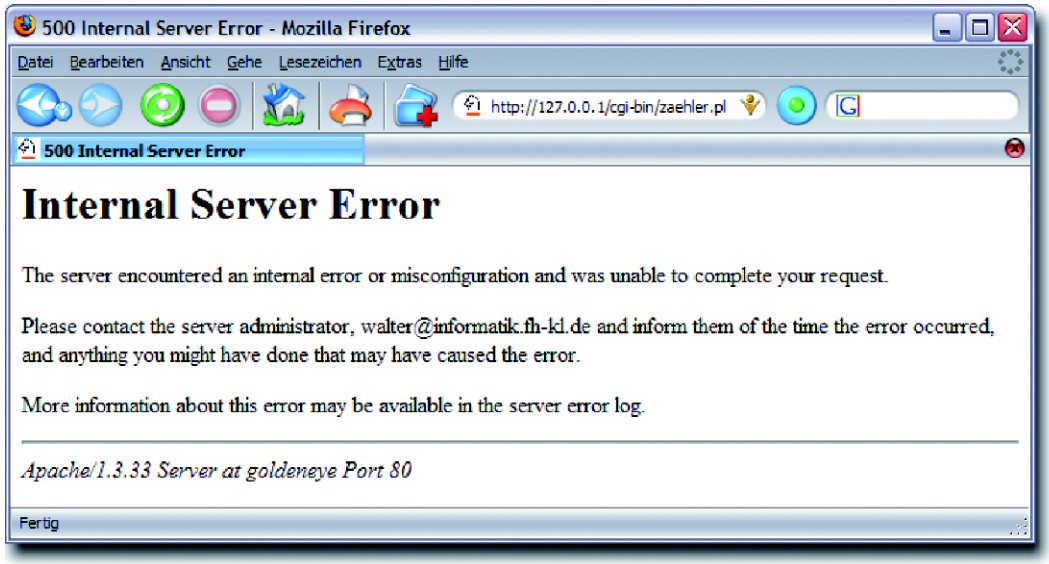



**Abbildung 1.13:** Alltag der Web-Programmierung: HTTP-Antwortcode 500 Internal Server Error

#### 1.6.5.1 Datenübertragung mittels `GET` und `POST`

Ein wesentlicher Bestandteil des HTTP-Protokolls ist die Datenübertragung zum Webserver, konkret die Übertragung eines ausgefüllten Formulars zum Server. Neben der Zieladresse für die Verarbeitung der Daten – etwa dem Namen eines CGI-Scriptes – gehört zur Übertragung ein Datenteil. Der Übertragungsmechanismus ist bei `GET` und `POST` unterschiedlich.

Ein derartiger Datenteil hat den Aufbau einer „assoziativen Liste“, wie wir sie noch häufig finden werden: eine Ansammlung von Key-Value-Paaren, von Paaren bestehend aus dem Namen des Formularfeldes und dem eingetragenen Wert. Diese Struktur der Art

```
feld1  wert1
feld2  wert2
...
```

wird mit den zwei Verfahren `GET` und `POST` unterschiedlich übermittelt:

- Bei `GET` wird der Datenteil getrennt durch das Sonderzeichen `?` an die Adresse angefügt; die einzelnen Key-Value-Paare werden durch `&` getrennt, so dass etwa folgender Request generiert wird:

  ```
  http://rechnername/scriptname?feld1=wert1&feld2=wert2
  ```

  Dieses Verfahren hat Vor- und auch Nachteile. Die so erzeugten Requests sind vollständig sichtbar und werden in der Browser-History hinterlegt, was für Debugging vorteilhaft ist, aber etwa für die Übermittlung von Kennwörtern ausscheidet. Ferner gibt es Probleme mit der Übermittlung sehr langer Datenblöcke und Sonderzeichen wie `=` und `&` müssen maskiert werden.

  Es ist sehr leicht, mit einfachen Clientscripten – etwa mit Perl zusammen mit dem LWP-Modul, vgl. 10.8 – derartige Requests zu erzeugen und damit praktisch beliebig viele ausgefüllte Formulare zu einem Server zu schicken.
- Bei `POST` wird parallel zum HTTP-Request zur Adresse ein separater Übermittlungskanal für die Daten genutzt, was wegen der bei `GET` beschriebenen Probleme vorteilhaft ist.

#### 1.6.5.2 Webserver-Zugriff mit `PUT`, `DELETE` und WebDAV

Mit der Version 1.1 stellt HTTP die Methoden `PUT` und `DELETE` bereit, und damit zumindest prinzipiell die Möglichkeit, mit HTTP Dateien auf einem Webserver abzulegen oder von diesem zu löschen; allerdings sind diese Methoden meistens nicht verfügbar und führen nur zum Antwortcode 405: Method Not Allowed.

*Web-based Distributed Authoring and Versioning* (WebDAV) stellt diese Funktionalität heute bereit und ist in vielen Clientprogrammen wie dem Internet-Explorer und Betriebssystemen integriert. Sinnvoller, wenn auch geringfügig schwieriger in der Bedienung, ist das sicherere SSH-Protokoll (vgl. 1.6.7.3).

| Code | Bedeutung |
|---|---|
| 200 | ok |
| 201 | created |
| 202 | accept |
| 204 | no content |
| 300 | multiple choices |
| 301 | moved permanently |
| 302 | moved temporarily |
| 304 | not modified |
| 400 | bad request |
| 402 | unauthorized |
| 403 | forbidden |
| 404 | not found |
| 500 | internal server error |
| 501 | not implemented |
| 502 | bad gateway |
| 505 | service unavailable |

**Tabelle 1.3:** HTTP-Antwortcodes

### 1.6.6 Domain Name Service (DNS)

Hinter dem *Domain Name Service* (DNS) verbirgt sich die im Zusammenhang mit dem ip beschriebene Auflösung der technischen ip-Adresse zum Namen.

Das nützliche Dienstprogramm `nslookup` stellt auf der Kommandozeile DNS bereit; Abbildung 1.14 zeigt die Namensauflösung für das Beispiel www.springer.de.

**Abbildung 1.14:** `nslookup` für www.springer.de: der Server hat die ip-Adresse 62.156.147.57; www.springer.de ist genauso wie www.springeronline.com ein Alias für www.springer.com

Sowohl unter Unix als auch unter Windows gibt es eine einfache Datei „Hosts“, welche die DNS-Abfrage unterstützt. Hier werden feste Namensauflösungen eingetragen, die stets bevorzugt verwendet werden. Für die praktische Web-Programmierung und -Administration ist es meistens sehr nützlich, die eigenen Systeme mit einem kurzen Akronym hier zu hinterlegen.

Unter Unix liegt diese Datei häufig im Ordner `/etc`, bei Windows unter `\WINDOWS\system32\drivers\etc`.

### 1.6.7 telnet, FTP und SSH

#### 1.6.7.1 telnet

*telnet* erlaubt das Einloggen auf einem entfernten Rechner. Da telnet aber ohne Verschlüsselung arbeitet und somit alle Verbindungsdaten im Klartext übertragen werden, nimmt es aktuell keine große Bedeutung mehr ein, sondern wurde weitgehend durch SSH abgelöst.

Während die Unix/Linux-Systeme in der Regel mit einem telnet-Client und einem telnet-Server ausgestattet sind, steht unter Windows nur der telnet-Client bereit.

**Auf Unix/Linux-Systemen sollte man sich aus Sicherheitsgründen genau überlegen, ob die unsicheren Dienste telnet und FTP benötigt werden; am besten, diese Dienste werden erst gar nicht installiert (mehr dazu in Kapitel 30).**

### 1.6.7.2 File Transfer Protocol (FTP)

Dient das telnet-Protokoll dem Arbeiten auf einem entfernten Rechner, so können mit dem *File Transfer Protocol* (FTP) Dateien übertragen werden. Über FTP ist es hingegen nicht möglich, eine Shell auf dem entfernten Rechner zu starten und dort Betriebssystem-Kommandos auszuführen.

Ebenso wie telnet arbeitet FTP unverschlüsselt, weshalb sein Einsatz kritisch zu bedenken ist. Für Windows steht standardmäßig kein FTP-Server bereit; der Webserver von Microsoft stellt einen FTP-Dienst aber bereit, ebenso wie gute und teils freie Zusatzprodukte (etwa Serv-U von Rob Beckers), die meistens aus Sicherheitsgründen und wegen der besseren Konfigurierbarkeit zu bevorzugen sind.

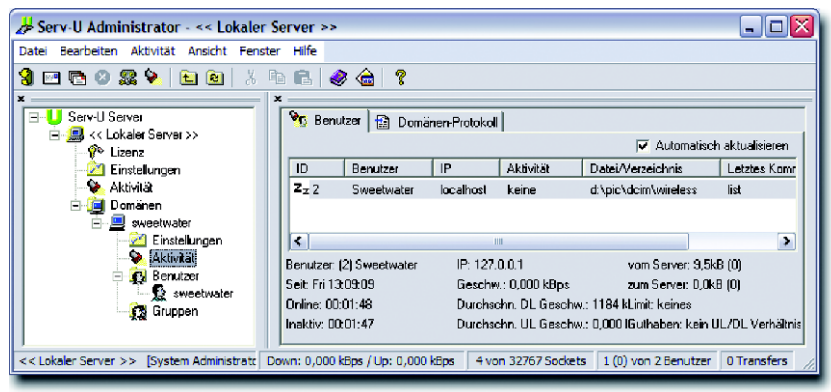



**Abbildung 1.15:**
FTP-Server Serv-U auf Windows mit einer aktiven Verbindung

Viele Internet-Provider bieten bis heute FTP-Dienste an, damit ihre Kunden ihre Seiten überspielen können. Teilweise wird sogar ausschließlich der FTP-Zugang für diesen Zweck angeboten. Dass man sich lokale Kopien dieser Daten halten muss, ist eine Selbstverständlichkeit.

Viele grafische FTP-Clients stehen für alle Betriebssysteme zur Verfügung; beliebt ist unter Windows etwa WS-FTP, dessen light-Version frei verfügbar war und von vielen Providern immer noch angeboten wird.

### 1.6.7.3 Secure Shell (SSH)

Das *Secure Shell-Protokoll* (SSH) erlaubt wie telnet das Arbeiten auf einem entfernten Rechner – im Gegensatz zu telnet wird der gesamte Datenverkehr aber verschlüsselt, so dass ein einfaches Abhören der Zugangsdaten nicht mehr möglich ist. Ebenso beinhaltet SSH aber auch die Möglichkeiten von FTP zum Transfer von Daten, allerdings sicher verschlüsselt. Somit ist SSH der Nachfolger beider Protokolle.

Die SSH-Anmeldung kann weiter deutlich sicherer gestaltet werden, wenn auf ein Kennwort verzichtet und ein Zertifikat verwendet wird. Hierbei handelt es sich letztlich um den öffentlichen Schlüssel eines Schlüsselpaares. Durch diese Zertifikate können keine Kennwörter mehr erraten werden, was wichtig ist, da

inzwischen zahllose automatisierte Angriffe dieser Art erfolgen.

Es stehen verschiedene Implementierungen von SSH bereit, etwa die freien openSSH und putty, aber auch das kommerzielle ssh von ssh.com. Leider sind die Zertifikate zwischen diesen nicht immer direkt austauschbar, die notwendige Anpassung ist aber einfach.

**SSH stellt die sichere Alternative zu telnet und FTP dar; SSH sollte mit Zertifikaten und nicht mit Kennwörtern verwendet werden.**

### 1.6.8 Simple Mail Transfer Protocol (SMTP)

Ein weiterer wichtiger Dienst im Internet ist das E-Mail. Die Vermittlung von E-Mails regelt das *Simple Mail Transfer Protocol* (SMTP). Das wichtigste Programm für Mailserver ist sendmail, dessen Konfigurationsdatei `sendmail.cfg` wegen seiner Komplexität berüchtigt ist.

Die Entwicklung von Web-Angeboten ist häufig auf eine Möglichkeit angewiesen, mittels SMTP E-Mails zu versenden. Verschiedene Möglichkeiten dafür werden wir kennenlernen.

In Zusammenhang mit SMTP steht das Post Office Protocol (POP3), welches regelt, wie Mailclients eingegangene E-Mails vom Mailserver abholen. Auch hier gibt es mit POP3s inzwischen eine verschlüsselte Alternative.

Das Internet Message Access Protocol (IMAP) ist eine weitere Alternative für POP3, die insbesondere dann von Vorteil ist, wenn mehrere Clients auf das gleiche Konto zugreifen sollen.

| Dienst | Port |
|---|---|
| ftp | 21 |
| SSH | 22 |
| telnet | 23 |
| smtp | 25 |
| http | 80 |
| https | 443 |

**Tabelle 1.4:** Netzwerkports der wichtigsten Dienste

### 1.6.9 Das Adressierungsschema im Web: URI und URL

Für beliebige Ressourcen im Web stellt der *Uniform Resource Identifier* (URI) eine eindeutige Bezeichnung bereit. Der Grundaufbau eines URI ist durch

```
<Schema>:<Schema-spezifischer Teil>
```

gegeben: durch „:“ wird der Typ, das Schema von den Detailangaben getrennt. Als Typ oder Schema sind dabei etwa `http`, `ftp` oder `file` möglich. Für `http` oder `ftp` kommt ein hierarchischer Aufbau der URI nach dem Prinzip

```
<Schema>://[<Benutzer>[:<Passwort>]@]<Server>
           [:<Port>]/[<Pfad>][?<Anfrage>][#<Fragment>]
```

zum Einsatz.

Eine spezielle Variante des URIs ist der *Uniform Resource Locator* (URL), der speziell für Ressourcen („Locations“) im Web mit Zugriff über HTTP oder FTP verwendet wird.

Eine weitere Variante des URI ist der Uniform Resource Name (URN); hier wird eine Ressource nach dem URI-Schema identifiziert, etwa passend zum Beispiel nach Kapitel 7 `urn:isbn`.

Insgesamt ist URI im RFC 1630 definiert, URL in RFC 1738.

### 1.6.10 Netzwerkkonfiguration

Jeder Internet-Rechner braucht eine spezifische Konfiguration für die Nutzung der vorgestellten Dienste. Hierzu zählt insbesondere:

- die lokale Netzwerkadresse;
- die Netzwerkmaske, die den Bereich des lokalen Netzwerkes festlegt;
- die Adresse des Gateways, welches die Pakete weiterleitet, welche nicht für das lokale Netzwerk bestimmt sind;
- die Adresse der DNS-Server.

Die verschiedenen Betriebssysteme stellen nützliche Tools bereit, um diese Konfiguration auf der Kommandozeile schnell abzurufen:

- Unix: `ifconfig` (vgl. Abbildung 1.16);
- Windows: `ipconfig` (vgl. Abbildung 1.17).

Allerdings ist `ifconfig` hauptsächlich für das Konfigurieren des Netzwerkinterfaces gedacht; der Schalter `-a` zeigt alle, auch die inaktiven, Netzwerkinterfaces an. Insgesamt kann jedes Netzwerkinterface auf Unix mittels `ifconfig` genau konfiguriert werden.

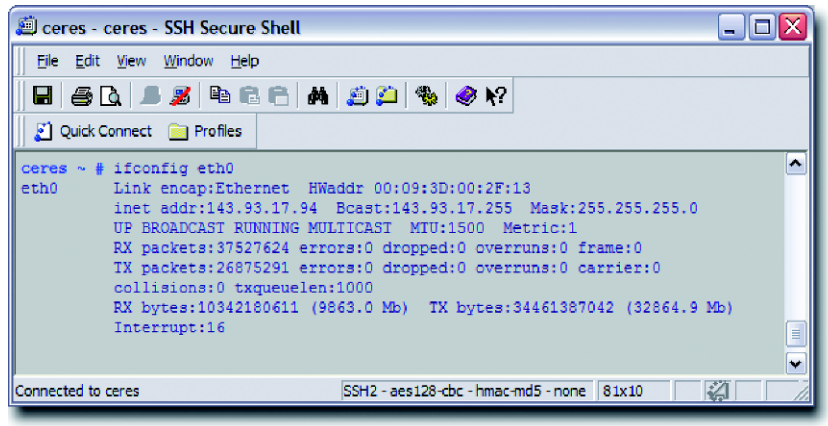



**Abbildung 1.16:** `ifconfig` auf Unix für das Netzwerkinterface `eth0`

Auf Windows stellt, wie erwähnt, das Dienstprogramm `ipconfig` eine schnelle Übersicht der Netzwerkinterfaces bereit; mit dem Schalter `/all` werden wichtige Zusatzinformationen, etwa die verwendeten DNS-Server (vgl. 1.6.6), ausgegeben.

Einige nützliche Optionen für `ipconfig` sind etwa:

- `/all`: vollständige Information;
- `/renew`: erneuert die ip-Adressen eines Netzwerkadapters;
- `/registerdns`: Erneuert die DHCP-Konfiguration;
- `/flushdns`: Löschen des DNS-Cache.

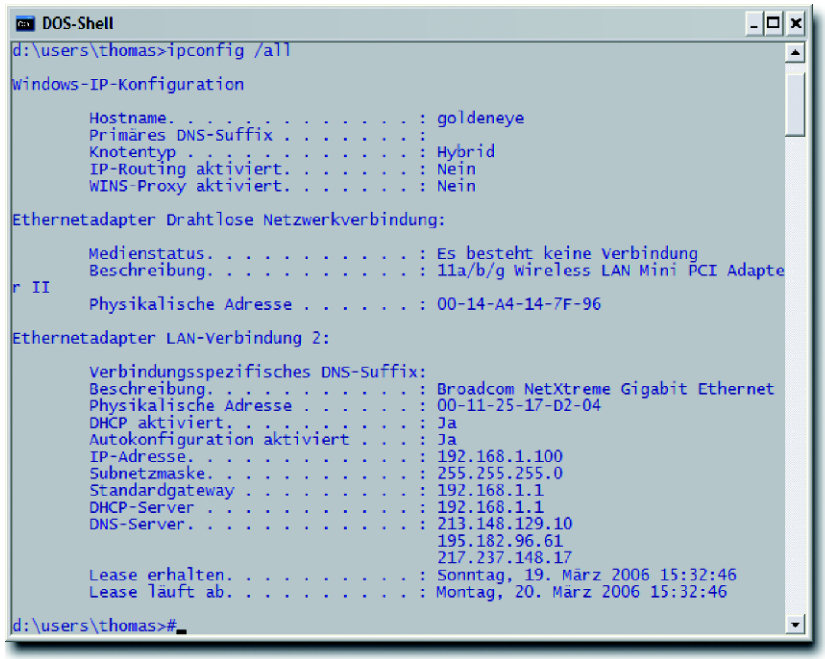



**Abbildung 1.17:** `ipconfig /all` auf Windows

Neben der Überprüfung der aktuellen Netzwerkkonfiguration ist die Übersicht der bestehenden Verbindungen häufig wichtig; dies kann sowohl unter Unix als auch unter Windows mittels `netstat` geschehen (Abbildung 1.18). Hier werden sowohl die bestehenden Verbindungen als auch die offenen, also „hörenden“ Netzwerkports ausgegeben.

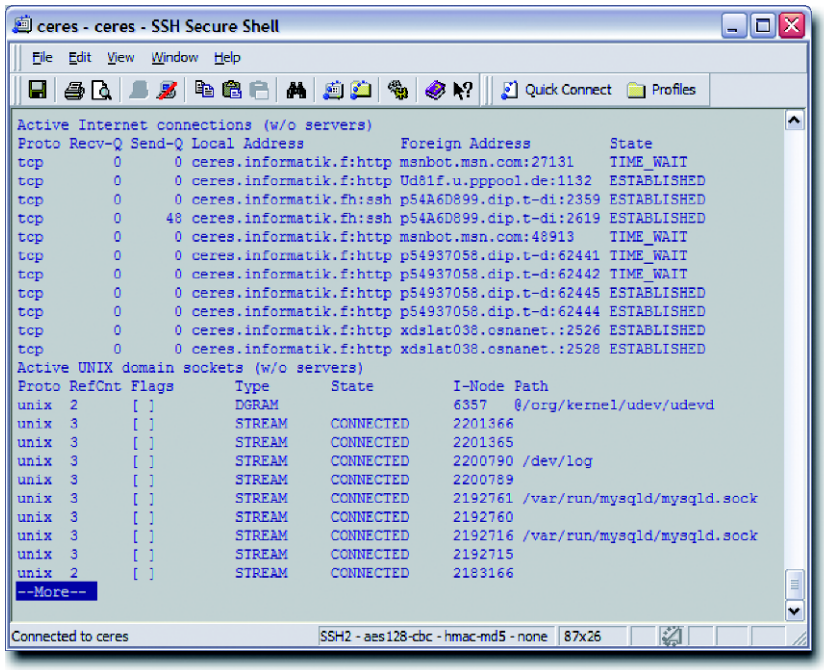



**Abbildung 1.18:** `netstat` auf Unix: es besteht beispielsweise eine http-Verbindung (tcp) zu msnbot.msn.com

Ein einfaches, aber auch in vielen Fällen unerlässliches Werkzeug sind die Dienstprogramme

- `tracepath` oder `traceroute` unter Unix;
- `tracert` unter Windows.

Beide zeigen den Weg zu einem entfernten Rechner über die einzelnen Stationen des Internets (vgl. Abbildung 1.19); dies ist etwa wichtig um Einschränkungen durch Firewalls auf diesem Weg zu sehen.

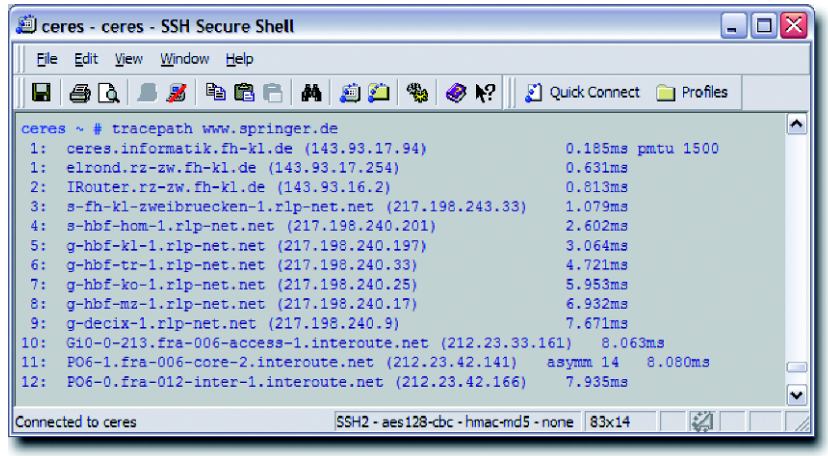



**Abbildung 1.19:** `tracepath` auf Unix: Anfang des Weges zu www.springer.de

Neben diesen einfachen – aber effizienten – Werkzeugen gibt es viele weitere; für das professionelle Arbeiten im Bereich der Web-Programmierung ist der Einsatz dieser Hilfsmittel unerlässlich, um die zugrundeliegende Netzwerktopologie zu verstehen.

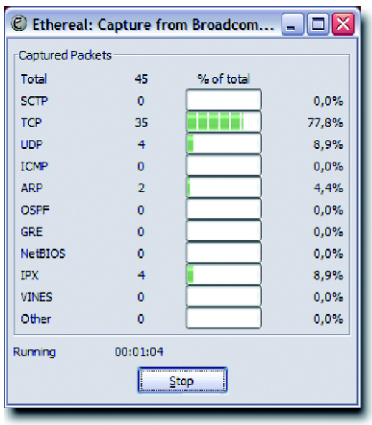



**Abbildung 1.20:** Aufzeichnung des Netzwerkverkehrs mit Ethereal

### 1.6.11 Netzwerkanalyse

Zum besseren Verständnis der Kommunikation zwischen Webserver und Client dient die *Netzwerkanalyse.* Hier wird der Verkehr der Netzwerkverbindung ausgewertet.

Netzwerkanalyse-Tools werden – eher abfällig – auch als „Sniffer“ bezeichnet, da mit ihnen der Netzwerkverkehr ja „ausgeschnüffelt“ wird. Es ist deshalb immer zu bedenken, dass der Einsatz dieser Technik mit dem lokalen Systemadministrator abgestimmt werden muss.

#### 1.6.11.1 Ethereal

*Ethereal* ist ein gutes, freies Tool für die Netzwerkanalyse sowohl für Unix als auch für Windows. Ethereal geht auf Gerald Combs zurück und basiert auf der libpcap-Bibliothek (Windows: WinPcap) für den Zugriff auf die Netzwerkkarten.

Die Problematik unverschlüsselter Protokolle kann damit direkt gezeigt werden, da hier die gesamte Kommunikation im Klartext mitgehört werden kann.

Die Arbeitsweise von Analysewerkzeugen wie Ethereal ist einfach: Zunächst wählt man dasjenige Netzwerkinterface aus, welches analysiert werden soll. Anschließend wird die Protokollierung gestartet, und nach deren Abschluss kann man die einzelnen Netzwerkpakete genau analysieren, wobei Filterregeln – etwa zur Auswahl nur eines Protokolls – wesentlich helfen. Abbildung 1.21 zeigt das Netzwerkpaket, welches den get-Request des HTTP-Protokolls

zu www.springer.de, ausgehend von einem Opera-Browser, enthält; im unteren Datenteil ist die HTTP-Syntax zu sehen. Abbildung 1.22 zeigt die gesamte Kommunikation zur Abfrage der Site www.springer.de.

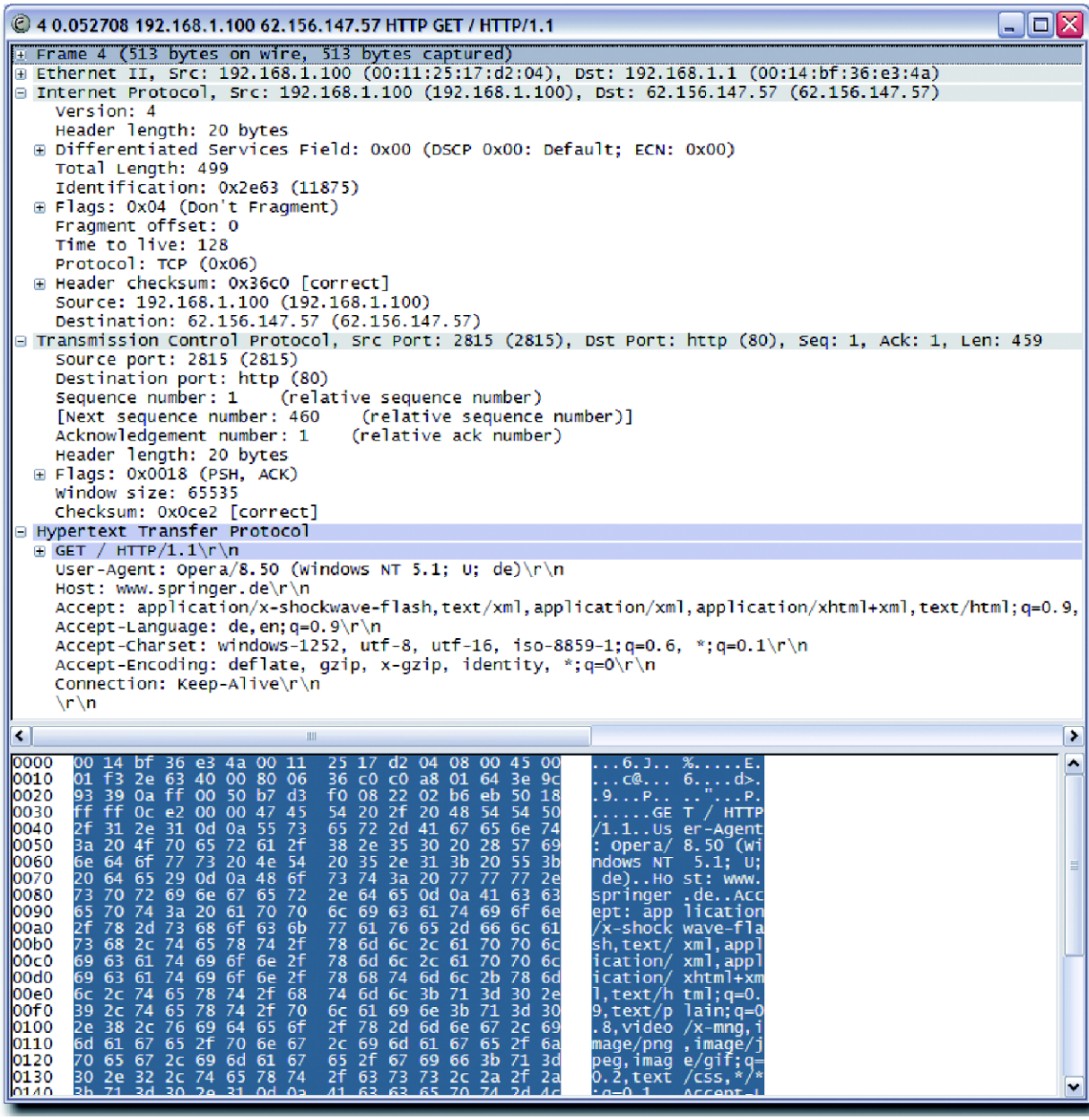



**Abbildung 1.21:**
HTTP-tcp-Paket, welches einen HTTP-get-Request zu www.springer.de von Opera 8.50 durchführt

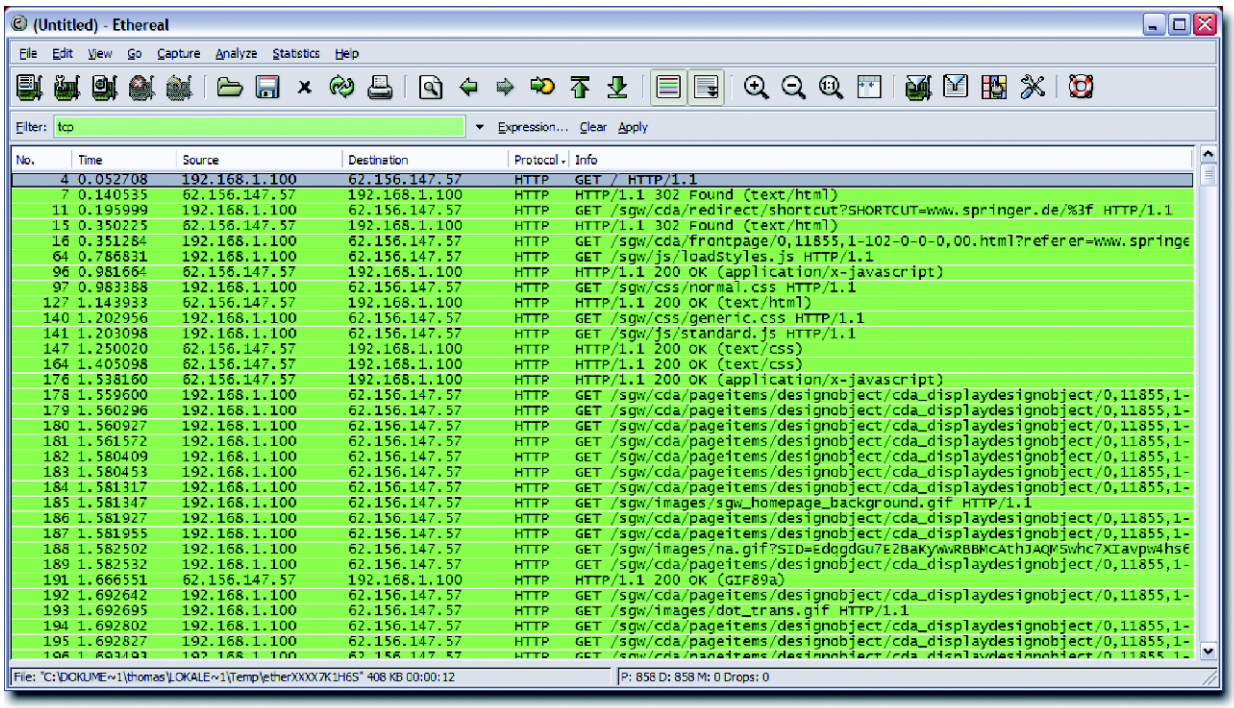



**Abbildung 1.22:**
Gesamte HTTP-Kommunikation für die Abfrage von www.springer.de (Anfang)

## 1.7 Sicherheit

In diesem Kapitel wurden an einigen Stellen Fragen der *Sicherheit* angesprochen. Dies ist heute für die Web-Entwicklung absolut unverzichtbar und muss stets mitbedacht werden. Aus diesem Grund sind in Kapitel 30 wichtige Punkte hierzu zusammengefasst.

# 2

# Darstellung im Web – Auszeichnungssprachen

Eine der Grundlagen der Web-Programmierung ist „HTML“. Da in diesem Buch immer wieder auf HTML und verwandte Techniken Bezug genommen wird, soll hier eine kurze Auseinandersetzung mit speziellem Fokus auf die später benötigten Aspekte wie beispielsweise Formulare erfolgen.

Für eine Vertiefung mit dieser Materie liegt umfassende Literatur zum Themenbereich „Webdesign“ vor.

## 2.1 Auszeichnungssprachen und die Trennung von Inhalt und Formatierung

HTML, die „Hypertext Markup Language“ ist zunächst eine *Auszeichnungssprache*. Es gibt viele Arten solcher Sprachen, hier spielen neben HTML insbesondere XML und ganz am Rande SGML eine Rolle.

Ein anderes Beispiel für solche Sprachen ist die Textverarbeitung mit TEX und LATEX (siehe [BLL06]), wie auch dieses Buch entstanden ist: Neben dem reinen Text gibt es Auszeichnungsattribute, welche die Struktur des Dokuments vorgeben. Im nächsten Schritt kann dann die strukturierte Information mit Formatierungsvorgaben verbunden und so entsprechend zur Anzeige gebracht werden. Im Falle von LATEX könnte das folgendermaßen aussehen:

```
\begin{center}
     zentrierter Text
\end{center}
```

Hier ist zunächst nur die Information durch die „Tags“ strukturiert, wobei diese Steuerelemente gleichzeitig direkt die Formatierung bestimmen; es geschieht das erwartete.

An diesem Beispiel sieht man auch direkt eine zentrale Eigenschaft: Das öffnende Tag muss sich später wieder schließen. Aber von dieser Regel kann es Ausnahmen geben; so wird in LATEX eine neue Seite mittels

```
\newpage
```

begonnen; hier fehlt das schließende Tag.

Einen Schritt weiter geht die Idee, die Formatierung separat zu definieren; am LATEX-Beispiel würde das bedeuten, dass man etwa in einer separaten Style-Datei die Bedeutung von `center` beliebig festlegen kann – und dann sogar die gleiche, strukturierte Information verschiedenartig, also mit verschiedenen Styles, anzeigen kann.

All dies findet sich im Bereich der Auszeichnungssprachen für das Web klar umgesetzt wieder.

## 2.2 SGML

Die *Standard Generalized Markup Language* (SGML) ist die breite Grundlage der verschiedenen Auszeichnungssprachen. SGML wurde 1986 im ISO-Standard 8870:1986 und später in einer DIN definiert und ist somit älter als HTML. Die Anfänge von SGML gehen bis in die sechziger Jahre des letzten Jahrhunderts zurück.

SGML selbst hat direkt nur eine ausgesprochen kleine Bedeutung. Wichtig sind vorallem eine Anwendung von SGML, nämlich HTML, und eine bewusste Reduzierung von SGML auf einen praktikablen Umfang: XML.

## 2.3 HTML

HTML ist genauso alt wie „das Web“, es wurde von Tim Berners-Lee 1989 vorgestellt (Abbildung 2.1).

Die „HTML-Normierung“ wird durch das World Wide Web Consortium (W3C) wahrgenommen [web].

Eine ausgesprochen praktische und ergiebige Quelle zu HTML ist das auf Stefan Münz zurückgehende Selfhtml [web]. Hier werden nur einige Grundprinzipien, insbesondere der Formularaufbau, vorgestellt.

### 2.3.1 Versionen von HTML

Die erste Version von HTML wurde im November 1992 offiziell vorgestellt (Abbildung 2.1). 1999 wurde die Version 4.01 veröffentlicht, welche bis heute aktuell ist, denn die Fortentwicklungen finden in XHTML statt (vgl. 2.4.7).

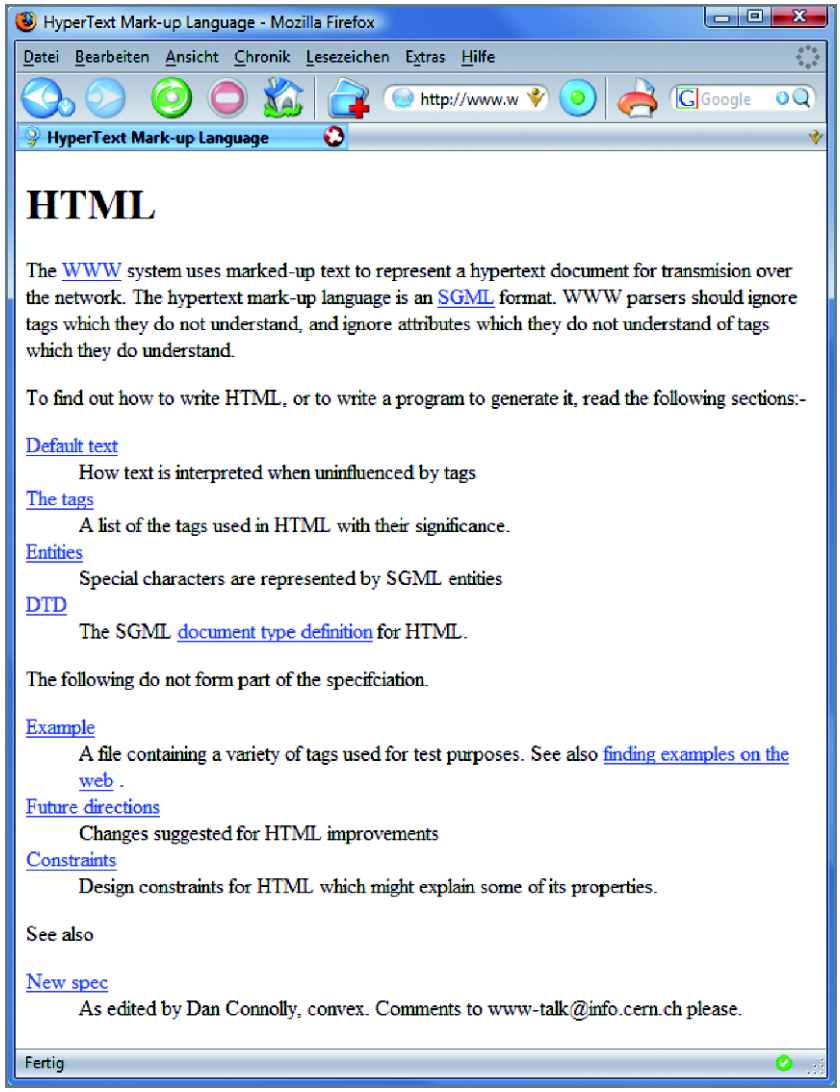



**Abbildung 2.1:**
HTML, Stand November 1992

### 2.3.2 Prinzip von HTML

Ein HTML-Dokument in der Version 4.01 hat folgenden prinzipiellen Aufbau:

```
Dokumententyp
<HTML>
   <HEAD>
      Header-Informationen
   </HEAD>
   <BODY>
      Body-Informationen
   </BODY>
</HTML>
```

HTML-Tags (Steuerelemente) erkennt man an der Schreibweise mit der spitzen Klammer `<...>`. Wie bei den Auszeichnungssprachen beschrieben, erkennt man die Struktur mit öffnenden und schließenden Tags, etwa `<BODY>` und `</BODY>`. HTML ist dabei (im Gegensatz zu XML und XHTML) nicht

case-sensitiv; man könnte auch <BODY> mit </body> schließen.

Ein HTML-Tag besteht aus dem eigentlichen Tag und einer Reihe von Attributen zu diesem Tag: diese Atribute setzen sich aus Namen-Wert-Zuweisungen der Art

```
Dokumententyp
<TAG attribut="wert" ...>
```

zusammen, wobei der `wert` stets in ‘‘ gesetzt wird.

HTML ist wie eingangs beschrieben eine Auszeichnungssprache – es wird also „geschrieben“ und nicht „programmiert“, auch wenn hier die Bezeichnungsweise nicht ganz konsequent angewendet wird. Im Sourcecode in HTML sind Kommentare möglich und auch erwünscht; sie werden durch

```
Dokumententyp
<!-- HTML-Kommentar -->
```

beschrieben.

#### 2.3.2.1 HTML-Dokumententyp und HTML-Varianten

Vor dem eigentlichen HTML-Dokument steht in den Versionen 4 und 4.01 von HTML eine *Dokumententyp-Deklaration*. Durch diese werden drei Varianten von HTML definiert:

- Strict: hier wird nur ein Kernbestandteil von HTML ermöglicht, viele Tags zur Formatierung wie etwa <FONT> fehlen; diese Formatierungen sind dann sauber in CSS (vgl. 2.5) auszulagern.
- Transitional: erlaubt gegenüber Strict zusätzliche Formatierungen älterer HTML-Versionen; derartig erklärte Seiten sind stärker abwärtskompatibel, es werden also auch ältere HTML-Sites vom Browser korrekt angezeigt;
- Frameset: hier sind ergänzend zu Transitional noch HTML-Frames möglich.

Nur die Variante Strict bedingt eine saubere Trennung der Formatierung vom strukturierten Inhalt; in XHTML hat deshalb nur diese eine Zukunft gefunden.

Die genaue Syntax der Dokumententyp-Definition lautet etwa für den Typ Strict:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
          "http://www.w3.org/TR/html4/strict.dtd">
```

Hier wird klar, wie die eigentlichen Typen definiert sind: Über eine öffentliche Document Type Definition (DTD, siehe 2.4.4) sind die Varianten sauber erklärt.

#### 2.3.2.2 HTML-Header

Im *HTML-Header* werden Metainformationen für das Dokument hinterlegt. Zu den wichtigsten gehören:

- Über das <TITLE>-Tag wird der Dokumententitel definiert, den üblicherweise der Browser in der Fenster-Titelzeile anzeigt.
- Metainformationen werden ebenfalls im Head erklärt; hierzu können Informationen zum Autor gehören, aber auch Keywords für Suchmaschinen, eine Kurzbeschreibung und mehr.
- Ein externes Style-Sheet (vgl. 2.5) wird im Head über

```
<link rel="stylesheet" type="text/css" href= "datei.css">
```

eingebunden.

- Genauso wie das CSS wird auch ein Favicon im Head festgelegt, jene kleinen Icons, welche modernere Browser zur Seite im Adressenfeld und im Fenstertab anzeigen. Dies geschieht über

  ```
  <link rel="shortcut icon" href="datei.ico">
  ```

- JavaScript-Methoden und zahlreiche weitere JavaScript-Elemente, etwa auch die Erzeugung eines Ajax-Objektes `XMLHttpRequest` findet sich im HTML-Header in `<SCRIPT>`-Tags (vgl. Kapitel 15 und 16).

#### 2.3.2.3 HTML-Body

Der *Body* enthält den eigentlichen Seiteninhalt. Für die Formatierung in HTML 4.01 sind dabei viele Möglichkeiten gegeben, die hier nicht alle aufgeführt werden können. Zu den wichtigsten zählen:

- Überschriften von `<H1>` bis `<H4>`:
- zentriert über `<CENTER>`; Aufzählungslisten nummeriert mittels `<OL>` und unnummeriert mittels `<UL>`; der einzelne Eintrag ist dann durch `<LI>`-Tags begrenzt.
- Hyperlinks auf externe Seiten:

  ```
  <a href="http://adresse" target="...">Linktext</a>
  ```

  Der Target steuert dabei die Anzeige in einem neuen Fenster (`_blank`) und mehr.

### 2.3.3 Formulare in HTML

*Formulare in HTML* werden in diesem Buch häufig eingesetzt, weshalb sie hier ausführlicher erläutert werden.

Die Grundsyntax eines HTML-Formulars ist

```
<FORM method="..." action="..." name="...">
   <!-- Formular-Elemente -->
</FORM>
```

Dabei bezeichnet das Attribut `method` die HTTP-Übertragungsmethode nach Abschnitt 1.6.5.1, also entweder GET oder POST. `action` legt das Ziel des Formulars fest, also das Script, an welches das ausgefüllte Formular übertragen wird. Ein weiteres Attribut ist `name`, der Name des Formulars; dies ist besonders dann wichtig, wenn eine HTML-Seite mehrere Formulare enthält: Eine Seite kann beliebig viele Formulare hintereinander enthalten, geschachtelte Formulare sind nicht möglich.

Weitere Attribute für das `<FORM>`-Tag sind:

- `accept-charset` zur Festlegung, welche Zeichensätze zugelassen werden;
- `onsubmit` zur Verbindung des Submit-Buttons mit einer JavaScript-Funktion (vgl. Kapitel 15).

#### 2.3.3.1 Eingabeelemente in HTML-Formularen

Ein einzeiliges *Eingabefeld* in einem HTML-Formular wird durch das Tag

```
<INPUT ...>
```

erzeugt. Dabei sind vielfältige Attribute möglich, zu ihnen zählen:

- `name` für den Namen des Eingabeelementes; hierfür wird inzwischen bei XHTML das Attribut `id` vorgesehen, allerdings verstehen einige der in diesem Buch vorgestellten CGI-Module für Scriptsprachen nur das Attribut `name` und nicht *id*.
- `align` für die Ausrichtung des Feldes, allerdings nicht im Typ Strict;
- `type` legt die Art des Eingabefeldes fest:
  - `text` für ein Textfeld;
  - ein `password`-Feld zeigt für jedes eingegebene Zeichen immer `*` an;
  - `checkbox` für eine Checkbox;
  - `radio` für einen Radiobutton;
  - `submit` für einen Absendeknopf;
  - `reset` für ein Eingabeelement zum Zurücksetzen des Formulars;
  - `hidden` für ein verstecktes Feld: die Werte werden übertragen, ohne dass das Feld angezeigt wird; dies ist beispielsweise für mehrere aufeinander folgende Formulare wichtig, damit das letzte die Eingaben der vorherigen kennt;
  - `button` für einen Eingabebutton;
  - `image` für das Einbinden einer Grafik (mittels `src`-Attribut);
  - `file` für eine Dateieingabe.
- `readonly` für ein unveränderliches, also nicht editierbares Eingabefeld;
- `maxlength` zum Festlegen der maximalen Länge der Eingabe;
- `size` legt die angezeigte Größe des Eingabefeldes fest;
- `value` zur Vorbelegung eines Eingabefelds; dies wird auch in Verbindung mit `readonly` eingesetzt.

Mehrzeilige Eingabebereiche werden ähnlich mittels

```
<TEXTAREA ...>
```

definiert.

### 2.3.4 HTML Tidy und HTML Validator

Ein sauberer HTML-Code ist für die Anzeige im Browser wünschenswert, aber nicht notwendig; die meisten Browser sind sogar sehr fehlertolerant und zeigen auch inkorrekte HTML-Dokumente vernünftig an, etwa, wenn das schließende `</HTML>`-Tag fehlt. Dennoch sollte derartiges unbedingt vermieden werden. Um den eigenen – oder fremden – (X)HTML-Code zu überprüfen, stellt das W3C einen webbasierten Dienst zum Testen der Seite bereit [web]. Den strengen Test bestehen die wenigsten Seiten, die aufgetreteten Fehler werden im Ergebnis genau aufgelistet (Abbildung 2.2). Korrekte Seiten können mit einem entsprechenden Logo gekennzeichnet werden.

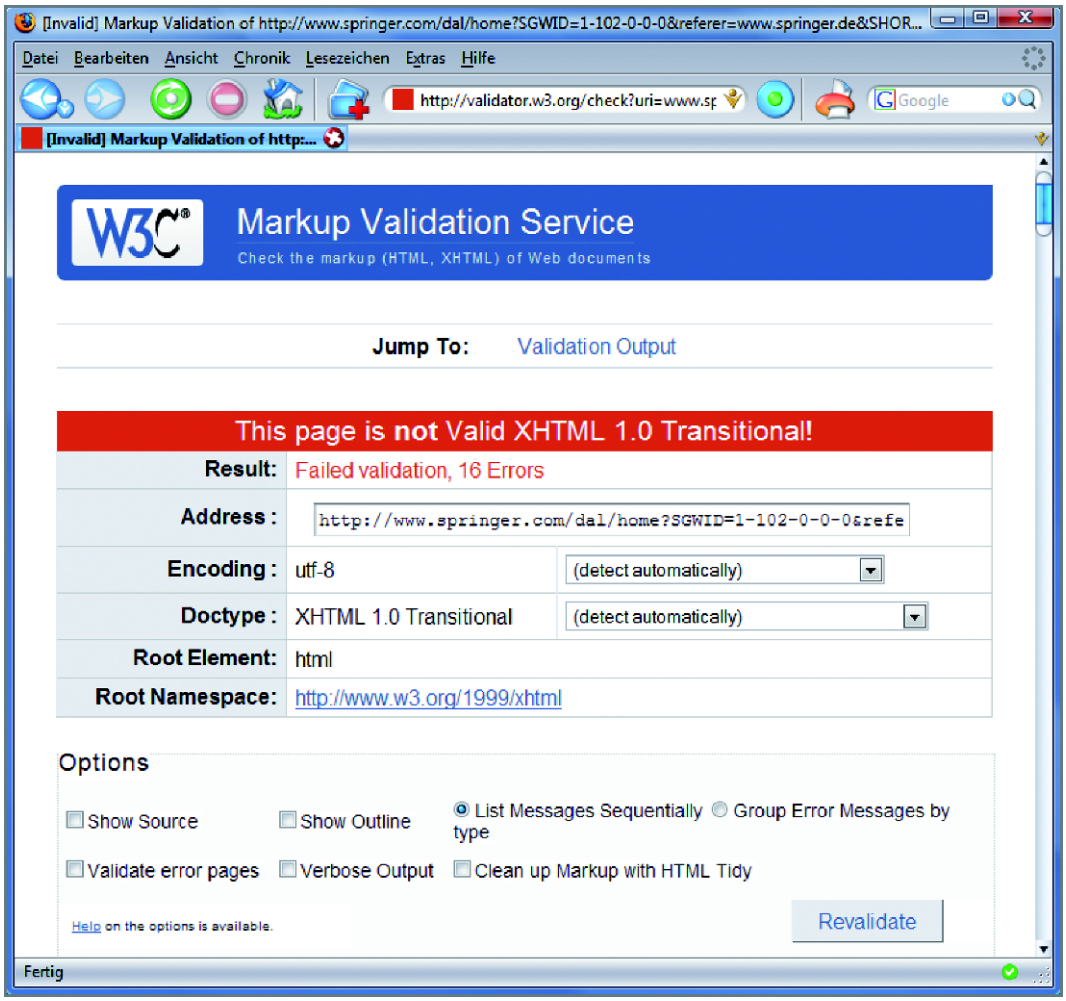



**Abbildung 2.2:** HTML-Validierung von www.springer.de: 16 Fehler bei strenger Prüfung, aber keine gravierenden Mängel

HTML Tidy [web], ein von Dave Raggett geschriebenes Programm für Unix/Linux und Windows, erleichtert das Schreiben von HTML-Code wesentlich, da es einen Test ähnlich zu HTML-Validate, nun aber auf dem lokalen Rechner, erlaubt (Abbildung 2.3), aber auch selbständig eine HTML-Datei reparieren kann. Es werden zahlreiche Parameter und wählbare Optionen angeboten.

Die in 8.1 vorgestellte Entwicklungsumgebung Eclipse kann durch Plugins auch für (X)HTML erweitert werden; diese Plugins nutzen häufig auch Tidy.

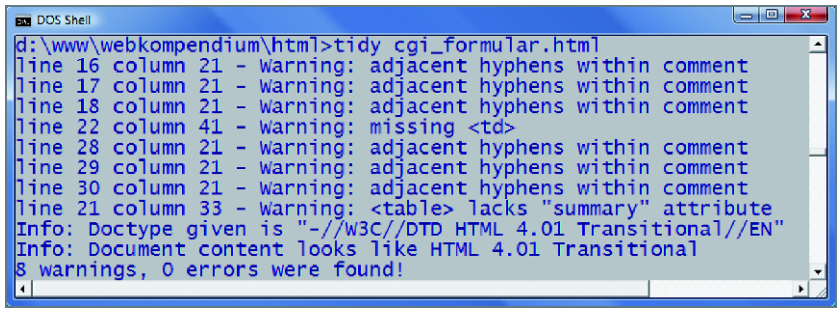



**Abbildung 2.3:** HTML-Tidy im Einsatz

### 2.3.5 Zeichencodierung im Web

Das Problem der Anzeige – oder auch nur der Übermittlung – von Zeichen, welche über die ersten 128 ASCII-Symbole hinausgehen, kann im „weltweiten Netz“ schnell zu wesentlichen Problemen führen.

HTML bietet hierfür die Umschreibung aller Symbole jenseits des einfachen ASCII an, etwa

```
&auml;  für ä
&szlig; für ß
```

definiert; so kann auch das €-Symbol erzeugt werden durch

```
&euro;
```

Allerdings ist dieses Verfahren auf HTML beschränkt und umständlich, weshalb inzwischen der offizielle Standard für die Codierung im Web UTF-8 ist: das 8-bit Unicode Transformation Format. Dieses Format, eine Erweiterung des klassischen Unicodes, verwendet eine Zeichenkette variabler Länge bis hin zu vier Byte zur Codierung von Zeichen. So können mit UTF-8 maximal 1.114.112 Zeichen dargestellt werden. Bei korrekter Anwendung von UTF-8 im Browser, im Webdokument und im Webserver kann dann beispielsweise das „ä“ direkt im HTML-Formular verwendet werden. Während man im Browser nur auf eine aktuelle Version der Software zurückgreifen muss, ist beim Dokument und beim Server mehr zu beachten:

- Im HTML-Header ist die Zusatzinformation `charset=utf-8` zu ergänzen:

  ```
  <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8">
  ```

- Der Webserver muss so konfiguriert werden, dass er UTF-8-Dokumente ausliefert, also diesen Zeichensatz im HTTP-Header ausweist. Im Falle des Apache bedeutet dies die Direktive der Art

  ```
  AddType "text/html; charset=UTF-8" html
  ```

## 2.4 XML

Wir haben HTML als eine gezielte Vereinfachung des komplexen SGML-Standards kennengelernt. *XML,* die Extensible Markup Language, ist ein im Februar 1998 verabschiedeter Standard, der zwischen beiden Wegen liegt: deutlich flexibler, insbesondere auch erweiterbar gegenüber HTML wird die unüberschaubare Komplexität von SGML bewusst vermieden. Die ursprüngliche Zielsetzung von XML war die Schaffung eines mächtigen, flexiblen Datenformats gerade im Hinblick auf den Austausch von Daten. Inzwischen wird XML auch in weiterem Kontext eingesetzt.

Dabei ist der reine Begriff „XML“ durchaus irreführend: Dahinter soll neben dem reinen XML-Format auch die ganze Familie von zugehörigen Standards wie XSL, XML Schema, DTD und viele andere verstanden werden.

Die XML-Standards werden, genauso wie die von HTML, durch das World Wide Web Consortium W3C [web] verwaltet.

### 2.4.1 Von HTML zu XML

XML erweitert den HTML-Standard zunächst in drei zentralen Punkten:

- HTML ist ein fest definierter Standard – XML erlaubt es, eigene Tags beliebig hinzuzufügen. Es ist damit keine „Menge von Anweisungen“, sondern ein offener Standard.
- HTML beschreibt die Struktur und die Formatierung für Webdokumente – wobei der Strict-Standard mit CSS hier bereits die Abspaltung der Formation erzwingt –, XML kann beliebige Daten strukturieren.
- XML erlaubt die Validierung von Daten: Anhand einer formalen Beschreibung der Datenstruktur kann überprüft werden, ob ein Datensatz formal korrekt (valid) ist.

XML ist damit aber keinesfalls nur eine „Erweiterung“ oder der „Nachfolger“ von HTML – das ist XHTML. XML ist viel mehr als HTML, ein Formalismus zur Beschreibung von Daten.

Wesentlich für XML ist gegenüber HTML die strenge Trennung der Daten (der eigentlichen xml-Datei) von der Darstellung/Formatierung (etwa in XSL) und der Beschreibung der Datenstruktur (DTD oder XML Schema). HTML 4 in der Variante Strict bildet hier den Übergang von HTML zur sauberen Trennung von XML.

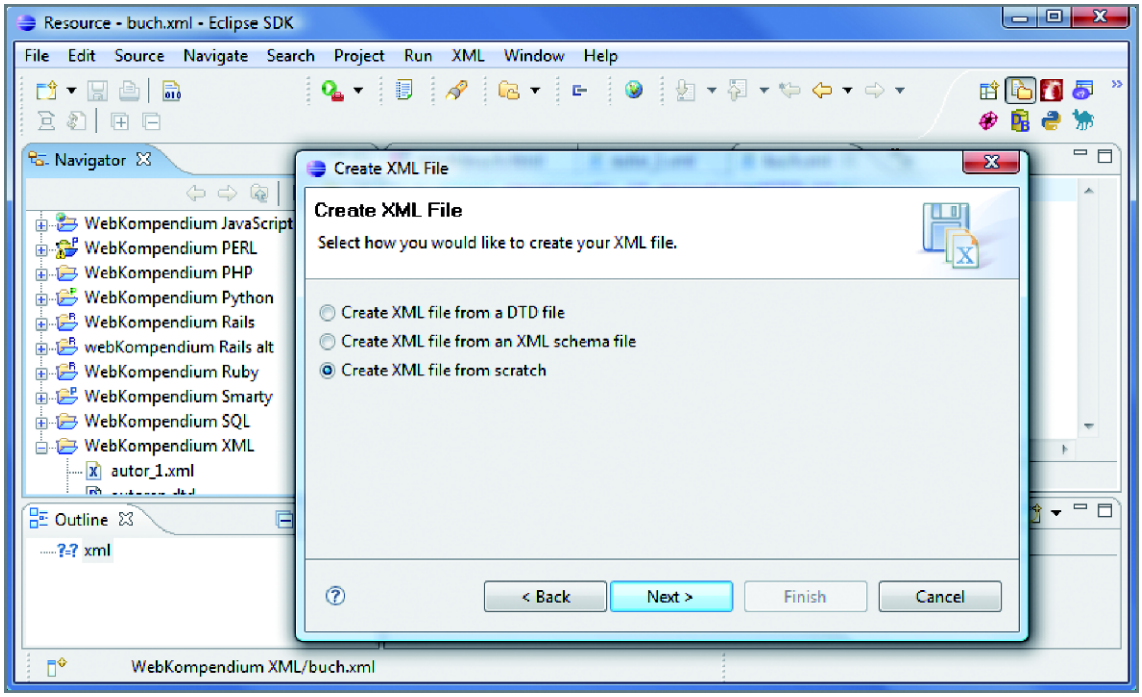



**Abbildung 2.4:**
Eclipse mit XML-Elementen des Webtools-Plugins

### 2.4.2 Editoren für XML

Für XML ist eine Vielzahl freier und kommerzieller Editoren verfügbar, welche insbesondere eine integrierte Überprüfung der XML-Daten, die Validierung, bieten. Ein gutes, häufig ausreichendes Werkzeug beinhaltet das Webtools-Plugin von Eclipse (vgl. 8.1), wie in Abbildung 2.4 zu sehen.

### 2.4.3 XML Syntax

Ein XML-Dokument beginnt mit dem Prolog

```
<?xml version="1.0" encoding="UTF-8"?>
```

welcher die Version und die Zeichencodierung festlegt. Dabei ist das Versions-Attribut zwingend.

Abschließend können beliebige Tags folgen, etwa

```
<buch>
</buch>
```

Dabei ist `<buch>` ein frei definiertes Tag.

Hierbei sind schon drei Besonderheiten gegenüber HTML zu bemerken:

- XML-Tags – außer dem öffnenden Prolog `<?xml ...?>` – müssen stets wieder geschlossen werden; die Schreibweise von HTML für eine horizontale Linie, `<hr>`, ist in XML falsch, es muss `<hr></hr>` geschrieben werden, was durch `<hr/>` abgekürzt werden kann.
- XML-Tags sind case-sensitiv; das Tag `<buch>` kann nicht durch `</BUCH>` geschlossen werden.
- Kommentare in XML werden wie in HTML mittels

  ```
  <!-- Kommentar -->
  ```

  geschrieben, allerdings ist `<!---->` in HTML ein korrekter Kommentar, in XML ist dies nicht möglich.

Nun ergänzen wir weitere Tags, um unser erstes Datenelement zu definieren (Abbildung 2.5).

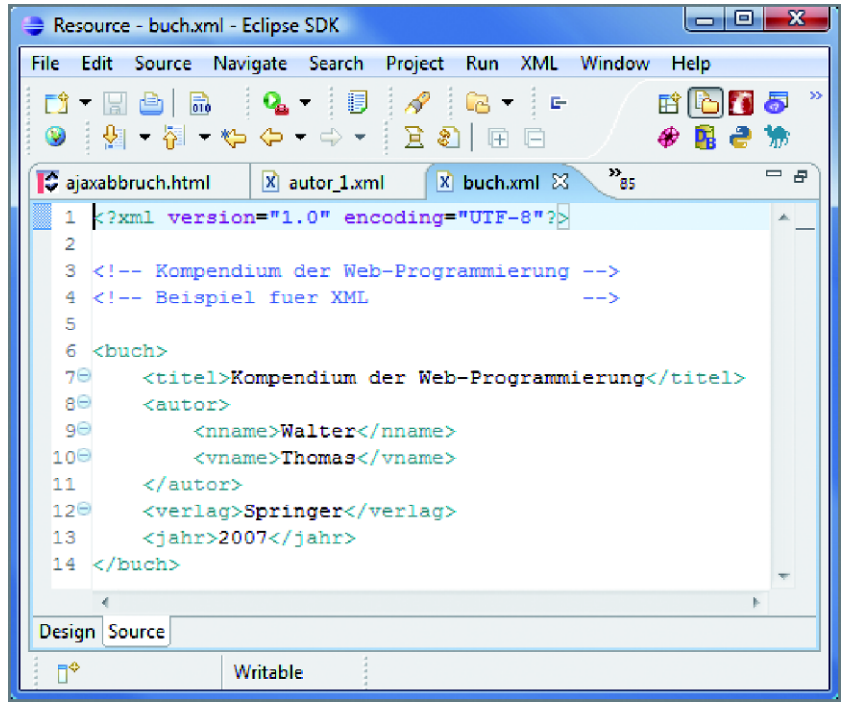



**Abbildung 2.5:** XML-Datensatz

Diese Datei kann in modernen Browsern angezeigt werden (Abbildung 2.6), was uns zu einer neuen Einsicht führt:

**Eine XML-Datei beschreibt immer eine Baumstruktur.**

Als nächstes soll die formale Korrektheit dieses Datensatzes überprüft werden.

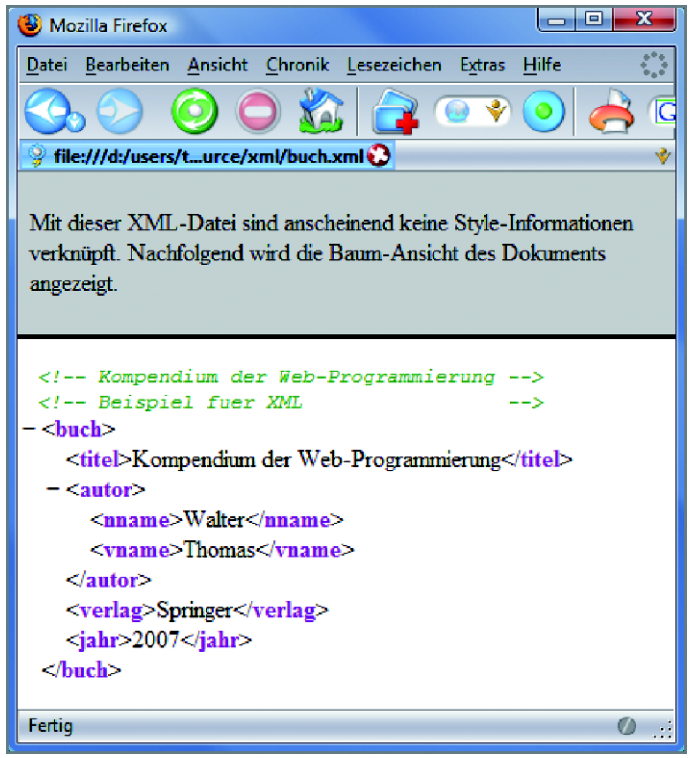



**Abbildung 2.6:** XML-Datensatz `buch.xml` nach Abbildung 2.5 im Browser

### 2.4.4 Document Type Definition

Eine Möglichkeit der formalen Festlegung einer XML-Struktur ist die *Document Type Definition* (DTD). Hier wird der formale Aufbau korrekter XML-Datenstrukturen festgelegt.

Eine DTD kann sowohl innerhalb der eigentlichen XML-Datei als auch in einer separaten Datei erfolgen. Eine interne DTD erfolgt durch die Syntax

```
<!DOCTYPE datentyp [ beschreibung ]>
```

Wichtiger ist das Einbinden einer externen DTD-Datei durch

```
<!DOCTYPE datentyp SYSTEM "dateiname.dtd">
```

Das Attribut `SYSTEM` bezeichnet dabei eine lokale DTD-Datei, es kann daneben auch `PUBLIC` für öffentliche, über das Web verfügbare DTD-Dateien verwendet werden, etwa zur Definition von XHTML-Dateien (vgl. 2.4.7).

#### 2.4.4.1 DTD Syntax

Die DTD erklärt eine Datenstruktur. Die Grundynytax hierfür lautet

```
<!ELEMENT datentyp(beschreibung)>
```

Als Beschreibung kann rekursiv ein Datentyp durch einen anderen erklärt werden – im Beispiel besteht ein Buch aus Titel, Autor (welcher wieder eine Struktur hat), Verlag und Jahr – oder er besteht aus den eigentlichen Daten. Diese werden durch `CDATA` und `PCDATA` gekennzeichnet: beide werden von XML übernommen, aber nur die zweite Gruppe wird vom XML wiederum verarbeitet, also auf Korrektheit (etwa verwendeter Zeichensatz) überprüft.

Abbildung 2.7 zeigt eine solche einfache DTD-Datei für den Datentyp `buch` nach dem Beispiel aus Abbildung 2.5.

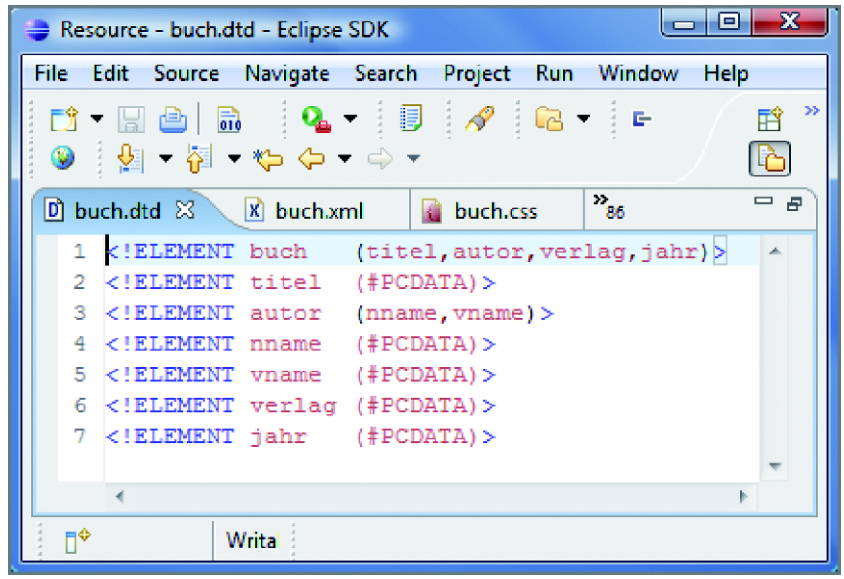



**Abbildung 2.7:**
DTD für die Struktur `buch`

Natürlich sind sehr zahlreiche weitere Möglichkeiten für die DTD gegeben, insbesondere Quantoren: Eine Struktur `bibliothek` kann aus beliebig vielen Elementen `buch` bestehen durch:

```
<!ELEMENT bibliothek(buch)+>
```

### 2.4.5 Wohlgeformt und gültig

Eine XML-Datei kann nun auf formale Korrektheit maschinell überprüft werden:

- *Wohlgeformt* (well-formed) ist XML, wenn es formal korrekt ist: richtiger Zeichensatz, alle Tags werden richtig geschlossen, Baumstruktur ist korrekt. Hierfür ist keine DTD notwendig.
- *Gültig* (valid) ist eine XML-Datei, wenn die DTD erfüllt ist.

Der XML-Parser überprüft well-formed und valid und überführt die XML-Daten in die durch sie präsentierte Baumstruktur; diese kann später etwa in DOM übernommen werden (vgl. 15.5).

### 2.4.6 Ergänzung der Formatierung

Genauso wie durch eine DTD die formale Struktur der XML-Daten erklärt werden kann, kann auch die XML-Datei mit einer Formatierung verbunden werden, etwa in Form einer CSS-Datei nach Abschnitt 2.5. Diese Verknüpfung erfolgt in einer zweiten Prolog-Zeile durch

```
<?xml-stylesheet href=css-Datei type="text/css"?>
```

Exemplarisch wird hier die CSS-Datei nach Abbildung 2.8 eingebunden.

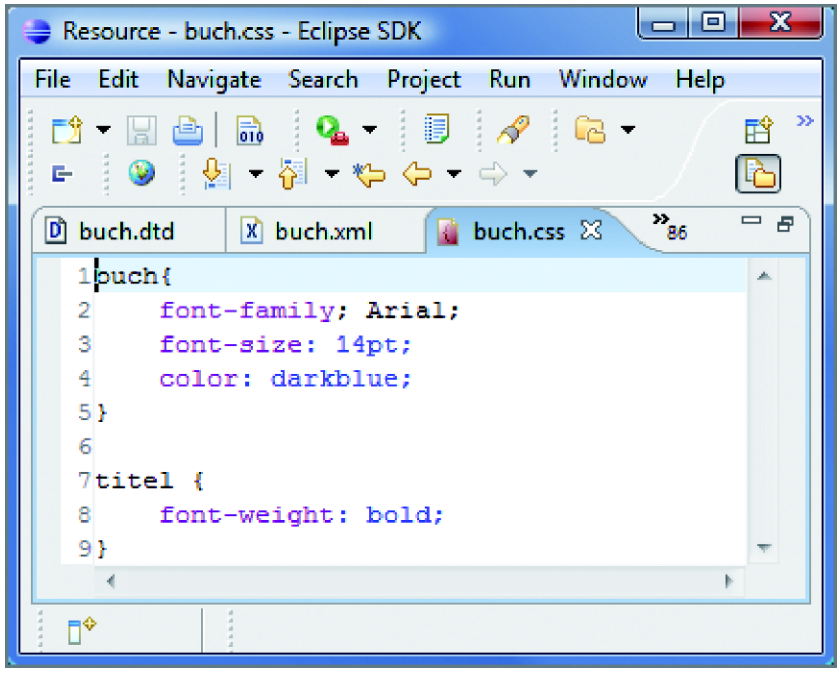



**Abbildung 2.8:**
Die CSS-Datei `buch.css`

Die fertige XML-Datei (Abbildung 2.9 verwendet nun sowohl die DTD- als auch die CSS-Datei; im Browser erfolgt die Anzeige nach Abbildung 2.10

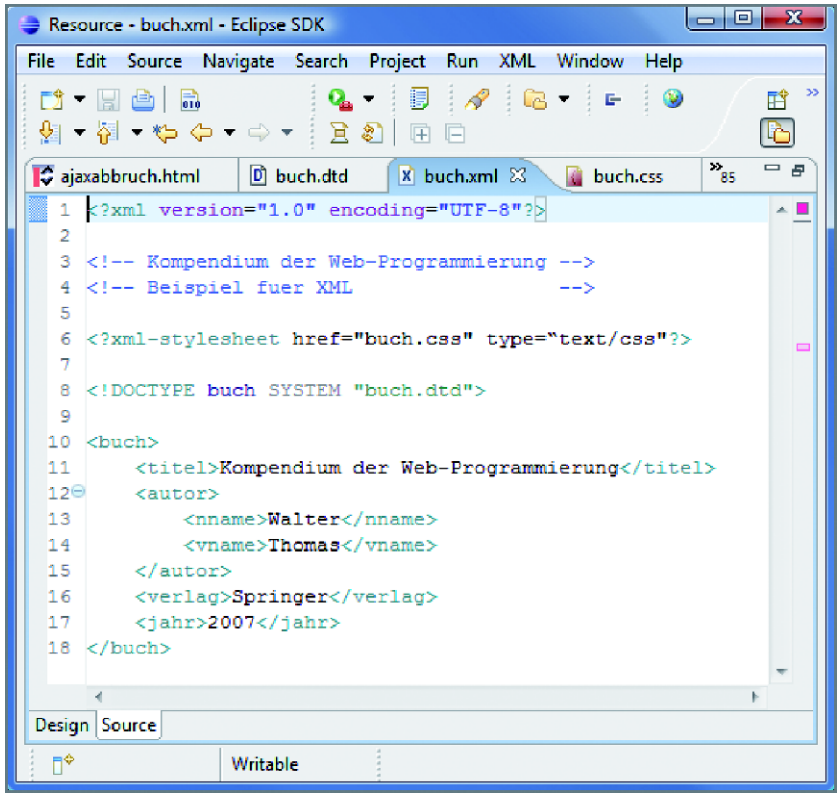



**Abbildung 2.9:**
Die fertige XML-Datei mit DTD und CSS `buch.xml`

**Abbildung 2.10:** Anzeige von `buch.xml` nach Abbildung 2.9 im Browser

### 2.4.7 XHTML

XHTML ist die konsequente Weiterentwicklung von HTML 4.01: Es ist „XML-konformes HTML“.

Konkret bedeutet dies, dass wie bei XML üblich die Korrektheit über eine DTD-Datei – hier nun eine öffentliche – definiert und überprüfbar ist, dass das Dokument well-formed ist, also beispielsweise alle Tags geschlossen werden, und dass die Formatierung sauber abgespaltet ist.

Eine XHTML-Datei beginnt entsprechend der XML-Syntax dann folgendermaßen:

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="de"
    xml:lang="de">
```

Inzwischen unterstützen die meisten Browser XHTML; Netscape seit der Version 6, der InternetExplorer seit der Version 5 und von Firefox alle Versionen.

Das Perl-Modul CGI.pm von Lincoln Stein (siehe 10.6) erzeugt standardmäßig direkt XHTML-Code.

## 2.5 Cascading Stylesheets – CSS: Format fürs Web

Eine inzwischen zentrale Technik im Web sind *Cascading Stylesheets* (CSS). Hierunter verbirgt sich eine eigene Sprache nur für die Formatierung von HTML und XHTML, welche zahlreiche Möglichkeiten bietet, insbesondere auch solche, die über die des HTML hinausgehen. Der zentrale Vorteil von CSS ist aber die Möglichkeit, die Inhalte im HTML-Dokument von der Formatierung in CSS zu trennen.

Die Formatierungsinformationen im CSS können dabei in einer CSS-Anweisung zu einem HTML-Attribut, in einem CSS-Block im HTML-Dokument oder auch extern in einer separaten CSS-Datei aufgeführt werden.

Mit CSS ist damit eine wichtige Trennung von Inhalt (Content) und Formatierung möglich. Werden zentrale CSS-Dateien verwendet, können diese Informationen für eine komplexe Site gleichzeitig formuliert werden, was eine einfache Konfiguration ermöglicht. Im Web [web] sind zahlreiche gute und umfassende Beispiele für CSS zu finden, daneben steht umfassende Literatur zu diesem Thema zur Verfügung, etwa [Häß03].

### 2.5.1 CSS: Versionen und Browser

Die erste Version von CSS, Version 1.0. stammt von 1996. Ihr folgte 1998 die Version 2.0, welche bis heute aktuell ist.

Die CSS-Technik ist besonders stark browserabhängig, insbesondere ältere Browser unterstützen CSS nicht oder nur teilweise; Tabelle 2.1 gibt eine grobe Übersicht. Die aktuellen Browsertypen unterstützen CSS sehr umfassend.

| Browser | Implementierung |
|---|---|
| Netscape 4.x | CSS 1 und CSS 2 teilweise |
| Netscape 6.x | CSS 1 und CSS 2 fast vollständig |
| Netscape 7.x | CSS 1 und CSS 2 fast vollständig |
| Opera 5.x | CSS 1 und CSS 2 fast vollständig |
| Opera 6.x | CSS 1 und CSS 2 fast vollständig |
| Internet Explorer 4.x | CSS 1 und CSS 2 teilweise |
| Internet Explorer 5.x | CSS 1 und CSS 2 fast vollständig |
| Internet Explorer 6.x | CSS 1 und CSS 2 fast vollständig |

**Tabelle 2.1:** CSS und ältere Browser

### 2.5.2 Platzierung von CSS-Formatinformationen

Es gibt drei Möglichkeiten, CSS-Formatinformationen aufzuführen:

- in einem eigenen Block in der HTML-Datei;
- als Attribute bei einem HTML-Element;
- in einer externen CSS-Datei.

Im ersten Fall wird der CSS-Block im HTML-Header aufgeführt. Die Syntax hierfür lautet:

```
<style type="text/css">
   ...
</style>
```

Im zweiten Fall wird ein einzelnes HTML-Tag mit einer CSS-Information als Attribut versehen, etwa:

```
<h1 style="...">...</h1>
```

Die größte Bedeutung kommt dem dritten Weg, der externen CSS-Datei zu. Diese hat typischerweise die Dateiendung `css` und wird ebenfalls im HTML-Header eingebunden:

```
<head>
  <title>...</title>

  <link rel="stylesheet" type="text/css" href= "datei.css">

</head>
```

Wichtig ist, dass diese drei Wege stets auch kombiniert werden können: Die aus der CSS-Datei importierten Formatierungen können im HTML-Header durch den style-Block für das ganze Dokument überschrieben werden und dann nochmals für ein einzelnes Element geändert werden.

### 2.5.3 CSS: Versionen und Browser

In einer HTML-Datei ist es möglich, je nach Zweck der Verwendung eine andere CSS-Datei zu laden. Dies geschieht durch das HTML-Attribut `media` im `link`-Tag. Folgende Typen sind vorgesehen:

- `all` für alle Fälle;
- `handheld` für die Ausgabe in einem Mobile Device;
- `print` für die Druckerausgabe;
- `projection` für eine Beamer-Projektion;
- `screen` für die Bildschirmausgabe.

Durch folgenden Header werden verschiedene CSS-Dateien eingebunden:

```
<link rel="stylesheet" media="print" type="text/css" href="style_print.css">
<link rel="stylesheet" media="screen" type="text/css" href="style.css">
```

Eine der zahllosen Möglichkeiten hiervon besteht etwa darin, einen Ausdruck einer Seite zu unterdrücken – oder zumindest zu erschweren – indem für den Druck die Ausgabe deaktiviert wird; dies geschieht durch ein spezielles Stylesheet mit der Anweisung:

```
body {
display:none;
}
```

Eine andere, häufiger verwendete Möglichkeit ist es hier, im Ausdruck die Navigationsinformationen zu unterdrücken.

### 2.5.4 Grundsyntax von CSS

CSS ist im Kern eine einfache Anweisungssprache. Zunächst bietet diese auch Kommentare. Diese werden wie in Java oder C durch

```
/*
    Kommentar
 */
```

geschrieben.

Die Grundsyntax einer CSS-Anweisung ist die Belegung von Eigenschaften von Elementen mit einem Wert. Die Elemente werden durch einen Selektor ausgewählt. Danach können Eigenschaften dieser Elemente Werte zugewiesen werden. Prinzipiell lautet die Syntax hierfür:

```
Selektor { Eigenschaft:Wert; }
```

Die Frage, inwieweit CSS dabei Groß- und Kleinschreibung unterscheidet, ist unertschiedlich zu beantworten:

- CSS für HTML ist unabhängig von der Groß- und Kleinschreibung, wie HTML selbst auch;
- CSS für XML und XHTML verhält es sich genau umgekehrt, hier wird die Schreibweise unterschieden.

### 2.5.5 Beispiel für CSS

Ein reales Beispiel soll die Anwendung von CSS zeigen: das CSS, welches in allen Beispielen in diesem Buch und zumindest in der ersten Version der Web-Site `http://www.webkompendium.de` verwendet wird. `webkompendium.css` (Abbildung 2.11).

Diese CSS-Datei definiert im wesentlichen:

- die Hintergundfarbe Orange mit dem Hex-Code `#FFA500`;
- die Schriftfarbe `#002A5A`, ein dunkles Blau;
- eine Formatierung von Tabellen mit einem grauen Hintergrund.

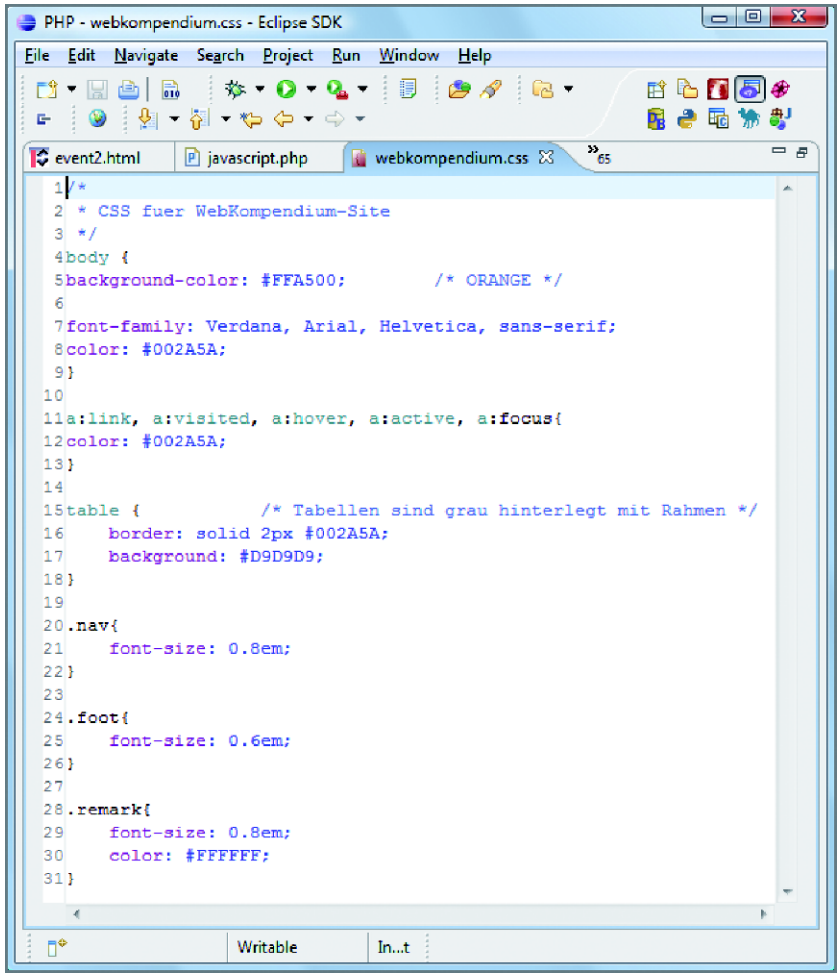



**Abbildung 2.11:** Das Style-Sheet `webkompendium.css`

Dieses Stylesheet wird über den beschriebenen Weg im HTML-Header eingebunden, im Browser wird die entsprechende Formatierung angezeigt (Abbildungen 2.12 und 2.13).

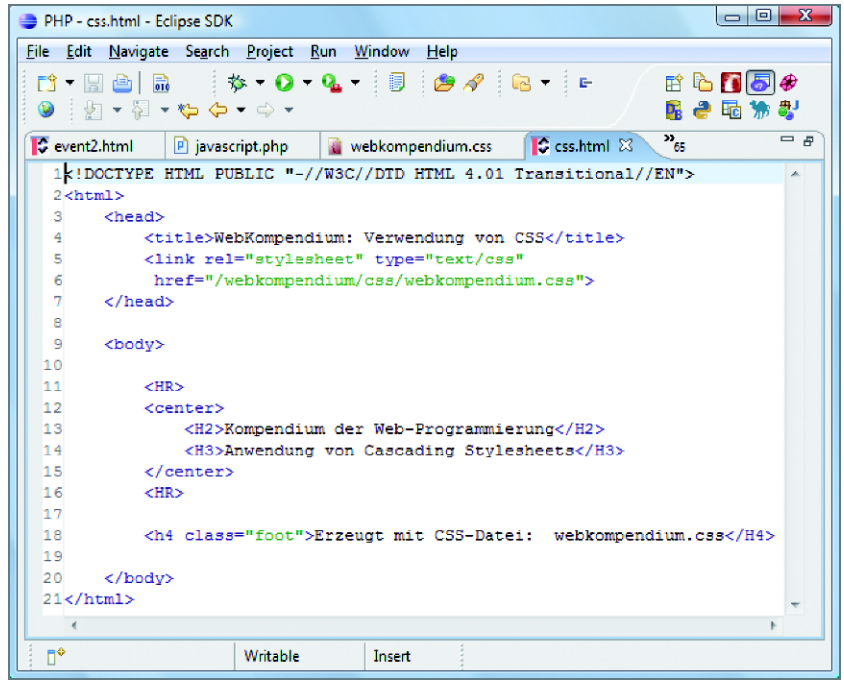



**Abbildung 2.12:** HTML-Datei, welche das Stylesheet verwendet: `css.html`

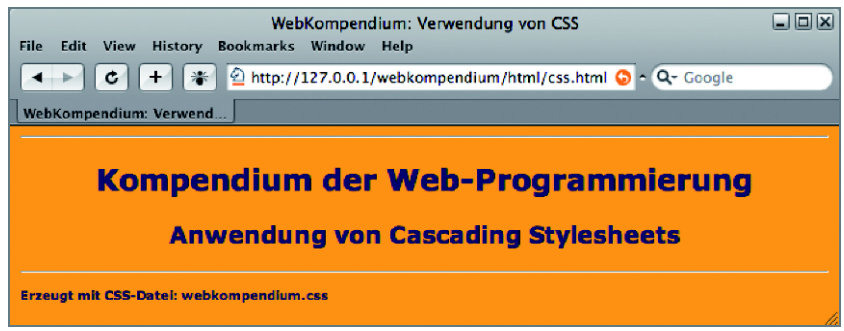



**Abbildung 2.13:** Ausführen von `css.html`

## 2.6 Gestaltung barrierefreier Webseiten

Bei der Gestaltung eines modernen Webangebotes ist neben den Fragen des Designs auch die Benutzbarkeit der Site heute ein wesentliches Kriterium. Dabei soll eine Siete insbesondere keine unnötigen Hürden aufbauen, also *barrierefrei* sein.

Als „barrierefrei“ werden Anwendungen bezeichnet, die auch von Menschen mit Einschränkungen, sei es eine körperliche Behinderung oder geringere technische Möglichkeiten, genutzt werden können. Als körperliche Behinderungen spielen vor allem Sehbehinderungen bis zur Blindheit, Taubheit oder Störungen der Feinmotorik eine Rolle. Zu den technischen Einschränkungen zählen neben veralteter Hardware und langsamen Internetanbindungen auch mobile Endgeräte, wie PDAs oder Handys, die in der Regel sehr kleine Displays und nur einfache Webbrowser haben, oder Textbrowser (etwa der Lynx-Browser nach 5.2.6), die bei einer Terminalverbindung die einzige Möglichkeit sind, Web-Sites vernünftig zu betrachten.

Um die Zugänglichkeit von Web-Sites für alle Zielgruppen zu fördern, wurde die Web Accessibility Initiative (WAI) [web] gegründet. Eines ihrer Hauptprojekte sind die Web Content Accessibility Guidelines, die seit 1999 in der Version 1.0 (WCAG1) bestehen. In Deutschland sind diese Regelungen in die Verordnung „Barrierefreie Informationstechnik Verordnung“ (BITV) eingegangen [web]. Diese Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz ist seit Juli 2002 in Kraft.

Es handelt sich dabei um 14 Richtlinien, deren Unterpunkte in den WCAG1 in drei, in der deutschen Verordnung in zwei Prioritätsstufen eingeteilt wurden. Bei den drei Stufen der WCAG1 unterscheidet man in

- Priority 1: Richtlinien, die auf jeden Fall eingehalten werden müssen;
- Priority 2: Richtlinien, die eingehalten werden sollen, aber nicht unbedingt müssen;
- Priority 3: Richtlinien, die empfohlen werden.

Die BITV unterscheidet in Richtlinien, die erfüllt werden müssen (Priorität I), und Richtlinien, die erfüllt werden können (Priorität II).

Seit Jahren ist eine neuere Version, die WCAG2, in Bearbeitung, aber noch nicht veröffentlicht. Diese wird die technischen Neuerungen für die Kommunikation im Internet berücksichtigen.

Die Forderungen der BITV sind:

1. Für jeden Audio- oder visuellen Inhalt sind geeignete äquivalente Inhalte bereitzustellen, die den gleichen Zweck oder die gleiche Funktion wie der originäre Inhalt erfüllen.
2. Texte und Grafiken müssen auch dann verständlich sein, wenn sie ohne Farbe betrachtet werden.
3. Markup-Sprachen (insbesondere HTML) und Stylesheets sind entsprechend ihrer Spezifikation und formalen Definitionen zu verwenden.
4. Sprachliche Besonderheiten wie Wechsel der Sprache oder Abkürzungen sind erkennbar zu machen.
5. Tabellen sind mittels der vorgesehenen Elemente der verwendeten Markup-Sprache zu beschreiben und in der Regel nur zur Darstellung tabellarischer Daten zu verwenden.
6. Internetangebote müssen auch dann nutzbar sein, wenn der verwendete Browser neuere Technologien nicht unterstützt oder diese deaktiviert sind.
7. Zeitgesteuerte Änderungen des Inhalts müssen durch den Nutzer kontrollierbar sein.
8. Die direkte Zugänglichkeit der in Internetangeboten eingebetteten Benutzerschnittstellen ist sicherzustellen.
9. Internetangebote sind so zu gestalten, dass Funktionen unabhängig vom Eingabegerät oder Ausgabegerät nutzbar sind.
10. Die Verwendbarkeit von nicht mehr dem jeweils aktuellen Stand der Technik entsprechenden assistiven Technologien und Browsern ist sicherzustellen, so weit der hiermit verbundene Aufwand nicht unverhältnismäßig ist.
11. Die zur Erstellung des Internetangebots verwendeten Technologien sollen öffentlich zugänglich und vollständig dokumentiert sein, wie z.B. die vom World Wide Web Consortium entwickelten Technologien.
12. Dem Nutzer sind Informationen zum Kontext und zur Orientierung bereitzustellen.
13. Navigationsmechanismen sind übersichtlich und schlüssig zu gestalten.
14. Das allgemeine Verständnis der angebotenen Inhalte ist durch angemessene Maßnahmen zu fördern.

Aus diesen Richtlinien lassen sich zum einen ganz konkrete Regeln für die Gestaltung barrierefreier Web-Sites ableiten, etwa die Vermeidung von Frames und die Verwendung von `alt`-Beschreibungen für Grafiken, damit diese auch eine textliche Repräsentation erhalten. Ein genauer Test einer Seite unter den Kriterien des BITV ist aber unerlässlich.

In Abschnitt 5.2.7 wird auf einige Möglichkeiten von Browsern für Menschen mit besonders starker Sehbehinderung eingegangen.

# 3

# Rechnersysteme für Webangebote

Für den Betrieb eines Webangebotes ist ein entsprechender Server notwendig. Für die Entwicklung von Web-Anwendungen ist das ebenso, nur sind hier wesentlich andere Kriterien ausschlaggebend. Ein drittes System neben dem Webclient kann wichtig sein – ein Testserver –, um Anwendungen vor ihrem Einsatz auszuprobieren.

Neben den Rechnersystemen für die Web-Applikation ist der Datenbankserver ein weiteres, entscheidendes Rechnersystem in der Gesamtarchitektur.

Netcraft [web] bietet unter „Whats that site running?“ eine Analyse praktisch aller Webserver samt einer zugehörigen Zeitreihe. Abbildung 3.1 zeigt diese Auswertung für `www.springer.de`.

## 3.1 Die Hardware

Auch in Zeiten einer immer stärkeren *Hardware* sind bei der Auswahl eines Systems für einen Webserver einige Punkte zu beachten:

- Der CPU kommt eine eher geringe Bedeutung zu; wichtig sind ein großer Speicher und viel Platz auf sehr schnellen Platten.
- Wichtig ist ein gutes Netzwerkinterface, möglicherweise auch mehrere, die parallel genutzt werden.
- Je nach Einsatzbereich ist die Ausfallsicherheit ein entscheidendes Kriterium. Im einfachsten Schritt werden über ein Hardware-RAID die Platten gespiegelt, in der Endstufe stehen zwei vollständig gespiegelte Rechnersysteme zur Verfügung, die sich gegenseitig immer ersetzen können.
- Bei hochwertiger Hardware ist an einen passenden Wartungsvertrag zu denken, der den Anbieter zu einer vorgegebenen Reaktionszeit verpflichtet.

Insgesamt stellt aber ein reiner Webserver keine hohen Ansprüche an die Hardware; in vielen Fällen leistet ein Apache-Server mit Linux einen sehr guten Dienst auf Hardware, die für den Büroalltag bereits nicht mehr genutzt wird.

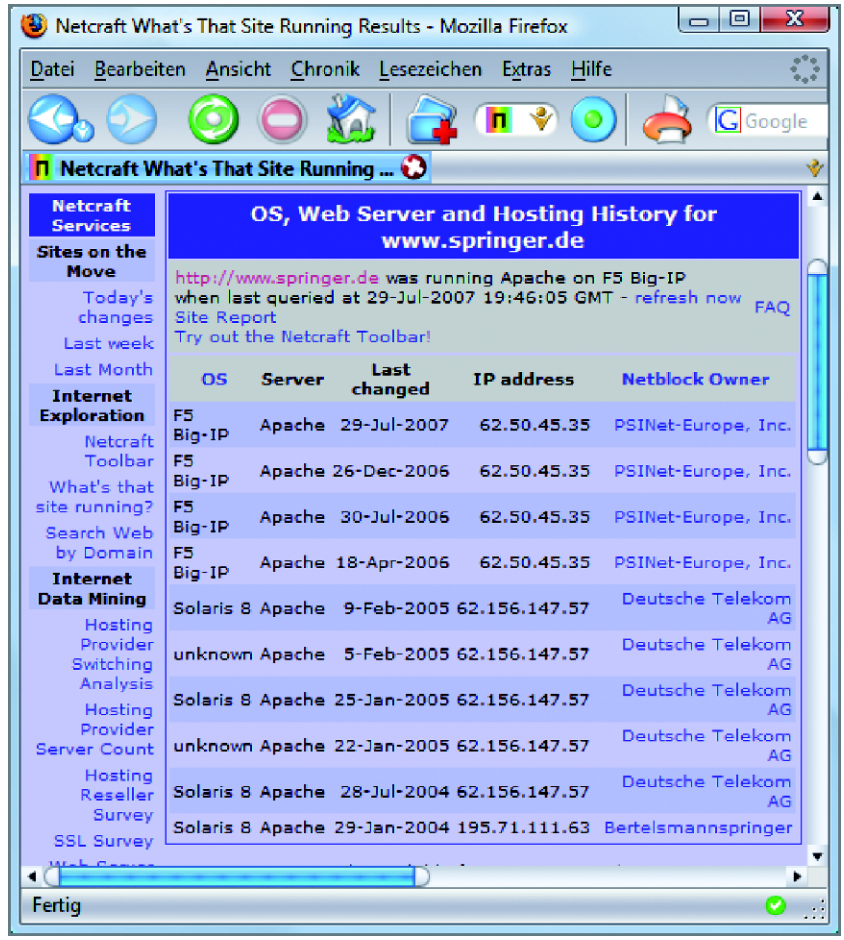



**Abbildung 3.1:** Netcraft-Analyse von `www.springer.de` – Apache/Solaris und der Münchner Provider F5

## 3.2 Betriebssysteme im Web

Für den Betrieb eines Webservers ist nach der reinen Hardware das *Betriebssystem* entscheidend.

Hier konkurrieren Unix/Linux-Systeme mit Windows, bei den Produktionssystemen ist aber die erste Gruppe stärker vertreten.

### 3.2.1 Unix

Unter den verschiedenen Unix-Derivaten ist das Betriebssystem Solaris von Sun Microsystems besonders stark vertreten, auch im Web-Umfeld [web].

Solaris wird sowohl für die Sun-eigene Linie der SPARC-Prozessoren als auch für x86-Prozessoren von intel und AMD angeboten. Insbesondere für SPARC bietet es einige Vorteile, etwa eine sehr ausgefeilte Virtualisierung.

### 3.2.2 Linux

Das auf Linus Torvalds zurückgehende freie Betriebssystem *Linux* erfreut sich seit 1991 eines sehr großen Zuspruchs und wird inzwischen in allen Bereichen

eingesetzt – bis hin zum Einsatz als Betriebssystem für das Handy „Neo 1973“.

Natürlich ist der Einsatz als Webserver eine Domäne von Linux. Dies resultiert aus zwei Gründen:

- Linux gilt als sehr sicher.
- Linux ist sehr performant, da vergleichsweise einfach nur diejenigen Dienste betrieben werden, die für den jeweiligen Einsatz benötigt werden (was zusätzlich auch aus Sicherheitsgründen wichtig ist, vgl. Kapitel 30). So braucht ein Webserver üblicherweise kein grafisches Benutzersystem, sondern wird – lokal oder remote – über die Kommandozeile administriert.

Inzwischen gibt es zahlreiche Dialekte des Betriebssystems Linux; im Folgenden sollen einige der wichtigsten für den Einsatz im Web vorgestellt werden.

**Abbildung 3.2:**
Das Linux-Maskottchen Tux

### 3.2.2.1 Debian

*Debian* [web] ist eine für den Betrieb als Server sehr verbreitete Linux-Variante. Der Grund hierfür liegt darin, dass Debian ein sehr gutes Update-Konzept bietet und sehr stabile Software-Pakete verwendet.

Debian wird in drei Varianten angeboten:

- stable: Diese Variante besteht nur aus ausgiebig erprobten Paketen; somit ist dieser Debian-Zweig sehr stabil, bietet aber nicht die neuesten Software-Versionen. Security-Updates werden aber sehr zügig eingearbeitet.
- testing: Dies ist eine Zwischenstufe zwischen stable und unstable.
- unstable: Dieser Zweig bietet jeweils die neuesten Software-Versionen, was etwa für den Entwickler wichtig sein kann, für einen stabilen Serverbetrieb aber nicht empfehlenswert ist.

Wichtig ist die in Debian integrierte automatisierte Update-Funktionalität, welche den Betrieb eines Debian-Servers für den Administrator wesentlich vereinfacht.

Die aktuelle stable-Version hat die interne Bezeichnung „Debian Etch“. Die Bezeichnungen der Debian-Versionen stammen vom ersten Pixar-Film „Toy Story“, nur die unstable-Version wird immer als „Sid“ bezeichnet.

### 3.2.2.2 Ubuntu und Kubuntu

Das Linux-Derivat *Ubuntu* basiert auf Debian, ist aber für den Anwender leichter zu handhaben, weshalb es als Einstiegsvariante in Linux gerne verwendet wird.

*Kubuntu* ist eine Ubuntu-Version, welche KDE als Windows-Manager verwendet (im Gegensatz zu Ubuntu mit Gnome).

### 3.2.2.3 Gentoo

*Gentoo* [web] ist eine hochspezialisierte Linux-Variante. Ihr Vorteil ist gleichzeitig auch ihr Nachteil: Gentoo ist sehr umfassend konfigurierbar, so dass ein optimales, hochperformantes Betriebsssytem vorliegt, welches aber einen erheblichen Administrationsaufwand bedingt.

### 3.2.3 Windows

Das Microsoft-Betriebssystem ist seit längerem auch im Serverbereich erfolgreich im Einsatz. Dabei kann Windows im Web sowohl mit dem Apache-Webserver als auch mit dem Microsoft-Server IIS eingesetzt werden.

Einige sehr große Sites, etwa `ebay.de`, werden mit Windows betrieben und zeigen somit, dass dies möglich ist. Als Nachteil von Windows als Betriebssystem im Web wird aber häufig der Sicherheitsaspekt genannt.

### 3.2.4 Virtualisierung

Moderne Betriebssysteme unterstützen eine *Virtualisierung:* Auf einer Hardware laufen parallel mehrere verschiedene Betriebssystem-Instanzen.

Dieser Ansatz ist für den Serverbetrieb heute zunehmend interessant, da es somit möglich ist, als Provider mehrere Domains zu hosten, die ein eigenes Betriebssystem verwenden. Der Vorteil liegt zum einen in der Sicherheit – die einzelnen Systeme sind geschützt voreinander –, zum anderen können so im Gegensatz zum virtuellen Host des Apache die einzelnen Apache-Instanzen unterschiedlich konfiguriert werden.

### 3.2.5 Resümee: Das Betriebssystem für die Web-Programmierung

Nun wurden hier einige Betriebssysteme kurz vorgestellt, es gibt aber eine ganz klare Empfehlung für das optimale Betriebssystem für die Web-Programmierung:

**Eine gute Web-Programmierung sollte stets unabhängig vom Betriebssystem und auf verschiedenen Systemen einsetzbar sein.**

Typisch ist die gemischte Situation: Entwicklung auf Windows, Produktivsystem auf Unix; dadurch ist die Portierbarkeit der Anwendung gesichert.

Entsprechend dieses Ziels sind die Beispiele in diesem Buch unabhängig vom Betriebssystem; folgende Betriebssysteme sind bei der Erstellung des Buches verwendet worden:

- Debian Etch (ein virtueller Host);
- Gentoo (der Webserver ceres.informatik.fh-kl.de);
- Windows XP professional SP2;
- Windows Vista business.

## 3.3 Datenbankserver

Neben dem eigentlichen Webserver ist der *Datenbankserver* zu beachten. Bei genauerer Betrachtung der Situation wird sogar klar, dass für die Performance einer datenbankbasierten Web-Applikation dem Datenbankserver eine wesentlich größere Bedeutung als dem Webserver zukommt.

Dabei gelten in Bezug auf das Betriebssystem zunächst die gleichen Bemerkungen wie bei dem Webserver. Hinzu kommt nun das Datenbankmanagementsystem (DBMS). Hier ist ein wichtiger Punkt im Vorfeld zu beachten: die Lizenzbedingungen für das DBMS.

Klassische Datenbanklizenzen basieren auf Modellen, die die gleichzeitige Benutzerzahl zugrunde legen; dies ist für Web-Applikationen so nicht mehr möglich, hier werden Lizenzmodelle nach CPU-Klassen verwendet. Als Konsequenz davon sind viele kommerzielle DBMS für den Einsatz im Web in den meisten Fällen zu teuer.

### 3.3.1 MySQL

Vermutlich das beliebteste Datenbankmanagementsystem für den Einsatz im Web ist *MySQL* [web]. Die Gründe hierfür sind vielfältig: MySQL ist sehr performant, mit vertretbarem Aufwand zu administrieren, es unterstützt inzwischen gut den SQL-Standard und es hat ein sehr offenes – aber nicht ganz freies – Lizenzmodell.

MySQL ist für viele Betriebssysteme verfügbar und leicht zu installieren. Ebenfalls über die MySQL-Site sind zahlreiche Tools und verschiedene Treiber für MySQL verfügbar.

Lange Zeit war die Version 3.23 von MySQL sehr verbreitet. Die neueren Versionen haben das DBMS aber erheblich in seinem Funktionsumfang erweitert, insbesondere

- unterstützt MySQL nun Transaktionen;
- ist der SQL-Sprachumfang von MySQL deutlich gewachsen.

Die Beispiele in diesem Buch beziehen sich meistens auf das DBMS MySQL.

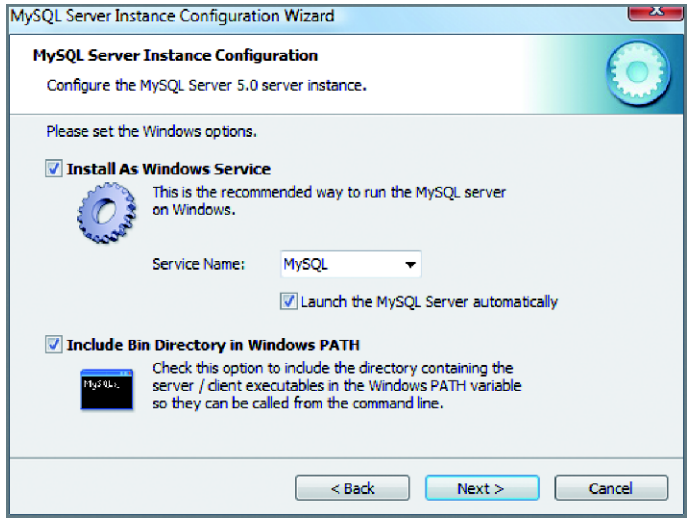



**Abbildung 3.3:** MySQL-Installation auf Windows Vista

Wichtig für MySQL, aber auch für viele andere DBMS, ist die Möglichkeit des direkten Arbeitens mit dem System über eine Shell, hier der „MySQL-Monitor“. Dieser wird durch die Anweisung

```
mysql -u user -p
```

gestartet; dabei besagt der Schalter `-u`, welche Benutzerkennung sich an MySQL anmelden möchte; `-p` erzwingt die Kennworteingabe. Danach kann mittels

```
use
```

eine Datenbank gewählt und mit dieser gearbeitet werden (Abbildung 3.4).

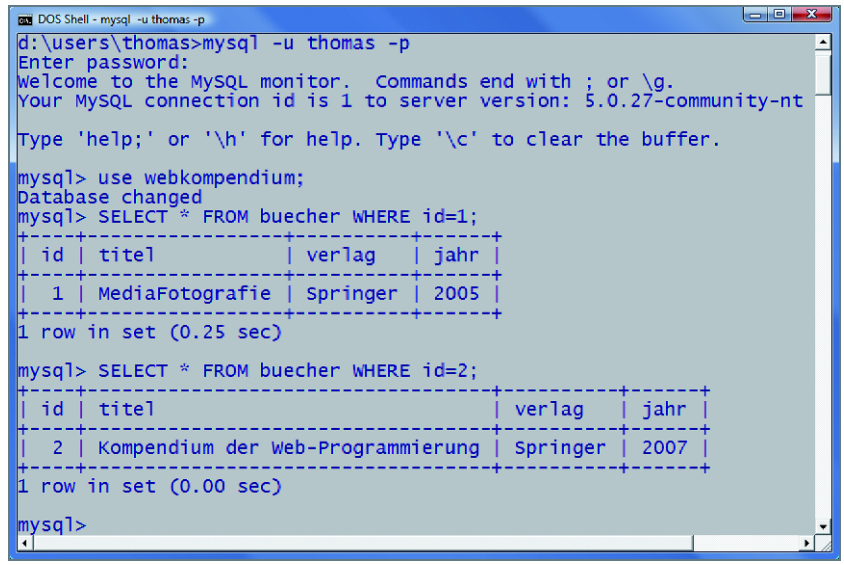



**Abbildung 3.4:**
Der MySQL-Monitor

Neben derartig einfachen Tools stehen zahlreiche grafische Hilfen bereit, um das DBMS MySQL zu administrieren; einige sind über die MySQL-Site zu beziehen, andere werden in 8.5 vorgestellt.

### 3.3.2 Postgres

Das DBMS PostgreSQL hat seinen Ursprung im akademischen Bereich, es ist eine Entwicklung der University of California at Berkeley (UCB). Heute ist PostgreSQL [web] im Open Source-Bereich die größte Alternative zu MySQL. Während in den älteren Versionen PostgreSQL einen stärkeren SQL-Sprachumfang, aber eine schwächere Performance gegenüber MySQL aufweist, haben sich inzwischen beide stark angenähert.

Ein starker Vorteil von PostgreSQL ist die gegenüber MySQL klarere und offenere Lizenz.

### 3.3.3 Die kleinen: SQLite, HSQLDB und Derby

Während MySQL und PostgreSQL vollwertige, komplexe Datenbankmanagementsysteme sind, bestehen einige wesentlich einfachere Systeme; zu diesen zählen *SQLite, HSQLDB und Derby* [web].

Die Systeme legen die eigentlichen Informationen in Textdateien ab. Ihr Nachteil liegt im kleineren Funktionsumfang, etwa in der eingeschränkten Benutzerverwaltung und in dem geringeren SQL-Sprachumfang. Auf der anderen Seite haben diese einfachen Systeme aber auch wichtige Vorteile:

- der Installationsaufwand ist minimal;
- sie sind teilweise sehr performant;
- ein Datenbank-Backup wird auf einfachste Weise durch Kopieren der Textdateien erzeugt.

SQLite ist inzwischen das Standardsystem für PHP in der Version 5 (vgl. Kapitel 11). HSQLDB ist eine Java-basiertes DBMS und ist inzwischen Teil von OpenOffice. Apache Derby, ebenfalls ein Java-basiertes DBMS, wird von der Apache Foundation herausgegeben und ist auch Teil des Java Development Kits in der Version 6.

### 3.3.4 Ergebnisse von Select: Resultset

Für viele Datenbank-Operationen wird eine SELECT-Anfrage in der SQL-Syntax an das DBMS gesendet. Die Art der Antwort ist eine Art Tabelle, welche hier als *Resultset* bezeichnet wird.

Im Java-Paket `java.sql` gibt es hierfür ein eigenes Interface `ResultSet`, welches die zentralen Methoden für den Umgang mit den Trefferlisten postuliert.

## 3.4 Alles aus einer Hand: XAMPP

Die Installation der einzelnen Komponenten, die für eine Web-Applikation benötigt werden, kann zeitaufwändig werden: Nach dem Betriebssystem das DBMS, der Webserver und die verschiedenen Sprachen. XAMPP [web] ist eine vorgefertigte Distribution, welche

- Apache
- MySQL
- PHP
- Perl

umfasst und für die Betriebssysteme Windows, Linux, Mac OS X und Solaris verfügbar ist. Zu XAMPP gehören auch gute Administrationswerkzeuge.

Für das schnelle Aufsetzen eines Entwicklungsrechners ist deshalb XAMPP durchaus von Vorteil, allerdings für Produktionssysteme oder individuell konfigurierte Systeme nicht zu empfehlen.

Der Anwender sollte auch bedenken, dass die selbständige Konfiguration sehr lehrreich ist und die einzelnen Systeme vertrauter macht.

# 4

# Softwarearchitektur für das Internet

Die Entwicklung moderner Web-Applikationen geht Hand in Hand mit Fragen der *Softwarearchitektur:* Wie wird eine Software „desgned“, also intern strukturiert, damit ein möglichst großer Nutzen besteht. Hierunter verbergen sich klassische Fragen und Ziele der Softwareentwicklung:

- Das Softwaresystem soll stabil, geprüft und performant sein; insbesondere der Performance kommt im Web eine große Bedeutung zu, da der Endnutzer eine sehr kurze Antwortzeit erwartet; hier gilt ein Wert von 5 Sekunden als Richtwert.
- Eine Software soll leicht zu warten, zu erweitern und wiederzuverwerten sein: Bei Änderungen in den Anforderungen sollen diese leicht und mit wenig Aufwand einzupflegen sein: So soll etwa eine Änderung am Design des Webauftritts keine Neuprogrammierung mit sich ziehen. Auch sollten relevante Softwareteile auch für andere Zwecke nutzbar sein

Es gibt viele Schritte und Teilschritte, wie diese Ziele heute erreicht werden. Grundlegend für alles ist eine systematische, durchdachte Softwareentwicklung und kein planloses „Herumprogrammieren“. Im ersten Schritt ist hierfür eine saubere Problemanalyse und die Fixierung der zu erreichenden Ziele/-Funktionalitäten in einem Pflichtenheft notwendig.

An diese erste Analysephase sollte sich die Modellierung des Problems etwa mit den sprachunabhängigen Methoden der Unified Modeling Language (UML, siehe etwa [BRJ05]) anschließen. Erst im letzten Schritt erfolgt eine Implementierung in der Software.

## 4.1 Projektmanagement für das Web

Für die erfolgreiche Realisierung eines Webprojektes ist wie für jedes Softwareprojekt ein *Projektmanagement* unerlässlich. Hierzu ist vielfältige, detaillierte Literatur verfügbar, siehe etwa [BBS04] und [Sto04].

An dieser Stelle soll nur darauf verwiesen werden, dass es ohne ein sauberes Projektmanagement nicht möglich ist, im Web erfolgreich tätig zu sein.

## 4.2 Programmierparadigmen für das Web

Ein klassischer Begriff im Zusammenhang mit der Softwarearchitektur ist das zugrundeliegende *Programmierparadigma.* Es gibt mehrere solcher Paradigmen, hier sind insbesondere das prozedurale und das objektorientierte von Bedeutung.

### 4.2.1 Prozedurale Programmierung

Die *prozedurale Programmierung* ist der Ansatz der frühen Informatik, in einem „Hauptprogramm“ mit „Unterprogrammen“, den Prozeduren, zu arbeiten. Die einzelnen Programme werden dabei in Bibliotheken gebündelt.

Dieser Ansatz der Softwareentwicklung ist immer noch stark verbreitet, etwa für Perl (vgl. Kapitel 10) ist dies weiterhin das vorherrschende Paradigma, die Objektorientierung hält nur langsam Einzug.

### 4.2.2 Objektorientierte Programmierung

Die *Objektorientierung* (OO) geht einen wesentlichen Schritt weiter als die prozedurale Programmierung und bündelt Software-Objekte, welche realen Objekten nachempfunden Eigenschaften und Verhalten haben. Diese Software-Objekte können dann sehr universell eingesetzt werden.

Entsprechend ihrer großen Bedeutung ist die Literatur zur Objektorientierung umfassend. Hier sollen im wesentlichen die Begrifflichkeiten, wie sie in diesem Buch verwendet werden, offengelegt werden:

- Eine Klasse ist die Beschreibung des Aufbaus aller Objekte eines bestimmten Typs, anschaulich der Bauplan für alle Objekte der entsprechenden Art.

  Innerhalb einer Klasse gibt es zwei Arten von Elementen:
  - die Klassenvariablen, auch als Membervariablen bezeichnet, präsentieren die Eigenschaften und den Zustand eines Objekts; sie werden auch als Attribute oder Felder bezeichnet.
  - die Methoden einer Klasse verändern den Zustand des Objektes, also den Wert der Membervariablen.

  Eine ganz besondere Methode ist der Konstruktor: diese Methode wird immer bei Erzeugung eines Objektes automatisch aufgerufen.

  Die Existenz einer Klasse bedeutet zunächst nur, dass dieser Typ definiert ist, es besteht aber noch keine „Variable vom Typ dieser Klasse“, das ist das Objekt.

  Es ist üblich, die Bezeichner von Klassen mit einem Großbuchstaben zu beginnen; dies wird in diesem Buch auch so gehandhabt.
- Ein Objekt ist die konkrete Instanz einer Klasse. Erst das Objekt belegt realen Speicherplatz.

  In vielen Sprachen wird ein Objekt durch einen Operator `new` erzeugt.
- Eine Referenz ist eine skalare Variable, welche auf ein Objekt, also auf eine konkrete Instanz einer Klasse, verweist. Erst durch diese Referenzen sind Objekte erreichbar.

  Hier unterscheiden sich die Objekte deutlich von einfachen Variablen eines elementaren Datentyps, da etwa zwei Referenzen auf das gleiche Objekt verweisen können, also den gleichen Speicherbereich adressieren.

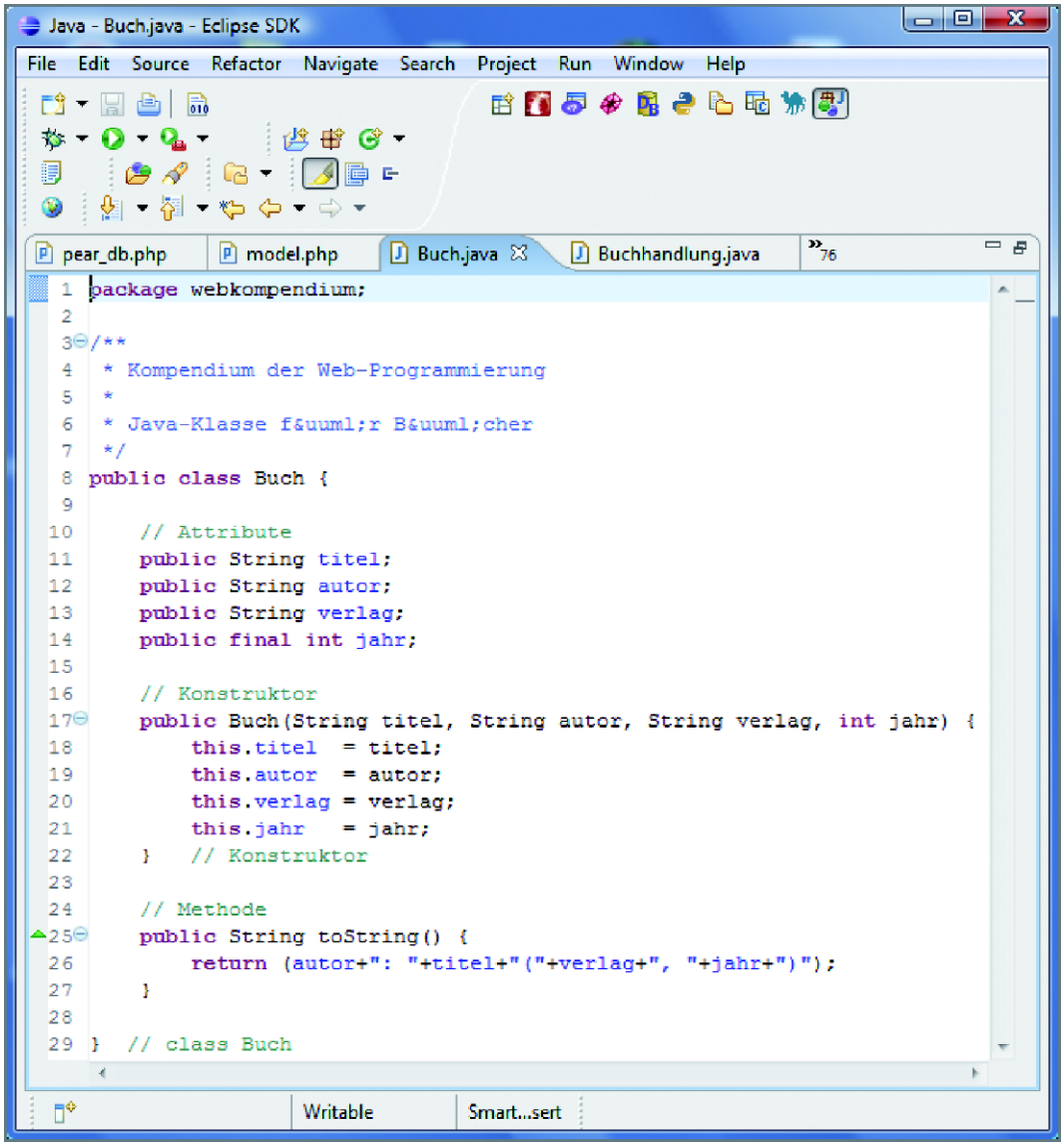
  Insgesamt wird dann über eine Syntax der Art

  ```
  $referenz = new Klasse();
  ```

  eine skalare Variable `$referenz` als Verweis auf ein Objekt der Klasse `Klasse` definiert; dabei wird der Konstruktor der Klasse `Klasse` aufgerufen, welchem auch Paramaterwerte übergeben werden können, typischerweise für die Initialisierung des neuen Objektes.

In diesem Buch wird ein Beispiel möglichst für alle Fälle behandelt: In Kapitel 7 wird die Datenstruktur „Buch“ vorgestellt. Hier zur Verdeutlichung der

Objektorientierung eine einfache Klasse `Buch` in Java: Es werden vier Attribute definiert, ein Konstruktor und eine Methode (Abbildung 4.1). `Buchhandlung.java` erzeugt nun zwei Instanzen, also zwei Objekte dieser Klasse und wendet implizit bei der Ausgabe die definierte Methode `toString()` an (Abbildung 4.2).




```
1  package webkompendium;
2
3  /**
4   * Kompendium der Web-Programmierung
5   *
6   * Java-Klasse f&uuml;r B&uuml;cher
7   */
8  public class Buch {
9
10     // Attribute
11     public String titel;
12     public String autor;
13     public String verlag;
14     public final int jahr;
15
16     // Konstruktor
17     public Buch(String titel, String autor, String verlag, int jahr) {
18         this.titel  = titel;
19         this.autor  = autor;
20         this.verlag = verlag;
21         this.jahr   = jahr;
22     }   // Konstruktor
23
24     // Methode
25     public String toString() {
26         return (autor+": "+titel+"("+verlag+", "+jahr+")");
27     }
28
29 }  // class Buch
```

**Abbildung 4.1:**
Die Java-Klasse `Buch`

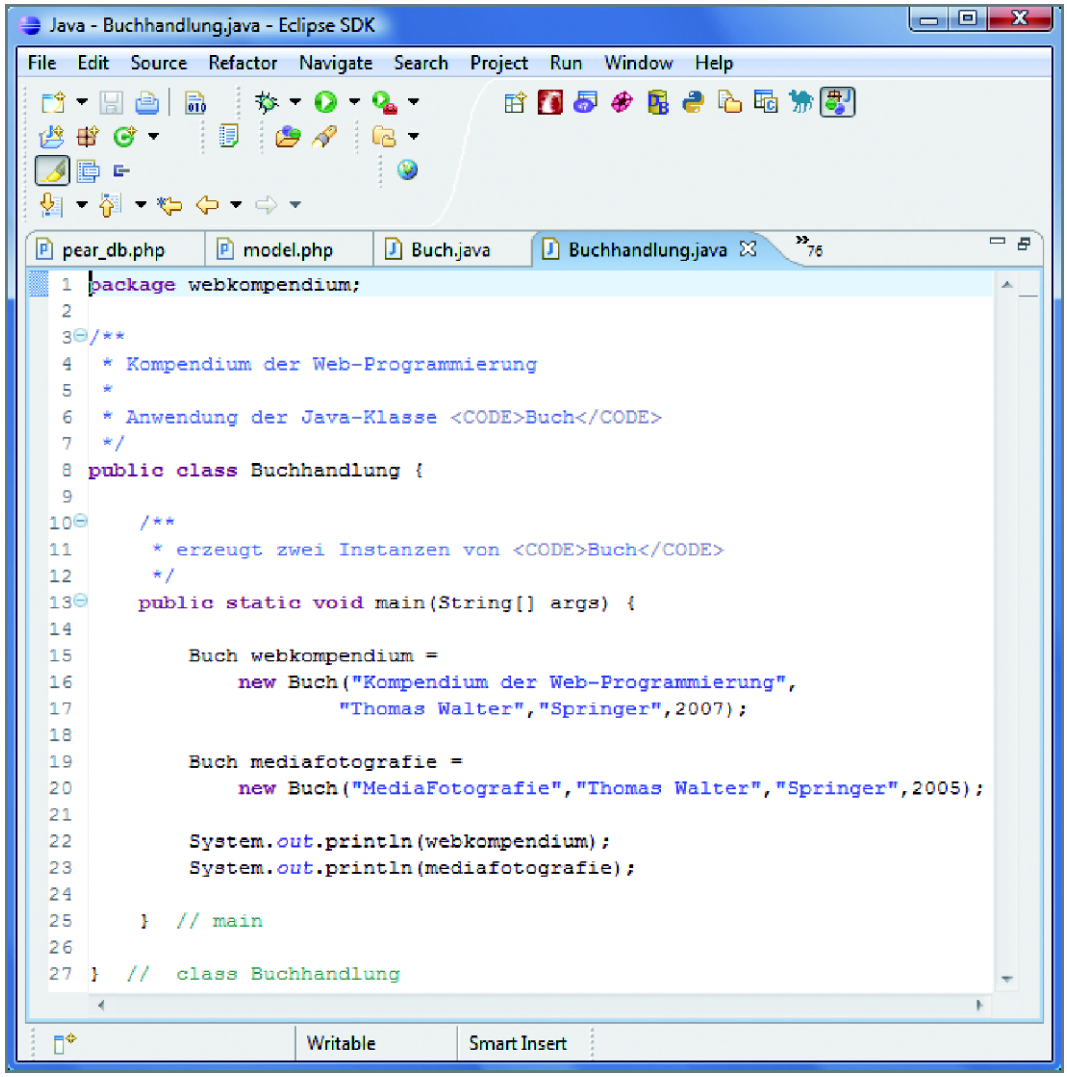



**Abbildung 4.2:** Erzeugung zweier Instanzen von `Buch`

Die Objektorientierung verfügt darüber hinaus über zahlreiche sehr wichtige und bewährte Konzepte, etwa die Datenkapselung für das Schützen der Eigenschaften eines Objektes und die Vererbung: Kindklassen erben mit einem sehr einfachen Mechanismus alle Eigenschaften und Methoden der Ausgangsklasse, was es erlaubt, einfach zu pflegende und übersichtliche Klassen zu entwerfen.

In diesem Buch werden viele verschiedene Programmiersprachen vorgestellt, die alle einen Bezug zum Web haben. Alle von diesen behaupten von sich, objektorientiert zu sein, was allerdings nur teilweise stimmt. Diese Sprachen implementieren alle Elemente der Objektorientierung, aber in stark unterschiedlichem Ausmaß. Vergleichsweise wenig entwickelt ist die Objektorientierung in Perl; bei PHP nimmt sie mit jeder neuen Version zu und erreicht fast das Niveau von Java (vgl. 11.6.1). Richtig stark in der Objektorientierung ist Ruby (vgl. 13.4), einer der Gründe, warum sich diese Sprache aktuell großer Beliebtheit erfreut.

Manche der hier behandelten Sprachen gehen auch einen eigenen Weg, so ist etwa die Objektorientierung in JavaScript sehr eigen implementiert: den Be-

griff der Klasse gibt es so nicht, es werden Klassen nur über spezielle Funktionen und darin vorgenommenen Zuordnungen definiert (vgl. 15.4).

### 4.2.3 Verteilte Systeme

*Verteilte Systeme* sind ein Zusammenschluss von interagierenden Prozessen auf verschiedenen Rechnersystemen. Dieser, auf Andrew Tanenbaum zurückgehende, Begriff ist für Betriebssysteme seit längerem von großer Bedeutung, da sich hiermit ausgesprochen leistungsfähige Architekturen mit vertretbarem Ressourceneinsatz bilden lassen.

Im Bereich der Web-Applikationen spielen verteilte Systeme eine große Rolle. Der erste Schritt ist die Abspaltung des Datenbankservers vom Webserver, was auch unter Sicherheitsaspekten wichtig ist. Im zweiten Schritt kann der Webserver von denjenigen Systemen, welche die eigentliche Web-Programmierung ausführen, getrennt werden; dies ist etwa mit fastCGI nach Kapitel 19 möglich. Von der Architektur her bildet auch eine Ajax-Anwendung ein verteiltes System (vgl. Kapitel 16), da sich hier die Web-Applikation fast schon klassisch auf den (Web-)Client und den (Web-)Server verteilt.

Load Balancing bezeichnet Verfahren zur gezielten Lastenverteilung auf mehrere Systeme, wobei auch mehrere Webserver zum Einsatz kommen können, die abwechselnd Requests beantworten.

Beispielsweise nutzt Google derartige Verfahren sehr intensiv, um stets ein schnelles Antwortverhalten bei sehr großer Last zu gewährleisten. Insgesamt soll Google momentan weltweit rund 450.000 Server im Einsatz haben, die Zahl ist aber nicht genau bekannt und steigt ständig.

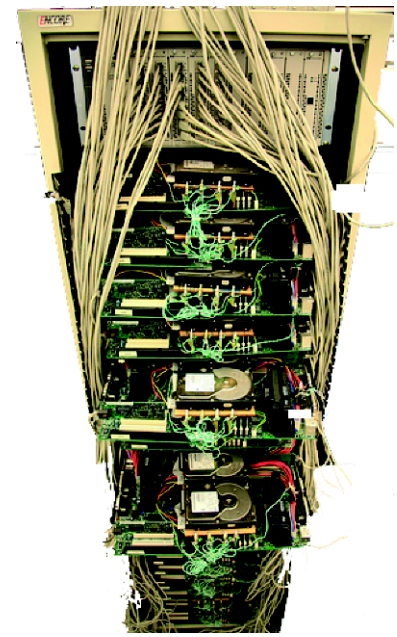


**Abbildung 4.3:**
Der erste produktive Google-Server (1999)

### 4.2.4 Schichtmodell

Ein wesentlicher Begriff für die Klassifizierung von Softwarearchitekturen für das Web ist das *Schichtmodell,* die Aufteilung der gesamten Software in *n* verschiedene Schichten.

Momentan sind für Web-Applikationen mindestens drei Schichten notwendig, man spricht dann vom „3-Tier Model“:

- der Client als getrennte Schicht;
- die Anwendungsschicht, in welcher die eigentliche (serverbasierte) Web-Programmierung liegt;
- die Datenhaltungsschicht: das DBMS.

Natürlich haben die einzelnen Schichten, insbesondere die mittlere, wiederum eine eigene Unterstruktur.

Einige Ansätze, wie sie hier vorgestellt werden, passen nicht streng in dieses Dreischichtmodell; so sind bei Applet Zweischichtenarchitekturen wie beim klassischen Client-Server-Modell möglich.

## 4.3 Das Entwurfsmuster Model-View-Controller

Die moderne Softwarearchitektur kennt zahlreiche Arten, wie ein Softwaresystem „designed“ werden kann. Das klassische Lehrbuch in diesem Bereich ist [GHJ04]. Die Ziele einer sauberen Softwarestruktur sind in erster Linie

- die gute Wiederverwertbarkeit der Entwicklung;

- ein möglichst geringer Pflegeaufwand;
- eine geprüfte und stabile Software.

Es gibt zahlreiche Entwurfsmuster für eine moderne Software, die diese Ziele umsetzen – vgl. [GHJ04] – allerdings hat sich seit langem ein System besonders bewährt: das 1979 von Trygve Reenskaug vorgestellte *Model-View-Controller* (MVC). Auch für Softwareentwicklungen für das Web ist MVC von zentraler Bedeutung.

MVC, im Deutschen als Modell, Präsentation und Steuerung bezeichnet, strukturiert die Software in drei Komponenten, welche alle interagieren:

- Model: Die Softwarekomponente „model“ behandelt die zu verwaltenden Daten an sich, unabhängig von ihrer Verwendung. Hier ist etwa auch eine Datenbankanbindung für die persistente Speicherung dieser Daten anzusiedeln.

  Diese Model-Klassen lassen sich nun unabhängig vom Einsatz schreiben; so können beispielsweise eine Web-Applikation und ein klassisches lokales Client-Programm die gleichen Model-Klassen nutzen.
- View: Die View-Klassen stellen die Daten des Models entsprechend dar und leiten ggf. Benutzer-Aktionen, die Events, an den Controller weiter.
- Controller: Der Controller steuert die Anwendung; in den Controller-Klassen ist die eigentliche Logik der Anwendung implementiert und sie geben die Model-Daten an die aufzurufenden Views entsprechend weiter. Der Controller verwaltet auch die Eingaben des Benutzers und reagiert entsprechend.

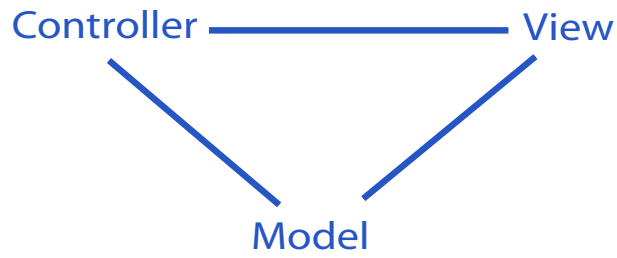



**Abbildung 4.4:**
Das MVC-Designpattern

Abschnitt 11.6.2 zeigt exemplarisch eine Web-Applikation nach MVC: Der HTTP-Request an den Controller bewirkt, dass die Model-Klasse instanziiert wird, was auch eine Abfrage der zugehörigen Datenbank bedeutet; die Anzeige dieser Model-Instanz wird durch eine Methode der Klasse View gesteuert (Abbildungen 11.53, 11.54, 11.55 und 11.56).

Viele der modernen Frameworks für das Web erleichtern die Softwareentwicklung nach MVC, erzwingen diese fast. Hier ist sicherlich Ruby on Rails zu erwähnen (vgl. 23), welches fest auf MVC basiert. Aber auch andere Frameworks wie Struts für Java (vgl. 24.5.3) oder django für Python (vgl. 22) führen direkt zu MVC.

Nur für die Gestaltung der View sind noch zwei für das Web zentrale Aspekte zu bemerken:

- Ein Teil der Abspaltung der View für die elementare Formatierung ist direkt durch das Stylesheet gegeben, vgl. 2.5).
- Nur für die Abkopplung der View sind Template Engines wie Smarty (vgl. 21) sehr hilfreich.

### 4.3.1 MVC im Web

Speziell für den Einsatz im Web kann es sinnvoll sein, die Controller-Klassen und die View-Klassen konkreter zu spezialisieren: Der Controller verwaltet HTTP-Requests und die View sendet die HTML-Response.

## 4.4 Entwicklungsumgebungen

### 4.4.1 Eclipse

Eine effiziente Programmentwicklung [web] bedient sich natürlich geeigneter Werkzeuge. Das wichtigste hiervon ist eine integrierte Entwicklungsumgebung, eine „IDE“. Aus einer Vielzahl unterschiedlicher Produkte in verschiedenen Leistungs- und Kostenbereichen setzt sich aktuell die Open Source-IDE *Eclipse* am meisten durch; diese wird in Abschnitt 8.1 weiter vorgestellt.

Eclipse ist direkt für den Einsatz mit der Programmiersprache Java vorgesehen, kann aber durch Erweiterung mit entsprechenden Plugins an andere Sprachen angepasst werden. Grundlegend für Eclipse ist der objektorientierte Ansatz, der schon in der Grundfunktionalität der Entwicklungsumgebung berücksichtigt ist.

#### 4.4.1.1 Plugins

Das reine Eclipse ist, wie schon erwähnt, für Java ausgelegt. Für den Einsatz mit anderen Programmiersprachen muss Eclipse durch geeignete *Plugins* erweitert werden.

Inzwischen liegen zahlreiche, meistens frei verfügbare Plugins für Eclipse vor. In diesem Buch wird die Softwareentwicklung stets mit Eclipse und den jeweiligen Plugins umgesetzt.

Weiteres zu den Plugins für Eclipse ist in 8.1.1 zu finden.

## 4.5 Dokumentation

Für eine professionelle Softwareentwicklung ist eine gute *Dokumentation* unverzichtbar: Die Entwicklerdokumentation soll stets so ausführlich sein, dass eine andere Person die Programmierung nachvollziehen kann.

**Ein Kommentaranteil im Sourcecode von 50 % ist ein grober Richtwert für eine ausreichende Dokumentation.**

Da die Erstellung der Dokumentation aber nicht zu den beliebtesten Aufgaben von Entwicklern zählt, stehen inzwischen auch gute Werkzeuge bereit, diese Arbeit zu erleichtern. Bei den jeweiligen Sprachen werden diese vorgestellt, hier sollen mit javadoc nur eine wegweisende Technik und mit Doxygen eine sprachübergreifende Lösung erläutert werden.

### 4.5.1 javadoc

*javadoc* ist von Anfang an Bestandteil des Java Entwicklerpaketes. Hintergrund ist ein Dienstprogramm `javadoc`, welches automatisiert aus den Java-Sourcen eine HTML-formatierte Online-Dokumentation für Entwickler erzeugt. Dabei

erkennt `javadoc` vieles aus der Programmierung selbst, etwa die genaue Deklaration von Methoden; der Programmierer muss dann nur noch in einem speziellen Kommentar einige Erläuterungen hinzufügen, etwa welche Bedeutung die Argumente einer Methode haben. Hierfür verwendet javadoc eigene Tags mit führendem „`@`“.

Diese speziellen javadoc-Kommentare haben die Syntax

```
/**
    hier ein <I>javadoc-Kommentar</I>
    @version 1.0
 */
```

und können HTML-Formatierungen enthalten.

Beispielsweise lautet der javadoc-Kommentar für die Methode `sendDocument` aus der Java-Klasse `SimpleHttpd` (Abbildungen 6.3 und 6.4)

```
/**
 * sendDocument-Methode: sendet Datei file an OutputStream out
 *
 * @param out  Ausgabestream
 * @param file Datei
 */
public static void sendDocument(DataOutputStream out, File file)
     throws IOException {
```

Die Entwicklungsumgebung Eclipse (vgl. 8.1) bindet javadoc geschickt ein; auch schreibt sie nach benutzerdefinierten Templates für alle Klassenelemente das Grundgerüst des javadoc-Kommentars automatisch.

Dabei bietet javadoc wesentliche Vorteile:

- Die erzeugte Kommentierung ist umfassend verlinkt.
- Ein alphabetischer Index wird automatisch erzeugt.
- Eine Vererbungshierarchie aller Klassen wird direkt erzeugt.

Das Aussehen der Kommentierung kann sogar individuell angepasst werden, wobei die Standardformatierung für die meisten Fälle völlig ausreichend ist.

Abbildung 4.5 zeigt exemplarisch eine derartige Kommentierung.

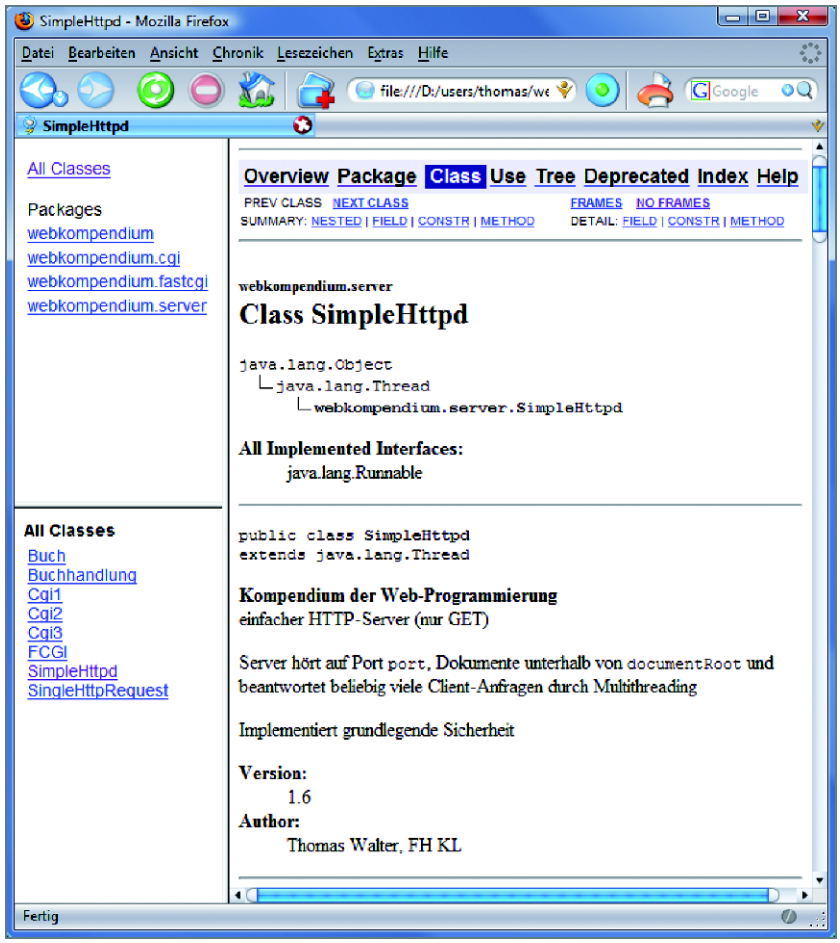



**Abbildung 4.5:** Beispiel für eine javadoc-Kommentierung

**Auf der Web-Site zum Buch ist die javadoc-Dokumentation aller Java-Beispiele in diesem Buch enthalten.**

### 4.5.2 Doxygen

Der Ansatz von javadoc ist bahnbrechend, da hiermit mit wenig Aufwand eine sehr praxisrelevante Entwicklerdokumentation erzeugt wird. Nachteile von javadoc, wenn man sie so bezeichnen will, sind:

- javadoc ist nur für Java anwendbar;
- es wird nur HTML-Dokumentation erzeugt.

Einige neuere, auf javadoc aufbauende Werkzeuge gehen nun in diesen zwei Punkten weiter; hierzu zählt auch das von Dimitri van Heesch entwickelte *Doxygen* [web].

Doxygen kann nach dem javadoc-Prinzip Dokumentation für C/C++, Java, Objective-C, Python und IDL erzeugen sowie mit etwas geringerem Funktionsumfang auch für PHP, C# und D. Als Ausgabeformate werden neben HTML zahlreiche andere Formate unterstützt, auch LaTeX, XML, RTF und PDF.

# 5

# Der Webclient – Browser

Für die Entwicklung erfolgreicher Web-Anwendungen ist die Berücksichtigung der Software „beim Kunden“ unverzichtbar: Der *Webclient* spielt eine wesentliche Rolle für den Erfolg einer Site.

## 5.1 Aufgaben und Arbeitsweise des Webclients

Der Webclient – der Browser – hat im Kern eine einfache Aufgabe: Er ist das Interface für den Menschen, um das HTTP-Protokoll zu bedienen. Dazu gehört das Absenden von HTTP-Requests an Webserver und die Verarbeitung der Antwort.

Von solchen Browsern ist eine Vielzahl verfügbar, die aber eine ausgesprochen unterschiedliche Marktrelevanz haben. Wichtig bei der Erstellung einer Web-Site ist es daher, dass „die richtigen“ Browser beachtet – also die Seite mit diesen getestet – werden:

**Der Entwickler von Web-Anwendungen muss stets verschiedene Browser testen und dabei den Einsatz bei seinen Kunden im Auge haben.**

## 5.2 Aktuelle Browser

### 5.2.1 Übersicht und Verteilung

Die Verteilung der Browser ist für den Entwickler von Web-Applikationen ein ausgesprochen wichtiges Kriterium, denn man muss seine Zielgruppe auch erreichen können. Nun gibt es für die aktuelle Verteilung der Browser keine endgültigen, exakt ermittelten Zahlen, allerdings gibt es zahlreiche Statistiken hierzu im Web, etwa auf heise.de.

Die aktuelle Verteilung ist überschlagsmäßig in Tabelle 5.1 angegeben.

| Region | Anteil IE (alle) | Anteil Firefox | restliche Browser |
|---|---|---|---|
| Deutschland | 65 % | 30 % | 5 % |
| Europa | 75 % | 20 % | 5 % |
| Weltweit | 85 % | 12 % | 3 % |

**Tabelle 5.1:** Verteilung der Browser

**Ein Web-Angebot muss heute auf jeden Fall sowohl mit IE (in den neueren Versionen 6 und 7) sowie mit Firefox getestet werden.**

Allerdings: Diese – zudem noch groben – Werte sind pauschale Näherungen: Für jedes konkrete Projekt können diese deutlich anders ausfallen. So wird etwa für eine Unix/Linux-spezifische Site der IE-Anteil sehr klein sein, während für Apple-spezifisches der Safari-Anteil signifikant wird.

Abbildung 5.1 zeigt eine Auswertung der Serverlogfiles des O'Reilly-Verlags. Hieran kann man zwei Dinge direkt beobachten:

- Obwohl weltweit die Verwendung des InternetExplorers bei über 80 % liegt (Tabelle 5.1), verzeichnet der auf Unix/Linux spezialisierte Verlag inzwischen mehr Zugriffe mit Firefox als mit IE: Das Kundenprofil wirkt sich direkt auf die Browserstatistik aus.
- Inzwischen sind die Anteile von IE in der Version 6 und in der Version 7 annähernd gleich, es wird also bald die neue Version stärker verbreitet sein als die alte.

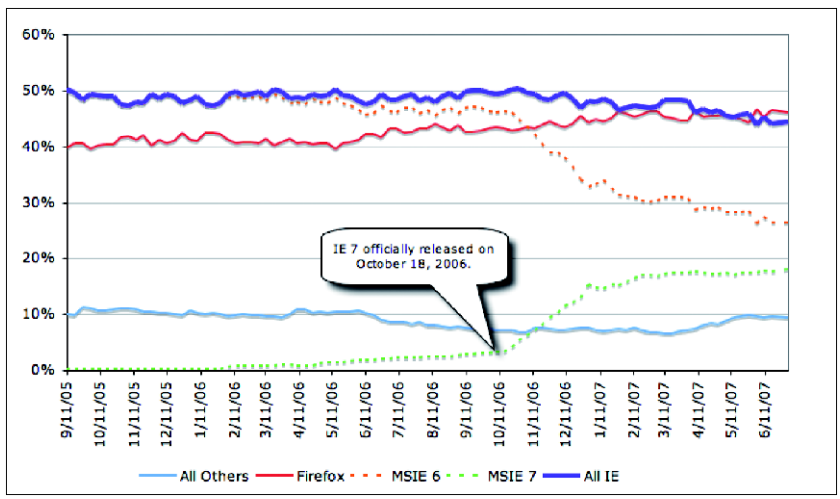



**Abbildung 5.1:** Browser-Verteilung des Webservers des O'Reilly-Verlags

### 5.2.2 MS Internet Explorer

In der zweiten Hälfte der neunziger Jahre erlebte die Internet-Gemeinde den „Browser-Krieg": zwei mächtige Anbieter, Netscape und Microsoft, rangen um die Marktbeherrschung bei den Browsern. Während es Microsoft bis heute nicht geschafft hat, einen größeren Anteil bei den Webservern zu erzielen, hat der Softwarehersteller aus Redmond den Browserkrieg eindeutig gewonnen – zumindest momentan.

Das Microsoft-Produkt ist der *Internet-Explorer,* welcher je nach Statistik (vgl. Tabelle 5.1) einen Marktanteil von über 80 % erreicht. Waren die frühen Versionen des IE noch recht eingeschränkt, wurde spätestens mit der Version 5 eine brauchbares Produkt vorgestellt.

Häufig diskutiert wurde die von Microsoft versuchte Verbindung von Browser und Betriebssystem, welche auch juristisch viele Fragen aufwarf.

Aufgrund seiner großen Verbreitung ist der IE das beliebteste Angriffziel – und er bietet auch häufig erschreckende Sicherheitslücken. Dies sollte bei seinem Einsatz stets bedacht werden.

Der IE wird nicht nur für Windows angeboten, es gab auch Versionen für andere Betriebssysteme, so etwa den IE 5 für Sun Solaris. Wichtig war hier insbesondere die Version für Apples Betriebssystem, beliebt bei Web-Designern, die damit ihre Entwicklung auf dem Mac im IE sehen konnten; diese Version hat aber Microsoft eingestellt. Allerdings kann nicht vollständig von der Ansicht des IE auf einem „fremden" Betriebssystem auf die Ansicht in Windows geschlossen werden – und die neue Version liegt nur noch für Windows vor.

#### 5.2.2.1 Version 6

Die Version 6 des IE war lange Zeit der verbreitetste Browser überhaupt. Microsoft hat, nachdem die Marktführerschaft errungen und Netscape sehr weit abgeschlagen war, hier lange nur noch das Allernötigste für die Fortentwicklung getan. So fehlt diesem Browser beispielsweise das heute sehr beliebte „Tabbed Browsing".

Seit Herbst 2006 wird deshalb der IE 6 durch den Nachfolger abgelöst; Windows Vista wird nur noch mit dem IE 7 ausgeliefert.

#### 5.2.2.2 Version 7

Seit Herbst 2006 bietet Microsoft die neue Version des InternetExplorers an. Nachdem der Hersteller vergleichsweise lange am IE 6 festgehlten hat, bietet die neue Version zahlreiche lange erwartete Verbesserungen, die die meisten Konkurrenzprodukte (Firefox, Opera) schon seit längerer Zeit bieten. Hierzu zählt insbesondere:

- deutlich verbesserte Sicherheit;
- Tabbed Browsing;
- bessere Unterstützung offener Standards, etwa des Standard-Ajax-Objektes von JavaScript `XMLHttpRequest` (vgl. Kapitel 16).

Momentan erfolgt eine rasche Ablösung der Version 6 durch die Version 7 (vgl. Abbildung 5.1).

### 5.2.3 Mozilla Firefox

Der Open Source-Browser *Firefox* ist inzwischen eindeutig der nach dem InternetExplorer wichtigste Browser, in einigen Bereichen überholt er sogar schon den IE (vgl. Abbildung 5.1).

Firefox ist aus der Auflösung des Mozilla-Projektes hervorgegangen, ebenso wie der Mailclient Thunderbird. Er basiert auf der Gecko-Rendering-Engine für die schnelle Darstellung von Webinhalten und ist frei für die meisten Betriebssysteme (u.a. Windows, Unix/Linux und Solaris) verfügbar.

Firefox bietet heute alle Elemente des modernen Browsens wie Browser-Tabs. Für die individuelle Nutzung kann sein Aussehen mit „Themes" verändert werden (Abbildung 5.2).

Für die Web-Entwicklung wichtig ist das Firebug-Plugin für Firefox, das in 8.3 vorgestellt wird.

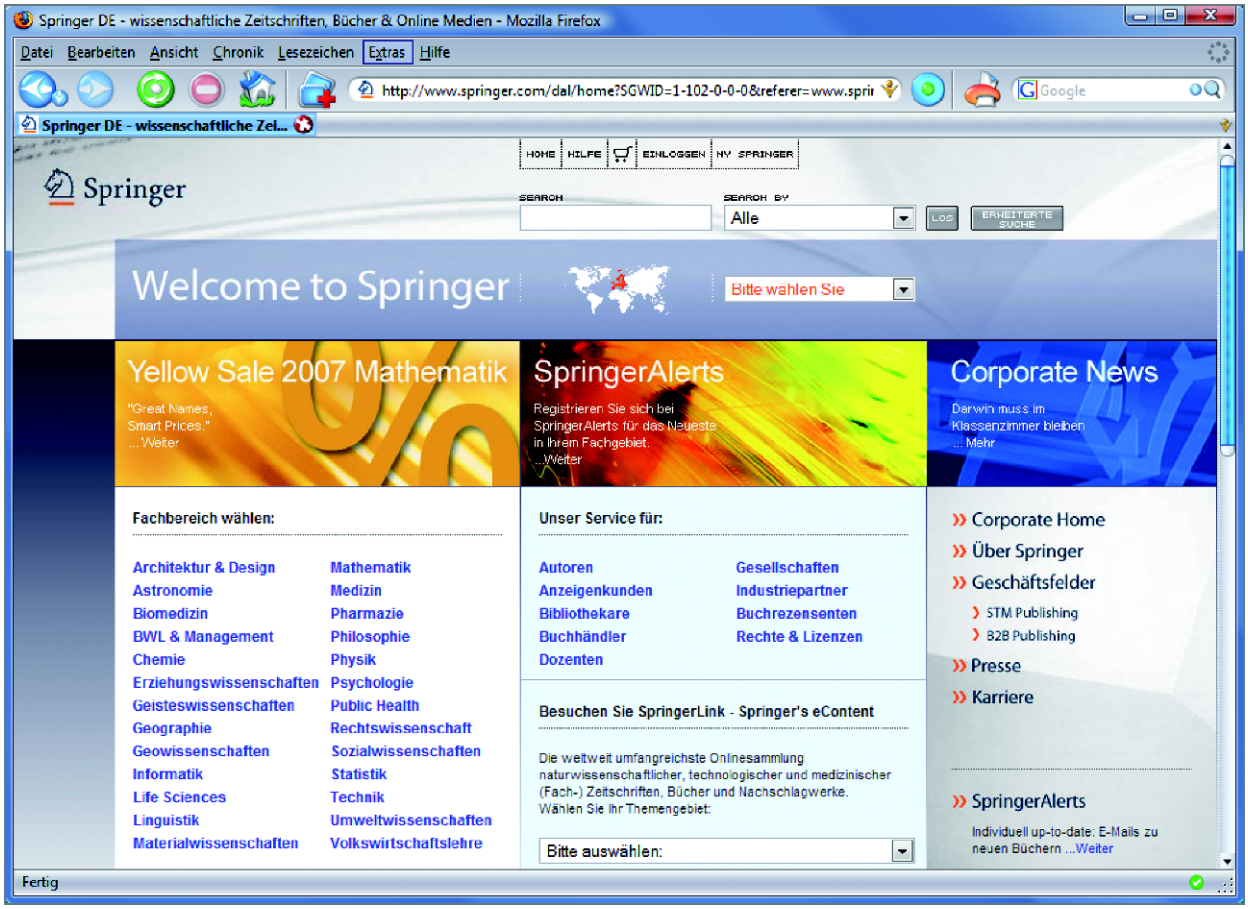



**Abbildung 5.2:** www.springer.de im Firefox-Browser mit dem Noia-Theme

### 5.2.4 Opera

Ein weiterer, für viele Betriebssysteme – auch mobile Endgeräte – verfügbarer Browser ist *Opera*.

Opera ist ein kommerzieller Browser, der inzwischen aber in den normalen Versionen frei verfügbar ist, auch ohne die früher obligatorischen Werbebanner.

Der Opera-Browser gilt als besonders konform zum HTML-/XHTML-Standard des W3C, weshalb er auch verstärkt zur Überprüfung von Webseiten eingestezt wird.

### 5.2.5 Safari

Der Apple-Browser *Safari* (Abbildung 5.3) ist speziell für OS-X entwickelt, ist aber inzwischen auch für Windows verfügbar, da er als Entwicklungsplattform für Software für Apples iPhone verwendet wird. Die Windows-Versionen sind momentan aber noch in einem vergleichsweise wenig ausgereiften Zustand.

Da insgesamt nicht so viele Browser für Apples Betriebssystem verfügbar sind, nimmt dort Safari eine vergleichsweise starke Stellung ein, insgesamt ist aber Safari im statistischen Mittel nicht nennenswert verbreitet.

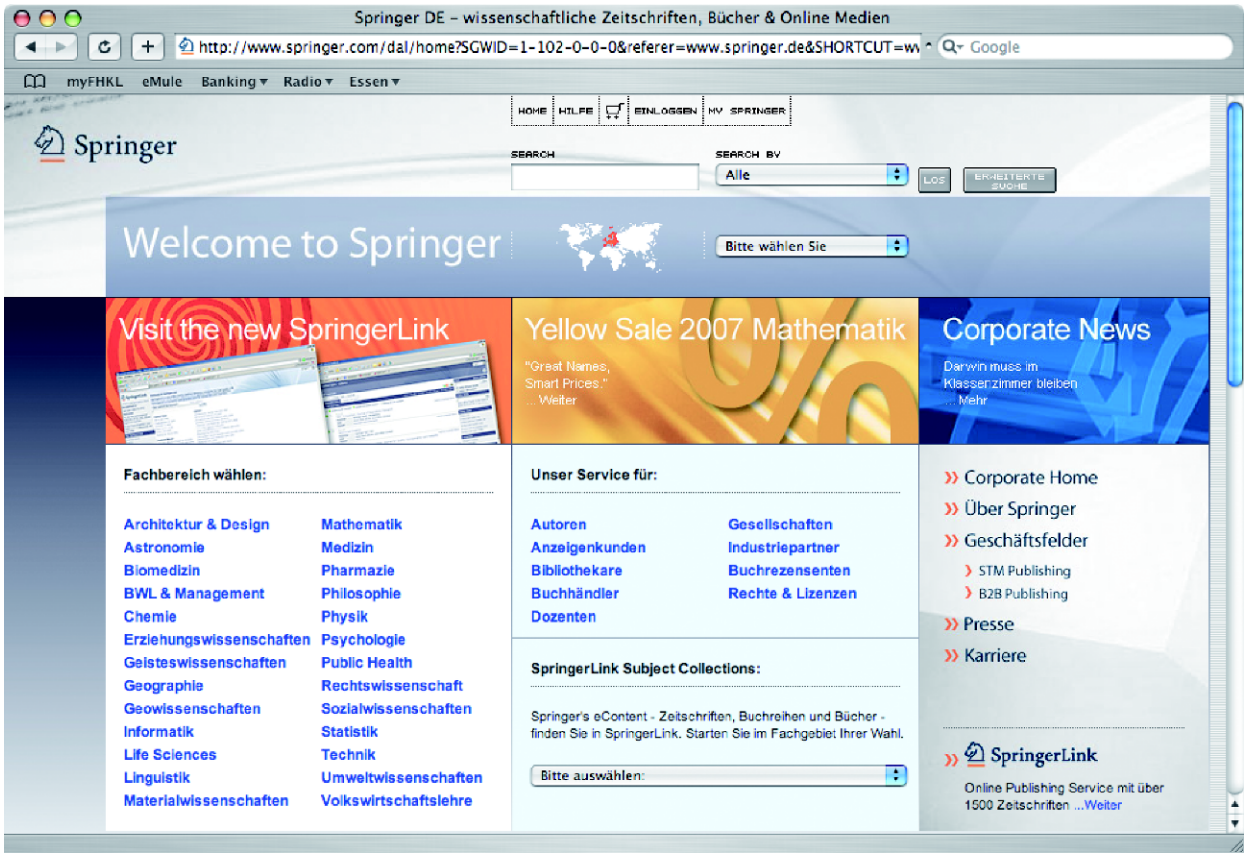



**Abbildung 5.3:**
Apples Safari-Browser auf Mac OS X 10.4

### 5.2.6 Lynx

*Lynx* ist ein ganz besonderer Browser: vollständig zeichenbasiert und damit in einer Kommandoshell ausführbar. Dies ist zum einen für den Systemadministrator von Bedeutung, da sich dieser Browser direkt in einer (SSH-)Shell auf einem entfernten Rechner ausführen lässt, zum anderen erlaubt es Lynx, einen einfachen Test der Barrierefreiheit durchzuführen.

Die Idee des Lynx-Browsers findet sich in einigen anderen Projekten wieder, etwa dem Links2-Browser.

**Der zeichenbasierte Browser Lynx erlaubt eine einfache Beurteilung, inwieweit eine Web-Site barrierefrei ist.**

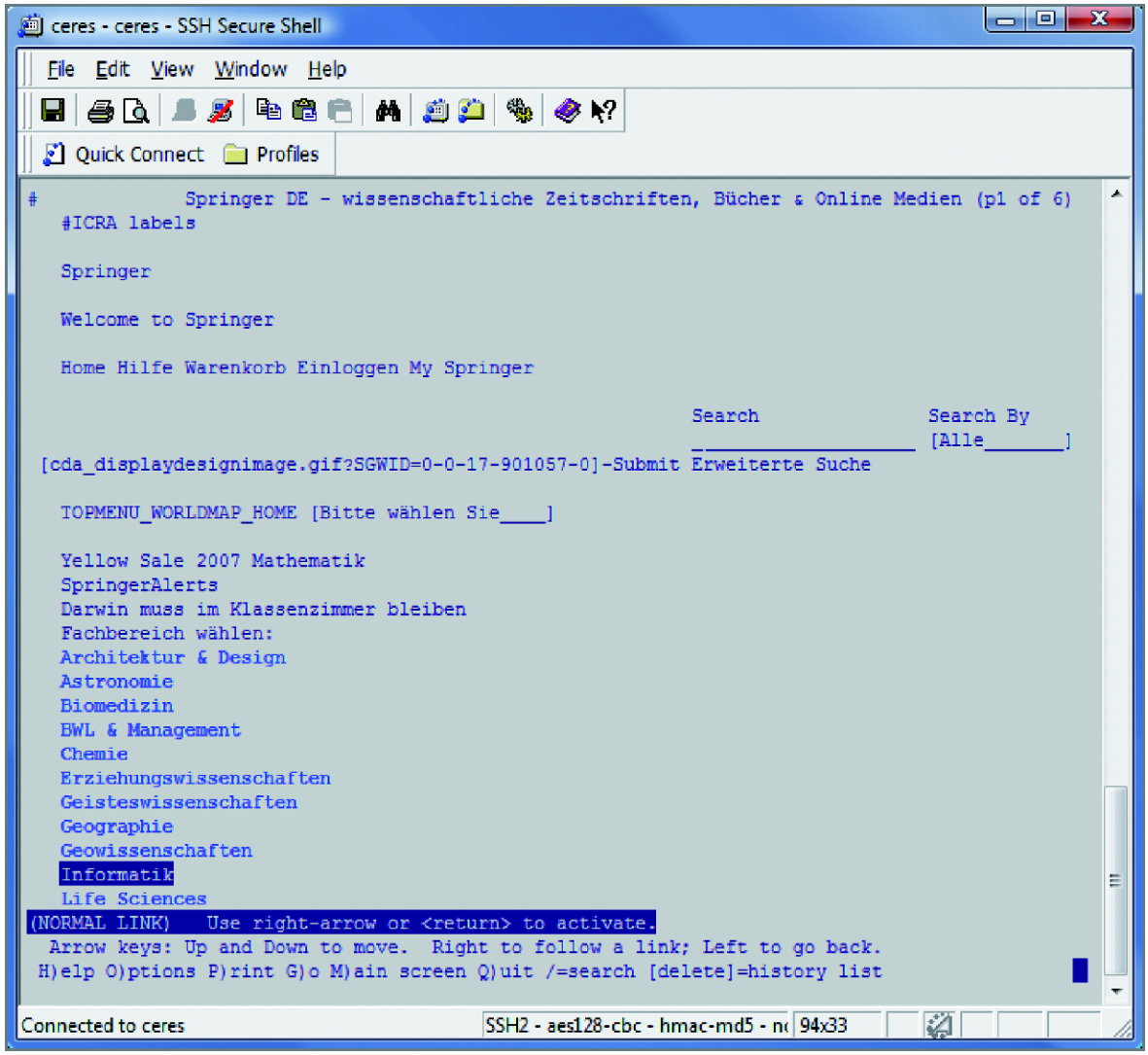



**Abbildung 5.4:** www.springer.de im Lynx-Browser

### 5.2.7 Festival

Eine andere Art von Browsern bilden sprachbasierte Systeme, welche eine Web-Site vorlesen. Hier zeigt sich die Bedeutung einer wirklich barrierefreien Seite nach Abschnitt 2.6 besonders deutlich.

Diese Clients werden als Voice Browser bezeichnet. Anstelle des Einsatzes eines speziellen Browsers kann auch ein allgemeiner Screen Reader verwendet werden, welcher Bildschirminhalte vorliest.

*Festival* [web] ist ein freier, von der Universität Edinburgh entwickelter Screen Reader.

## 5.3 Browser-Tests

Die aktuellen Browser sind leider in den meisten Fällen immer noch fehlerbehaftet, was die Darstellung gängiger Techniken (CSS etc.) anbelangt. Das bedeutet konkret, dass die Vorgaben von (X)HTML und CSS durch das W3C

nicht korrekt von den Browsern umgesetzt werden, was zu sehr hohem Entwicklungsaufwand für ein funktionierendes Web-Design für viele Browsertypen führt.

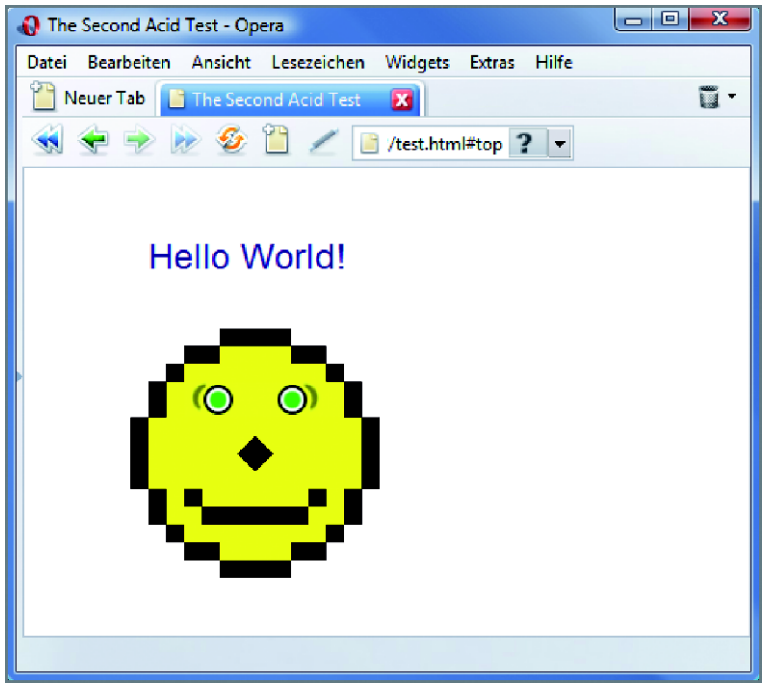



**Abbildung 5.5:** Bestanden: ACID2 in Opera

Ein beliebter Test, um den Browser zu überprüfen, ist ACID2 [web]. Momentan bestehen die aktuellen Versionen des InternetExplorers (Abbildung 5.6) und des Firefox diesen Test nicht (die nachfolgende Firefox-Version soll dies aber leisten); Opera und Safari haben keine Probleme mit dem Test (Abbildung 5.5).

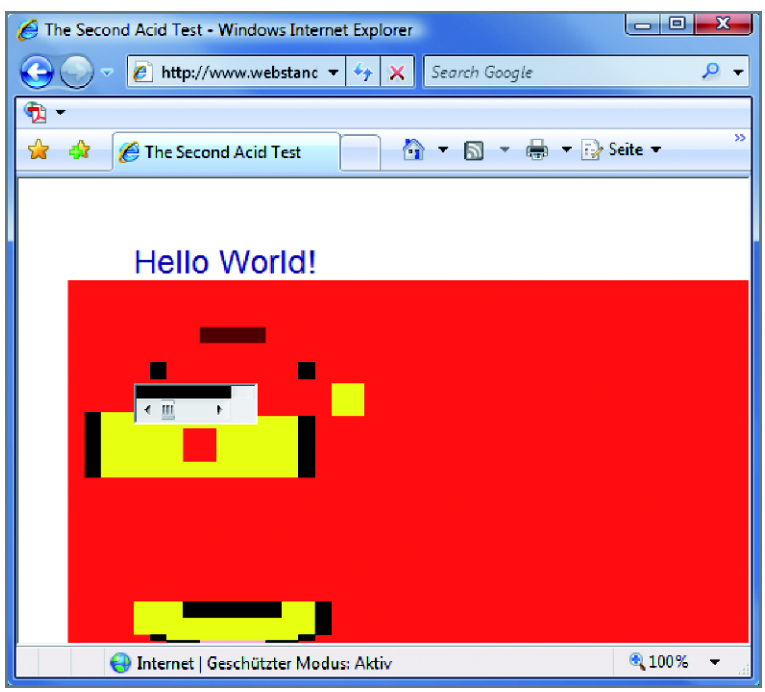



**Abbildung 5.6:** Durchgefallen: ACID2 im InternetExplorer

## 5.4 Die Browser in diesem Buch

Der ACID2-Test nach Abschnitt 5.3 zeigt sehr deutlich, wie unterschiedlich die Browser auch heute noch sind. Wie eingangs beschrieben muss eine Web-Site mit den gängigen und den „erwarteten“ Browsern getestet werden.

In diesem Buch kommen verschiedene Browser gemischt zum Einsatz; die vorgestellten Beispiele sind nicht an einen speziellen Browsertyp gebunden.

# 6

# Der Webserver

Web-Programmierung ist ohne einen *Webserver* nicht möglich, und viele Probleme aus dem Alltag ergeben sich aus einem unvollständigen Verständnis des Webservers.

In diesem Kapitel soll deshalb der Webserver genauer betrachtet und einige Praxis-Tipps für den Webserver gegeben werden.

## 6.1 Aufgaben und Arbeitsweise

Die Aufgabe des Webservers ist einfach zu beschreiben:

**Ein Webserver ist ein Serverdienst, welcher das HTTP-Protokoll (vgl. 1.6.5) bedient.**

Konkret bedeutet dies, dass der Server (typischerweise) den tcp-Port 80 belegt und alle eingehenden/ausgehenden Anfragen, die im HTTP-Protokoll nach 1.6.5 formuliert sein müssen, beantwortet; dazu gehört, dass er die einzelnen HTTP-Direktiven beherrscht und die Anfragen entsprechend verstehen kann.

Eine weitere wichtige Anforderung an den Webserver ist eine einfache Konfigurierbarkeit seiner Eigenschaften. Zu diesen Eigenschaften zählen insbesondere:

- der tcp-Port, auf welchem der Server arbeitet: der Parameter `port` (Default-Wert für HTTP ist 80);
- das Wurzelverzeichnis der Web-Dokumente: dieser Parameter wird als `document-root` bezeichnet; Zugriffe sollen dann nur unterhalb dieses Startverzeichnisses möglich sein.

Neben diesen zwei zentralen Parametern gibt es zahlreiche weitere; die Konfigurationsdatei eines Webservers umfasst häufig weit über 1.000 Zeilen.

## 6.2 Ein einfacher Webserver in Java

Um die grundlegende Funktionsweise eines Webservers zu verstehen, wollen wir zuerst einen solchen selbst implementieren. Überraschender Weise ist dies in modernen Programmiersprachen vergleichsweise leicht, da hier die notwendige Netzwerkfunktionalität direkt zur Verfügung steht; Beispiele für solche Sprachen sind Java und Ruby.

Hier soll ein Webserver in Java implementiert werden. Dieser beherrscht zunächst nur eine einzelne HTTP-Anweisung – `GET` – und nicht das vollständige HTTP. Ferner ist er im Sourcecode einfach konfigurierbar: Sowohl der Netzwerkport als auch der Stammordner der HTML-Dokumente können gewählt werden.

### 6.2.1 Sockets

Zuerst benötigen wir einen Endpunkt für die Netzwerkverbindung: einen *Socket*. Dies lässt sich in Java – aber auch in vielen anderen modernen Sprachen wie Ruby, vgl. Abschnitt 13.5.4 – ausgesprochen leicht realisieren, da im Java-Paket `java.net` Sockets für die wichtigsten Protokolle vorhanden sind.

Als Protokoll auf der Transport-Schicht (ISO/OSI-Schicht vier, vgl. 1.6.1) kommt nur tcp in Frage, da eine technisch gesicherte Übertragung unverzichtbar ist. tcp-Sockets sind in den Klasssen `java.net.ServerSocket` und `java.net.Socket` implementiert (udp-Sockets hingegen in `java.net.DatagrammSocket`).

Die Klasse `ServerSocket` verfügt über zahlreiche (fast zehn) Konstruktoren; außer dem unspezifischen Default-Konstruktor verwenden alle ein integer-Argument für den verwendeten tcp-Port.

Für eine erfolgreiche tcp-Kommunikation ist serverseitig zuerst ein Endpunkt der Kommunikation mit dem Client notwendig, eine Instanz der Klasse `Socket`; diese gewinnen wir nicht über einen Konstruktor, sondern warten auf eine Anfrage und erzeugen damit den Socket. Dafür erzeugen wir auf dem festgelegten tcp-Port eine Instanz der Klasse `java.net.ServerSocket`, auf welche wir die Methode `accept()` anwenden. Diese Methode wartet auf dem entsprechenden Serversocket, bis eine Anfrage erfolgt; dann wird eine Instanz der Klasse `java.net.Socket` erzeugt, über welche die weitere Kommunikation erfolgt.

**Sockets in Java sind mit den Klassen `java.net.Socket` (tcp) und `java.net.DatagramSocket` (udp) einfach zu implementieren.**

### 6.2.2 Beantwortung einer einzelnen Anfrage

Zuerst soll nur eine einzelne GET-Anfrage an den Server beantwortet werden und der Server endet. Dazu erzeugen wir wie beschrieben einen `ServerSocket` und wenden auf diesen die Methode `accept()` an.

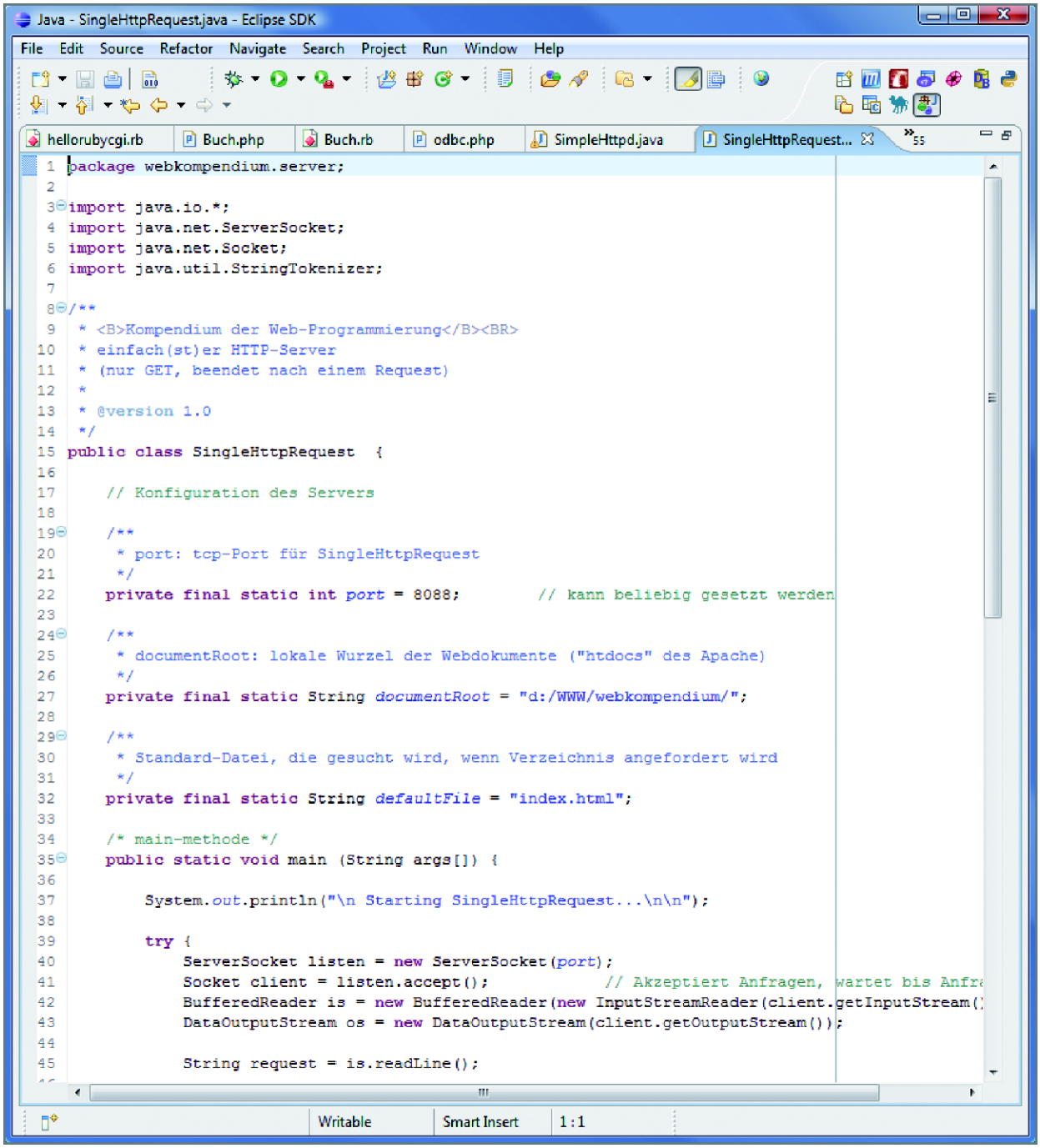
Der über diesen Socket eingehende Request besteht nach dem HTTP-Protokoll aus drei Teilen:

- der HTTP-Anweisung GET;
- dem angeforderten Dokument (URL);
- der verwendeten HTTP-Version (1.0 oder 1.1).

Um dies zu verarbeiten, wird vom eingehenden Socket ein `DataInputStream` gelesen (über einen `BufferedReader`): mittels der Methode `readLine()` wird eine Zeile des `DataInputStream` gewonnen. Diese Zeile wird über `StringTokenizer` nach dem Blank in einzelne Bestandteile aufgespalten. Sind es drei Bestandteile, und das erste lautet „GET“, dann erfolgt eine weitere Verarbeitung: Die definierte Methode `sendDocument` wird aufgerufen und

über diese wird die angeforderte Datei über einen `DataOutputStream` an den Socket und damit zum Client gesendet.

Die Abbildungen 6.1 und 6.2 zeigen die entsprechende Java-Klasse.




```
1  package webkompendium.server;
2
3  import java.io.*;
4  import java.net.ServerSocket;
5  import java.net.Socket;
6  import java.util.StringTokenizer;
7
8  /**
9   * <B>Kompendium der Web-Programmierung</B><BR>
10  * einfach(st)er HTTP-Server
11  * (nur GET, beendet nach einem Request)
12  *
13  * @version 1.0
14  */
15 public class SingleHttpRequest  {
16
17     // Konfiguration des Servers
18
19     /**
20      * port: tcp-Port für SingleHttpRequest
21      */
22     private final static int port = 8088;        // kann beliebig gesetzt werden
23
24     /**
25      * documentRoot: lokale Wurzel der Webdokumente ("htdocs" des Apache)
26      */
27     private final static String documentRoot = "d:/WWW/webkompendium/";
28
29     /**
30      * Standard-Datei, die gesucht wird, wenn Verzeichnis angefordert wird
31      */
32     private final static String defaultFile = "index.html";
33
34     /* main-methode */
35     public static void main (String args[]) {
36
37         System.out.println("\n Starting SingleHttpRequest...\n\n");
38
39         try {
40             ServerSocket listen = new ServerSocket(port);
41             Socket client = listen.accept();            // Akzeptiert Anfragen, wartet bis Anfra
42             BufferedReader is = new BufferedReader(new InputStreamReader(client.getInputStream()
43             DataOutputStream os = new DataOutputStream(client.getOutputStream());
44
45             String request = is.readLine();
```

**Abbildung 6.1:**
Die Java-Klasse `SingleHttpRequest` im Paket `webkompendium.server`, Teil I

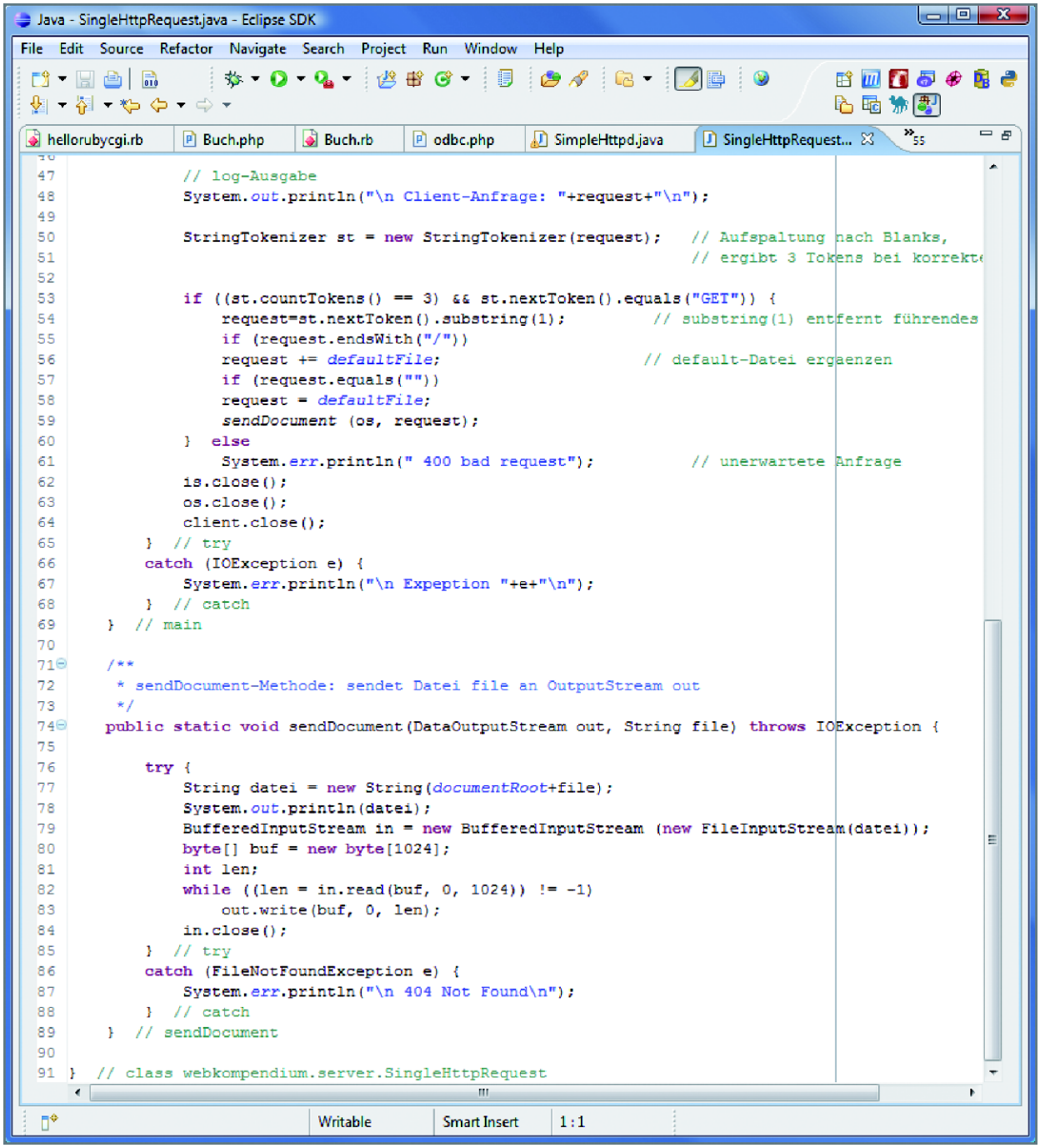



**Abbildung 6.2:** Die Java-Klasse `SingleHttpRequest` im Paket `webkompendium.server`, Teil II

### Grenzen der Anwendung

Diese einfache Java-Applikation erlaubt bereits die Beantwortung eines GET-Requests – mehr aber nicht; die Hauptmängel liegen in folgenden Punkten:

- dem Beenden nach nur einer Anfrage: dies lässt sich durch Einbetten in eine Endlosschleife beheben;
- wie verhält sich die Anwendung bei mehreren parallelen Zugriffen? Dies kann durch den Einsatz von parallel laufenden Threads gelöst werden.

- es wird kein Logfile geschrieben;
- es gibt ein klassisches Sicherheitsproblem: Ein Zugriff über die Adresse der Art `../../etc/passwd` wäre denkbar und die Datei würde – wenn so vorhanden – vom Server ausgeliefert werden.

Dies wird in der nächsten Java-Applikation besser.

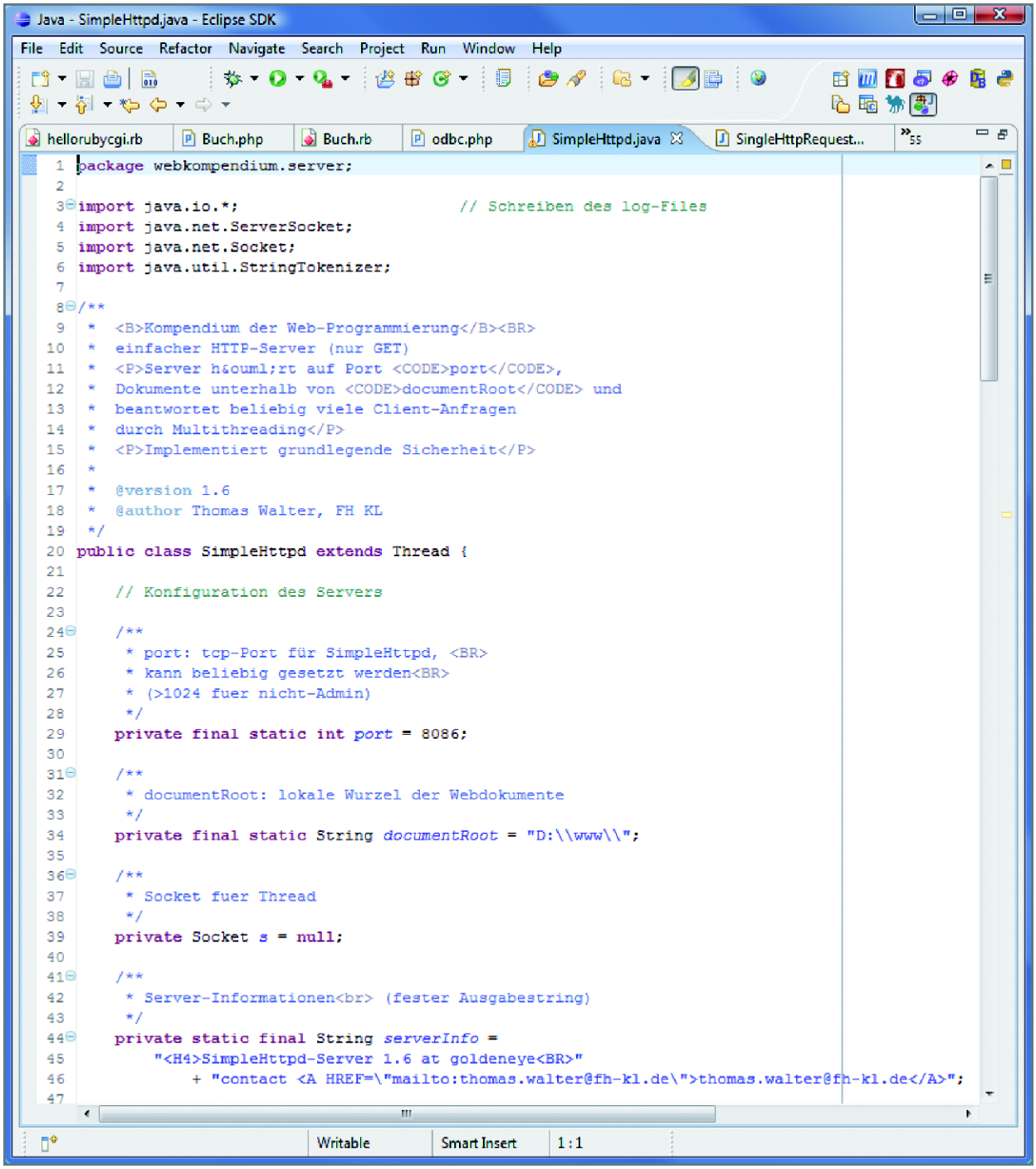
### 6.2.3 Ein einfacher Webserver

Die Klasse `webkompendium.server.SimpleHttpd` liefert ein Grundgerüst für einen tatsächlich funktionierenden Webserver – der in mancher Hinsicht vielen anderen sogar überlegen ist, da er durch seinen minimalistischen Aufbau einen hohen Grad an Sicherheit bietet.

Im Einzelnen erweitert die neue Klasse den bisherigen Server folgendermaßen:

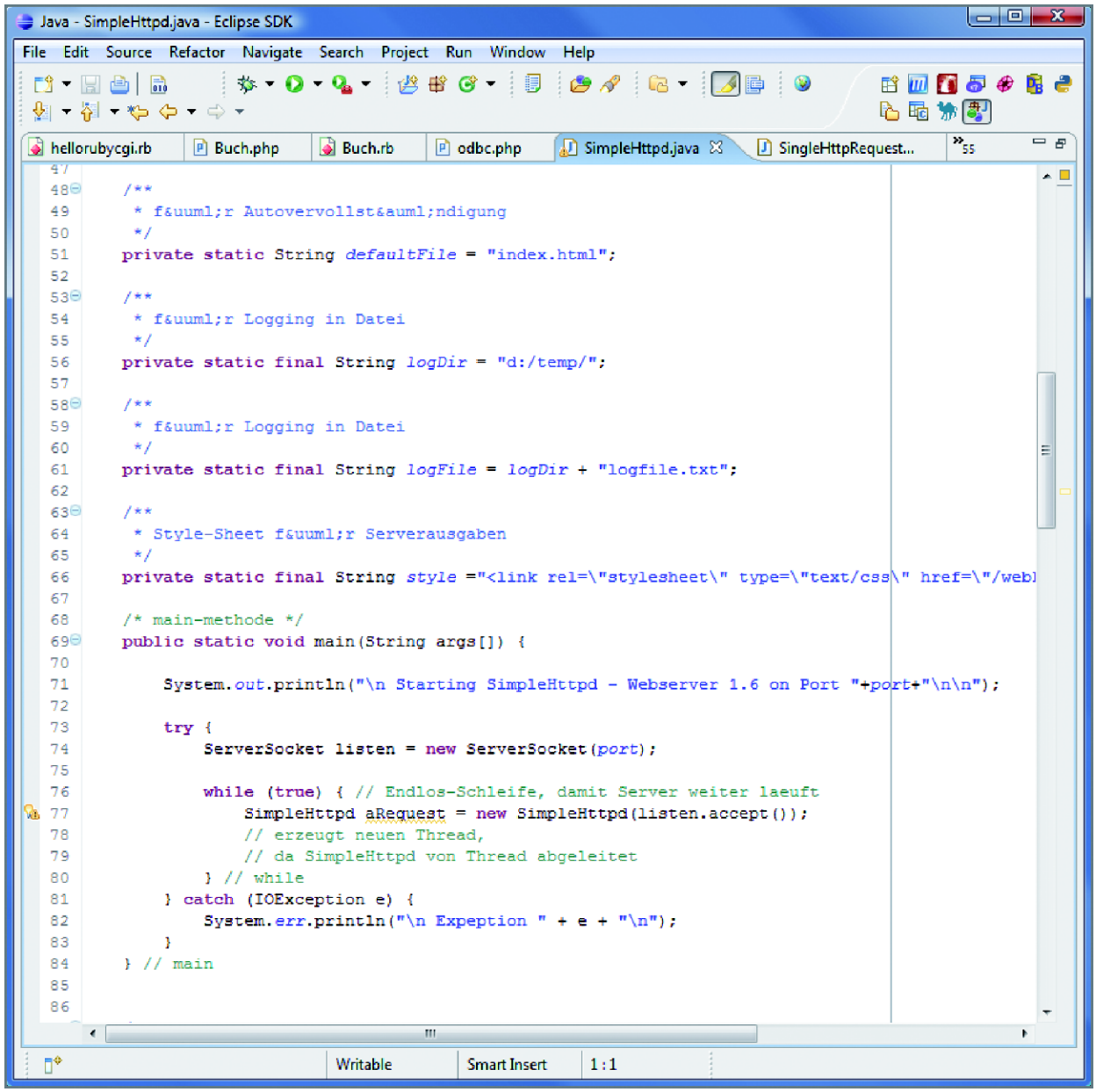
- Durch den Einsatz von Threads werden parallele Zugriffe unterstützt.
- Die Sicherheit – Zugriffsverweigerung auf übergeordnete Verzeichnisse – wird durch „Bereinigung“ der angeforderten URL mittels `canonical path` ermöglicht.
- Eine Endlosschleife verhindert das Beenden.

Die Abbildungen 6.3 bis 6.7 zeigen die entsprechende Java-Klasse.



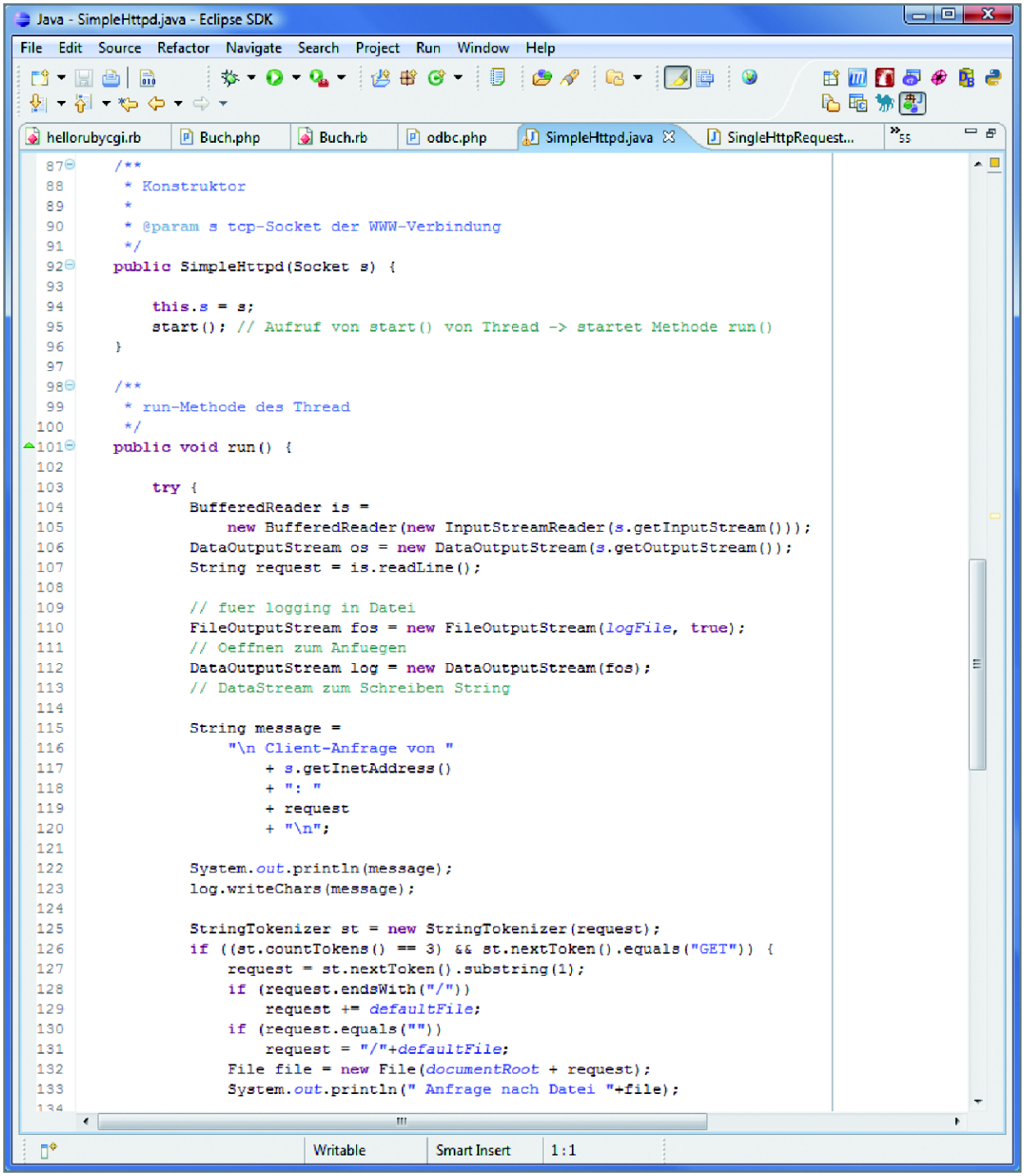

```java
package webkompendium.server;

import java.io.*;                       // Schreiben des log-Files
import java.net.ServerSocket;
import java.net.Socket;
import java.util.StringTokenizer;

/**
 *  <B>Kompendium der Web-Programmierung</B><BR>
 *  einfacher HTTP-Server (nur GET)
 *  <P>Server h&ouml;rt auf Port <CODE>port</CODE>,
 *  Dokumente unterhalb von <CODE>documentRoot</CODE> und
 *  beantwortet beliebig viele Client-Anfragen
 *  durch Multithreading</P>
 *  <P>Implementiert grundlegende Sicherheit</P>
 *
 *  @version 1.6
 *  @author Thomas Walter, FH KL
 */
public class SimpleHttpd extends Thread {

    // Konfiguration des Servers

    /**
     * port: tcp-Port für SimpleHttpd, <BR>
     * kann beliebig gesetzt werden<BR>
     * (>1024 fuer nicht-Admin)
     */
    private final static int port = 8086;

    /**
     * documentRoot: lokale Wurzel der Webdokumente
     */
    private final static String documentRoot = "D:\\www\\";

    /**
     * Socket fuer Thread
     */
    private Socket s = null;

    /**
     * Server-Informationen<br> (fester Ausgabestring)
     */
    private static final String serverInfo =
        "<H4>SimpleHttpd-Server 1.6 at goldeneye<BR>"
            + "contact <A HREF=\"mailto:thomas.walter@fh-kl.de\">thomas.walter@fh-kl.de</A>";

```

**Abbildung 6.3:**
Die Java-Klasse `SimpleHttpd` im Paket `webkompendium.server`, Teil I



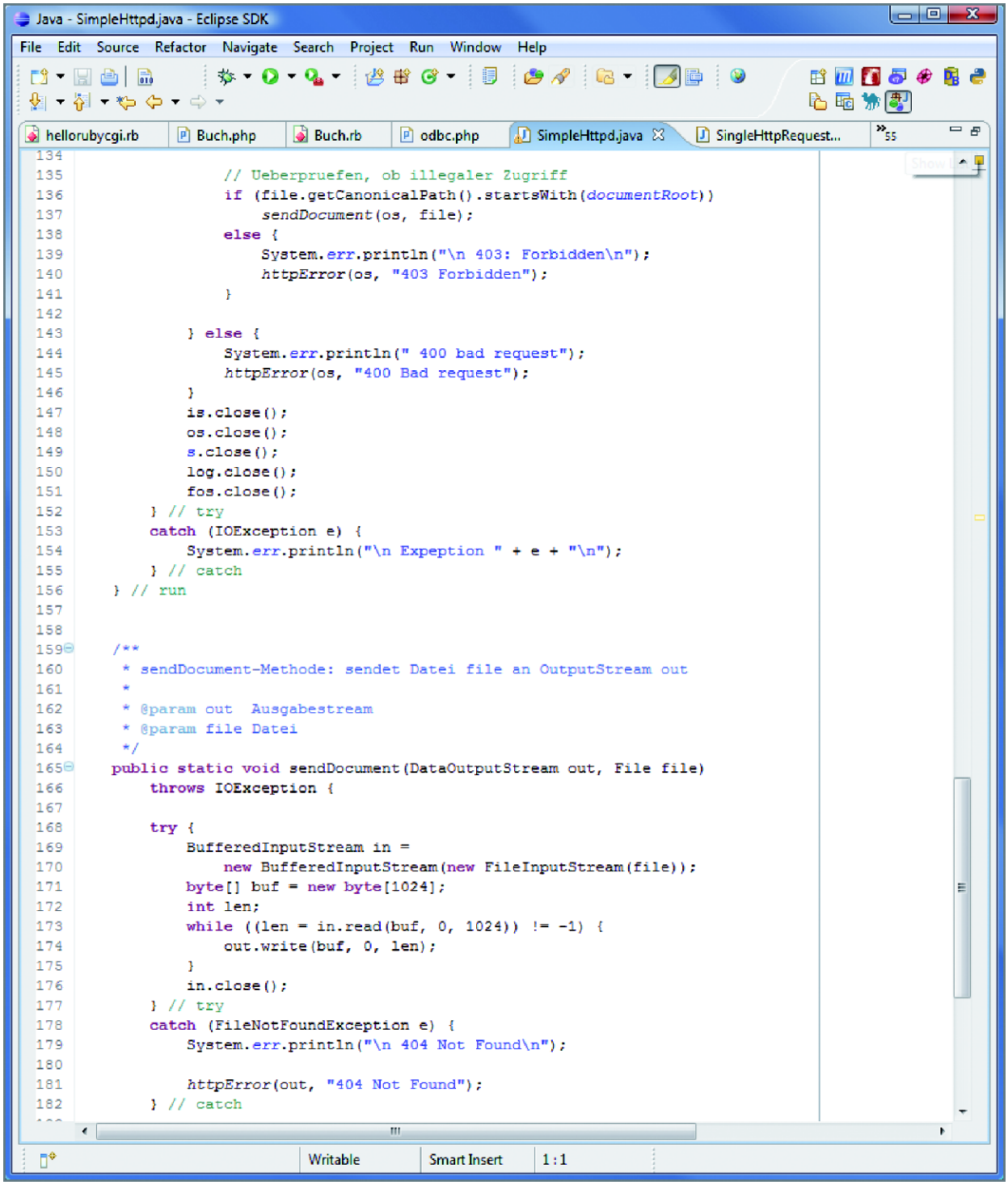

```
47
48    /**
49     * f&uuml;r Autovervollst&auml;ndigung
50     */
51    private static String defaultFile = "index.html";
52
53    /**
54     * f&uuml;r Logging in Datei
55     */
56    private static final String logDir = "d:/temp/";
57
58    /**
59     * f&uuml;r Logging in Datei
60     */
61    private static final String logFile = logDir + "logfile.txt";
62
63    /**
64     * Style-Sheet f&uuml;r Serverausgaben
65     */
66    private static final String style ="<link rel=\"stylesheet\" type=\"text/css\" href=\"/webl
67
68    /* main-methode */
69    public static void main(String args[]) {
70
71        System.out.println("\n Starting SimpleHttpd - Webserver 1.6 on Port "+port+"\n\n");
72
73        try {
74            ServerSocket listen = new ServerSocket(port);
75
76            while (true) { // Endlos-Schleife, damit Server weiter laeuft
77                SimpleHttpd aRequest = new SimpleHttpd(listen.accept());
78                // erzeugt neuen Thread,
79                // da SimpleHttpd von Thread abgeleitet
80            } // while
81        } catch (IOException e) {
82            System.err.println("\n Expeption " + e + "\n");
83        }
84    } // main
85
86
```

**Abbildung 6.4:**
Die Java-Klasse `SimpleHttpd` im Paket `webkompendium.server`, Teil II




```
87    /**
88     * Konstruktor
89     *
90     * @param s tcp-Socket der WWW-Verbindung
91     */
92    public SimpleHttpd(Socket s) {
93
94        this.s = s;
95        start(); // Aufruf von start() von Thread -> startet Methode run()
96    }
97
98    /**
99     * run-Methode des Thread
100    */
101   public void run() {
102
103       try {
104           BufferedReader is =
105               new BufferedReader(new InputStreamReader(s.getInputStream()));
106           DataOutputStream os = new DataOutputStream(s.getOutputStream());
107           String request = is.readLine();
108
109           // fuer logging in Datei
110           FileOutputStream fos = new FileOutputStream(logFile, true);
111           // Oeffnen zum Anfuegen
112           DataOutputStream log = new DataOutputStream(fos);
113           // DataStream zum Schreiben String
114
115           String message =
116               "\n Client-Anfrage von "
117                   + s.getInetAddress()
118                   + ": "
119                   + request
120                   + "\n";
121
122           System.out.println(message);
123           log.writeChars(message);
124
125           StringTokenizer st = new StringTokenizer(request);
126           if ((st.countTokens() == 3) && st.nextToken().equals("GET")) {
127               request = st.nextToken().substring(1);
128               if (request.endsWith("/"))
129                   request += defaultFile;
130               if (request.equals(""))
131                   request = "/"+defaultFile;
132               File file = new File(documentRoot + request);
133               System.out.println(" Anfrage nach Datei "+file);
```

**Abbildung 6.5:**
Die Java-Klasse `SimpleHttpd` im Paket `webkompendium.server`, Teil III




```
134
135                 // Ueberpruefen, ob illegaler Zugriff
136                 if (file.getCanonicalPath().startsWith(documentRoot))
137                     sendDocument(os, file);
138                 else {
139                     System.err.println("\n 403: Forbidden\n");
140                     httpError(os, "403 Forbidden");
141                 }
142
143             } else {
144                 System.err.println(" 400 bad request");
145                 httpError(os, "400 Bad request");
146             }
147             is.close();
148             os.close();
149             s.close();
150             log.close();
151             fos.close();
152         } // try
153         catch (IOException e) {
154             System.err.println("\n Expeption " + e + "\n");
155         } // catch
156     } // run
157
158
159     /**
160      * sendDocument-Methode: sendet Datei file an OutputStream out
161      *
162      * @param out  Ausgabestream
163      * @param file Datei
164      */
165     public static void sendDocument(DataOutputStream out, File file)
166         throws IOException {
167
168         try {
169             BufferedInputStream in =
170                 new BufferedInputStream(new FileInputStream(file));
171             byte[] buf = new byte[1024];
172             int len;
173             while ((len = in.read(buf, 0, 1024)) != -1) {
174                 out.write(buf, 0, len);
175             }
176             in.close();
177         } // try
178         catch (FileNotFoundException e) {
179             System.err.println("\n 404 Not Found\n");
180
181             httpError(out, "404 Not Found");
182         } // catch
```

**Abbildung 6.6:**
Die Java-Klasse `SimpleHttpd` im Paket `webkompendium.server`, Teil IV

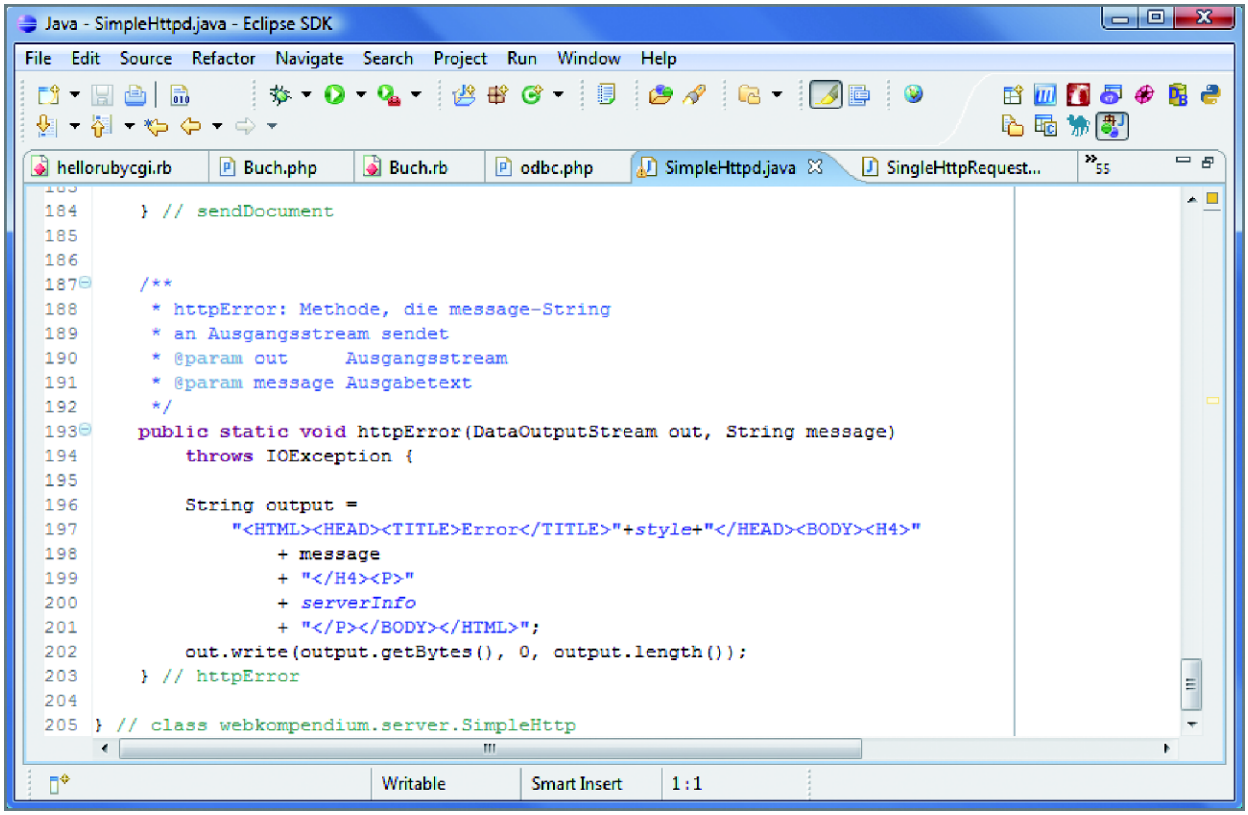



**Abbildung 6.7:** Die Java-Klasse `SimpleHttpd` im Paket `webkompendium.server`, Teil V

Diese Klasse `webkompendium.server.SimpleHttpd` ist ähnlich zum einfachen `SingleHttpRequest` aufgebaut, allerdings nun von der Klasse `java.lang.Thread` abgeleitet. Die main-Methode bettet das Abfangen der eingehenden Anfragen in eine Endlosschleife `while(true)` ein; in dieser Schleife wird wieder die Methode `accept()` auf einen `ServerSocket` angewendet; bei eingehender Anfrage wird der über dieses `accept` gewonnene Socket als Argument dem Konstruktor der Klasse `SimpleHttpd` selbst übergeben. Der Konstruktor der Klasse belegt das private Attribut für den Socket mit diesem Argument und ruft die Methode `start()` des Threads auf, wodurch letztlich die Methode `run()` aufgerufen wird. `run()` überschreibt die entsprechende Methode der Klasse `java.lang.Thread` und geht jetzt ähnlich vor wie die main-Methode der Klasse `SingleHttpRequest`.

Zur Erleichterung und Erweiterung sind in die Klasse `webkompendium.server.SimpleHttpd` weitere Methoden integriert, etwa `httpError(...)` für die Generierung einer Fehlerseite.

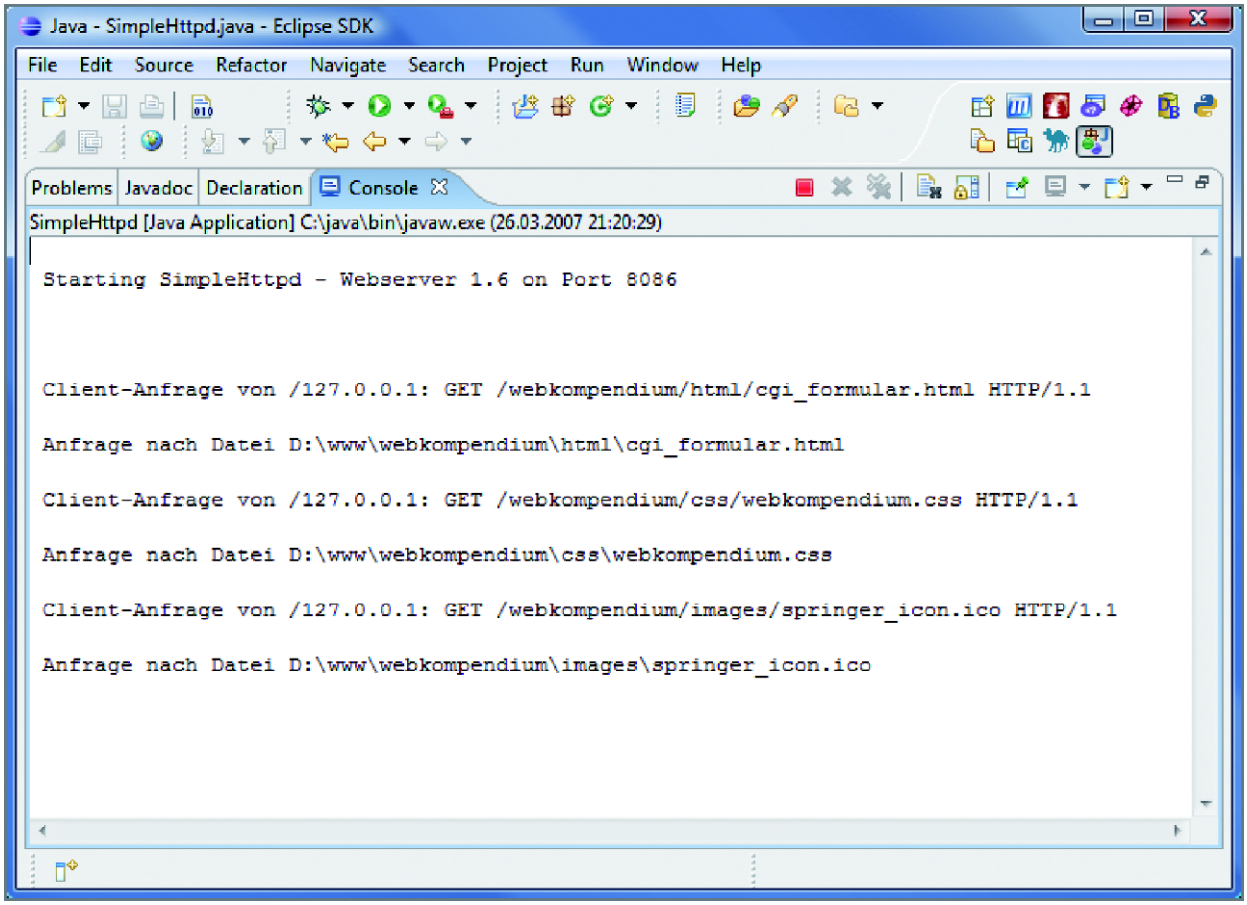



**Abbildung 6.8:**
Ausführen von `webkompendium.server.SimpleHttpd`

Mit der Klasse `webkompendium.server.SimpleHttpd` liegt ein im wesentlichen funktionierender Webserver vor; was noch fehlt ist etwa folgende Funktionalität:

- Schreiben eines besseren Logfiles;
- Auslagerung der Konfiguration in eine externe Datei (etwa in XML-Form);
- Wird nur ein Verzeichnis angefragt, in dem eine index.html nicht vorhanden ist, sollte ein Directory-Listing ausgegeben werden;

Das ist alles nicht mehr viel, was zeigt, wie effizient die modernen Programmiersprachen in der Netzwerkanbindung sind.

## 6.3 Der Apache Webserver

Der erste Webserver ist naturgemäß so alt wie das Word Wide Web: Tim Berners-Lee entwickelte am CERN den CERN-Webserver (Abbildung 6.10). Der Apache-Webserver [web] basiert auf dem von Rob McCool um 1995 entwickelten NCSA-Webserver, dem Nachfolger des CERN-Servers. Nachdem McCool das NCSA verlassen hat, erfolgte die Fortentwicklung dieses Servers in verschiedenen Gruppen, die jeweils Patches geliefert haben, was zu dem Namen des beliebten Webservers geführt hat:

„a patchy server“ = Apache

**Abbildung 6.9:**
Apache Software Foundation

**Abbildung 6.10:** Immer noch online: der CERN-Webserver unter `http://www.w3c.org/Daemon`

Bereits im Dezember 1995 wurde die offizielle Version 1.0 des Apache freigegeben, und seit Ende 1996 ist Apache der weltweit am meisten eingesetzte Webserver.

Apache hat seither seinen Erfolg kontinuierlich fortgesetzt und ist nach der verbreiteten Netcraft-Analyse [web] heute mit einem Marktanteil deutlich über 70 % stark vertreten (Abbildung 6.11).

Die Gründe für diesen großen und anhaltenden Erfolg des Apache sind vielfältig. Zum einen ist die freie (auch für den kommerziellen Einsatz) Lizenz zu nennen, aber auch die Verfügbarkeit für praktisch alle Betriebssysteme bei weitgehend gleichartiger Konfiguration ist wichtig. Außerdem ist der Apache ausgesprochen stabil und durch sein Modulkonzept flexibel erweiterbar.

**Es gibt – wenn überhaupt – nur sehr wenige Gründe, die für Alternativen zum Apache-Webserver sprechen.**

### 6.3.1 Versionen des Apache

Momentan sind drei Hauptversionen des Apache von Relevanz: der „Klassiker“ 1.3, die „Normalversion“ 2.0 und die neue Version 2.2. Die Beispiele in

diesem Buch beziehen sich auf das aktuelle Release des Apache 2.2, sind aber auf den anderen Versionen auch einsetzbar; auf 1.3 wurden die meisten Fälle explizit getestet.

Obwohl es die neuere Version schon seit einigen Jahren gibt, wird die alte ebenfalls weiter gepflegt. Während die Version 2 die modernere Architektur und etwas einfachere Konfiguration bietet (etwa sind keine `AddModule`-Regeln mehr nötig), gibt es nach wie vor einige Zusatzmodule, die nur für 1.3 zur Verfügung stehen.

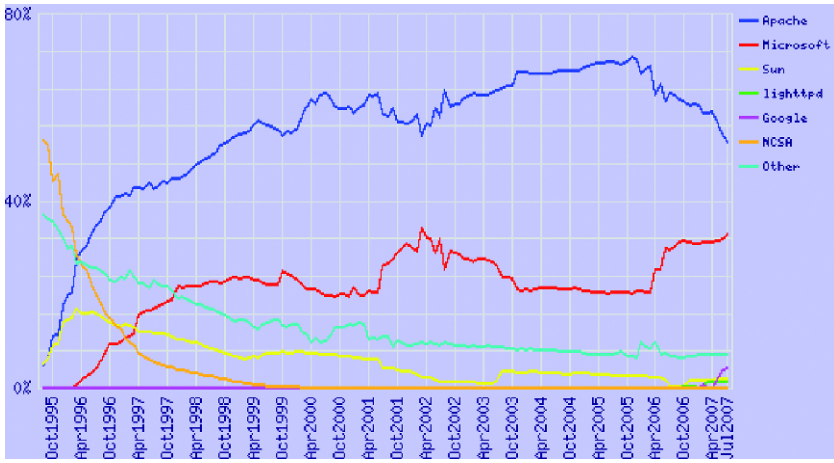



**Abbildung 6.11:** Verteilung der Webserver (Quelle: Netcraft)

Startpunkt aller Auseinandersetzung mit Apache ist das umfassende Web-Angebot unter `http://www.apache.org`; dort findet sich neben allen Sourcen und auch bereits compilierten Versionen insbesondere eine umfangreiche Online-Dokumentation zu dem Webserver, die für die tägliche Arbeit des Webmasters unerlässlich ist.

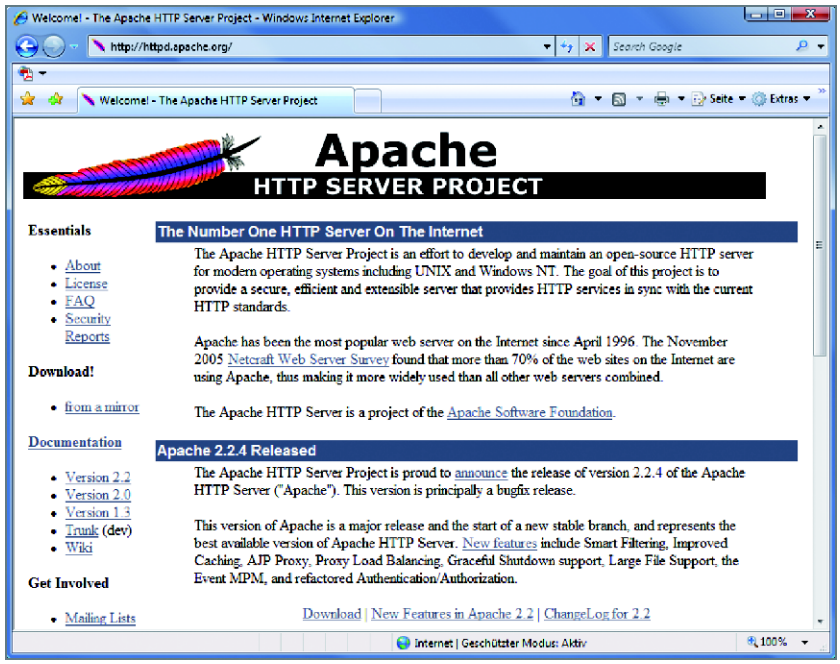



**Abbildung 6.12:** Der Apache-Server im Web

### 6.3.2 Architektur

Apache in der Version 2 (und neuer, sowie unter Windows auch schon die Version 1.3) verwendet das Thread-Modell für die Beantwortung von Anfragen: Beim Starten des Webservers wird der Root-Prozess gestartet, welcher in Threads in einer konfigurierbaren Benutzerkennung die weiteren Anfragen beantwortet.

Zentral für das Verständnis des Apache ist sein Modulkonzept (vgl. 6.3.7): Die Kernfunktionalität des Servers kann durch Hinzufügen von Modulen wesentlich erweitert werden. Hierzu zählen zunächst die Standard-Module, welche von der Apache-Group herausgegeben werden, und dann Module von Drittanbietern wie PHP.

### 6.3.3 Installation

Die Installation des Apache ist völlig klassisch: Unter Windows eine msi-Installerdatei, unter Unix/Linux entweder eine Paketinstallation oder die Installation „von Hand“ durch Herunterladen und Entpacken der Sourcen, Konfiguration und Compilation.

#### 6.3.3.1 Installation unter Unix/Linux

Unter Unix/Linux stehen prinzipiell zwei Wege bereit: Die Installation eines für das jeweilige System bereitstehenden Paketes mit Tools wie `pkgadd`, `aptitude` oder `yast` oder das Vorgehen „von Hand“: Entpacken, Konfigurieren, Compilieren und Installieren durch Kopieren.

Während Entpacken aus der tar.gz- oder bz2-Distribution und das Compilieren über `make` sowie das Installieren/Kopieren über `make install` einfach ist, liegt in der Konfiguration eine ausgesprochene Besonderheit des Apache: Hier stehen sehr zahlreiche festzulegende Parameter bereit. Zum Kern der zu konfigurierenden Parameter gehören die zum Server gehörenden Apache-Module (vgl. 6.3.7).

Eine Übersicht über die zahlreichen Möglichkeiten der Konfiguration ergibt (Abbildung 6.13)

```
configure --help
```

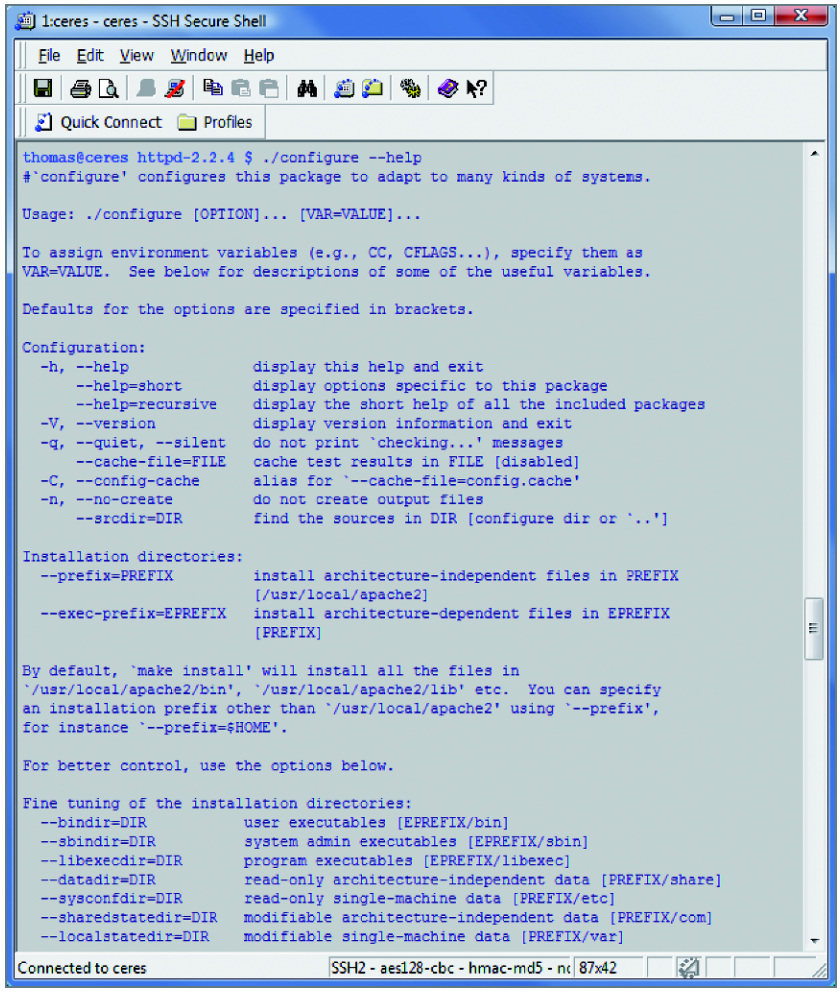



**Abbildung 6.13:** `configure --help` gibt eine Übersicht über die Parameter für das Compilieren

**Wichtig für eine optimale Installation des Apache ist der sinnvolle Einsatz der Compiler-Flags: Durch Angabe des Optimierungsgrades, des CPU-Designs und weiterer Parameter für den C-Compiler kann der erzeugte Binärcode wesentlich an das konkrete System angepasst werden.**

Nach der Konfiguration für die Installation steht das eigenliche Compilieren und das Installieren/Kopieren wie unter Unix üblich an:

```
make            ### Compilieren
make install    ### Installieren/Kopieren
make clean      ### Loeschen
```

#### 6.3.3.2 Installation unter Windows

Die Installation unter Windows ist im Kern herzlich einfach: Der Apache wird mit einer msi-Installer-Datei ausgeliefert, die auszuführen ist. Dabei fragt der Apache einige grundlegende Parameter ab: Ob es eine Systeminstallation (als

NT-Dienst und auf Port 80) oder eine benutzerspezifische Installation (als Anwendung auf Port 8080) ist, die wichtigsten Pfade und einige zentrale Parameter wie die E-Mail-Adresse des Administrators, die gleich in die Konfiguration geschrieben wird.

Unter Windows Vista ist momentan die Situation etwas schwieriger, wenn das Rechtemanagement verwendet wird: Die Installation läuft weitgehend problemlos, allerdings ist es nicht möglich, die NT-Dienste zu registrieren. Der Ausweg kann darin bestehen, eine Shell als Administrator zu öffnen und aus dieser heraus die Apache-Installation zu starten.

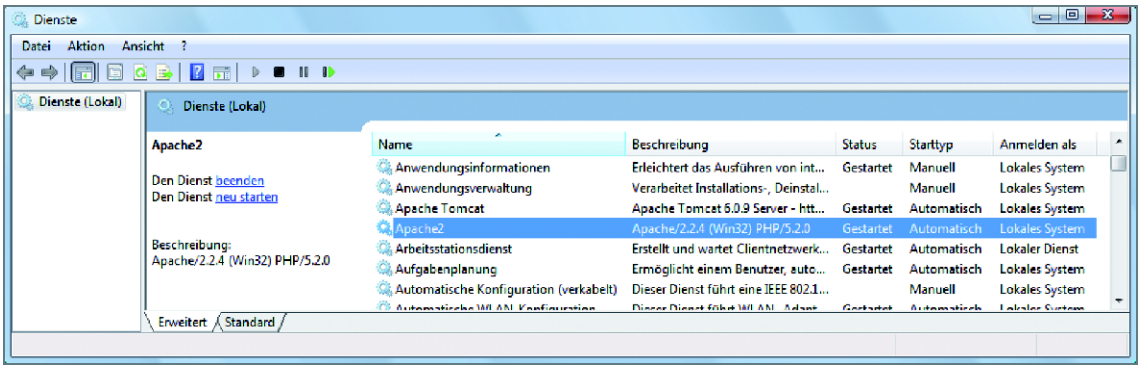



**Abbildung 6.14:**
Der Apache-Dienst unter Vista

#### 6.3.3.3 Apache Friends

Eine einfache Variante der Installation von „fast allem“ rund um die Web-Programmierung ist XAMPP (vgl. 3.4): Hier wird unter Windows ein Paket mit Apache, der MySQL-Datenbank, PHP und Perl auf einem Schritt installiert; dies ist schön, wenn man in Zeitnot lebt, bietet sich aber im allgemeinen nicht an, da bei der Installation der Einzelkomponenten vieles besser gemacht werden kann. Von daher ist dieser Weg – selbst für reine Entwicklungssysteme – nicht empfehlenswert.

### 6.3.4 File-System des Apache

Nach der Installation hat der Apache – wird er in einem einzelnen Verzeichnis wie `/user/local/apache` installiert – eine typische Ordnerstruktur (unter Unix/Linux genauso wie unter Windows):

- `bin` für ausführbare Programme einschließlich des eigentlichen Apache;
- `cgi-bin` für CGI-Scripte (vgl. Kapitel 9);
- `conf` für die Konfigurationsdateien;
- `htdocs` als Ordner für HTML-Dokumente;
- `logs` zum Ablegen der vom Server geschriebenen Logfiles.

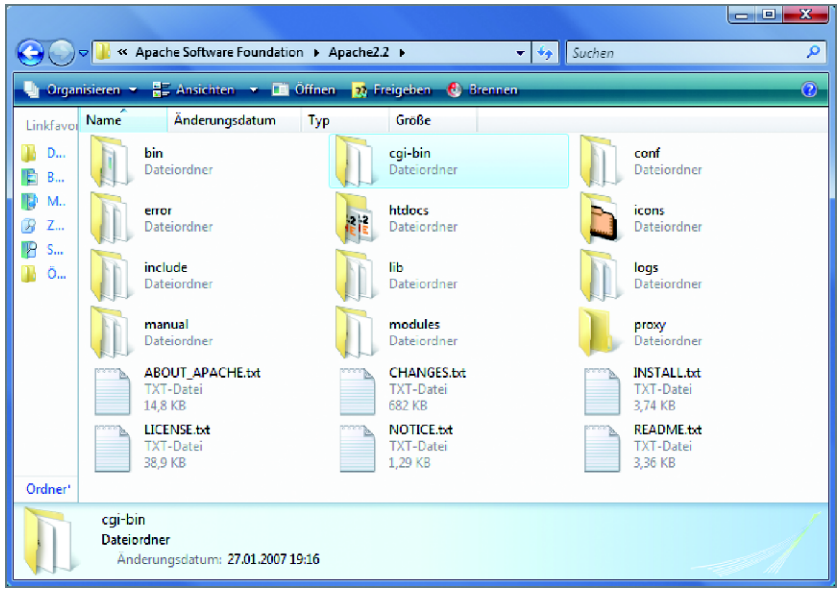



**Abbildung 6.15:**
Apache-Verzeichnisstruktur

Viele Linux-Derivate wie Debian verteilten die Installation auf verschiedene Systemverzeichnisse, etwa

- `/usr/sbin` für das Apache-Programm;
- `/etc/httpd` für Konfigurationsdateien;
- `/var/log/httpd` für Logfiles;
- `/usr/local/httpd/htdocs` für die Dokumente.

### 6.3.5 Konfiguration

Nach der Installation und vor dem eigentlichen Betrieb – aber immer auch während dessen – ist der Apache-Webserver grundlegend zu *konfigurieren*. Dies geschieht typischerweise in der zentralen Konfigurationsdatei `httpd.conf` oder in neueren Versionen in der Datei `apache2.conf`. Diese fasst die früher beim NCSA-Webserver üblichen Konfigurationsdateien `srm.conf` und `access.conf` zusammen und hat schon im Grundzustand eine Länge von rund 1.000 Zeilen; umfassende Webserver haben eine wesentlich größere Konfigurationsdatei.

Die Konfigurationsdatei umfasst drei grundlegende Bereiche:

- zentrale Parameter, die den gesamten Webserver betreffen;
- Konfigurationen für den Hauptserver;
- Konfigurationen der virtuellen Hosts.

Allgemein sind drei Punkte zu bemerken:

- die Konfigurationsdatei httpd.conf besteht aus sehr vielen Kommentaren – eingeleitet durch # –;
- alle verwendeten Regeln sind case-sensitiv;
- auch unter Windows wird der Slash / und nicht \ als Pfadtrenner verwendet.

Die Regeln innerhalb der httpd.conf werden üblicherweise als Direktiven bezeichnet.

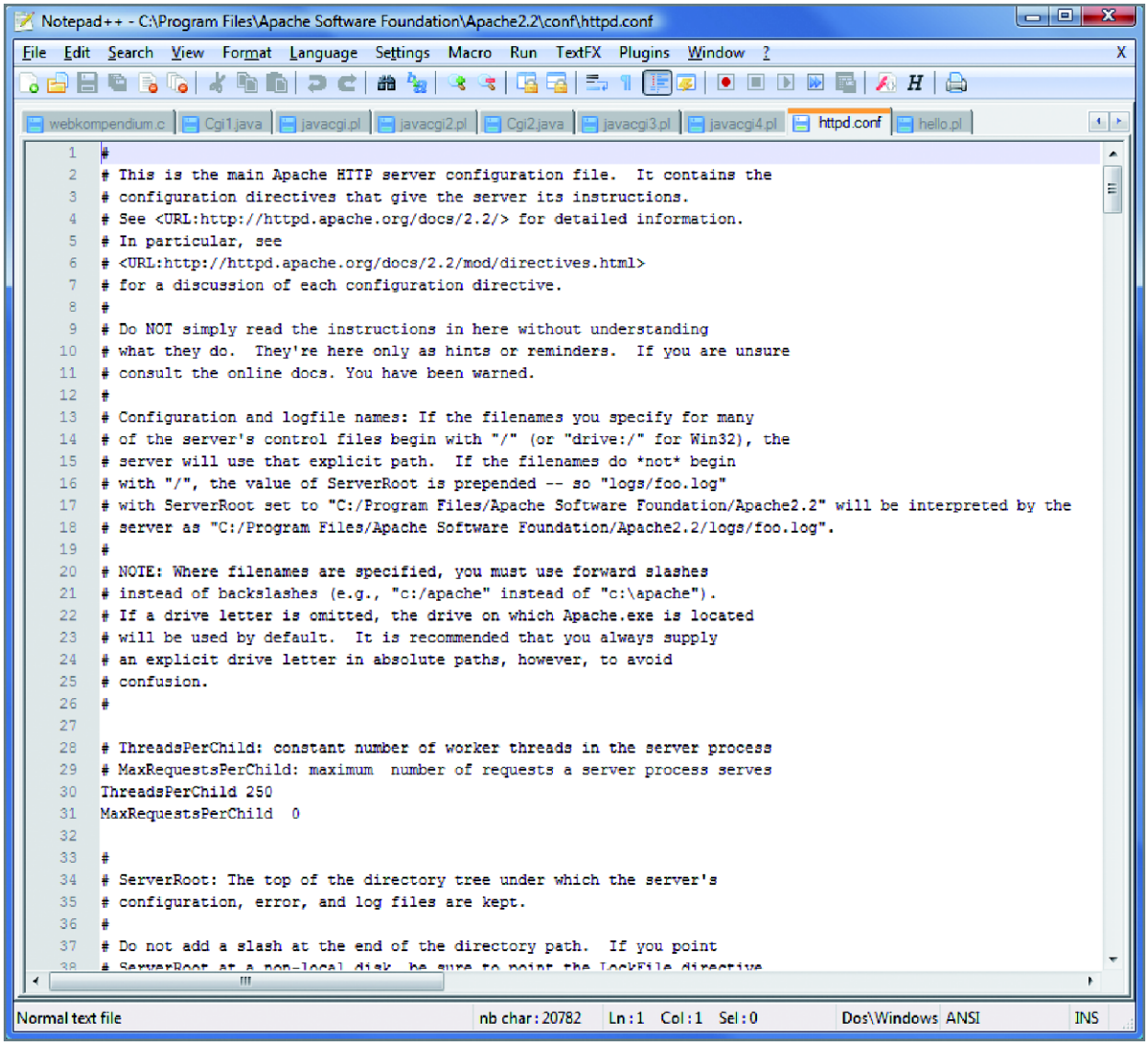



**Abbildung 6.16:** Anfang der `httpd.conf`

### 6.3.5.1 Globale Konfiguration

Wesentliche Parameter des ersten Abschnitts der httpd.conf sind die globalen Parameter, die den gesamten Webserver betreffen. Die wichtigsten hiervon sind:

- Die Prozesssteuerung: Angabe der Höchstgrenze der Threads, die ein Kindprozess beantworten kann, bevor er neu gestartet wird.
- `ServerRoot`: Angabe des Pfades zur Apache-Installation.
- `Listen`: Angabe des tcp-Ports.
- `LoadModule`-Liste: Angabe der Module, die dynamisch (vgl. 6.3.7) geladen werden.

Wesentlicher Bestandteil der globalen Konfiguration ist das Laden der Module (vgl. 6.3.7); dies geschieht bei den neueren Versionen des Apache nur noch dynamisch mittels der LoadModule-Direktive

```
LoadModule alias_module
                /usr/lib/apache2/modules/mod_alias.so
```

Die einzelnen Load-Anweisungen werden inzwischen bei Apache unter Unix/Linux in einen speziellen Unterordner `modules-enabled` ausgelagert; dort werden auch spezifische Konfigurationen zu den Modulen systematisch gesammelt (vgl. 6.3.7).

#### 6.3.5.2 Konfiguration des Hauptservers

Der *Hauptserver* ist derjenige, der die Anfrage an `http://<rechnername>/` auf dem zentralen Port, der bei den allgemeinen Parametern festgelegt wurde, beantwortet.

Hier ist zunächst der Administrator durch die Direktive

```
ServerAdmin mailadresse@server
```

festzulegen, sowie anschließend der Name des Servers.

Zentral ist anschließend die Definition der Wurzel des Verzeichnisbaums für die Webdokumente: der Parameter `DocumentRoot`. Zugriffe sind dann nur unterhalb dieses Verzeichnisknotens möglich.

Danach können für alle Verzeichnisse – teilweise auch für einzelne Dateien – Rechte gesetzt werden. Diese Einzeldirektiven arbeiten hierarchisch: Was in einem oberen Knoten gilt, gilt auch in allen unteren – es sei denn, es wird dort abweichend definiert.

Für den Zugriff auf Verzeichnisse gibt es zwei Möglichkeiten:

- Über den Block `Directory`:

```
<Directory "/data/www">
        Options Indexes
        AllowOverride None
        Order allow,deny
        Allow from all
</Directory>
```

- `Location`: Während der Directory-Block eine absolute Pfadangabe relativ zum gesamten File-System nutzt, verwendet `Location` die URL, löst also relativ zum zuvor angegebenen `DocumentRoot` die Adresse auf; dies kann den Administrationsaufwand deutlich reduzieren.
- Eine dritte Möglichkeit ist `File`, wenn Regeln für einzelne Dateien gesetzt werden; diese gelten dann nicht für nachfolgende Verzeichnisse.

In den jeweiligen Verzeichnissen stehen viele Konfigurationsmöglichkeiten zur Verfügung; die wichtigsten sind:

- `Options`
  - `None` verbietet alles;
  - `Indexes` erlaubt das Directory-Listing;
  - `FollowSymLinks` erlaubt die Verwendung von symbolischen Links auf Unix/Linux, die dann auch aus dem Webverzeichnisbaum herausspringen können (Vorsicht, hier kann die Sicherheit gefährdet sein);
  - `ExecCGI` erlaubt das Ausführen von CGI-Scripten in diesem Ordner (auch hier ist die Sicherheit zu bedenken);
  - `Includes` erlaubt SSI in diesem Ordner und allen Unterordnern (vgl. Kapitel 14);

  - `IncludesNOEXEC` erlaubt SSI ohne das Ausführen mit der `EXEC`-Direktive;
  - `All` beinhaltet `Indexes`, `FollowSymLinks`, `ExecCGI` und `Includes`; dies ist der Default-Fall.
- Mittels `Allow` und `Deny` können Rechner/Netzwerke zugelassen und verboten werden; dies kann sowohl über den Namen als auch über die ip-Adresse erfolgen; `Order allow,deny` (oder umgekehrt) legt die Reihenfolge der Überprüfung fest, wobei anders als beim Cisco IOS der letzte Treffer zählt.

Mittels `ServerSignature` kann der Hauptserver – und dann entsprechend die virtuellen Server – eine individualisierte Ausgabe erzeugen.

Eine häufig angewendete Direktive ist `Alias`, durch welche eine Weiterleitung einer URL erfolgen kann; leistungsfähiger ist noch das Standard-Modul mod_rewrite, hier können mittels regulärer Ausdrücke komplexere Umschreibungen der URL erfolgen.

#### 6.3.5.3 Konfiguration der virtuellen Hosts

Neben dem Hauptserver kann jeder moderne Apache beliebig viele weitere Server hosten. Grundlage hierfür ist insbesondere die Möglichkeit des HTTP-Protokolls in der Version 1.1, bei dem nach dem GET-Request der Name der angefragten Domain übermittelt wird. Eine weitere Möglichkeit ist das Binden des virtuellen Servers an eine ip-Adresse oder einen speziellen tcp-Port.

#### 6.3.5.4 Testen der Konfiguration

Die jeweilige Konfiguration des Apache kann einfach getestet werden:

- unter Unix mit `apachectr -configtest`;
- unter Windows je nach Version mittels `http.exe -t` oder `apache.exe -t`.

### 6.3.6 Starten des Apache

Gestartet wird der Apache unter Unix mit dem Dienstprogramm `apachectr`, welches mehrere Möglichkeiten anbietet: `-start`, `-stop` und `-restart` sind die wichtigsten.

Unter Windows sollte Apache als NT-Dienst registriert sein und über die Dienstesteuerung gestartet werden (Abbildung 6.14).

### 6.3.7 Das Modulkonzept des Apache

Module bei modernen Apache werden dynamisch geladen, also nicht statisch gelinkt. Welche Module hinzugefügt werden, entscheiden die `LoadModule`-Direktiven der Apache-Konfiguration.

Ein modularer Apache in Linux-Distributionen verwendet zwei Unterordner für die Konfiguration der dynamischen Module:

- `mods-available` mit den verfügbaren Modulen;
- `mods-enabled` mit den konkret verwendeten Modulen.

In mods-enabled finden sich nur symbolische Links zum Laden und Konfigurieren der Module. Die Anweisung zum Laden des jeweiligen Moduls ist die LoadModule-Direktive, welche beispielsweise für das Modul mod_alias dann lautet:

```
LoadModule alias_module
                /usr/lib/apache2/modules/mod_alias.so
```

#### 6.3.7.1 Standardmodule

Das Modulkonzept des Apache ist ausgesprochen vielfältig; als *Standard-Module* werden die Module bezeichnet, die direkt von der Apache Group herausgegeben werden; für den aktuellen Apache 2.2 sind dies rund 40 verschiedene Module.

Der Bereich der Kernmodule umfasst neun Stück, darunter das zentrale Modul core sowie spezielle Module für optimierte Multiprozessor-Nutzung.

Zu den weiteren Modulen gehören viele, die für einen normalen Serverbetrieb unerlässlich sind, etwa mod_cgi für CGI-Funktionalitäten. Besondere Bedeutung kommt mod_so zu, da dieses Modul für das dynamische Laden weiterer Module verantwortlich ist.

**Der Apache-Webserver kann durch die Apache-Standardmodule wesentlich in seinem Funktionsumfang verändert werden; mehr Informationen dazu sind online etwa unter** `http://httpd.apache.org/docs/2.2/mod/` **zu finden.**

#### 6.3.7.2 mod_php

Zu den wichtigsten Modulen, welche nicht zur Gruppe der Standard-Module gehören, zählt das PHP-Modul *mod_php*. Zwar kann PHP auch als cgi betrieben werden (vgl. Kapitel 11), aus Performance-Gründen soll es aber unbedingt als Webserver-Modul eingerichtet werden.

#### 6.3.7.3 mod_ssl

Das verschlüsselte https-Protokoll wird durch das Modul *mod_ssl* ermöglicht; dieses Modul ist für Apache 1.3 ein Fremdmodul, seit der Version 2 des Apache gehört es zur Standard-Distribution. Es baut auf dem OpenSSL-Paket auf und wird von Ralf Engelschall entwickelt.

#### 6.3.7.4 Module für Scriptsprachen: mod_perl, mod_ruby und andere

Für fast alle im Web gebräuchlichen Scriptsprachen existieren Servermodule: Das bedeutet, dass die Funktionalität des Scriptsprachen-Interpreters in den Webserver direkt integriert wird, also nicht bei jedem Aufruf des Scriptes der Interpreter neu geladen werden muss. Dadurch beschleunigt sich die Ausführung von Serverscripten wesentlich.

Die Idee, die letztlich hinter diesen Konzepten steht, hat viel mit dem fastCGI-Modul zu tun; in Kapitel 19 wird dies genauer vorgestellt.

### 6.3.8 Auswertung von Logfiles

Der Apache-Webserver schreibt sehr ausführliche, zudem genau konfigurierbare, Logfiles. Für den professionellen Betrieb eines Webservers ist die regelmäßige und genaue Auswertung dieser Logfiles essentiell.

Standardmäßig verteilt der Apache die Logging-Information auf mehrere Dateien im Unterordner `log`:

- `access.log` ist die zentrale Zugriffsdatei; sie protokolliert welcher Client (ip-Adresse oder Name) wann welchen HTTP-Request zum Apache geschickt hat; der HTTP-Antwortcode wird ebenfalls festgehalten, genauso wie die Länge der ausgelieferten Datei;
- `referer.log` hält fest, von welcher Adresse der Client kam;
- `agent.log` protokolliert den user-agent, also letztlich die Meldung, unter der sich der Browser bei dem Server meldet;
- `error.log` hält die serverseitigen Fehler fest, angefangen von einem „not found“ bis hin zum „internal server error“; der Grad der Protokollierung kann in der httpd.conf über die Direktive

  ```
  LogLevel warn
  ```

  dabei eingestellt werden, wobei acht Stufen von `debug` bis `emerg` möglich sind.

Alle diese Dateien sind sehr wichtig; häufig werden die ersten drei auch zu einer einzelnen Datei zusammengefasst. Für die praktische Web-Programmierung ist gerade die `error.log` in vielen Fällen sehr wichtig, um zusätzliche Debugging-Informationen zu bekommen.

#### 6.3.8.1 Konfiguration der logs

Über die Direktive

```
LogFormat
```

kann die Art der Protokollierung des Apache in der httpd.conf-Datei umfassend verändert werden.

#### 6.3.8.2 Tools: Webalizer, awstat & Co

Der Apache schreibt somit umfassende Logging-Informationen in Textdateien. Deren Auswertung ist allerdings rein textbasiert etwas mühsam, hier helfen spezielle Werkzeuge.

Zu den bekanntesten Tools für die Analyse der Logfiles gehören *webalizer* und *awstats* [web]. Besonders das in Perl geschriebene awstats wird häufig eingesetzt; es bietet zahlreiche Auswertungs- und auch Konfigurationsmöglichkeiten.

Die awstats-Analyse umfasst beispielsweise eine zeitliche Verteilung der Anfragen, eine geographische Verteilung und insbesondere eine Analyse, von welchen Suchmaschinen mit welchen Suchbegriffen die Besucher kamen.

Google bietet mit dem Dienst Google Analytics eine externe Analyse von Web-Sites an, die auf einem einfachen JavaScript-Anteil basiert; hier werden extern Daten über die Zugriffe der eigenen Site gesammelt, was bequem, aber auch datenschutzrechtlich nicht unbedenklich ist.

# 7

# Das Beispiel

Durchgängig in diesem Buch betrachten wir ein Beispiel, an welchem die verschiedenen Techniken jeweils gezeigt und erprobt werden sollen; wir bauen unsere eigene kleine (leider virtuelle) Bibliothek auf, wir wollen also Bücher verwalten.

Bücher sind sehr geeignete Objekte für unser Beispiel. Wir verwalten sie natürlich in einem relationalen Datenbankmanagementsystem, wobei die Auswahl des Systems letztlich eine sehr untergeordnete Rolle spielt; wir wollen sogar meistens so entwickeln, dass unsere Anwendungen unabhängig vom jeweiligen DBMS sind, dieses also austauschbar wäre.

## 7.1 Die Literatur-Datenbanktabellen

Das Beispiel ist an reale Bibliothekssoftware angelehnt, für die Zwecke hier aber vereinfacht. Folgende Eigenschaften sollen die Bücher in unserer Datenbank haben:

- eine durchlaufende Nummerierung, welche zugleich den Primärschlüssel liefert;
- einen Titel;
- einen Verlag;
- ein Erscheinungsjahr;
- beliebig viele Autoren.

Um die Eigenschaften relationaler Datenbanken zu nutzen, muss ein Literatureintrag mindestens einen, kann aber beliebig viele Autoren haben; diese pflegen wir in einer zweiten Tabelle mit folgenden Attributen:

- eindeutige Kennungs-ID, der Prinärschlüssel;
- Nachname;
- Vorname (optional, da nicht immer verfügbar, etwa bei „Benedikt XVI.“);
- Position in der Aufzählungsreihenfolge der Autoren; diese Reihenfolge ist für wissenschaftliche Publikationen wichtig, da hierdurch der Anteil der Beteiligten am Werk beschrieben wird;
- Bezug zum Titel, zu welchem der Autorendatensatz gehört; hier handelt es sich um einen Fremdschlüssel.

Die Daten werden in einer relationalen Datenbank in zwei Tabellen abgelegt, den Tabellen `buecher` und `autoren`; die Datenbank selbst soll im Folgenden immer `webkompendium` heißen.

### 7.1.1 Das Datenmodell

Um diese Information zu hinterlegen, wird eine Tabelle `buecher` mit den eigentlichen Buchinformationen (Titel, Verlag, Jahr) verwendet. Eine zweite Tabelle `autoren` speichert die Auorendaten (Name und Reihenfolge) und verwendet einen Fremdschlüssel für den Verweis zum zugehörigen Buch. Abbildung 7.1 zeigt eine grafische Darstellung der Tabellenstruktur.

Dabei kann eine leistungsstarke Datenbank viele sinnvolle Überprüfungen der Einzeldaten auf der Datenbankebene bereits beinhalten, etwa:

- Jede der Tabellen soll über einen Primärschlüssel verfügen, welcher die Einträge durchzählt und automatisch (also vom Datenbankmanagementsystem) belegt wird; im Falle von MySQL steht hierfür `AUTO_INCREMENT` zur Verfügung;
- wird ein Buch gelöscht, sollen auch automatisiert die zugehörigen Autoren gelöscht werden;
- die Reihenfolge der Autoren beginnt bei 1 und muss für jedes Buch eindeutig sein.

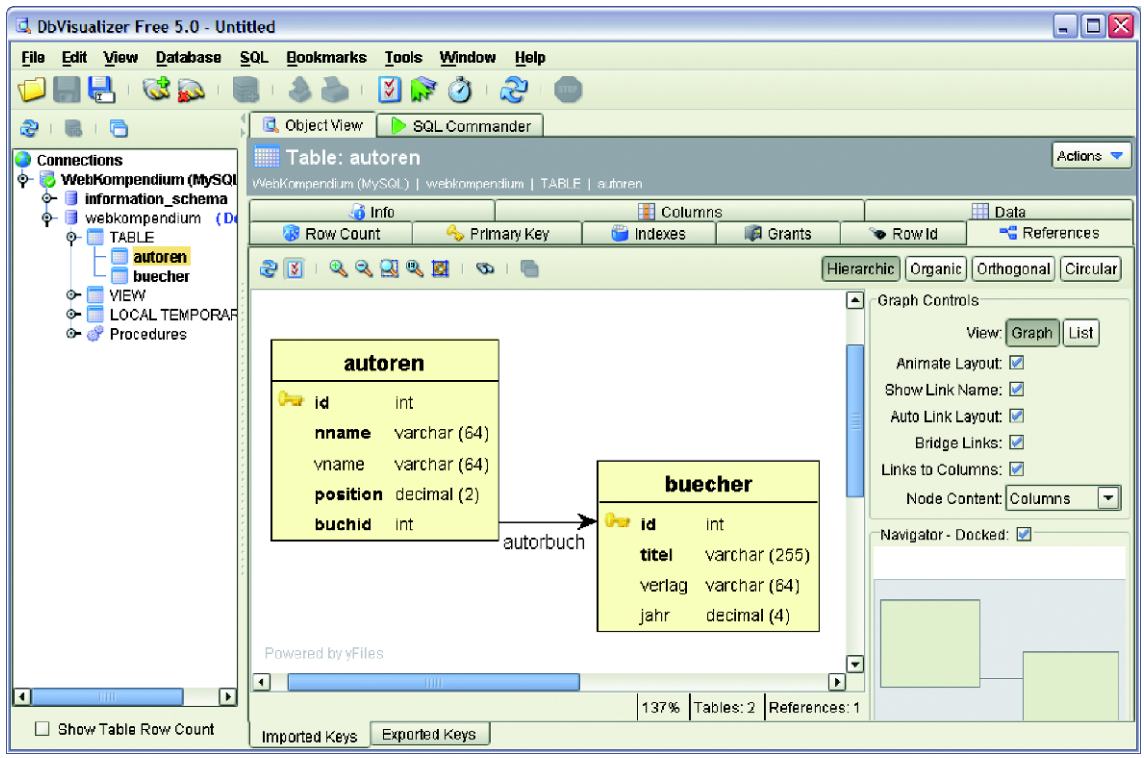



**Abbildung 7.1:** Visualisierung der Fremdschlüsselbeziehung mit DbVisualizer

### 7.1.2 SQL-Script

Das SQL-Script in Listing 7.1 baut unsere zwei Datenbanktabellen auf. Um mit MySQL Funktionalitäten wie Fremdschlüssel nutzen zu können, muss die Tabelle vom Typ `INNODB` sein; der Standardtyp für Tabellen ist `MyISAM`, welcher weniger Funktionalität, aber eine (deutlich) höhere Performance bietet.

| Typ | Bedeutung |
|---|---|
| MyISAM | Standardtyp, keine Transaktionen, performant |
| MERGE | Zusammenfassung mehrerer MyISAM-Tabellen |
| HEAP | Tabellen werden im Arbeitsspeicher verwaltet |
| InnoDB | Transaktionen, Fremdschlüssel |

**Tabelle 7.1:** Tabellentypen in MySQL

```
/* Buecher-Tabelle */
CREATE TABLE buecher (
  id       INT AUTO_INCREMENT PRIMARY KEY,
  titel    VARCHAR (255) NOT NULL,
  verlag   VARCHAR (64),
  jahr     DEC (4)
) ENGINE=INNODB;

/* Autoren-Tabelle */
CREATE TABLE autoren (
  id       INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  nname    VARCHAR (64) NOT NULL,
  vname    VARCHAR (64),
  position DEC (2) NOT NULL CHECK position > 0,
  buchid   INT NOT NULL,
  CONSTRAINT autorbuch FOREIGN KEY (buchid)
             REFERENCES buecher (id) ON DELETE CASCADE,
  UNIQUE (buchid, position)
) ENGINE=INNODB;
```

**Listing 7.1:** Aufbau der zwei Datenbanktabellen `buecher` und `autoren` (buecher.sql)

Einige grundlegende Einträge, die gerade Fälle mit mehrfachen Autoren beinhalten, wird durch folgendes SQL-Script erreicht:

```
insert into buecher (titel, verlag, jahr) values
       ("MediaFotografie","Springer",2005);
insert into buecher (titel, verlag, jahr) values
       ("Kompendium der Web-Programmierung","Springer",2007);
insert into buecher (titel, verlag, jahr) values
       ("Grundlagen der Informatik","Hanser",2003);
insert into buecher (titel, verlag, jahr) values
       ("Computergrafik","Hanser",2005);
insert into buecher (titel, verlag, jahr) values
       ("Mathematik fuer Informatiker","Hanser",2004);
insert into buecher (titel, verlag, jahr) values
       ("Multimedia-Recht","Springer",2002);
insert into buecher (titel, verlag, jahr) values
       ("Kompendium der Mediengestaltung","Springer",2003);


insert into autoren (nname, vname, position, buchid) values ("Walter", "Thomas", 1,1);
insert into autoren (nname, vname, position, buchid) values ("Walter", "Thomas", 1,2);
insert into autoren (nname, vname, position, buchid) values ("Walter", "Thomas", 1,3);
insert into autoren (nname, vname, position, buchid) values ("Bender", "Michael",1,4);
insert into autoren (nname, vname, position, buchid) values ("Brill", "Manfred",2,4);
insert into autoren (nname, vname, position, buchid) values ("Brill", "Manfred",1,5);
insert into autoren (nname, vname, position, buchid) values ("Merx", "Oliver",1,6);
insert into autoren (nname, vname, position, buchid) values ("Tandler", "Ernst",2,6);
insert into autoren (nname, vname, position, buchid) values ("Hahn", "Heinrich",3,6);
insert into autoren (nname, vname, position, buchid) values ("Boehringer", "Joachim",1,7);
insert into autoren (nname, vname, position, buchid) values ("Buehler", "Peter",2,7);
insert into autoren (nname, vname, position, buchid) values ("Schlaich", "Patrick",3,7);
insert into autoren (nname, vname, position, buchid) values ("Ziegler", "Hanns-Juergen",4,7);
```

**Listing 7.2:** Füllen der Datenbanktabellen `buecher` und `autoren` (buecher_inhalte.sql)

### 7.1.3 Normalform

Die *Normalform* bei relationalen Datenbanken bestimmt allgemein, welche Abhängigkeiten zwischen Attributen vorhanden sind. Ziel der Normalform ist ein redundanzfreies Datenmodell, sie führt zur Zerlegung in viele kleine Relationen.

Die in diesem Beispiel entstandene Datenbank ist bereits in der ersten Normalform formuliert, also grundlegend normalisiert:

- die Tabellenstruktur entspricht den „Realen Objekten“, hier dem Buch und dem Autor;
- in jeder Relation (Tabelle) kommt jedes Attribut nur ein mal vor (keine Redundanz);
- jedes Attribut hat (höchstens) einen Wert.

Dies beschreibt die erste Normalform; die zweite Normalform fordert weiter eine klare Verwendung der Primärschlüsselfelder. Die weiteren Normalformen führen zu immer deutlicher gegliederten Datenbankstrukturen.

**Für die Entwicklung performanter Web-Applikationen ist ein wohlüberlegtes Datenmodell absolut grundlegend; diesem sollte keinesfalls zu wenig Beachtung geschenkt werden.**

## 7.2 Reale Beispiele: Dublin Core

Dieses Beispiel findet sich im Kern in vielen Bibliothekssystemen wieder, allerdings ist unsere Implementierung noch viel zu einfach: Nicht nur, dass viele Attribute – wie die für neuere Literatur im allgemeinen verfügbare ISBN-Nummer – fehlen, auch sind etwa Probleme wie die Frage der Trennung von Autoren und Herausgebern nicht berücksichtigt. Dennoch ist dieses Beispiel in dieser Form gut geeignet, die typischen Probleme aufzuzeigen und Lösungen in den verschiedensten Implementierungen zu finden.

Ein realer Standard für bibliografische Datensätze im deutschsprachigen Raum ist das MAB-Format (Maschinelles Austauschformat für Bibliotheken), welches letztlich ein Austauschformat für Daten beschreibt. Ein alternatives Format wird etwa durch MARC (Machine Readable Cataloging) definiert.

Ein modernerer Ansatz baut auf ein Datenaustauschformat in XML (vgl. 2.4): die Dublin Core Metadata Initiative (DCMI). Seit 1994 werden von mehreren Arbeitsgruppen hier Standards für Metadaten entwickelt.

Insgesamt besteht DCMI momentan (Version 1.1, ISO 15836) aus 15 Kernelementen aus den Bereichen

- technische Daten (format, type, language);
- Inhaltsbeschreibung (title, subject, coverage, description);
- Personen und Rechte (creator, publisher, contributor, rights);
- Vernetzung (source, relation);
- Kennzeichnung (id) und Lebnszyklus (date).

Dublin Core-Elemente können etwa in den Meta-Informationen eines HTML-Dokumentes verwendet werden; für diese Attribute ist der Namensraum `DC` üblich:

```
<head profile="http://dublincore.org/documents/dcq-html/">
  <title>Dublin Core</title>
  <link rel="schema.DC"      href="http://purl.org/dc/elements/1.1/" />
  <link rel="schema.DCTERMS" href="http://purl.org/dc/terms/" />
  <meta name="DC.format"        scheme="DCTERMS.IMT"      content="text/html" />
  <meta name="DC.type"          scheme="DCTERMS.DCMIType" content="Text" />
  <meta name="DC.publisher"     content="Springer" />
  <meta name="DC.title"         content="Kompendium der Web-Programmierung" />
  <meta name="DC.creator"       content="Thomas Walter" />
  <meta name="DC.date"          content="2007" />
 </head>
```

Die Bedeutung der XML-basierten Technologien wird auch durch die Weiterentwicklung der Formate hin zu MABxml und MARCXML in jüngerer Zeit unterstrichen.

## 7.3 Anwendungsfälle

Wir wollen im weiteren Verlauf des Buches häufig auf dieses Beispiel zurückgreifen. Dazu werden wir folgende typische Fälle betrachten:

- Ausgabe aus der Datenbank: Unsere Datenbankeinträge sollen ausgewertet werden; dazu werden je nach Fall verschiedene Trefferlisten gebildet: alle Einträge in der Datenbank oder spezielle Suchergebnisse etwa nach einem Verlag oder nach einem Autor.

  Es werden nach den Regeln der alphabetischen Katalogisierung (RAK) zwei Fälle unterschieden:
  - Im einfachen Fall wird zu jedem Titel nur der erste Autor ausgegeben;
  - Nach RAK korrekt ist folgendes Vorgehen: Bei einem bis drei Autoren werden diese in ihrer Reihenfolge aufgeführt, bei vier und mehr Autoren wird nur der Erstautor mit dem Hinweis „et al.“ (et alii) angegeben.
- Eintrag in die Datenbank: Wir wollen dazu ein Formular im Browser ausfüllen und dieses serverbasiert verarbeiten, so dass unsere Datenbank korrekt gefüllt wird (vgl. Abschnitt 26.2.3).

### 7.3.1 SQL-Syntax für die SELECT-Abfrage

Ein Beispiel, welches häufiger behandelt wird, ist die einfache Abfrage über die mit dem SQL-Script nach Listing 7.1 aufgebauten Datenbanktabellen; dabei soll – im einfachen Fall – zu jedem Buch nur der Autor mit der Position 1 ausgegeben werden.

Zwei Wege führen zu diesem Ziel:

1. Abfrage innerhalb der Applikation: Hier werden zwei geschachtelte SQL-Abfragen verwendet; die Äußere fragt die Tabelle `buecher` ab:

   ```
   SELECT id,titel,verlag,jahr FROM buecher;
   ```

   Das Ergebnis dieser Abfrage, ein Resultset (vgl. 3.3.4), wird zeilenweise verarbeitet. Innerhalb dieses Abarbeitens erfolgt eine zweite SQL-Abfrage, nun an die Tabelle `autoren`:

   ```
   SELECT nname,vname FROM autoren
   WHERE position=1
   AND buchid=(Wert des id-Feldes von buecher);
   ```

   Hier wird die Abfragelogik also in die Anwendung gelegt.
2. Der zweite Ansatz – für das gleiche Ergebnis – verwendet das kartesische Produkt der Datenbanktabellen `buecher` und `autoren` über `JOIN`:

   ```
   SELECT buecher.titel, autoren.nname, autoren.vname,
          buecher.verlag, buecher.jahr
   FROM autoren
   JOIN buecher
   WHERE buecher.id=autoren.buchid AND autoren.position=1
   ORDER BY buecher.titel ASC;
   ```

Beide Wege führen zu dem gleichen Ergebnis, der erste ist flexibler, der zweite performanter. In den nachfolgenden Kapiteln werden beide Lösungswege benutzt.

### 7.3.2 Datenbankeintrag

Der allgemeine Fall eines neuen Datenbankeintrages mit beliebig vielen Autoren wird in Abschnitt 26.2.3 diskutiert; hier werden alle Daten in einer Session erfasst.

# 8

# Wichtige und nützliche Werkzeuge für die Web-Entwicklung

Softwareentwicklung und Designgestaltung für das Web verwenden heute *Werkzeuge,* die den Entwicklungsprozess deutlich effizienter gestalten. In diesem Kapitel wird eine Auswahl der wichtigsten Werkzeuge vorgestellt, die in diesem Buch auch weiter genutzt werden.

## 8.1 Die Entwicklungsumgebung Eclipse

Das zentrale Werkzeug für die Softwareentwicklung ist die integrierte Entwicklungsumgebung (IDE: Integrated Development Environment). Eine IDE bietet zahlreiche praktische Hilfen für den Entwickler:

- Ein Basiswerkzeug ist ein leistungsstarker Editor mit einem guten Syntax-Highlighting für die jeweilige Sprache.
- Zu diesem Editor gehört die Integration des Compilers bzw. des Interpreters, so dass dieser einfach aus der IDE zu bedienen ist und Fehlermeldungen direkt zur entsprechenden Stelle im Sourcecode springen.
- Wichtig sind einfache Hilfen wie die automatisierte Vervollständigung: Schon nach Eingabe des Anfangs eines Bezeichners einer Variablen oder Methode werden die zu diesem Anfang passenden Möglichkeiten aufgelistet und können per Mausklick gewählt werden.
- Ein umfangreicher, gut integrierter Debugger gehört ebenfalls zu jeder IDE.
- Für die Programmentwicklung mit objektorientierten Sprachen wird eine Darstellung der OO-Architektur der jeweiligen Klassen visualisiert.
- Häufig enthalten IDEs für die Entwicklung von Client-Applikationen einen GUI-Editor, mit welchem eine Oberfläche in einem Baukastenprinzip per Maus erzeugt wird; die IDE schreibt dann automatisch den benötigten Code dazu.
- Die Anbindung an Systeme zur Verwaltung von Sourcecode etwa für die Teamarbeit wie CVS ist ebenfalls Bestandteil jeder guten IDE.
- IDEs erleichtern die Erstellung der Entwicklerdokumentation durch automatisches Generieren von Kommentarvorlagen aus Templates und durch Integration von Dokumentationssoftware wie javadoc (vgl. 4.5.1).

**Mit einer leistungsstarken Entwicklungsumgebung wird die Programmentwicklung wesentlich erleichtert.**

Es sind viele solcher IDEs verfügbar, kommerzielle und freie. Verbreitet und bekannt sind etwa Visual Studio von Microsoft (insbesondere für die Sprachen C, C++, C# und VisualBasic) und die Borland-Produkte wie der JBuilder für Java.

Seit Ende 2001 bekommt ein Open Source-Produkt stark zunehmende Bedeutung und löst in weiten Bereichen die kommerziellen Produkte ab: *Eclipse* [web]. Eclipse ist zunächst für die Programmiersprache Java ausgelegt, es besteht aber die einfache Möglichkeit, über die Erweiterung über Plugins praktisch für jede moderne Sprache Eclipse zu verwenden. Gerade für die attraktiven Sprachen der Web-Programmierung stehen zahlreiche Eclipse-Plugins zur Verfügung (vgl. 8.1.1).

**Abbildung 8.1:** Eclipse-Logo

Eclipse selbst ist Nachfolger der IBM-Entwicklung Visual Age for Java. IBM hat die Rechte an die eigenständige Eclipse-Foundation übertragen, welche nun die Software weiterentwickelt.

Eclipse ist in vielen Fällen ein sehr überzeugendes und ausreichendes Werkzeug. Für die meisten konkreten Fälle gibt es noch leistungsstärkere und komplexere Werkzeuge, mit Eclipse liegt aber eine universelle IDE für nahezu alle Anwendungsfelder vor, welche zudem noch frei verfügbar ist.

Eclipse wird frei für alle relevanten Betriebssysteme angeboten.

Abbildung 8.2 zeigt die IDE Eclipse im Einsatz. Eclipse strukturiert, wie die meisten IDEs, die Entwicklungsvorhaben in Projekten (in Abbildung 8.2 das linke Fenster). Bereits beim Anlegen eines solchen Projektes ist die Umgebung zu wählen, also etwa ein Java-Projekt oder ein PHP-Projekt.

Für die jeweiligen Sprachen stellt Eclipse dann, wenn das notwendige Plugin installiert ist, eine Perspektive zur Verfügung: Darunter versteht sich eine für die jeweilige Sprache angepasste Oberflächengestaltung von Eclipse. Abbildung 8.2 zeigt die Java-Perspektive mit speziellen, auf Java ausgerichteten Fenstern wie die OO-Struktur rechts und javadoc in der Mitte unten. Die Perspektiven werden über die Icons oben rechts ausgewählt.

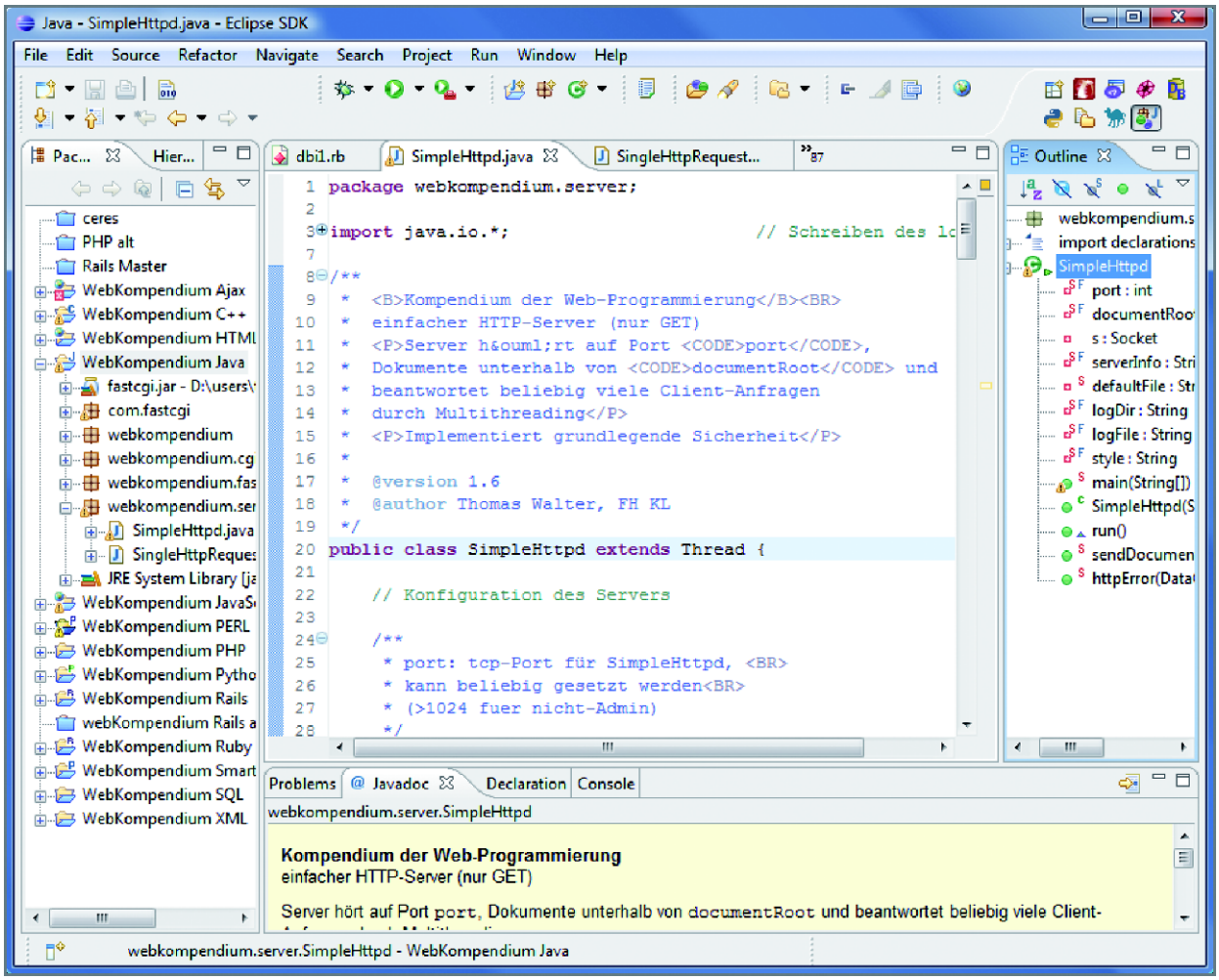



**Abbildung 8.2:**
Eclipse mit der Java-Perspektive

### 8.1.1 Eclipse Plugins allgemein und für die Web-Programmierung

Eclipse kann aufgrund eines klaren Schnittstellenkonzeptes einfach über Plugins erweitert werden. Hauptziel dieser Erweiterungen ist die Unterstützung weiterer Sprachen. In diesem Buch wird hiervon sehr intensiv Gebrauch gemacht, zu jeder Sprache wird mindestens ein passendes Eclipse-Plugin vorgestellt und verwendet.

Die bekanntesten Plugins sind CDT für C/C++ und das Meta-Plugin Callisto, welches viele Plugins bündelt, darunter EMF, GEF (Graphical Editing Framework) und die praktische Web Tools Platform (WTP).

#### 8.1.1.1 Ein Datenbank-Plugin

Ein für die praktische Arbeit besonders nützliches Plugin für Eclipse ist der „SQLExplorer“ [web]. Dieser liefert eine eigene Perspektive, welche den Umgang mit Datenbanken erleichtert (Abbildung 8.3).

Dieses Plugin ist ein sehr einfaches, aber für viele Fälle durchaus ausreichendes Werkzeug. Es unterstützt beliebige Datenbankmanagementsysteme, Voraussetzung ist lediglich ein JDBC-Treiber.

Im ersten Schritt sind die Treiber für die benötigten DBMS einzurichten, danach ist ein Alias für die jeweilige Datenbank zu konfigurieren. Anschließend ist es möglich, sich mit der Datenbank zu verbinden (Fenster oben links in Abbildung 8.3), es können parallel mehrere Verbindungen zu verschiedenen Datenbanken offen gehalten werden (Fenster unten links). Die zwei rechten Fenster zeigen die zugreifbaren Datenbanken in ihrer ganzen Struktur (oben) und für ausgewählte Tabellen deren Aufbau bzw. ausgewählte Datensätze (unten). Im mittleren Bereich kann in einem SQL-Editor (oben) eine SQL-Anweisung geschrieben und ausgeführt werden; das Ergebnis ist in dem Fenster darunter zu sehen.

Somit ist es einfach möglich, im Rahmen der Entwicklung einer Web-Applikation in Eclipse eine SQL-Anweisung direkt in der verwendeten IDE zu testen.

Über dieses einfache Plugin hinaus gibt es viele, deutlich leistungsfähigere Tools; dennoch ist für die praktische Arbeit der freie SQLExplorer von großem Nutzen.

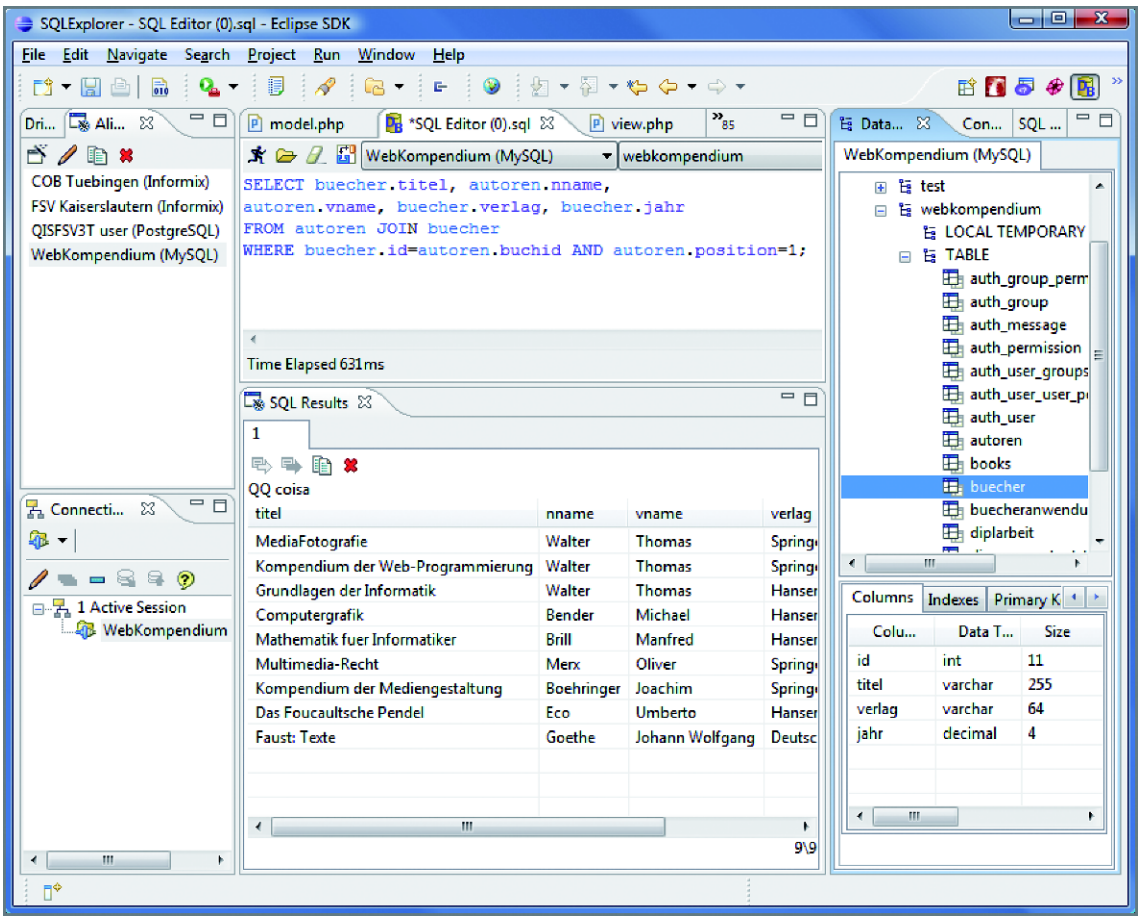



**Abbildung 8.3:**
SQLExplorer-Perspektive in Eclipse

## 8.2 Webeditoren

Die Frage, mit welchen Editoren sinnvollerweise Web-Sites gestaltet werden sollten, kennt viele Antworten. „Hardliner“ verwenden vergleichsweise einfache Editoren wie VI und XEmacs, die immerhin ein Syntax Highlighting für

HTML bieten. Hier stehen auch wieder zahlreiche Eclipse-Plugins zur Verfügung, welche ebenfalls HTML unterstützen, etwa das PHP-Plugin aus Abschnitt 11.3.3.

Häufiger ist in der Praxis der Einsatz sehr fortgeschrittener Webeditoren, welche die Gestaltung einer Site nach dem WYSIWYG-Prinzip erlauben; bekannte Beispiele für diese Editoren sind Dreamweaver und GoLive.

### 8.2.1 Macromedia Dreamweaver

Ein mehr im Bereich der Windows-Systeme verbreiteter und leistungsfähiger Webeditor ist *Dreamweaver,* ursprünglich von Macromedia, inzwischen durch die Übernahme ein Adobe-Produkt.

Auch der Dreamweaver verwaltet die Entwicklungsvorhaben in Projekten. Bereits beim Start stellt er eine aufschlussreiche Frage: die Auswahl zwischen der Sicht für den Designer oder für den Coder (Abbildung 8.4). In der Praxis ist häufig die „gemischte“ Arbeitsweise mit Code- und Vorschauansicht am sinnvollsten (Abbildung 8.5).

Abbildung 27.2 zeigt einen typischen Dreamweaver-Dialog.

**Abbildung 8.4:**
Dreamweaver-Startup: Designer oder Coder?

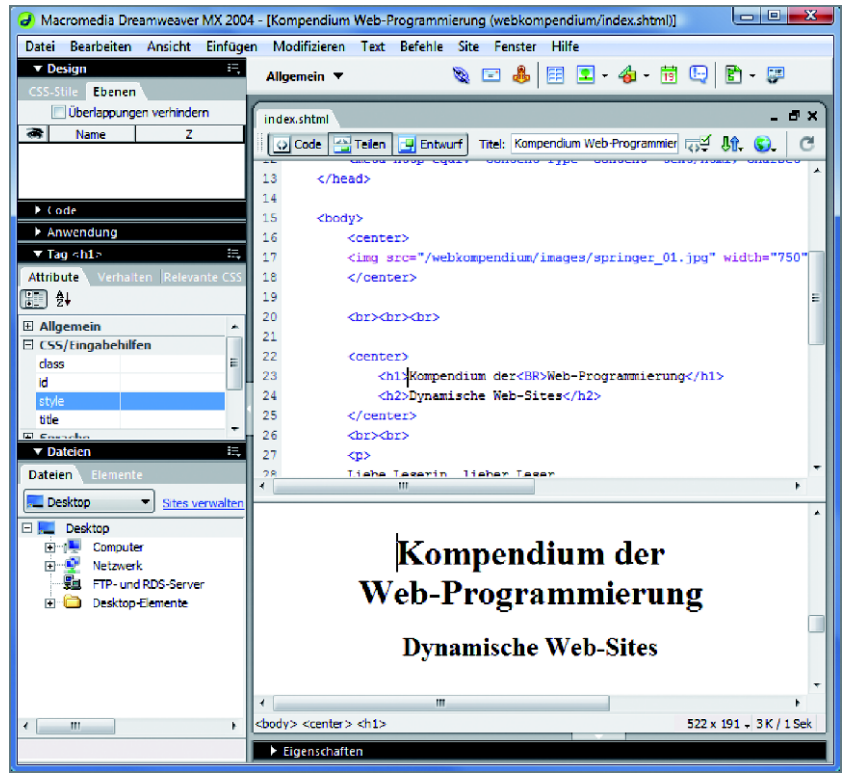



**Abbildung 8.5:**
Dreamweaver im Einsatz

### 8.2.2 Adobe GoLive

Ähnlich leistungsfähig wie Dreamweaver ist Adobes GoLive (Abbildung 8.6), wobei dieses Programm stärker im Bereich des Apple Betriebssystems verbreitet ist.

Abbildung 27.1 zeigt typische GoLive-Dialoge.

### 8.2.3 Vorteile starker Webeditoren

Derart umfassende Werkzeuge wie Dreamweaver und GoLive sind zunächst in der Lage, eine Vielzahl von Aufgaben zu übernehmen. Hierzu zählt die Erstellung von Stylesheets, die automatische Ergänzung einfacher JavaScripts für grafische Effekte im Browser bis hin zur Einbindung einfacher PHP-Codes.

Je mehr Code man mit diesen Tools erzeugt, desto genauer sollte aber auch kontrolliert werden, welcher Art das Ergebnis ist. So ist ein Workflow, sich von den Werkzeugen bei der Erstellung einer Site unterstützen zu lassen, vor Veröffentlichung aber den generierten Code von Hand im „normalen“ Editor zu bereinigen, häufig anzutreffen.

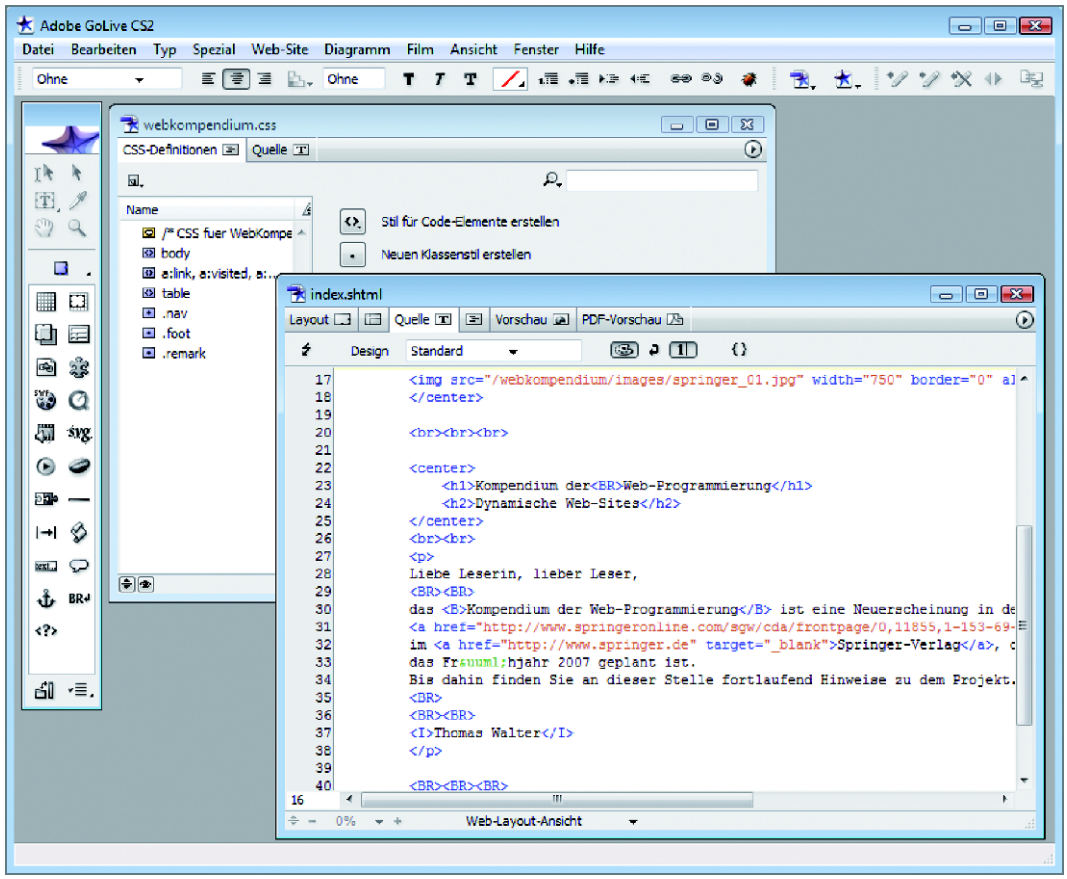



**Abbildung 8.6:**
GoLive im Einsatz

## 8.3 Firebug

Ein wesentlich einfacheres, aber hocheffizientes Werkzeug ist *Firebug* [web]. Hierbei handelt es sich um eine Erweiterung des populären Firefox-Browsers (vgl. 5.2.3), welches für den Entwickler zahlreiche Vorteile bringt:

- mit Firebug kann der HTML-Code der angezeigten Seite angezeigt und direkt geändert werden;[1]
- genauso kann die zugrundeliegende CSS-Datei angezeigt und geändert werden;
- die Netzwerkaktivität beim Laden der Seite kann analysiert werden;
- JavaScript-Anteile in der Seite können getestet werden, wobei einfache Debugging-Möglichkeiten bestehen;
- JavaScript-Funktionen können auch direkt ausgeführt werden;
- die DOM-Baumstruktur einer Seite (vgl. 15.5) wird durch Firebug grafisch visualisiert.

Abbildung 8.7 zeigt den Einsatz von Firebug für die Site springer.de.

[1] dadurch ändert sich natürlich nicht der Code auf dem Server, aber der vom Browser angezeigte Fensterinhalt.

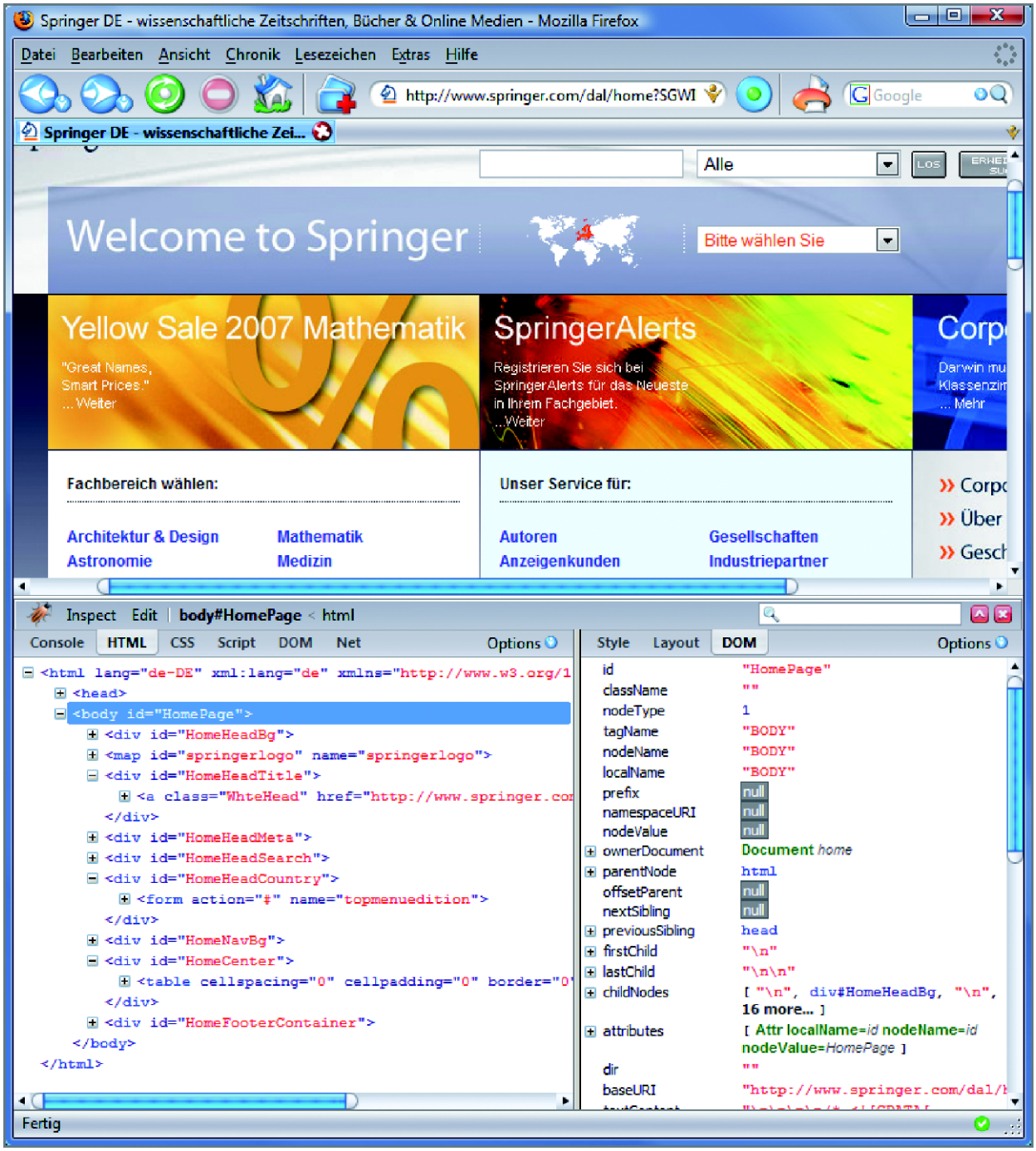



**Abbildung 8.7:** Firebug, hier mit HTML- und DOM-Ansicht

## 8.4 Server-Logs

Die Auswertung der auf dem Webserver geschriebenen Logfiles ist von besonderer Bedeutung; in Abschnitt 6.3.8.2 wurden hierzu einige nützliche Werkzeuge wie awstats vorgestellt.

## 8.5 Datenbank-Tools

Zentral für viele Bereiche von Web-Applikationen ist der Einsatz von Datenbanken. Auch hierfür stehen nützliche, die Arbeit wesentlich beschleunigende Werkzeuge bereit. Einige typische werden hier kurz vorgestellt.

### 8.5.1 Eclipse-Plugin SQLExplorer

Eine einfache und nützliche Grundfunktionalität liefert das *Eclipse-Plugin SQLExplorer* (Abbildung 8.3), welches in 8.1.1.1 genauer vorgestellt wurde.

### 8.5.2 DbVisualizer

Ein typisches, leistungsfähiges Tool ist der *DbVisualizer* von Minq Software [web]. Hier werden u.a. Datenbankrelationen grafisch dargestellt. Abbildung 7.1 zeigt die Anwendung von DbVisualizer.

### 8.5.3 phpMyAdmin

Ein anderes, beliebtes Werkzeug für das Datenbankmanagement ist *phpMyAdmin* [web]. Dieses ist, entgegen den vorher vorgestellten, webbasiert, die Datenbank wird also über den Browser administriert.

**Beim Einsatz dieses nützlichen Tools sind insbesondere Sicherheitskriterien zu beachten, damit das damit verwaltete DBMS vor unberechtigtem Zugriff geschützt ist.**

# Teil II

# Klassische Web-Programmierung: CGI, PHP und moderne Scriptsprachen

# 9

# CGI: das Common Gateway Interface

## 9.1 Dynamik im Web: ein Prozess auf dem Webserver

Mit den ursprünglichen, von Tim Berners-Lee entwickelten Web-Technologien – insbesondere HTTP, HTML und URL – können Dokumente mit multimedialen Inhalten wie Formatierungen und Grafiken angezeigt und mit Hyperlinks einfach verbunden werden (vgl. Abschnitt 1.1), es kann aber keine echte *Dynamik* integriert werden in dem Sinne, dass abhängig von Benutzereingaben etwas „berechnet“ wird.

Hier wird der älteste und bis heute am weitesten verbreitete Ansatz für serverseitige Dynamik im Web vorgestellt: das *Common Gateway Interface* (CGI).

CGI nimmt immer noch eine wichtige Rolle ein, da es ein sehr einfacher, gut vorkonfigurierter Mechanismus ist – nicht performant, aber stabil, einfach und praktisch.

## 9.2 Der CGI-Mechanismus

Hinter CGI steckt im Kern ein Mechanismus, der gesteuert durch einen HTTP-Request auf dem Webserver einen Prozess starten kann und diesem mögliche Parameterpaare, die zum Datenteil des Requests gehören (vgl. 1.6.5.1), sowie weitere für den Prozess wichtige Daten übergeben kann. Genauso muss die Ausgabe des Scriptes über den Webserver wieder an den Client ausgeliefert werden. In vielen Fällen wiederum kommuniziert der CGI-Prozess mit einer Datenbank.

CGI ist nicht neu [web], seit 1996 liegt die Version 1.1 vor, seither hat es am eigentlichen CGI keine Veränderungen mehr gegeben, die Fortentwicklung fand in performanteren Alternativen wie fastCGI statt.

Voraussetzung für CGI ist die Unterstützung durch den Webserver (jeder Client „beherrscht CGI“, da letztlich ein HTTP-GET-Request hierfür genügt). Bei Apache (vgl. 6.3) bedeutet das, dass das CGI-Modul mod_cgi installiert und der Server entsprechend konfiguriert ist. CGI-Scripte liegen üblicherweise in einem separaten Ordner `cgi-bin`.

### 9.2.1 Die Ausgabe eines CGI-Prozesses

Ein CGI-Prozess erzeugt letztlich eine Text-Ausgabe an die Standardausgabe, welche dann über den Webserver an den anfragenden Client gesendet wird.

Diese Ausgabe muss zuerst Information über das nachfolgende Dokument enthalten, nämlich welche Formatierung verwendet wird (der MIME-Typ, vgl. 27.1). Folgende zwei Möglichkeiten sind üblich:

```
Content-type: text/html

Content-type: text/plain
```

Beide müssen durch eine Leerzeile vom nachfolgenden Dokument abgesetzt werden.

## 9.3 Kommunikation zwischen CGI und Webserver

Die Kommunikation zwischen Webserver und dem CGI-Prozess läuft über die Umgebungsvariablen des Servers. So legt der Webserver ein ganzes Set spezieller CGI-Umgebungsvariablen an, auf welche der CGI-Prozess direkt zugreifen kann (Abbildung 9.1).

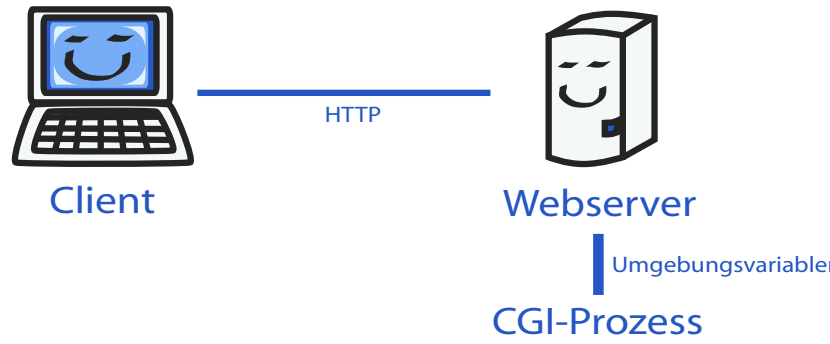



**Abbildung 9.1:** Prinzip von CGI

### 9.3.1 Umgebungsvariablen

CGI legt alle wesentlichen Informationen in speziellen Umgebungsvariablen ab. Zu den wichtigsten hiervon zählen:

- SERVER_NAME
- SERVER_PROTOCOL
- SERVER_PORT
- PATH_INFO
- SCRIPT_NAME
- QUERY_NAME
- REMOTE_HOST
- REMOTE_ADDR
- REMOTE_USER

Abbildung 9.2 zeigt die Ausgabe eines Standard-Scriptes des Apache-Webservers, welches alle Umgebungsvariablen mit Namen und Wert ausgibt.

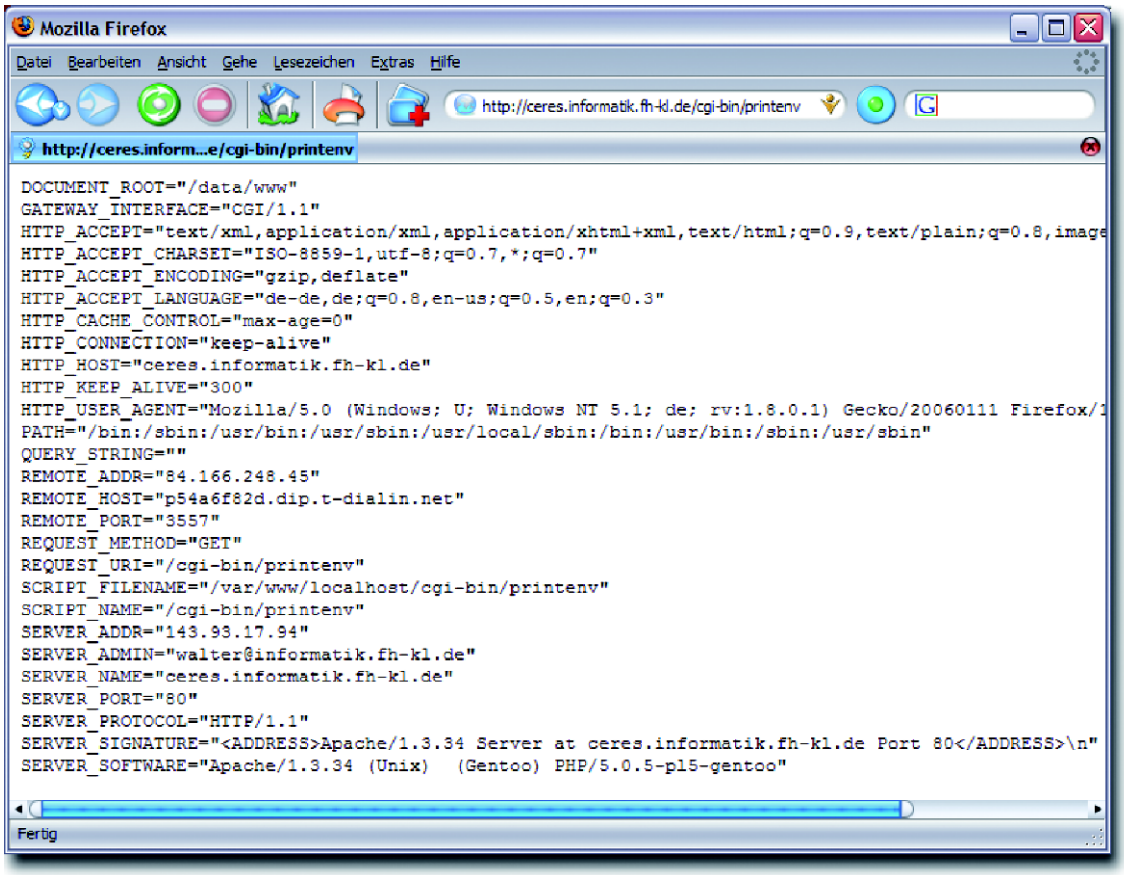



**Abbildung 9.2:**
Das Standard-Script `printenv`: alle Umgebungsvariablen

In 1.6.5.1 haben wir gesehen, wie Parameterpaare beim HTTP-GET-Request Bestandteil der URL werden; geschieht dies, so wird durch CGI einfach eine Umgebungsvariable `QUERY_STRING` mit dem übergebenen Datenteil belegt. Exemplarisch bedeutet dies die Belegung

```
QUERY_STRING="key1=value1&key2=value2"
```

wenn bei einem GET-Request die CGI-Adresse mit Datenteil

```
http://<rechnername>/cgi-bin/<scriptname>?key1=value1&key2=value2
```

angefordert wird.

## 9.4 Beispiele

CGI-Programme können in allen Programmiersprachen formuliert werden, die direkt zu einem ausführbaren Programm führen. Hierzu gehören zu Maschinencode compilierte Sprachen wie C und C++, aber auch interpretierte Sprachen mit eingebundenen Interpretern wie Perl oder Ruby.

### 9.4.1 Ein Beispiel in C

Ein erstes Beispiel zeigt ein in C geschriebenes, minimalistisches CGI-Programm (Abbildung 9.3).

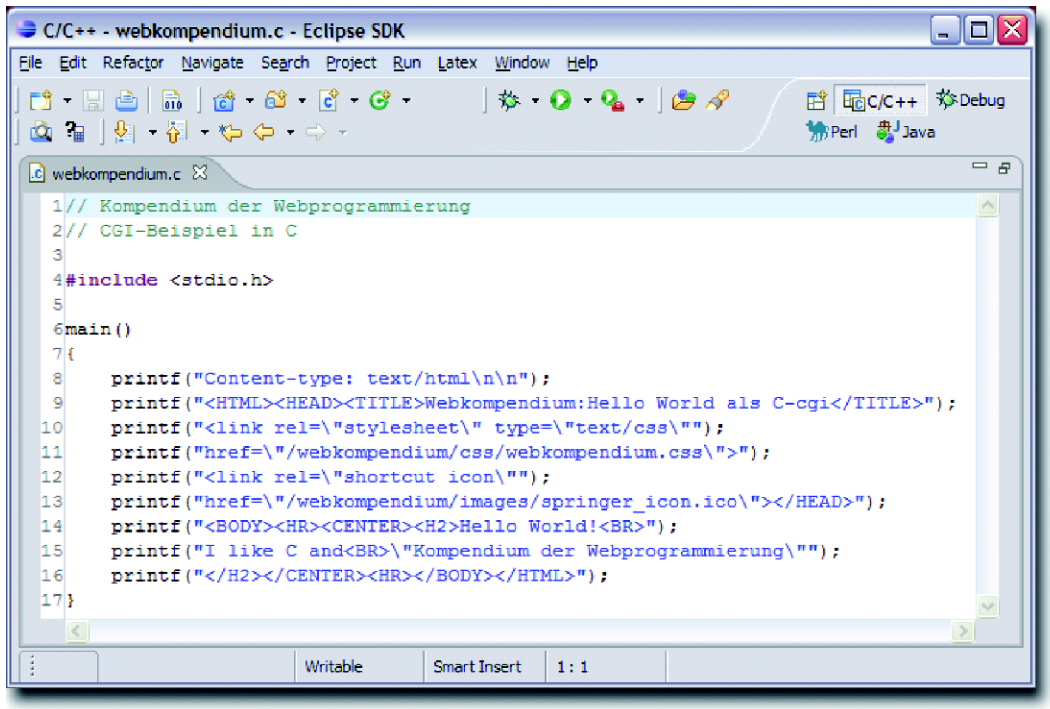



**Abbildung 9.3:** Ausgabe des Scriptes `webkompendium.c` in Eclipse mit dem C-Plugin cdt

Dieses wird mittels eines normalen C-Compilers, etwa `gcc`, in eine ausführbare Binärdatei übersetzt, die wiederum in den cgi-bin-Ordner des Webservers kopiert werden muss; über die URL

```
http://<rechnername>/cgi-bin/<scriptname>
```

kann das CGI-Programm dann ausgeführt und das Ergebnis angezeigt werden; Abbildung 9.4 zeigt eine entsprechende Ausgabe.

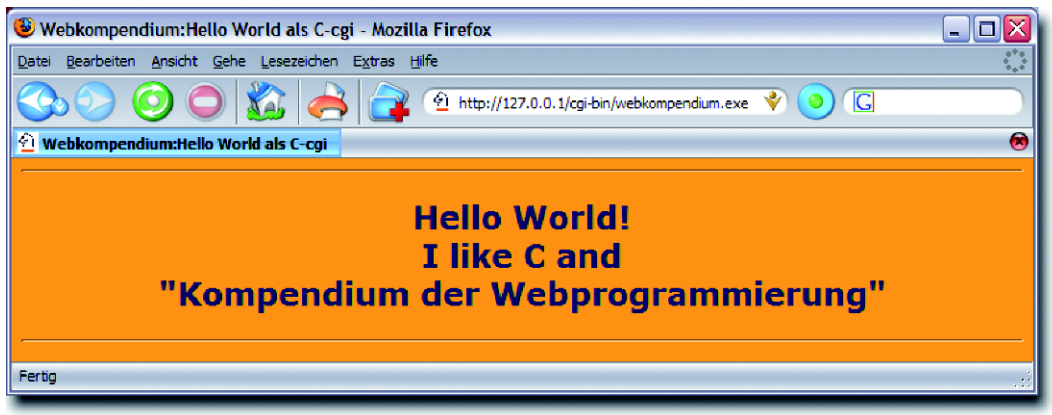



**Abbildung 9.4:** Ausgabe des Scriptes `webkompendium.c`

### 9.4.2 Ein Beispiel in Java

Einer der zentralen Vorteile der Programmiersprache Java ist die Plattformunabhängigkeit: Ein in Java geschriebenes Programm wird durch den Java-Compiler `javac` in eine plattformunabhängige class-Datei übersetzt; diese wird von der Java Virtual Machine (JVM) auf dem jeweiligen System ausgeführt (interpretiert).

Dieser Vorgang ist für die Erstellung von CGI-Programmen sehr hinderlich, da mit Java keine direkt ausführbaren Dateien entstehen. Um dennoch mit Java nach dem CGI-Muster arbeiten zu können, kann ein einfaches Script verwendet werden, welches die JVM startet; hierzu machen wir einen Vorgriff auf die Programmiersprache Perl.

Im ersten Szenario wird über den CGI-Mechanismus das Script `javacgi.pl` aufgerufen; dieses startet die Java-Klasse `webkompendium.cgi.Cgi1`, welche eine HTML-formatierte HelloWorld-Message ausgibt.

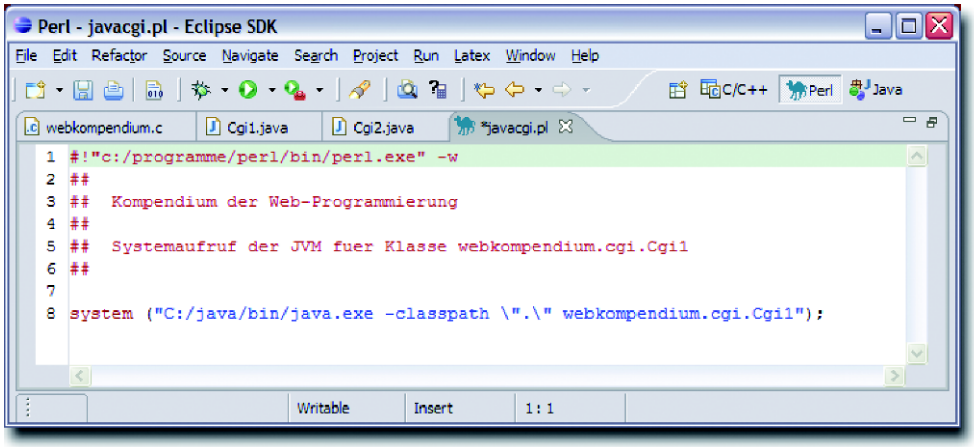



**Abbildung 9.5:**
Perl-Script `javacgi.pl` zum Starten der JVM

Die Java-Klasse `webkompendium.cgi` leistet, wie bereits erwähnt, nicht mehr als eine HTML-formatierte Begrüßung auszugeben, der aber das für CGI notwendige „Content-type“ vorausgeht (Abbildung 9.7).

**Abbildung 9.6:**
Ausgabe der von dem CGI-Script `javacgi.pl` ausgeführten Klasse `Cgi`

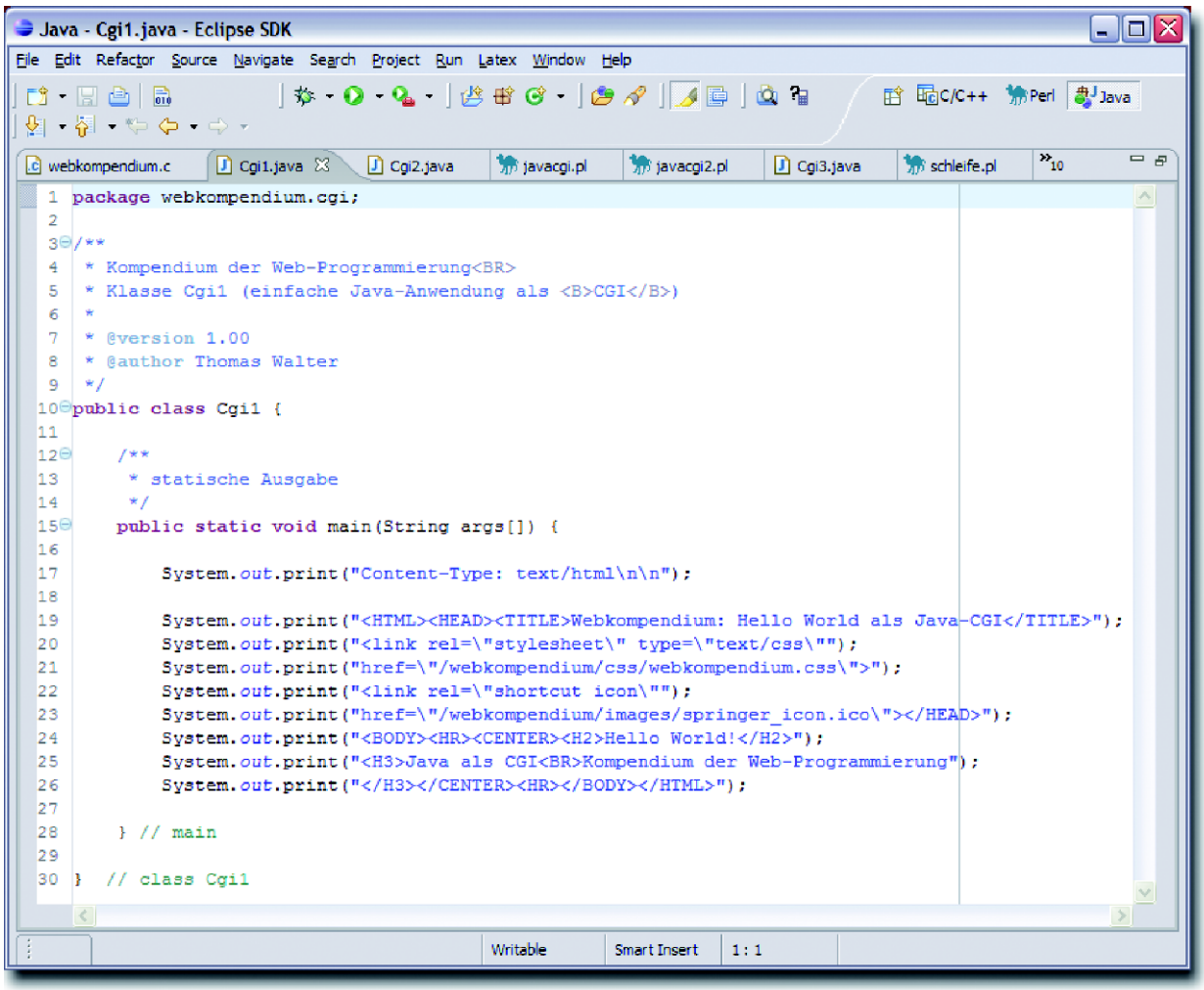



**Abbildung 9.7:**
Die Klasse `Cgi1`

Im zweiten Szenario werden beide Programmierungen verbessert; das Script `javacgi2.pl` erlaubt die flexiblere Behandlung der Parameter, während die Klasse `webkompendium.cgi.Cgi2` sowohl die Zeit auf dem Server als auch einige Standard-Systemvariablen ausgibt; Abbildung 9.9 zeigt das Ergebnis.

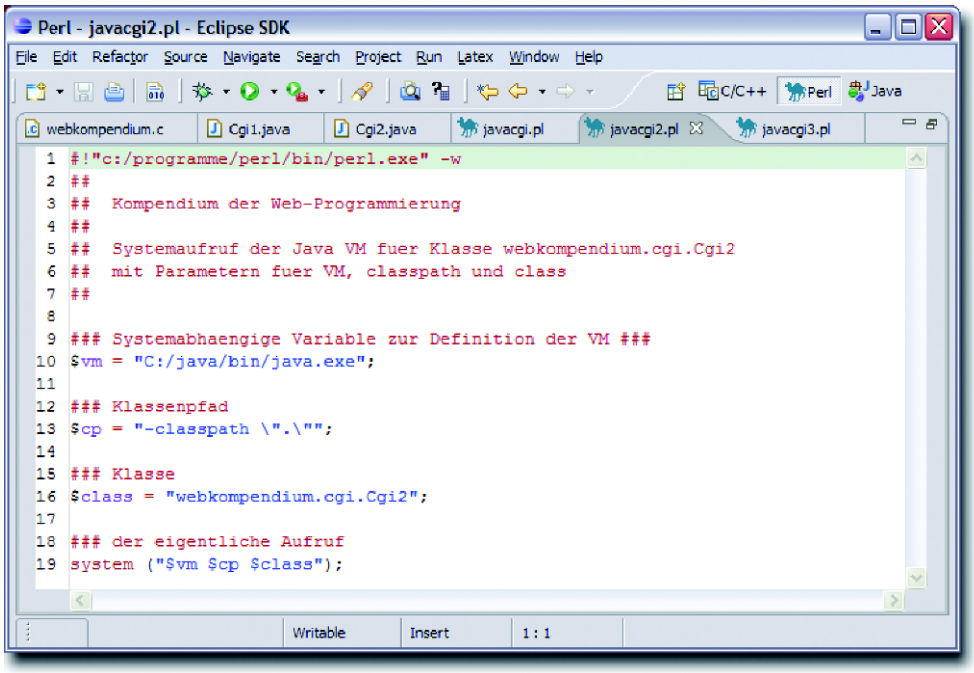



**Abbildung 9.8:**
Perl-Script `javacgi2.pl`

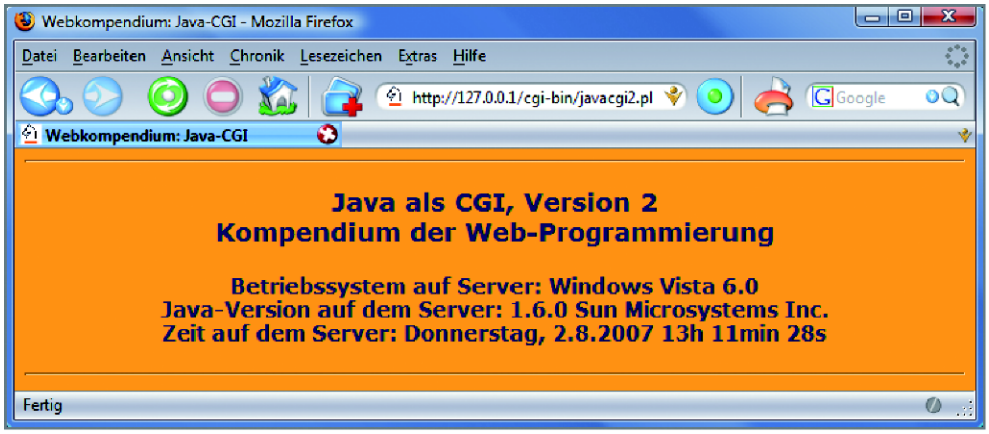



**Abbildung 9.9:**
Ausgabe der von `javacgi2.pl` ausgeführten Klasse `Cgi2`

Das dritte Script verarbeitet ein ihm übergebenes Formular mit einem Feld mit der Bezeichnung „class“; hier soll der Name der Klasse übergeben werden, die dann ausgeführt wird.

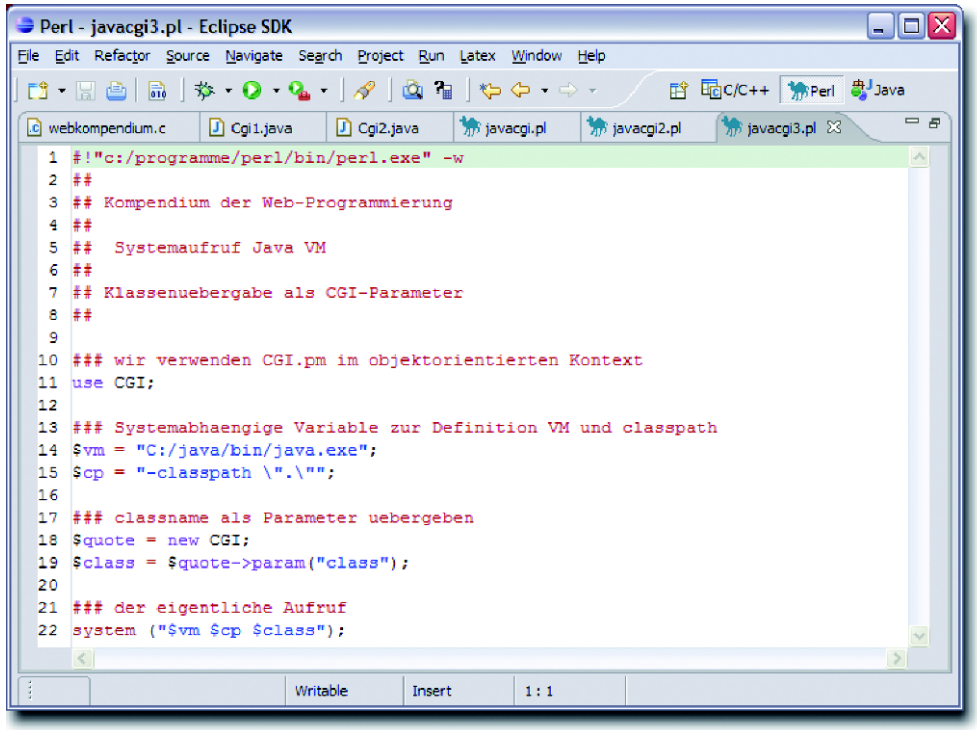



**Abbildung 9.10:**
Perl-Script `javacgi3.pl`

### 9.4.3 Java mit Formularübergabe

Abschließend wollen wir noch einen Schritt weiter gehen, um den CGI-Mechanismus richtig zu verstehen, und ein übertragenes Formular – eine Sammlung von key-value-Paaren – in einer Java-Klasse verarbeiten.

Ausgangspunkt hierfür ist die bereits geschilderte Tatsache, dass das übergebene Formular durch CGI in der Umgebungsvariablen `QUERY_STRING` abgespeichert wird. Beim Starten der JVM kann diese über den Schalter `-D` ein Wert für eine Systemvariable übergeben werden:

```
java -Dkey=value <class>
```

Hier wird für die Umgebungsvariable der Bezeichner `cgi.query_string` gewählt.

Damit kann einer Java-Klasse der Wert der Umgebungsvariablen `QUERY_STRING` übergeben werden, welche dort mit der Methode `getProperty` ausgewertet werden kann:

```
private static String cgi_query =
System.getProperty("cgi.query_string");
```

ausgelesen werden kann. Das Perl-Script `javacgi4.pl` (Abbildung 9.11) leistet genau dieses und übergibt den Wert der durch CGI gesetzten Umgebungsvariablen `QUERY_STRING` mit dem Schlüssel `cgi.query_string` an die JVM.

Die von diesem Perl-Script ausgeführte Java-Klasse `webkompendium.cgi.Cgi4.pl` verwendet den aus der Klasse `Cgi2` bekannten Mechanismus, um den Wert der Umgebungsvariablen `QUERY_STRING` auszulesen. Mittels der im Java-Grundframework bereits umfassend implementierten regulären Ausdrücke (vgl. 10.3.15) wird die so gewonnene Zeichenkette

- zunächst an den Fundstellen von & in die einzelnen Key-Value-Paare aufgespalten;
- danach jedes einzelne Paar am Zeichen = in die Bestandteile Key und Value getrennt.

Das Ergebnis wird dann in einfacher HTML-Formatierung als Tabelle ausgegeben (Abbildung 9.12).

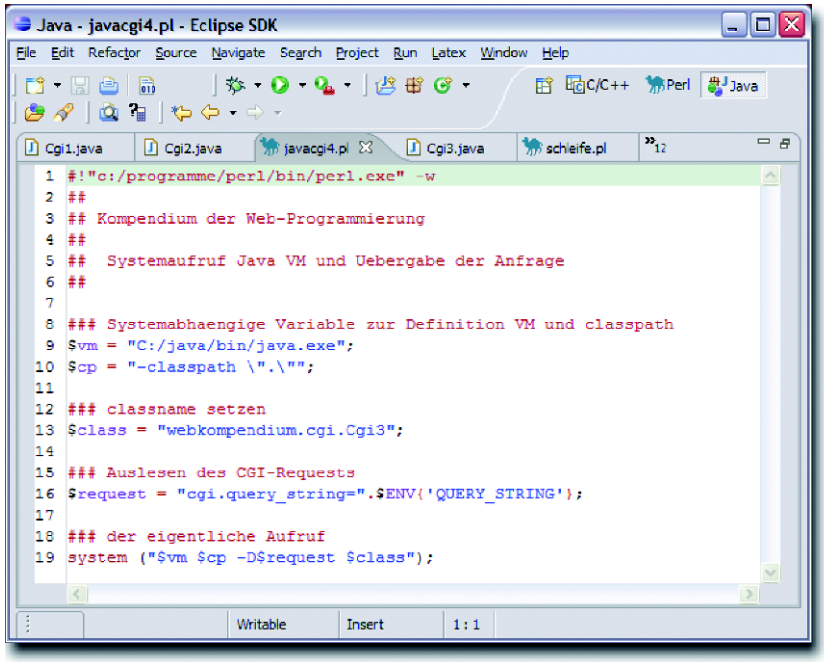



**Abbildung 9.11:**
Perl-Script `javacgi4.pl` zum Ausführen der Java-Klasse `Cgi3` in der JVM und Übergabe von `QUERY_STRING`

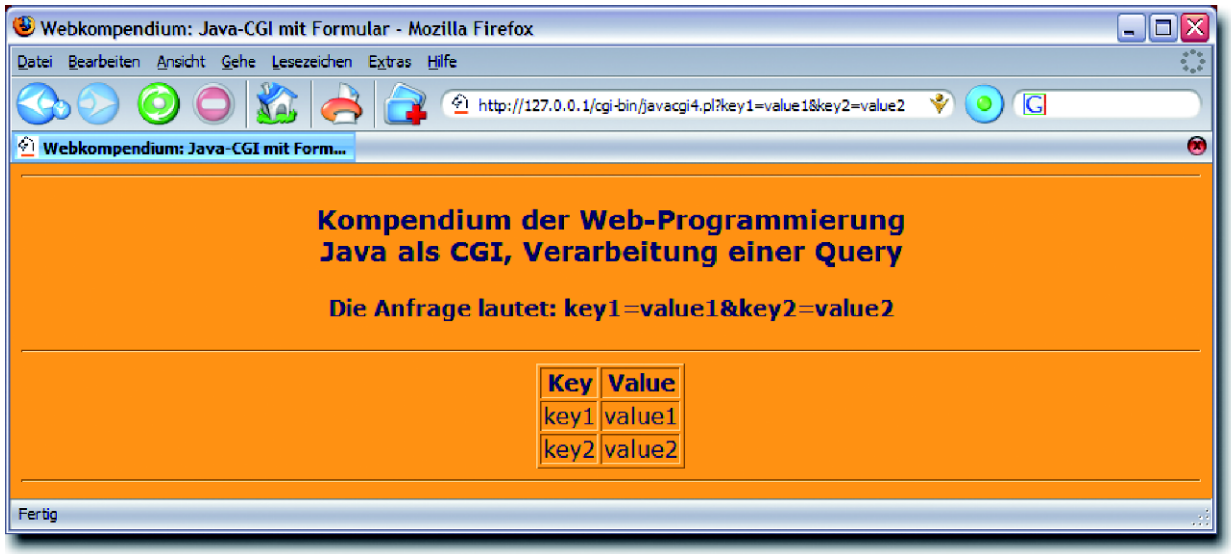



**Abbildung 9.12:**
Ausführen der Klasse `Cgi3` über das Perl-Script `javacgi4.pl` mit zwei über `GET` übergebenen Key-Value-Paaren

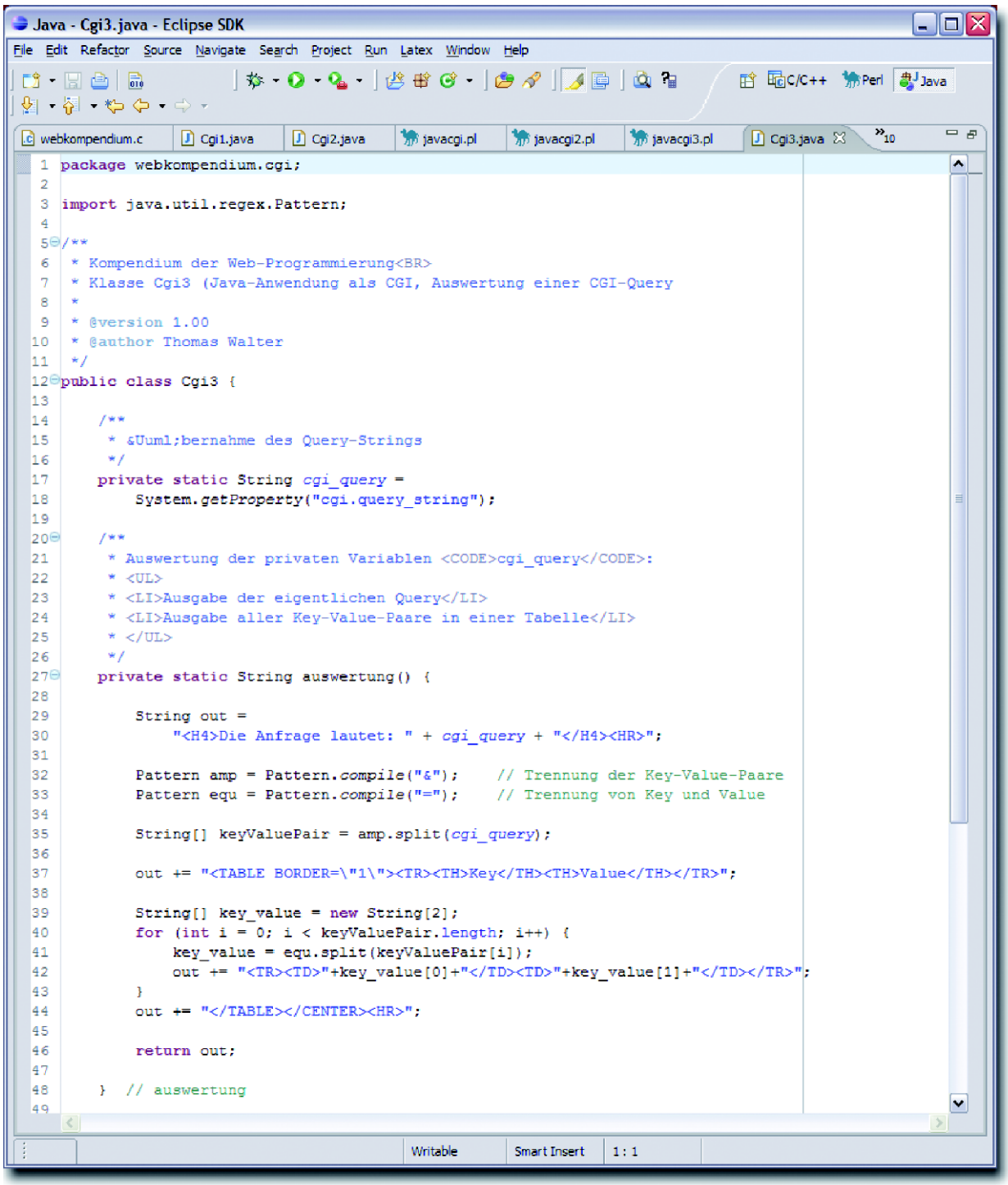



**Abbildung 9.13:** Die Klasse `Cgi3` aus dem Paket `webkompendium.cgi`, Teil 1

Das hier implementierte Grundschema

- Übergabe der durch CGI belegten Umgebungsvariablen, insbesondere von `QUERY_STRING` durch einen geeigneten Mechanismus
- Auswertung dieser Umgebungsvariablen in der jeweiligen Programmiersprache

ist der Kern des CGI-Mechanismus „von Hand implementiert“; wir werden im folgenden dieses auf bequemere Weise erledigen, indem wir geeignete Module verwenden, die uns diese Arbeit deutlich erleichtern. Ein wichtiges und typisches Beispiel dafür ist das Perl-Modul `CGI.pm`, welches in Abschnitt 10.6 vorgestellt wird. Auch komplexere Anwendungsserver wie Tomcat (Abschnitt 24.2.2) leisten (unter anderem) die Aufgabe, auf einfache Weise die übergebenen Wertepaare für die Programmierung zur Verfügung zu stellen.

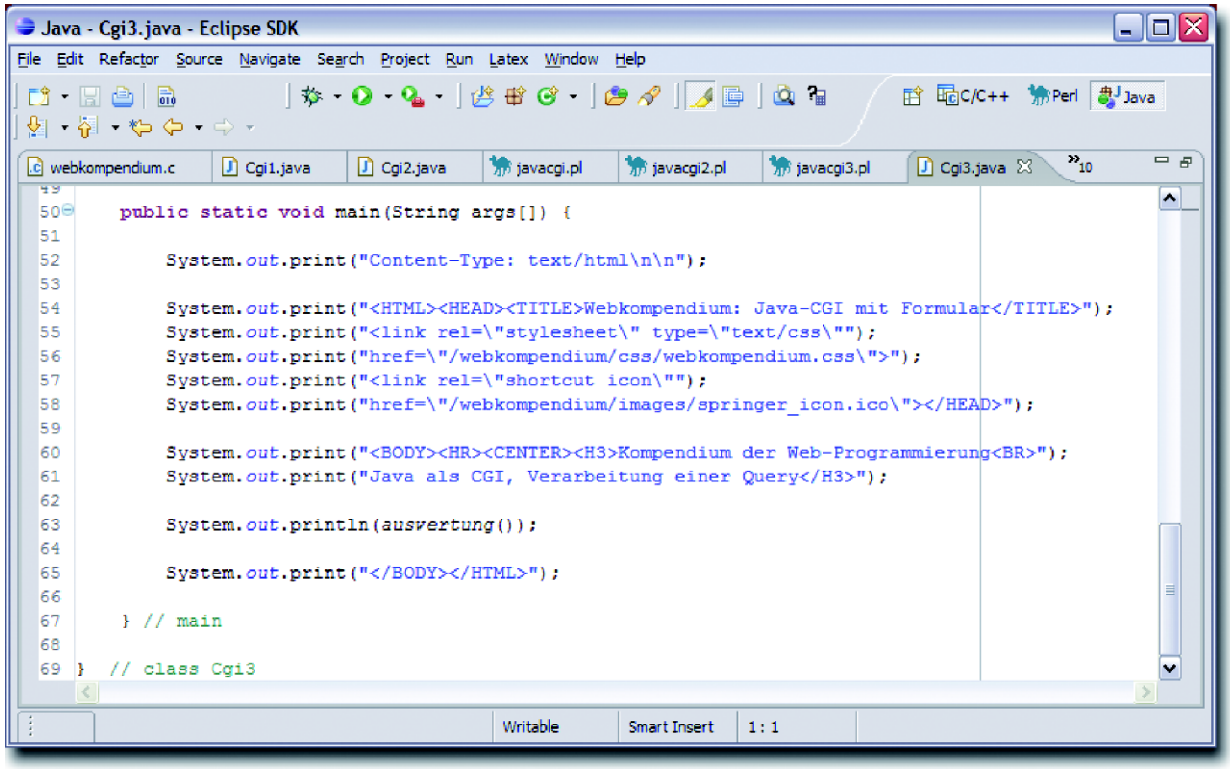



**Abbildung 9.14:**
Die Klasse `Cgi3` aus dem Paket `webkompendium.cgi`, Teil 2

# 10

# Perl

Die Programmiersprache *Perl* wird häufig als das „Schweizer Taschenmesser des Systemadministrators“ bezeichnet – eine sehr treffende, aber auch die volle Bedeutung dieser Sprache verniedlichende Bezeichnung, denn Perl kann noch sehr viel mehr.

Im weiten Feld der Scriptsprachen nimmt Perl eine herausgehobene Rolle ein, da es die am stärksten verbreitete Sprache ist; dabei wird sie sowohl im Web für die CGI-Programmierung als auch als einfache Shell-Programmiersprache verwendet. Ein beliebtes Bonmot zu Perl verdeutlicht dies recht treffend: „Perl ist der russische Traktor der Web-Programmierung: nicht sehr elegant und schnell, aber unverwüstlich“.

Typische Sprachkonstrukte, die uns bei den anderen Scriptsprachen immer wieder begegnen werden, betrachten wir am Beispiel Perl am gründlichsten und kommen später jeweils kurz darauf zurück.

**Abbildung 10.1:** Das Perl-Logo

## 10.1 Die Scriptsprache Perl

Die Sprache Perl ist das Kind von Larry Wall, der bis heute maßgeblich die Sprache fortentwickelt. Perl wurde 1987 vorgestellt, seit 2003 ist die Version 5.8 aktuell. “Perl” steht für “Practical Extraction and Report Language”, manche reden aber auch wegen der teilweise syntaktischen Kuriosität von “Pathologically Eclectic Rubbish Lister”.

Perl ist immer noch die am meisten verbreitete Programmiersprache im Internet, und auch insgesamt eine der am häufigsten eingesetzten Programmiersprachen. Wir werden Beispiele für den Einsatz von Perl in ganz verschiedenen Bereichen – als Standalone-Applikation, als CGI-Programm, als Webclient für die Performance-Messung und für die Systemadministration – kennenlernen.

Perl ist eine plattformunabhängige Sprache und für praktisch alle Betriebssysteme verfügbar. In Perl geschriebene Scripte können meistens problemlos auf verschiedenen Plattformen betrieben werden.

Ein Standardwerk zu Perl, welches intensiv auf die Sprache, aber wenig auf den Einsatz im Internet eingeht, ist [WCS00]. Weiteres und Ergänzendes ist etwa in [Zie02] und [PI99] zu finden.

## 10.2 Quellen und Installation

Offizielle Quelle im Web ist `http://www.Perl.org`, die vom O'Reilly-Verlag betriebene Site `http://www.Perl.com` bietet eine gute Ergänzung. Zentral ist das freie Archiv für Perl, `http://www.cpan.org`, welches ähnlich wie ctan für LaTeX eine unüberschaubare Vielzahl freier Ergänzungen mit einer guten Suchfunktion verbindet.

ActiveState bietet für verschiedene Betriebssysteme gute Perl-Distributionen an. Im Anhang A ist eine Übersicht über weitere Ressourcen im Web zu finden.

### 10.2.1 Lizenz

Perl ist, wie die meisten Scriptsprachen für das Web, frei verfügbar: Open Source. Es unterliegt der offenen „Artistic License" oder wahlweise der „GNU General Public License".

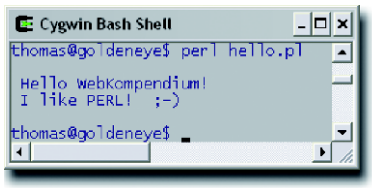



**Abbildung 10.2:** Ausführen von `hello.pl` auf Windows/Cygwin

## 10.3 Grundlegende Syntax

Wir betrachten zuerst das typische Beispiel einer HelloWorld-Ausgabe:

```
# Kompendium der Web-Programmierung
# Kommentar: hier das erste Perl-Script

"\n Hello WebKompendium!\n I like Perl!  ;-)\n";
print $phrase;
```

**Listing 10.1:** HelloWorld in Perl (`hello.pl`)

Dieses einfachste Script speichern wir in der Datei `hello.pl` ab und führen es zunächst von Hand auf der Konsole aus:

```
thomas@goldeneye$ perl hello.pl

 Hello WebKompendium!
 I like Perl!  ;-)
```

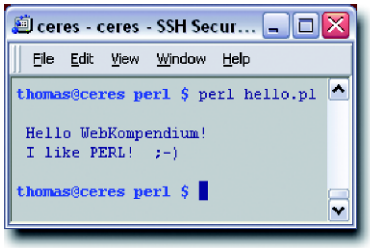



**Abbildung 10.3:** Ausführen von `hello.pl` auf Unix

Was passiert hier? Wir haben mit der Anweisung `Perl` den Perl-Interpreter von Hand gestartet und diesem als Argument unsere Textdatei `hello.pl` übergeben; der Interpreter hat diese Datei komplett übersetzt und dann ausgeführt, wobei die zweizeilige Ausgabe entstanden ist.

Damit dies so geschieht, muss das Betriebssystem den Perl-Interpreter finden, das Verzeichnis, in dem dieser abgelegt wird, muss also Bestandteil der Umgebungsvariablen PATH sein.

**Unter Unix können Sie über die Anweisung `which` feststellen, ob ein ausführbares Programm mit einem bestimmten Namen vorhanden ist und wenn ja, welches verwendet wird, falls es mehrere gibt (es ist nicht selten, auf einem Unix-Rechner mehrere Perl-Versionen zu finden).**

### 10.3.1 Der Perl-Interpreter

Der Perl-Interpreter kann über zahlreiche Parameter gesteuert werden; die wichtigsten davon sind die Schalter

- `-v` : Ausgabe der Version der Perl-Version;
- `-V` : vollständige Information über das Perl-System samt der installierten Module;
- `-w` : Ausgabe der wichtigsten Warnungen;
- `-help` : Übersicht über alle Optionen;
- `-c` : Überprüft die Syntax des Scriptes, ohne dieses auszuführen;
- `-e` : Übergabe einer Perl-Anweisung, die direkt ausgeführt wird.

Wie bereits erwähnt, arbeitet `Perl` nicht wie ein klassischer Interpreter, sondern compiliert das Script vollständig vor dem Ausführen. Als Konsequenz dieses Vorgehens treten zur Laufzeit niemals Syntaxfehler auf. Allerdings verfügt Perl nicht über die Möglichkeit, den – temporären – Binärcode persistent abzuspeichern, wodurch ja bei einem erneuten Aufruf die weitere Compilation entfallen würde. Hier sind modernere Sprachen weiter, etwa Python, was wir in Kapitel 12 genauer sehen werden.

Über den Schalter `-e` kann eine einzelne Anweisung in Perl direkt ausgeführt werden:

```
thomas@goldeneye$ perl -e "print (4711/42);"
112.166666666667
```

### 10.3.2 Die integrierte Dokumentation: `perldoc`

Zu Perl gehört eine integrierte „Direkthilfe“: `perldoc`. Dieses Dienstprogramm ist sehr ähnlich zu den man-pages von Unix und ausgesprochen praktisch.

Das Dienstprogramm hat verschiedene Syntaxoptionen; der Grundaufruf erfolgt nach

```
perldoc [options] PageName|ModuleName|ProgramName...
perldoc [options] -f BuiltinFunction
perldoc [options] -q FAQRegex
```

Die erste Variante liefert die Dokumentation für ein Perl-Modul oder ein Perl-Programm, die zweite Variante erklärt eine der Perl-Standardfunktionen während die dritte die Perl-FAQ nach einem regulären Ausdruck (mehr dazu in 10.3.15) durchsucht.

Die wichtigsten Optionen sind:

- `-h` : Hilfe zu `perldoc`;
- `-i` : Suchausdruck unabhängig von Groß- und Kleinschreibung;
- `-q` : durchsucht die Fragen (nicht die Antworten) in der Perl-FAQ.

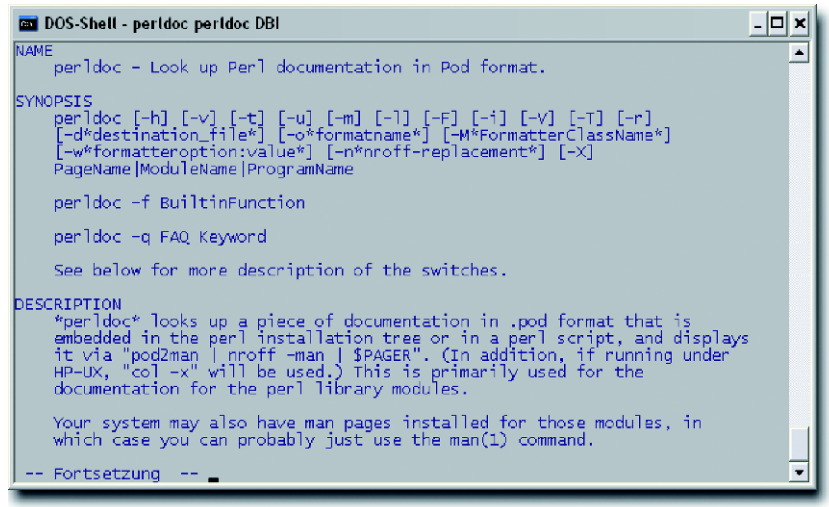



**Abbildung 10.4:**
Beispiel für `perldoc`

### 10.3.2.1 Hintergrund

`perldoc` ist durchaus im Zusammenhang mit den Tools für die automatisierte Dokumentation zu sehen, wie sie in 4.5 vorgestellt wurden. Allerdings fehlt hier das automatisiert erzeugte Grundgerüst der Dokumentation.

Hinter `perldoc` steht die Ausgabe von Dokumentation im `.pod`-Format. Dieses wird durch `pod2man` umformatiert.

### 10.3.3 Kommentare in Perl

Die Abstammung von den Shell-Scripten zeigt Perl sehr deutlich an der Art der *Kommentierung,* die von der heute in den meisten Programmiersprachen üblichen abweicht: als Kommentieriungszeichen wird # verwendet, wie wir schon in Listing 10.1 gesehen haben.

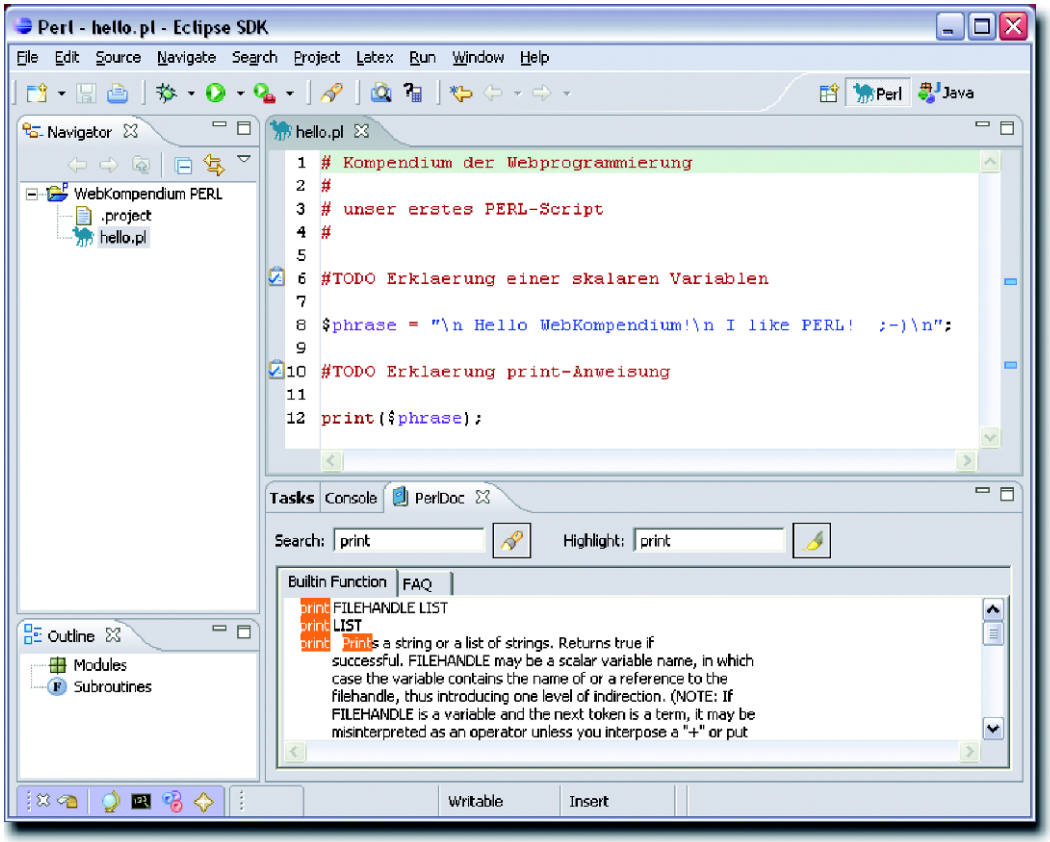



**Abbildung 10.5:**
Die IDE Eclipse mit EPIC-Plugin und geöffnetem Perl-Projekt

### 10.3.4 Entwicklungsumgebungen für Perl

Es gibt einige IDEs, die für den Einsatz von Perl entwickelt wurden.

Im professionellen Einsatz ist die Entwicklungsumgebung Komodo von ActiveState sehr verbreitet.

Eine sehr gute und kostenfreie Alternative ist ein Perl-Plugin für die beliebte IDE Eclipse, die schon in 8.1 vorgestellt wurde: EPIC (Abbildung 10.5). Das Plugin kann einfach über Internet [web] bezogen werden und lässt sich direkt über den Eclipse-Updatemanager installieren. Es bietet unter anderem folgende Funktionalität:

- Syntax Highlighting für Perl;
- Integration von `perldoc` in die IDE;
- Syntaxcheck bei der Programmierung;
- integriertes Perl-Debugging;
- direktes Ausführen von Perl-Scripten innerhalb von Eclipse.

Nach der Installation ist zunächst ein Perl-Projekt (etwa über File|New|Project|Perl|Perl Project) zu erzeugen. Anschließend sollte in die Perl-Perspektive gewechselt werden.

#### 10.3.4.1 Ausführen von Perl-Scripten in EPIC

Es ist möglich, Perl-Scripte direkt in der IDE EPIC auszuführen. Dazu wählt man in Eclipse Run|Run|Perl Local und gelangt zum Dialog nach Abbildung

10.6. Dort ist nur zunächst das Perl-Projekt und dann die Perl-Datei in diesem Projekt zu wählen.

Verwendet das Perl-Script eine Tastatureingabe, so sollte vor dieser Eingabe der Kommandozeilenpuffer von Eclipse deaktiviert werden; mittels der Perl-Anweisung

```
$| = -1;
```

wird die Standardausgabe auf unbuffered gestellt.

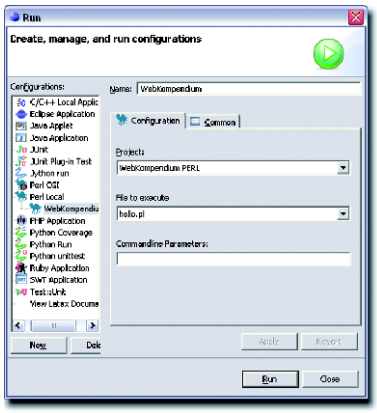



**Abbildung 10.6:** Ausführen eines Perl-Scriptes mit EPIC-Eclipse

### 10.3.5 Datenstrukturen in Perl

Zu Perl gehören die für eine Scriptsprache üblichen *Datenstrukturen*, wie sie uns noch häufiger begegnen werden.

#### 10.3.5.1 Skalare Variablen

Eine *skalare Variable* in Perl verfügt über Eigenschaften, die wir sehr häufig bei Scriptsprachen antreffen werden:

- skalare Variablen sind nicht typisiert: eine Deklaration (vor Verwendung) ist unnötig, ein Datentyp kann nicht festgelegt werden;
- der Bezeichner beginnt mit einem führenden $;
- skalare Variablen werden kontextabhängig interpretiert;
- eine automatische Initialisierung auf Null-Werte erfolgt.

Listing 10.2 zeigt den intuitiven Einsatz von lokalen Variablen, die dabei von Typ Zeichenkette zu Ganzzahl zu Gleitkommazahl umgewandelt werden.

```
# Kompendium der Web-Programmierung
#
# skalare Variablen in Perl

$variable = "42";

print  'Die Variable $variable hat die Laenge '.length($variable)."\n";
$variable += 1;
print 'Der neue Wert von $variable ist '.$variable."\n";
$variable *= 3.1415926;
print '...und nun ist $variable ein Gleitkommawert: '.$variable. "\n";
```

**Listing 10.2:** Einsatz von skalaren Variablen in Perl (skalareVariable.pl)

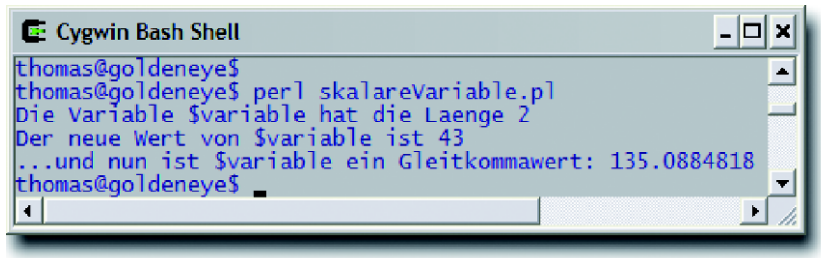



**Abbildung 10.7:** Ausführen eines Perl-Scriptes `skalareVariable.pl`

### Umgang mit numerischen Variablen

Für numerische Variablen stehen Operatoren bereit, wie sie von vielen Sprachen bekannt sind; hinzu kommt ein Operator zum Potenzieren, der bereits in Fortran verwendet wurde. Eine Übersicht ist in Tabelle 10.1 enthalten.

### Umgang mit Zeichenketten

Der Umgang mit Zeichenkettenliteralen in Perl ist typisch für die meisten Programmiersprachen, hat aber in Perl auch eine zentrale Besonderheit.

Zeichenkettenliterale werden mit den Begrenzern ‘‘ und ’ begonnen und beendet. Um innerhalb eines solchen Literals nun selbst dieses Zeichen zu verwenden, muss es durch Voranstellen des Backslash \ maskiert werden.

Genauso wie diese Maskierung gibt es weitere; eine Übersicht ist in Tabelle 10.2 enthalten.

Die Bedeutung der zwei Arten der Begrenzungszeichen hat aber in Perl – und einigen anderen Scriptsprachen – eine grundlegende Unterscheidung (Listing 10.3): Im Fall der doppelten Anführungszeichen findet eine Ersetzung von Variablennamen durch deren Wert statt, bei einfachen bleibt der Variablennamen erhalten.

| Operatoren | Bedeutung |
|---|---|
| `$x += $y` | `$x = $x + $y` |
| `$x -= $y` | `$x = $x - $y` |
| `$x *= $y` | `$x = $x * $y` |
| `$x /= $y` | `$x = $x / $y` |
| `$x %= $y` | `$x = $x modulo $y` |
| `$x++` | `$x = $x + 1` |
| `$x- -` | `$x = $x - 1` |
| `$x ** $y` | `$x hoch $y` |

**Tabelle 10.1:** Operatoren in Perl

```
$phrase = 'WebKompendium';
print("I like $phrase \n")      # ergibt: I like WebKompendium  (Ersetzung)
print('I like $phrase');        # ergibt: I like $phrase        (keine Ersetzung)
```

**Listing 10.3:** Ersetzung in String-Literalen (`stringliterale.pl`)

Wurde der Variablen, deren Bezeichner im Literal mit ‘‘ verwendet wird, kein Wert zugewiesen, wird die Vorbelegung – der leere String – ausgegeben.

Das Zusammenfügen von Zeichenketten erfolgt in Perl (wieder wie in den meisten Scriptsprachen, aber etwa unterschiedlich zu Java) mit dem Kompositionsoperator .:

```
print("Hello " . "WebKompendium");
```

| Sonderzeichen | erzeugender Code |
|---|---|
| Backslash | \\ |
| neue Zeile | \n |
| ‘‘ | \‘‘ |
| Warnton | \a |
| Tabulator | \t |
| Hex-Zeichen hh | \xhh |

**Tabelle 10.2:** Sonderzeichenmaskierung

### Tastatureingabe

Die Eingabe über die Tastatur, also über die „Standardeingabe“, erfolgt in Perl mittels der Anweisung `<STDIN>`. Praktisch sind in diesem Zusammenhang die Perl-Methoden

- `chomp`;
- `chop`.

Während die erste das letzte Zeichen der Zeichenkette nur dann entfernt, falls es ein \n ist, entfernt die zweite Variante immer das letzte Zeichen der Zeichenkette, auf die es angewendet wird (es sind eigentlich Operatoren, die das Argument verändern, auf die sie angewendet werden).

Mit dem nun vorhandenen Wissen ist es möglich, einfache und nützliche Perl-Programme zu schreiben. Listing 10.4 zeigt ein Beispiel, die Berechnung der Fläche und des Umfangs eines Kreises, dessen Radius einzugeben ist.

```
# Kompendium der Web-Programmierung
#
# Perl-Script zur Kreisberechnung

$|=1;  #  STDOUT unbuffered (Eclipse only)

print("\n Radius des Kreises: ");   # Einlesen
$r = <STDIN>;                       # des Radius von
chomp($r);                          # von Standardeingabe

$pi = 3.1415926;                    # Definition PI

$umfang = 2 * $pi * $r;             # Berechnung Umfang

print("\n Umfang  = $umfang \n");

$flaeche = $pi * ( $r**2);          # Berechnung Flaeche

print(" Flaeche = $flaeche \n\n");
```

**Listing 10.4:** Perl-Script zur Kreisberechnung (`kreis.pl`)

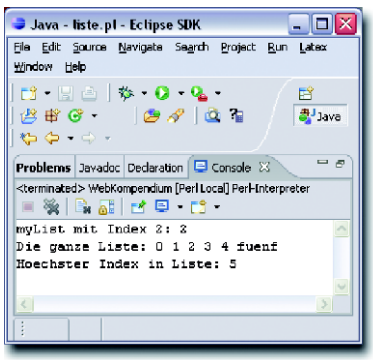



**Abbildung 10.8:** Ausführen des Scriptes aus Listing 10.5 in Eclipse

### 10.3.5.2 Listen in Perl

Perl kennt das klassische, „statische“ Array fester Größe nicht, sondern geht direkt vom Konzept der dynamischen, also erweiter- und verkleinerbaren, Liste aus. Bezeichner von Listenvariablen erkennt man am führenden Zeichen, dem `@`. Das Prinzip des Umgangs mit Listen zeigt das Beispiel 10.5.

```
# Kompendium der Web-Programmierung
#
# Beispiel fuer den Umgang mit Listen

### Deklaration und Initialisiserung der Liste
@myList =(0,1,2);
print ("myList mit Index 2: $myList[2]\n");

### Erweiterung, Variante 1
@myList[3] = 3;

### Erweiterung, Variante 2
@myList = (@myList,4);

### Erweiterung Variante 3
push(@myList,"fuenf");

print ("Die ganze Liste: @myList ");

print("\nHoechster Index in Liste: $#myList");
```

**Listing 10.5:** Listen in Perl (`liste.pl`)

**Das Element einer Liste zu einem bestimmten Index ist ein Skalar; deshalb wird die Syntax `$liste[0]` verwendet.**

### List-Operatoren

Für lineare Listen gibt es Operatoren zum Ablegen auf der Liste und zum Entnehmen von der Liste: `push` und `pop`. Damit entsteht der übliche Stapelspeicher, das Stack. Perl stellt aber noch die zwei ergänzenden Operatoren bereit, um auf das andere Ende der Liste zuzugreifen: `shift` und `unshift`. Damit kann eine Warteschlange (Queue) gebildet werden.

- `push(@liste,$skalar)` : erweitert die Liste `@liste` um das Element `$skalar`, legt also diesen Wert auf der Liste ab;
- `$skalar = pop(@liste)` : entnimmt das zuletzt mit `push` auf der Liste `@liste` abgelegte Element (verkürzt die Liste) und legt diesen Wert in der Variablen `$skalar` ab;
- `unshift(@liste,$skalar)` : erweitert die Liste `@liste` an ihrem Fuß um das Element `$skalar`, legt also diesen Wert am Ende der Liste ab;
- `$skalar = shift(@liste)` : entnimmt das zuerst mit `push` auf der Liste `@liste` abgelegte Element (verkürzt die Liste) und legt diesen Wert in der Variablen `$skalar` ab.

### Weiteres zu Listen in Perl

Neben dem bereits Erwähnten gibt es noch einiges nützliches und bemerkenswertes für den Umgang mit Listen in Perl:

- innerhalb einer Liste `@liste` kann der höchste Index über die Syntax

  ```
  @#liste
  ```

  abgefragt werden;
- da skalare Variablen in Perl nicht typisiert sind, kann eine Liste Werte verschiedenen Typs beinhalten: im Beispiel aus Listing 10.1 sind sowohl numerische Werte als auch Zeichenketten in der Liste enthalten;
- wie in modernen Sprachen üblich beginnt der Index bei `0`;
- es gibt einen Listkonstruktor: Durch

  ```
  @liste = (0..313);
  ```

  wird die Liste `@liste` mit 314 Werten von 0 bis 313 erzeugt;
- mehrdimensionale Listen können ebenfalls nach dieser Logik erzeugt werden: Es sind einfach Listen von Listen.

Einige dieser Besonderheiten sind in Listing 10.5 enthalten.

### Die Liste `@ARGV`

Perl verfügt über einige ausgesprochen nützliche Standardvariablen, zu denen `@ARGV` gehört. In dieser Liste werden die dem Perl-Script als Parameter übergebenen Werte abgelegt; ein Aufruf der Art

```
perl myScript a b c
```

hat zur Folge, dass diese Liste die Belegung

```
@ARGV = ('a','b','c');
```

hat. Eine vergleichbare Rolle hat etwa das Array args in einer Java-Anwendung mit der `main`-Methode

```
public static void main(String[] args)
```

#### 10.3.5.3 Assoziative Listen in Perl

Listen, wie wir sie nun kennengelernt haben, verwenden einen bei 0 beginnenden numerischen (ganzzahligen, positiven) Index. Häufig ist es einfacher, anstelle eines numerischen Index einen „sprechenden Schlüssel“ zu verwenden, der eine direkte Assoziation zur Bedeutung des hier abgelegten Wertes enthält. Eine solche Datenstruktur wird als *assoziatives Array* bezeichnet, es besteht aus einer ungeordneten Menge von Key-Value-Paaren. Scriptsprachen bezeichnen diese Strukturen häufig als Hash, womit aber nicht der klassische Hash-Wert zur praktisch eindeutigen Kennzeichnung eines Elementes im Sinne eines Fingerprints gemeint ist.

Eine ideale Anwendung der assoziativen Arrays werden wir bei der Verarbeitung von HTML-Formularen, wie sie in 2.3.3 vorgestellt wurden, kennenlernen.

Der Bezeichner eines assoziativen Arrays in Perl beginnt mit dem Sonderzeichen %; insgesamt lautet die Syntax in einem Beispiel:

```
%buecher = ('liebling','MediaFotografie,
            'auchGut','WebKompendium');
```

Hier werden zwei Key-Value-Paare mit den Keys `liebling` und `auchGut` angelegt; der Zugriff auf ein Value dazu erfolgt nach

```
$buecher{'liebling'}
```

Dabei können die Hochkommata ' und ‘‘ verwendet werden – oder auch gar keine Zeichenkettenbegrenzer, meistens der bequemste Weg.

Perl stellt hier noch eine alternative Syntax zur Verfügung über die Zuweisung mittels =>; das gleiche Beispiel kann damit auch folgendermaßen definiert werden:

```
%buecher = ('liebling'=>'MediaFotografie,
            'auchGut'=>'WebKompendium');
```

Wie bei den indizierten Listen ist das einzelne Element ein Skalar, hat also ein führendes $. Es können wieder die verschiedenen Datentypen beliebig gemischt werden, nun nicht nur für die Values, sondern auch für die Keys, es sind also in einem Hash sowohl numerische als auch alphanumerische Keys möglich.

Die Kennzeichnung des Keys als Zeichenkette durch begrenzende Hochkommatas kann auch entfallen; es können sogar gemischt numerische und alphanumerische Indices verwendet werden.

**Hash-Funktionen**

Für assoziative Arrays stehen spezielle Funktionen bereit, die insbesondere den Übergang zur „normalen“ Liste ermöglichen. Einige davon sind:

- `keys` : liefert eine Liste mit den Keys des assoziativen Arrays;
- `values` : liefert eine Liste mit den Values des assoziativen Arrays;
- `revers` : liefert ein neues assoziatives Array mit vertauschten Keys und Values in jedem Datenpaar; hier ist darauf zu achten, dass zwar die Values, nicht aber die Keys eindeutig sind, so dass das Vertauschen nicht vollständig möglich sein muss.

Das Script `assArray.pl` (Abbildung 10.9) zeigt ein exemplarisches Beispiel für den Umgang mit assoziativen Arrays in Perl.

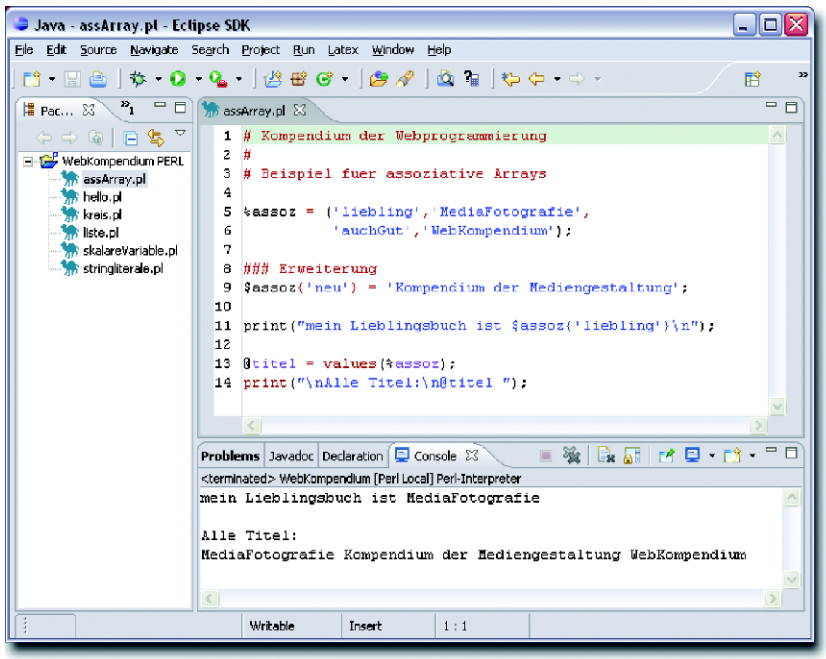



**Abbildung 10.9:**
Scipt `assArray.pl` in Eclipse mit EPIC

### 10.3.6 Kontrollstrukturen in Perl

*Kontrollstrukturen* steuern den Ablauf eines Programmes. Sie ähneln sich in den gängigen Programmiersprachen vergleichsweise stark, weshalb sie hier am Beispiel Perl ausführlicher behandelt werden.

#### 10.3.6.1 Logik und Entscheidungen

Bevor die eigentlichen Kontrollstrukturen vorgestellt werden, müssen zunächst grundlegend logische Operationen erklärt werden. Perl verhält sich bei logischen Variablen wie die meisten Scriptsprachen – anders als moderne Hochsprachen:

**Perl verfügt nicht über einen Datentyp für logische Variablen („`boolean`“); stattdessen gilt:**

- **eine numerische Variable mit dem Wert `0` ist falsch, alle anderen sind wahr;**
- **die leere Zeichenkette ist falsch, alle anderen sind wahr.**

Entscheidungen haben eine Besonderheit in Perl, da sie „doppelt“ implementiert sind: Es gibt die Vergleiche numerisch und alphanumerisch. Tabelle 10.3 gibt eine Übersicht.

Die Unterscheidung zwischen dem numerischen und dem alphanumerischen Vergleich liegt darin, dass im ersten eine automatisierte Typumwandlung zu numerischen, im zweiten Fall zu alphanumerischen Werten erfolgt, wovon das Ergebnis wesentlich abhängt.

Abbildung 10.10 zeigt das Beispielscript `ifElse.pl`, welches die Bedeutung der unterschiedlichen Vergleichsoperatoren verdeutlicht.
Einzelne logische Ausdrücke können in Perl wie üblich durch logische Operatoren wie „oder“, „und“ und „nicht“ verknüpft werden. Auch hierfür stehen

in Perl Syntaxalternativen bereit, die sich in ihrer Priorität unterscheiden. Eine Übersicht ist in Tabelle 10.4 enthalten.

| Vergleich | Bedeutung |
|---|---|
| == | Gleichheit (numerisch) |
| eq | Gleichheit (alpha) |
| != | Ungleichheit (numerisch) |
| ne | Ungleichheit (alpha) |
| < | kleiner (numerisch) |
| lt | kleiner (alpha) |
| <= | kleinergleich (numerisch) |
| le | kleinergleich (alpha) |
| > | größer (numerisch) |
| gt | größer (alpha) |
| >= | größergleich (numerisch) |
| ge | größergleich (alpha) |

**Tabelle 10.3:** Vergleiche in Perl

### 10.3.7 Sequentielle Komposition

Die einfachste, aber auch wichtigste Kontrollstruktur ist das Zusammenfassen mehrerer Einzelanweisungen zu einem Block: die *sequentielle Komposition.* Perl verwendet hier die Nomenklatur der meisten modernen Sprachen, ein Paar geschweifte Klammern:

```
# Block in Perl
{
   Anweisung_1;
   ...
   Anweisung_n;
}
```

Die Scriptsprachen Python und Ruby haben für dieses Konstrukt eine abweichende Terminologie, was wir noch kennenlernen werden.

### 10.3.8 Entscheidungen

Die „klassische“ Kontrollstruktur ist die „wenn-dann-sonst“-Entscheidung. In Perl wird dies durch die Syntax

```
if (bedingung)
{
   wahr_anweisungen;
}
else
{
   falsch_anweisungen;
}
```

formuliert. Zu bemerken ist hier, dass die Paare geschweifter Klammern in Perl stets vorhanden sein müssen, auch wenn nur eine Anweisung folgt; der else-Teil kann selbstverständlich entfallen.

Diese Kontrollstruktur kann in Perl noch erweitert werden: Hat `bedingung` den logischen Werte false, so können weitere Bedingungen überprüft werden:

```
if (bedingung1)
{
   wahr_anweisungen;
}
elsif (bedingung2)
{
   falsch_wahr_anweisungen;
}
elsif (bedingung3)
{
   falsch_falsch_wahr_anweisungen;
}
...
```

| Schreibweise | Bedeutung |
|---|---|
| && | und |
| and | und (niedrigere Priorität) |
| \|\| | oder |
| or | oder (niedrigere Priorität) |
| ! | Negation |
| not | Negation (niedrigere Priorität) |

**Tabelle 10.4:** logische Operationen in Perl

```
else
{
   falsch_anweisungen;
}
```

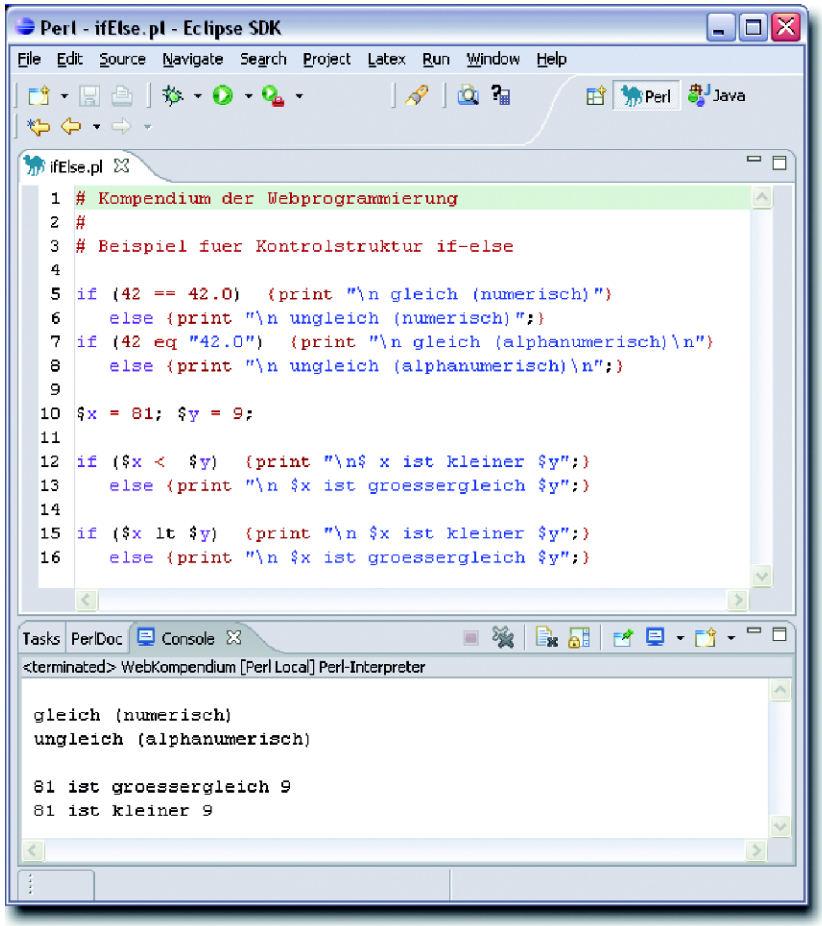



**Abbildung 10.10:**
Scipt `ifElse.pl` in Eclipse mit EPIC

### 10.3.9 Schleifen in Perl

Als letzte Kontrollstruktur fehlt uns noch die *Schleife*. Diese benötigen wir für die wiederholte Ausführung von Anweisungen, etwa für die Berechnung einer Summe wie

$$S(n) = \sum_{i=0}^{n} i \quad . \tag{10.1}$$

Je nach Programmiersprache stehen verschiedene Schleifentypen zur Verfügung; praktisch einheitlich ist die while-Schleife, die for-Schleife liegt gerade bei Scriptsprachen in unterschiedlicher Syntax vor, hinzu kommen weitere Typen je nach Sprache.

Einheitlich für alle Schleifentypen ist die Steuerung über eine logische Bedingung; die Schleifen werden je nach Wert dieser Bedingung weiter ausgeführt oder beendet.

Die Summierung (10.1) ist übrigens eine berühmte, mit dem Namen Carl-Friedrich Gauß verbundene Summe und es gilt

$$S(n) = \sum_{i=0}^{n} i = \frac{n(n+1)}{2} \quad .$$

Sie wird als Beispiel für die jeweiligen Schleifentypen in diesem Buch in den verschiedenen Sprachen verwendet.

#### 10.3.9.1 Die while-Schleife

Dieser Schleifentyp bildet den Grundtypus der Schleifen in fast allen Programmiersprachen. Die Anweisungen des Schleifenrumpfes werden dabei so lange ausgeführt, wie die Bedingung im Schleifenkopf den logischen Wert true hat.

Die Grundsyntax dieser Schleife in Perl lautet:

```
while (bedingung) {
   schleifenanweisung;
}
```

Wie in den meisten Programmiersprachen steht die Bedingung in runden Klammern; anders als üblich ist das Paar geschweifte Klammern in Perl zwingend, auch wenn der Schleifenkörper nur aus einer einzelnen Anweisung besteht – wie wir es schon bei der if-else-Entscheidung kennengelernt haben.

Ein Beispiel für die Berechnung der Formel (10.1) lautet in Perl:

```
# Kompendium der Web-Programmierung
#
# Perl: Kontrollstruktur while-Schleife

$n = 100; $i = 0; $summe = 0;

while($i <= $n) {
   $summe += $i;
   $i++;
}

print("\nErgebnis mit while-Summierung:  $summe");
```

**Listing 10.6:** Beispiel der while-Schleife in Perl (`schleife.pl`, Teil 1)

Die while-Schleife ist eine kopfgesteuerte Schleife, da hier die Bedingung am Kopf der Schleife vor deren erstem Durchlauf getestet wird; es kann deshalb vorkommen, dass der Schleifenkörper nie durchlaufen wird. Der nächste Schleifentyp erzwingt hingegen immer mindestens einen Durchlauf.

#### 10.3.9.2 Die do-while-Schleife

Die do-while-Schleife ist eine fußgesteuerte Schleife, die Bedingung wird erst am Ende der Schleife überprüft. Deshalb wird die Schleife immer mindestens einmal durchlaufen.

Das Beispiel lautet für diesen Schleifentyp:

```
$n = 100; $i = 0; $summe = 0;

do {
   $summe += $i;
   $i++;
```

```
} while($i <= $n);

print("\nErgebnis mit do-while-Summierung:  $summe");
```

**Listing 10.7:** Beispiel der do-while-Schleife in Perl (`schleife.pl`, Teil 2)

#### 10.3.9.3 Die until-Schleife

Eine eher seltene Schleife ist die in Perl implementierte *until-Konstruktion;* sie entspricht der while-Schleife, nur mit umgekehrter Auswertung der Logik: Die Schleife wird ausgeführt, so lange die Schleifenbedingung falsch ist.

#### 10.3.9.4 Die for-Schleife

Nach der while-Schleife ist die *for-Schleife* der am meisten verbreitete Typ. Sie dient dazu, den typischen „Wiederholungsfall“ einfacher zu behandeln.

Die Grundsyntax dieser Schleife in Perl und vielen anderen Sprachen lautet:

```
for(init; bedingung; inkrement)  {
   schleifenanweisung;
}
```

Dabei ist `init` eine Anweisung, die einmalig vor dem ersten Schleifendurchlauf ausgeführt wird; die Schleife wird so lange ausgeführt, wie `bedingung` den logischen Wert true hat, und nach jedem Durchlauf wird die `inkrement`-Anweisung einmal ausgeführt, um die Schleifenvariablen zu verändern.

Das Beispiel der Implementierung der Beziehung (10.1) mit diesem Schleifentyp in Perl lautet etwa:

```
$summe = 0;
for ($i = 0; $i <= $n; $i++)
{
   $summe += $i;
}
print("\nErgebnis mit for-Summierung: $summe");
```

**Listing 10.8:** Beispiel der for-Schleife in Perl (`schleife.pl`, Teil 3)

#### 10.3.9.5 Die foreach-Schleife

Eine für Scriptsprachen typische weitere Schleifenform erlaubt das Durchschreiten einer Liste, wie wir sie in 10.3.5.2 kennengelernt haben. In Perl ist dies die *foreach-Schleife.* Sie hat die Grundsyntax

```
foreach $i (@liste)
{
   anweisung;
}
```

Dabei nimmt die Variable `$i` jeden Wert aus der Liste `@liste` in der Reihenfolge steigender Indices genau einmal an.

Unser Beispiel der Summierung (10.1) kann man mit diesem Schleifentyp natürlich ebenfalls implementieren:

```
@summe = (0..$n); $summe = 0;
foreach $i (@summe)
{
   $summe += $i;
}
print("\nErgebnis mit foreach-Summierung: $summe");
```

**Listing 10.9:** Beispiel der foreach-Schleife in Perl (`schleife.pl`, Teil 4)

Abbildung 10.11 zeigt das vollständige Beispiel mit allen vier Schleifentypen in Perl mit Eclipse und EPIC.

#### 10.3.9.6 Vorzeitiges Verlassen von Schleifen

Schleifen dienen dem wiederholten Ausführen von Anweisungen und werden durch eine logische Bedingung gesteuert. Nun gibt es häufig den Fall, dass abhängig von weiteren Bedingungen die *Schleifenausführung vorzeitig beendet werden soll.* Je nach Programmiersprache stehen hierfür verschiedene Anweisungen zur Steuerung des Programmflusses bereit.

| Anweisung | Bedeutung |
|---|---|
| `last` | beendet Schleife |
| `next` | nächster Durchlauf |
| `redo` | wiederholt aktuellen Durchlauf |

**Tabelle 10.5:** Schleifenabbruch in Perl

In Perl bedeutet `last`, dass die aktuell ausgeführte Schleife beendet wird. Es entspricht somit dem `break` der Programmiersprache Java. Bei geschachtelten Schleifen wird durch `last` die innere Schleife beendet.

Die Perl-Anweisung `next` erzwingt den nächsten Schleifendurchlauf (mit erneuter Überprüfung der Bedingung); es entspricht dem `continue` von Java.

`redo` wiederholt den aktuellen Schleifendurchlauf; es hat keine Entsprechung in Java.

### 10.3.10 Programmsteuerung in Blöcken

Moderne Programmiersprachen verwenden häufig Ausnahmen (Exceptions), die ausgelöst werden, wenn gewisse Sondersituationen auftreten. Dann wird der normale Programmfluss verlassen und in die Ausnahmebedingung verzweigt. Java verwendet hierfür Exceptions (Klassen, die von `java.lang.Throwable` abgeleitet sind und die Kontrollstruktur `try - catch - finally` haben); der try-Block wird bei Auslösen der Exception dann verlassen.

Perl bietet hier lediglich die Möglichkeit, die Auswertung eines Ausdrucks (oder einen ganzen Block) zu kapseln

```
eval  Ausdruck
```

Bei Abbruch von `Ausdruck` wird dann nur die Ausführung dieses Blockes beendet, ähnlich zum try-Block von Java.

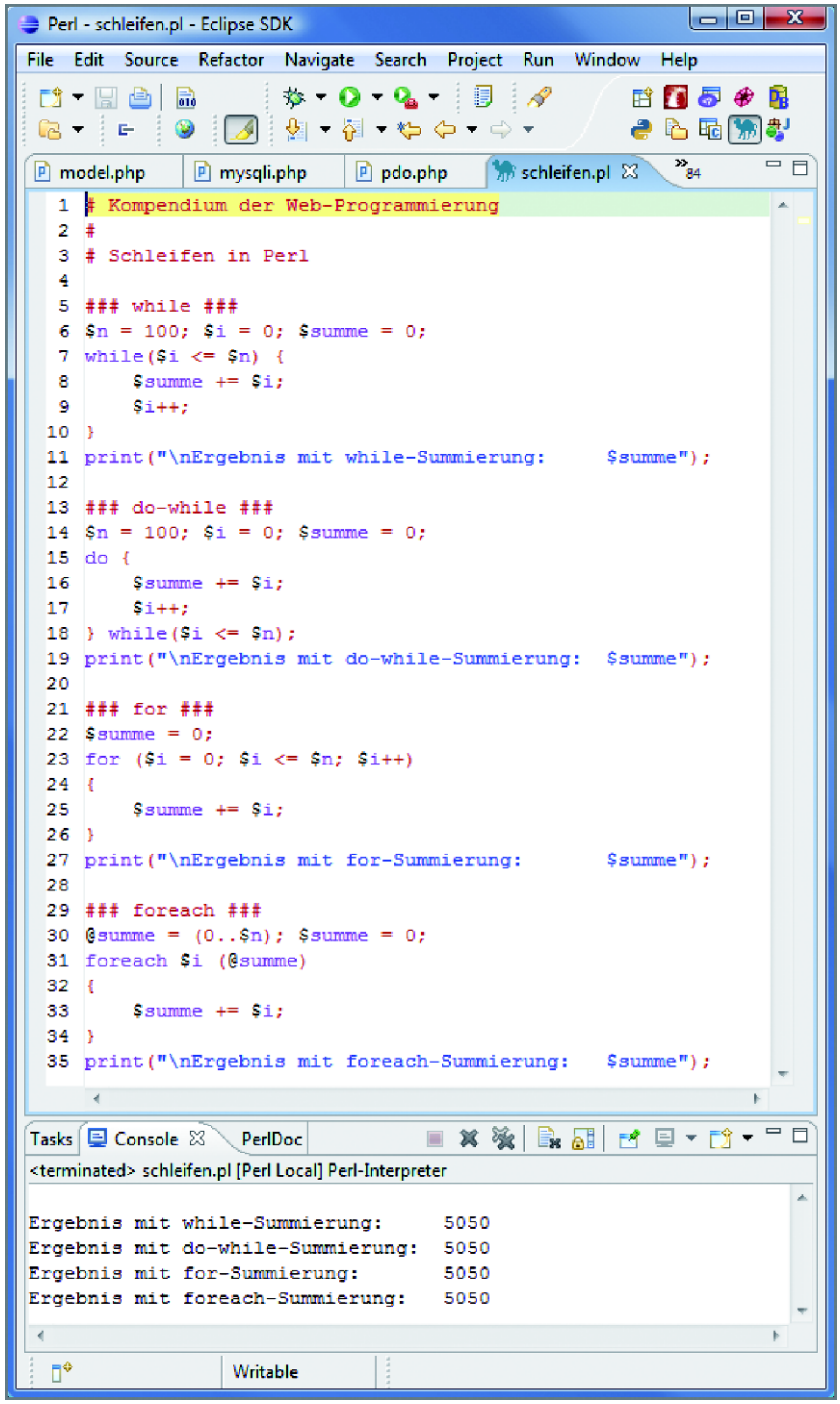



**Abbildung 10.11:**
Die vier Schleifentypen in Perl anhand des Beispiels (10.1) (Script `schleifen.pl`)

### 10.3.11 Beenden von Perl

Die Ausführung eines Perl-Scriptes kann gezielt *beendet* werden. Hierfür stellt Perl zwei unterschiedliche Anweisungen bereit:

- `exit expr` wertet den Ausdruck `expr` aus und beendet die Progammausführung mit der Ausgabe der Auswertung von `expr`; wird auf `expr` verzichtet, wird der Rückgabewert 0 ausgegeben;

- mit `die` wird das Perl-Script direkt beendet und die optionale Information auf STDERR (vgl. 10.3.13.1) ausgegeben.

Trotz ihrer vermeintlichen Gleichheit unterscheiden sich diese zwei Methoden, wenn sie im Zusammenhang mit einem `eval`-Block nach 10.3.10 verwendet werden: `exit` beendet die Ausführung des gesamten Scripts, während `die` nur den eval-Block beendet.

Häufig werden diese Anweisungen zur Progammbeendung in einer oder-Konstruktion der Art

```
open(H "meineDatei.txt") or die("Datei nicht geoeffnet");
```

verwendet (die `open`-Methode lernen wir in 10.3.13 kennen).

#### 10.3.11.1 Fehlermanagement in Perl

Perl stellt einige Funktionalität zur *Fehlerbehandlung* bereit, auch wenn es nicht so weit geht wie andere moderne Sprachen. Bereits angesprochen wurde das Kapseln von Blöcken durch `eval`.

Die Perl-Variable

```
$!
```

speichert die letzte Fehlermeldung und ermöglicht so eine gesteuerte Fehlerausgabe. Hierfür können auch

```
__FILE__
```

für den Namen des Scriptes und

```
__LINE__
```

für die Zeilennummer des Fehlers genutzt werden.

### 10.3.12 Funktionen in Perl

#### 10.3.12.1 Integrierte Funktionen

Die Programmiersprache Perl verfügt über eine Vielzahl *integrierter Funktionen,* wovon wir schon einige kennengelernt haben. Eine vollständige Übersicht über die Standardfunktionen ist in [WCS00] zu finden; diese werden ergänzt um weitere, die in den gängigen Perl-Modulen enthalten sind und noch weiter durch die globale Perl-Bibliothek, die CPAN (wie 10.5.4) bietet.

Hier kann keine auch nur annähernd vollständige Auflistung der Perl-Funktionen erfolgen; einige typische werden vorgestellt.

**Basisfunktionalität**

Hierzu zählen elementare Ausgabe-Funktionen, Zeichenkettenmanipulation und mehr:

- Ausgabe: `print` und synonym dazu `echo`;
- Entfernen des letzten Zeichens, wenn es das Symbol für neue Zeile ist: `chomp`;
- Entfernen des letzten Symbols: `chop`;
- Teilzeichenkette: `substr($string,$pos,$laenge)` bestimmt aus der Zeichenkette `$string` eine Teilzeichenkette ab der Position `$pos` – wobei die Nummerierung mit 0 beginnt – der Länge `$laenge`.

**Mathematisches**

Mathematische Basisfunktionen gehören zur Grundausstattung von Perl; Beispiele hierfür sind:

- Absolutwert `abs`;
- Exponentialfunktion `exp`, natürlicher Logarithmus (Basis *e*) `log`;
- trigonometrische Funktionen mit Argument in Bogenmaß: `sin`, `cos`, `tan`;
- hexadezimale Darstellung einer ganzen Zahl: `hex`.

**Listenfunktionen**

Dies sind die Funktionen, die wir in 10.3.5.2 und 10.3.5.3 kennengelernt haben:

- `push` und `pop` für ein klassisches Stack (LIFO);
- `unshift` und `shift`;
- Funktionen für assoziative Arrays wie `values` und `keys`.

**Weiteres**

Neben den vorgestellten Funktionsbereichen gibt es zahlreiche weitere, einige lernen wir hier kennen.

- Methoden für modulare Software: `use`;
- Netzwerkfunktionalität: `gethostbyaddr`;
- Zeitfunktionen, vgl. 10.4.4.

#### 10.3.12.2 Selbstdefinierte Methoden

Neben den vorgegebenen Funktionen in Perl können natürlich eigene Methoden geschrieben werden. Diese können wiederum in sinnvollen Bibliotheken gebündelt, verteilt und eingebunden werden. Perl verwendet hierfür Module.

Die Möglichkeiten der strukturierten Erstellung von wiederverwertbarem Code in Perl werden in 10.3.14 vorgestellt.

#### 10.3.12.3 Überladen

Als *Überladen* bezeichnet man die Möglichkeit, verschiedene Methoden mit dem gleichen Namen zu implementieren. Ermöglicht wird dies in Hochsprachen durch die dann notwendig unterschiedliche Signatur, also eine unterschiedliche Kombination von Methodennamen und Anzahl/Typ der Argumente.

Perl bietet dieses klassische Überladen nicht, wohl aber eine alternative Form: Es wird unterschieden, ob eine Methode im skalaren oder im Listen-Kontext aufgerufen wird; man kann also die zwei Aufrufe

```
$x = f();     # skalarer Kontext
@x = f();     # List-Kontext
```

mit unterschiedlichen Methoden `f` implementieren.

Perl stellt mit der Methode

```
wantarray
```

eine Möglichkeit bereit abzufragen, ob eine Methode im List-Kontext aufgerufen wurde; dies ermöglicht die Implementierung dieser Art von Überladung in selbstdefinierten Methoden.

### 10.3.13 Dateizugriff mit Perl

Perl verhält sich bei einem *Dateizugriff* vergleichsweise einfach: Jede IO-Verbindung wird in einem Handle verwaltet. Für diese werden üblicherweise Bezeichner in Großbuchstaben verwendet. Eine IO-Verbindung mit dem Handle `H` wird etwa durch

```
open(H, $file);
```

erzeugt, wobei noch festzulegen ist, ob diese Verbindung zum Lesen oder Schreiben ist. Diese Methode liefert einen Rückgabewert, der als logischer Wert false bedeutet, wenn das Öffnen fehl schlug.

Umgekehrt wird diese Verbindung durch

```
close(H);
```

wieder geschlossen und die allokierten Systemresourcen werden freigesetzt.

Das Öffnen einer Datei ist letztlich der Zugriff auf eine sequentielle Datenstruktur. Dieser Zugriff positioniert einen virtuellen Dateizeiger, zunächst auf den Anfang der Datei. Es ist schon beim Öffnen der Datei die Art des Zugriffes zu unterscheiden:

- Lesender Zugriff: dies ist der Standardfall; die Datei kann nur gelesen werden beginnend von der ersten Position.

  Syntax: lesender Zugriff wird ohne weitere Kennzeichnung oder alternativ durch ein dem Dateinamen vorangestelltes < gekennzeichnet:

  ```
  open(H, $file)
  open(H, "<$file")
  ```

- Schreibender Zugriff: existiert eine Datei gleichen Namens wird diese gelöscht und wir beginnen ab der ersten Position die Datei neu zu schreiben.

  Syntax: schreibender Zugriff wird durch ein dem Dateinamen vorangestelltes > gekennzeichnet:

  ```
  open(H, ">$file")
  ```

- Anhängen am Dateiende: die dritte klassische Variante beim Zugriff auf sequentielle Strukturen wie Dateien ist das Anfügen: nach dem Öffnen steht der Dateizeiger bereits am Ende der Datei; ab dieser Position wird geschrieben.

Das Schreiben an einen Handle `H` geschieht in Perl durch die Syntax

```
print(H $zuschreibendes)
```

wobei auch mehrere, dann durch Komma getrennte Argumente an den Handle `H` geschrieben werden können.

Gelesen werden Dateien zeilenweise; durch die Syntax

```
$zeile = <H>;
```

wird eine Zeile von Handle `H` gelesen. Diese Operation kann auch sehr praktisch im Listenkontext (vgl. 10.3.12.3) verwendet werden durch

```
@datei = <H>;
```

Hier wird die durch den Handle `H` geöffnete Datei komplett eingelesen und jede Zeile dieser Datei in einem Element der Liste `datei` in der Reihenfolge ihres Auftretens abgelegt.

Neben den beschriebenen Möglichkeiten, eine Datei zum Lesen, Schreiben oder Anfügen zu öffnen, bietet Perl eine weitere Möglichkeit, um zunächst zu lesen und dann zu schreiben. Dies wird durch die Syntax

```
open(H, "+>$file");
open(H, "+<$file");
```

erreicht. Eine vollständige Übersicht über die Modi der `open`-Methode in Perl gibt Tabelle 10.6.

| Syntax | Bedeutung |
|---|---|
| `datei`<br>`<datei` | Lesen |
| `>datei` | Schreiben<br>bestehende Datei<br>wird gelöscht |
| `»datei` | Anfügen an Datei<br>oder Anlegen neue Datei |
| `+<datei`<br>`+datei` | Lesen und Schreiben<br>Fehler, wenn Datei<br>nicht existiert |
| `+>datei` | Lesen und Schreiben<br>existierende Datei<br>wird gelöscht |
| `+»datei` | Lesen und Schreiben<br>beginnt am Dateiende<br>oder erzeugt Datei |

**Tabelle 10.6:**
Modi der Perl-Methode `open`

#### 10.3.13.1 Standard-Handles

Die Programmiersprache Perl definiert drei *Standard-Handles* für die Eingabe, Ausgabe und Fehlerausgabe:

- `STDIN` ist die Standardeingabe, also die Eingabe über die Tastatur;
- `STDOUT` ist der Handle zur Standardausgabe (Konsole);
- über `STDERR` wird die Default-Ausgabe für Fehlermeldungen angesprochen.

Damit ist die in 10.3.5.1 auf Seite 131 eingeführte Syntax für das Lesen über die Tastatur klar geworden:

```
$eingabe = <STDIN>;
```

bedeutet also, dass genau eine Zeile über die Tastatur eingelesen wird.

#### 10.3.13.2 Beispiel

Ein typisches Beispiel für die Verwendung von Dateien ist ein einfacher Seitenzähler auf einer Web-Site; dieses Beispiel erfordert noch etwas CGI-Kenntnisse, weshalb wir es in 10.4.5 kennenlernen werden.

#### 10.3.13.3 Filetests

Perl bietet sehr viele nützliche Methoden, die uns die praktische Arbeit erleichtern. Dazu gehören die *Filetests,* mit deren Hilfe wir die Existenz und die Zugriffsrechte von Dateien und Verzeichnissen einfach überprüfen können.

Ein Filetest wird als boolescher Wert interpretiert. Exemplarisch wird durch die Syntax

```
-e $datei
```

überprüft, ob die Datei, deren Name in der Variablen `$datei` abgelegt ist, existiert.

Tabelle 10.7 gibt eine Übersicht über die Filetests von Perl.

Zur Verdeutlichung betrachten wir wieder das Beispielscript `zaehler.pl` aus Abschnitt 10.4.5 und schreiben ein Script zur Überprüfung, ob die notwendige Zählerdatei korrekt angelegt werden kann. Dieses Script kann folgendermaßen formuliert werden:

```
# Kompendium der Web-Programmierung
#
# Filetest in Perl: Testen der Datei zaehler.txt
#                   fuer CGI-Script zaehler.pl

$spooldir = '/data/www/tmp/';
$file = $spooldir . 'zaehler.txt';

if (-e $file)  {
  print "\n Zaehlerdatei vorhanden!\n";
}  else  {
  die "\n Zaehlerdatei fehlt!\n";
}

if ((-r $file) && (-w $file))  {
  print "\n Kann Datei lesen und schreiben!\n\n";
}  else  {
  print "\n Zugriff fehlerhaft!\n\n";
}
```

**Listing 10.10:** Filetest für die Zähler-Datei (Script `filetest.pl`)

| Syntax | Bedeutung |
|---|---|
| `-e` | Existenz |
| `-r` | Datei lesbar |
| `-w` | Datei schreibbar |
| `-x` | Datei ausführbar |
| `-d` | Verzeichnis |
| `-l` | symbolischer Link |
| `-M` | Anzahl Tage seit letzter Änderung |

**Tabelle 10.7:** Filetests in Perl

Bei der Anwendung von Script `filetest.pl` ist aber darauf zu achten, dass das Script in der gleichen Benutzerkennung ausgeführt wird, die auch später den entsprechenden CGI-Prozess betreibt, denn nur dann können die überprüften Rechte für diese Kennung gewährleistet sein.

### 10.3.14 Strukturierte Programmierung in Perl

Um sinnvoll mit der Programmiersprache Perl umzugehen, benötigen wir noch Kenntnisse, wie in Perl strukturierte Anwendungen erstellt werden können. Zunächst werden wir dazu eigene Methoden in der Art klassischer, prozeduraler Unterprogramme erstellen, danach die Ansätze zur Objektorientierung in Perl kennenlernen und die Verteilung von Code besprechen.

#### 10.3.14.1 Subroutines in Perl

Perl verwendet für die Deklaration von Unterprogrammen das Schlüsselwort `sub`. Ein solches Unterprogramm mit dem Bezeichner `fakultaet` wird beispielsweise durch

```
sub fakultaet {
   ### Anweisungen
}
```

deklariert. Bemerkenswert an dieser Deklaration ist:

**Subroutines in Perl haben keine explizite Parameterliste, sondern verwalten die ihnen übergebenen Parameter in einer internen Liste mit dem Bezeichner `@_`.**

Unterprogramme können an beliebiger Stelle, auch nach ihrer Verwendung, im Perl-Code vorkommen. Es ist gute Konvention, am Ende des Scriptes die Unterprogramme übersichtlich aufzuführen.

Der Aufruf von Unterprogrammen in Perl ist sehr flexibel; für das angegebene Beispiel sind folgende drei Möglichkeiten gegeben:

```
$ergebnis = fakultaet(5);
$ergebnis = &fakultaet;
$ergebnis = fakultaet;
```

Die erste Art des Aufrufs ist die heute gewöhnliche, bei der eine Liste von Parametern dem Unterprogramm übergeben werden kann. Diese Parameter werden dann in der Liste

```
@_
```

in ihrer Reihenfolge abgelegt. Hier kann etwa das übergebene Argument `5` über das erste Element der Liste abgerufen werden, also über

```
$_[0]
```

Die zweite Variante des Aufrufs ist die ältere Perl-Syntax, die `&` als Kennzeichnung von Unterprogrammen verwendete; hier ist ein gesonderter Mechanismus zur Übergabe der Parameter, üblicherweise über globale Variablen, notwendig. Die dritte Version entspricht der zweiten ohne die besondere Kennzeichnung.

Abbildung 10.12 zeigt das klassische Beispiel der Informatik: die rekursive Berechnung der Fakultät aus

$$\text{fakultaet}(n) = \begin{cases} n \cdot \text{fakultaet}(n-1) \text{ für } n > 1 \\ 1 \text{ für } n = 0 \end{cases} \tag{10.2}$$

Dies kann in Perl nun direkt implementiert werden (Script `subroutine.pl` aus Abbildung 10.12).

In diesem Beispiel zeigt sich bereits eine wichtige Eigenschaft der Liste `@_`: Offensichtlich existiert für jeden Aufruf des Unterprogramms eine eigene Version dieser Liste; dies wird in 10.3.14.2 genauer erklärt.

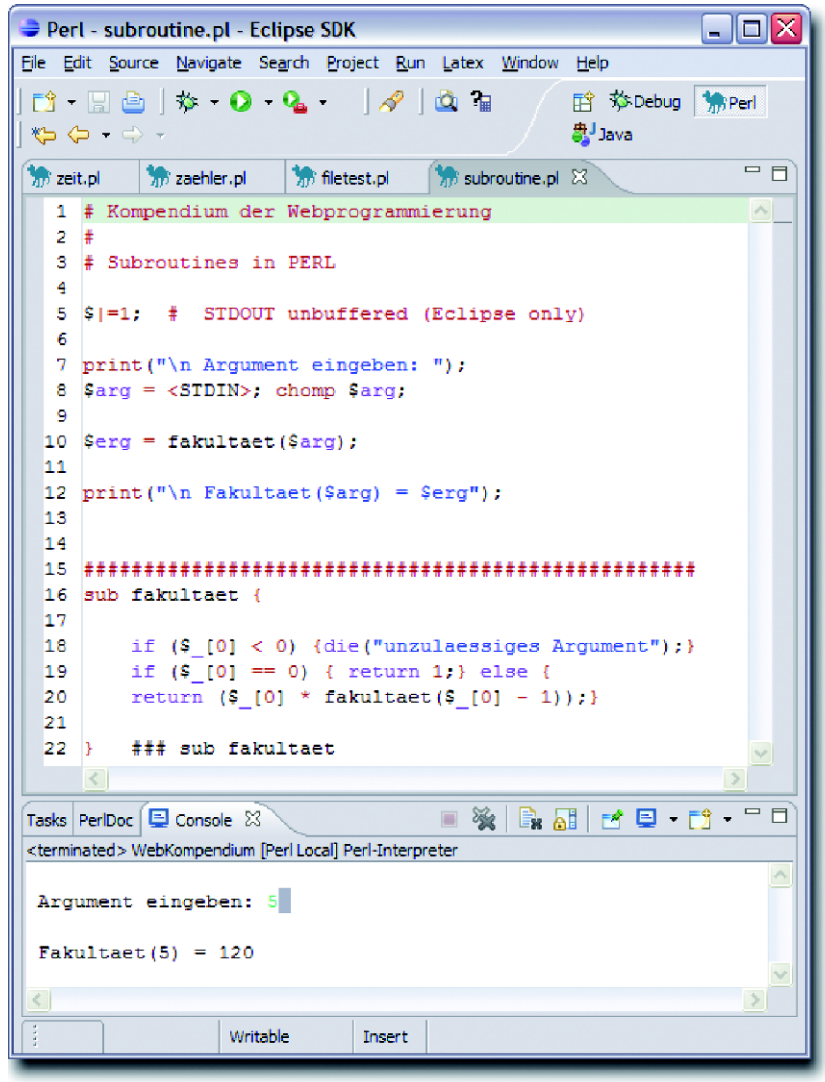



**Abbildung 10.12:** Rekursive Berechnung der Fakultät in Perl mit Unterprogramm (Script `subroutine.pl`)

### 10.3.14.2 Lokale und globale Variablen in Perl

Eine Variable ist *global,* wenn sie ohne Übergabemechanismen in allen Programmsegmenten gültig, also zugreifbar, ist. Der Vorteil globaler Variablen ist die Bequemlichkeit, der Nachteil ist die Fehleranfälligkeit sowie die schlechte Wartbarkeit des Codes.

**Variablen (ohne die später beschriebenen Einschränkungen `local` und `my`) in Perl sind global.**

Perl verhält sich hier abweichend zu den meisten modernen Programmiersprachen, die stets von der *lokalen Variablen* ausgehen.

Die Bedeutung dieser Aussage soll an dieser Stelle veranschaulicht werden: Der Code aus Abbildung 10.13 zeigt ein Script mit zwei einfachen Unterpro-

grammen, welche eine Variable `$n` setzen – und damit die Variable aus dem Hauptprogramm mit dem gleichen Bezeichner überschreiben.

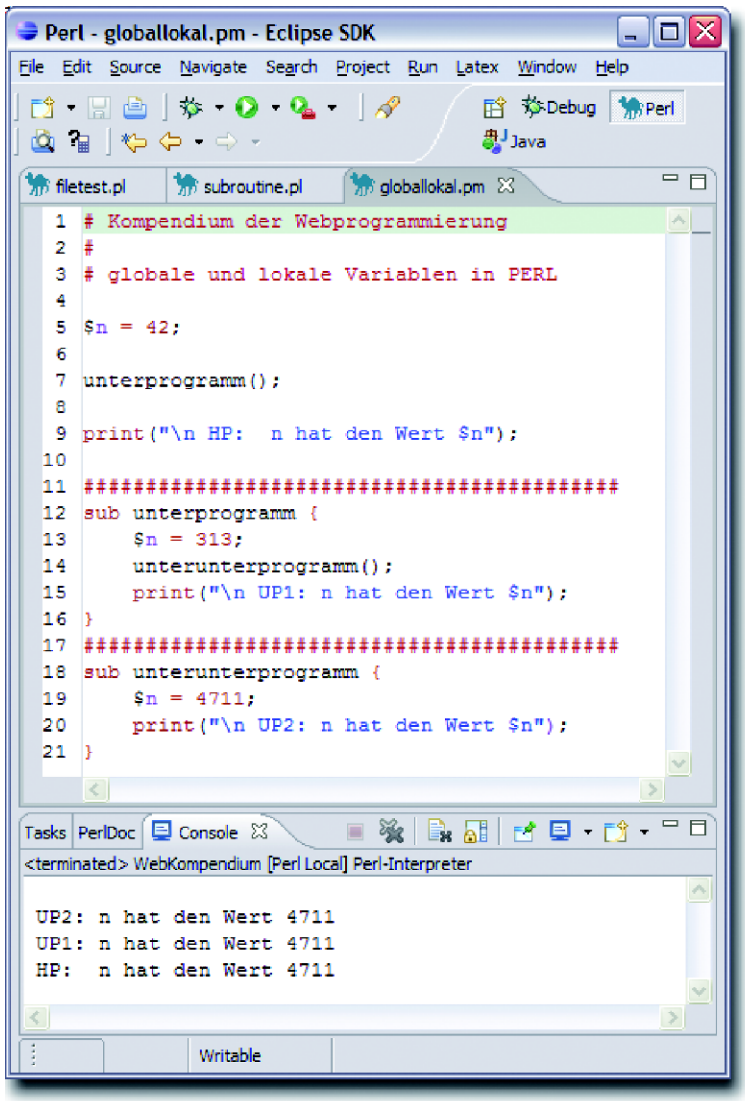



**Abbildung 10.13:**
Bedeutung der globalen Variablen (Script `globallokal.pl`)

Es ist aber auch möglich, in Perl Variablen als lokal zu deklarieren; hierfür stehen zwei Schlüsselwörter bereit:

- `my` : hierdurch wird im Hauptprogramm die Gültigkeit der Variablen so eingeschränkt, dass die Variable nur im Hauptprogramm und in den von diesen aufgerufenen Unterprogrammen gültig ist; `my` in einem Unterprogramm schränkt die Gültigkeit nur auf dieses Unterprogramm ein;
- `local` : erzeugt eine lokal gültige Kopie der Variablen, die nur im jeweiligen Unterprogramm gültig ist.

Die Bedeutung dieser Schlüsselwörter kann mit dem Beispielscript aus Abbildung 10.13 verdeutlicht werden:

- die Deklaration in Zeile 5 kann mit `my` und `local` versehen werden, ohne dass sich etwas ändert;
- wird in Zeile 13, bei der Deklaration der Variablen im ersten Unterprogramm, `my` verwendet, so ändert sich nur die Ausgabe dieses Segmentes

auf 313; wird an dieser Stelle `local` verwendet, so gibt das Hauptprogramm wieder 42 aus, da beide Unterprogramme mit einer lokalen Kopie von `$n` arbeiten;
- wird im zweiten Unterprogramm bei der Deklaration von `$n` in Zeile 19 `my` oder `local` verwendet, ändert sich dadurch nur die Ausgabe dieses Unterprogramms – dieses Unterprogramm ändert dann also nicht mehr die vorher gesetzten Werte.

#### 10.3.14.3 Namensräume

Voraussetzung für die Bereitstellung von wiederverwendbarem Code ist die Existenz separater *Namensräume,* damit es nicht zu Überschneidungen von Variablen in verschiedenen Segmenten wie im Script nach Abbildung 10.13 kommt.

Die Schlüsselwörter `my` und `local` stellen in Perl diese Namensräume bereit.

#### 10.3.14.4 Referenzen in Perl

Ähnlich wie C stellt Perl (seit der Version 5) Referenzen bereit. Eine *Referenz* ist ein Verweis auf eine Variable (skalarer Wert, (assoziative) Liste) oder auf ein Unterprogramm; die Referenz selbst ist aber eine skalare Variable, da sie nur einen Wert (einen Verweis) enthält.

Wie in Java und abweichend zu C merkt sich Perl, wieviele Referenzen auf ein Objekt zeigen in einem Referenzenzähler. Verweist keine Referenz auf ein Objekt, kann dieses speicherbereinigt, also gelöscht, werden.

Erzeugt wird eine Referenz durch den Referenzierungsoperator `\`. Im Beispiel

```
my $pi  = 3.1415926;
my $ref = \$pi;
```

ist `$pi` eine normale, skalare und lokale Variable, während `$ref` eine Referenz auf diese Variable ist. Der Wert dieser Referenz kann direkt ausgegeben werden, und man erhält etwa

```
Wert von $ref: SCALAR(0x2257c8)
```

bei der Ausgabe von `$ref`: Ein skalarer Wert an der hexadezimalen Adresse 2257c8.

Um den Wert der über eine Referenz verwiesenen Variablen zu erhalten, muss die Referenz dereferenziert werden. Dies geschieht bei einer Referenz auf eine skalare Variable durch Voranstellen des `$`:

```
print($$ref);
```

liefert im Beispiel den Wert der Variablen `$pi`.

Referenzen auf Listen funktionieren nach dem gleichen Prinzip; mittels

```
$listeref = \@liste;
```

wird die skalare Referenz `$listeref` als Referenz auf die Liste `@liste` erzeugt. Dereferenzierung erfolgt nun zu einem bestimmten Index der referenzierten Liste, wofür zwei Syntaxvarianten zur Verfügung stehen:

```
$$listeref[0]
$listeref->[0]
```

Das gleiche Prinzip gilt für Referenzen auf assoziative Listen, die mit

```
$hashref = \%hash
```

erzeugt werden. Dereferenzierung etwa zum Key `springer` erfolgt dann über eine der zwei Alternativen

```
$$hashref{"springer"}
$hashref->{"springer"}
```

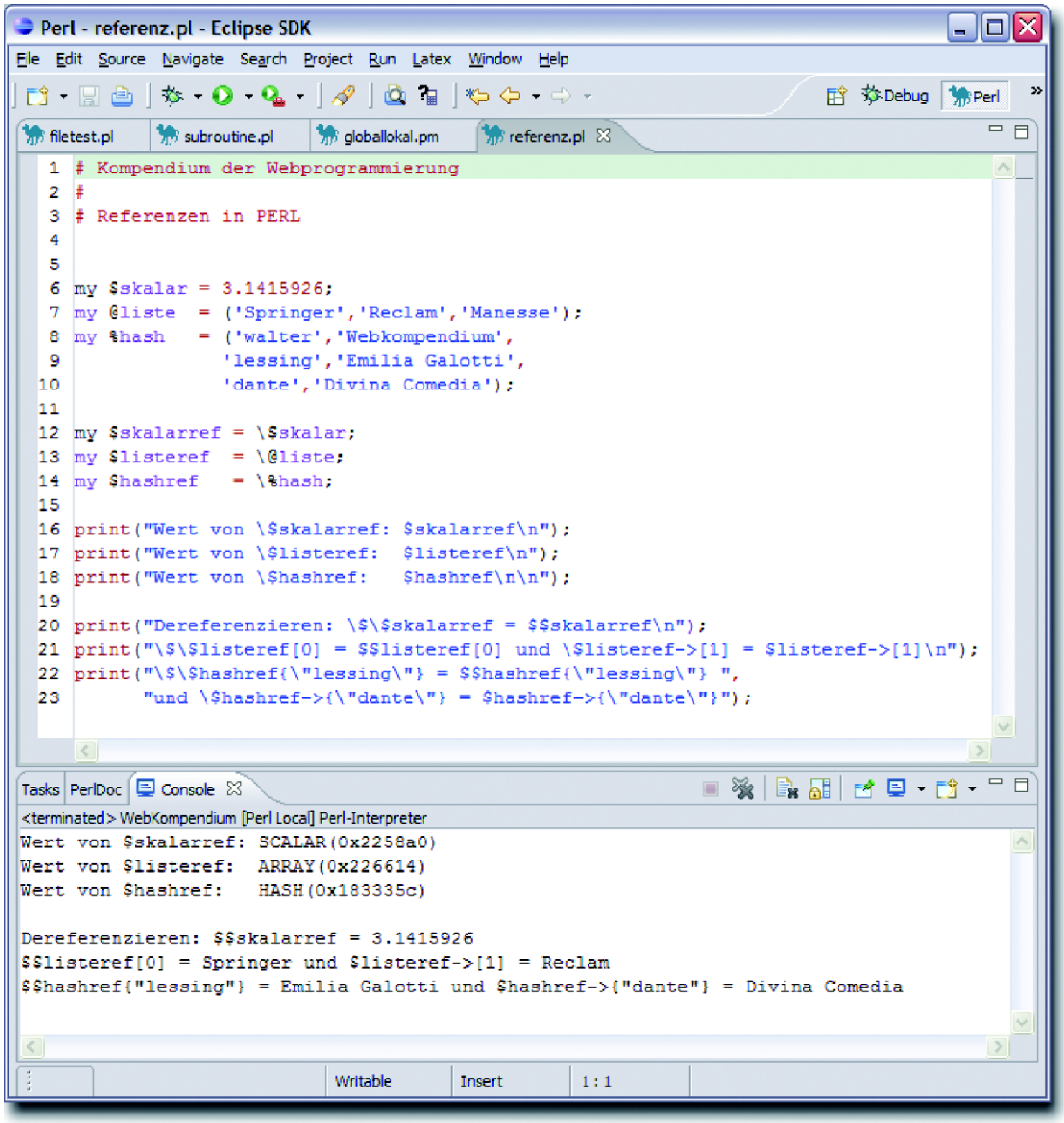



**Abbildung 10.14:** Referenzen auf Skalar, Liste und assoziative Liste und Dereferenzierung (Script `referenz.pl`)

Das Script `referenzen.pl` aus Abbildung 10.14 zeigt ein Gesamtbeispiel.

Wir werden bei der Verarbeitung von HTML-Formularen in Abschnitt 10.6 genauer mit Referenzen auf assoziative Listen arbeiten.

### 10.3.15 Reguläre Ausdrücke

*Reguläre Ausdrücke* sind eine der Stärken der Programmiersprache Perl, was nicht verwundert, da Perl ja von der Shell-Programmierung und Unix kommt und damit mit Tools wie awk verwandt ist.

Ein regulärer Ausdruck beschreibt Regeln zum Bilden von Untermengen von Zeichenketten. Sie bilden in der Informatik eine formale Sprache.

Es steht eine größere Anzahl von Möglichkeiten für das Bilden von regulären Ausdrücken zur Verfügung, die Gesamtzahl und genaue Syntax variiert etwas mit dem verwendeten Betriebssystem bzw. der verwendeten Programmiersprache. Die wichtigsten (und überall identischen) sind:

- `.` beliebiges Zeichen (außer Zeilenende)
- `[AB]` entweder A oder B
- `[0-9]` Zeichen von 0 bis 9
- `[A-Z0-9]` ein Großbuchstabe oder eine Ziffer
- `[^a]` beliebiges Zeichen außer a
- `?` der voranstehende Ausdruck kann 0 oder 1 mal vorkommen
- `+` der voranstehende Ausdruck kann 1 oder mehrfach vorkommen
- `*` der voranstehende Ausdruck kann 0 oder beliebig oft vorkommen
- `{n}` der voranstehende Ausdruck genau n mal
- `{n,}` der voranstehende Ausdruck mindestens n mal
- `{n,m}` der voranstehende Ausdruck mindstens n und höchstens m mal
- `(...)` Klammerung: zusammenfassen von Sequenzen
- `|` Alternative
- `^` Zeilenanfang
- `$` Zeilenende
- `\d` Ziffer, also gleichbedeutend zu `[0-9]`
  `\D` ist das Komplement dazu, also ein beliebiges Zeichen außer einer Ziffer
- `\w` Wortsymbol, also gleichbedeutend zu `[0-9a-zA-Z]`; deutsche Umlaute oder etwa „ß“ sind nicht enthalten;
  `\W` ist das Gegenteil von `\w`
- `\` maskiert nachfolgendes Zeichen

In Perl wird ein regulärer Ausdruck durch den Slash gekennzeichnet:

```
/buch/
```

ist der reguläre Ausdruck genau für die Zeichenkette „buch“. Ein solcher regulärer Ausdruck kann noch mit Optionen (Flags) verwertet werden, etwa

```
/buch/g
```

Die wichtigsten Optionen sind:

- `g`: globale Suche;
- `i`: ignoriert Groß- und Kleinschreibung;
- `m`: ermöglicht mehrzeilige Eingaben;
- `s`: ermöglicht nur einzeilige Eingaben;
- `x`: ermöglicht erweiterte reguläre Ausdrücke mit weiteren Sprachregeln.

Sollen Zeichen im regulären Ausdruck enthalten sein, welche eines der Symbole mit Sonderbedeutung enthalten, so ist dieses Symbol durch Voranstellen eines Backslash zu maskieren.

Mittels der Überprüfung `=~` kann getestet werden, ob etwa eine Textvariable einen regulären Ausdruck enthält; die Negation wird durch `!~` codiert. Beispiel `regex.pl` aus Abbildung 10.15 zeigt eine solche Anwendung in Perl.

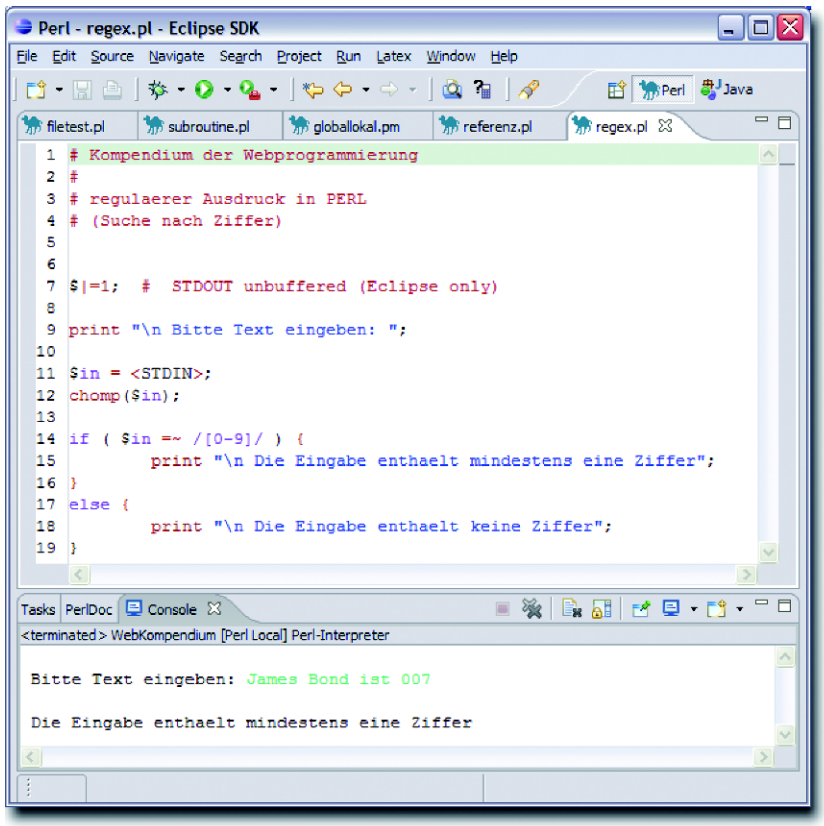



**Abbildung 10.15:** Anwendung von regulären Ausdrücken in Perl (Script `regex.pl`)

Eine ganz zentrale Anwendung von regulären Ausdrücken stellt die Aufspaltung von Zeichenketten anhand des Auftretens von regulären Ausdrücken dar; hierfür bietet Perl etwa die Methode `split`; diese spaltet das zweite Argument (sonst die Variable `$_`) anhand des als erstes Argument übergebenen regulären Ausdrucks auf und liefert eine entsprechend untergliederte Liste zurück. Etwa liefert

```
@teile = split(/[0-9],$eingabe);
```

eine Liste, indem an jeder Ziffer der Wert der Variablen `$eingabe` aufgespalten wird.

Besonders nützlich ist auch der Substitutionsoperator

```
s/regex/Ersetzung/
```

Hier wird bei jedem Auftreten des regulären Ausdrucks `regex` dieser durch `Ersetzung` ersetzt.

Der Substitutionsoperator wirkt default-mäßig auf die Variable `$_`, kann aber mittels

```
=~
```

auch auf andere Variablen angewendet werden, etwa

```
$string = "Murmeltier";
$string =~ s/tier/teil/;
print $string;
```

liefert die Ausgabe „Murmelteil“.

## 10.4 Einfachste CGIs mit Perl

Nach dieser „Schnelleinführung“ kennen wir die verbreitete Scriptsprache Perl grundlegend; wir wollen nun mit diesem Wissen eine *einfache CGI-Anwendung* schreiben und auf einem Apache-Webserver implementieren.

Um eine CGI-Anwendung in Perl auszuführen, muss aber der Webserver den Perl-Interpreter starten können; hierzu muss er wissen, wo er diesen Interpreter findet. Dafür gibt es die Shebang-Zeile.

### 10.4.1 Die Shebang-Zeile

Damit der Webserver das Perl-Script ausführen kann, ändern wir es so ab, dass das Perl-Script formal keine zu interpretierende Script-Datei, sondern ein ausführbares Programm wird. Dazu nutzen wir eine Eigenschaft der Betriebssystem-Shell: Beginnt ein Shell-Script mit dem Sonderkommentar `#!`, so wird der Rest dieser Zeile als Interpreterdatei verstanden, man spricht von einer *Shebang-Zeile*. Typische Shebang-Zeilen unter Unix/Linux und Windows lauten:

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

#!"c:/programme/Perl/bin/Perl.exe" -w
```

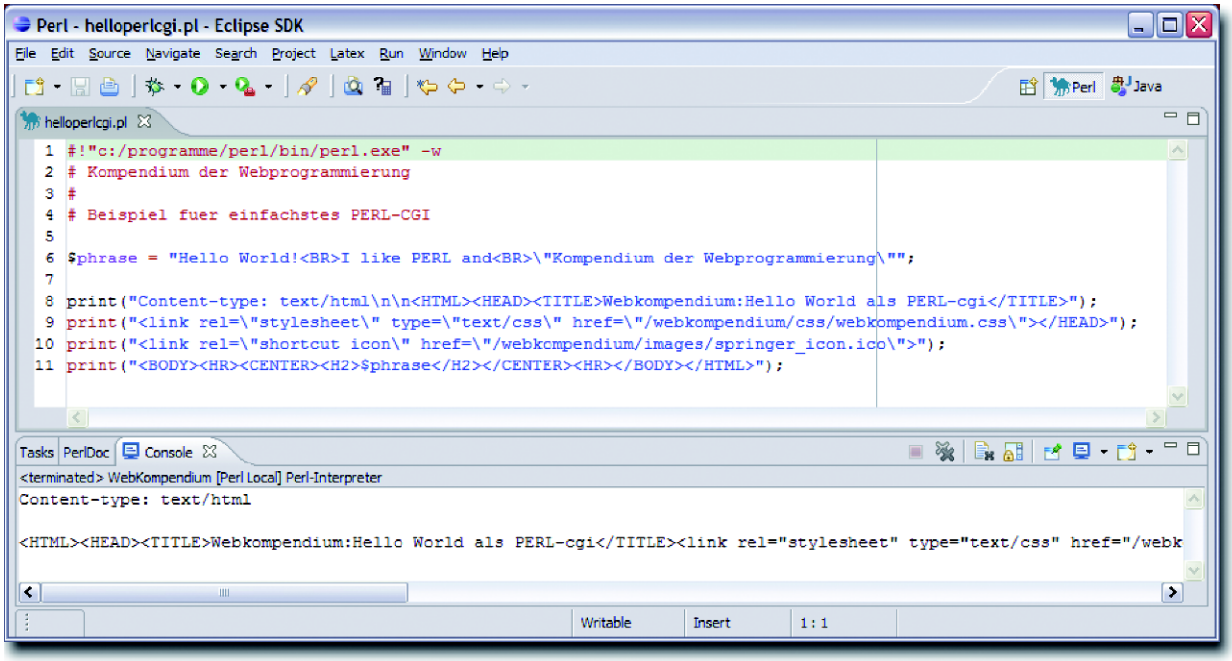



**Abbildung 10.16:** Einfaches Hello-World-CGI mit Perl (Script `hellocgi1.pl`)

### 10.4.2 Das erste Perl-CGI

Unser erstes CGI mit Perl soll nun nichts weiter tun, als uns in guter „Hello-World-Manier“ zu begrüßen, dabei aber schon unsere CSS-Datei aus Abschnitt 2.5 sowie ein typisches Favicon verwenden. Das Perl-Script aus Abbildung 10.16 leistet dies.

Diese Anweisung kann nun zunächst direkt auf der Kommandozeile oder im Eclipse-Plugin ausgeführt werden und man bekommt als Ausgabe den durch

das Perl-Script „berechneten“ HTML-Code (Abbildung 10.17). Um das Script im Browser auszuführen, ist es notwendig, die Perl-Datei in den Unterordner `cgi-bin` des Apache-Verzeichnisses zu kopieren. Über den HTTP-GET-Request der Browser-URL
`http://<rechnername>/cgi-bin/helloperlcgi.pl` geschieht dann folgendes:

- der Apache-Webserver startet einen CGI-Prozess;
- wegen der Shebang-Zeile wird der Perl-Interpreter gestartet, das Script übersetzt und dann ausgeführt;
- die Ausgabe wird an den Webserver und durch diesen als HTTP-Response an den Webclient weitergegeben.

Schließlich entsteht die Ausgabe nach Abbildung 10.17.

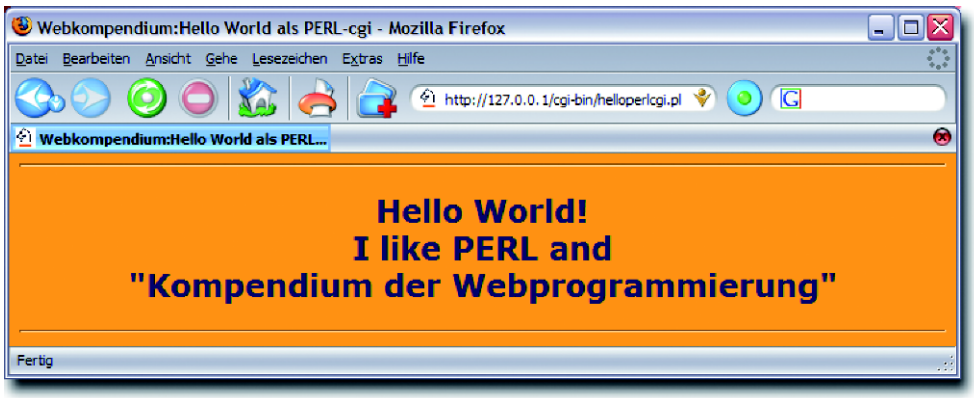



**Abbildung 10.17:** Ausführen von `helloperlcgi.pl` im Browser

### 10.4.3 Einfacher: Here-Documents

Das Beispiel aus dem Script `helloperlcgi.pl` zeigt, dass wir es häufig mit der Ausgabe von Texten zu tun haben. Dort haben wir diese jeweils zeilenweise in print-Methoden verpackt; hierfür bieten die meisten Scriptsprachen einen einfacheren Weg, die *Here-Documents*.

Ein Here-Document bettet „here“ ein auszugebendes Dokument in den Perl-Sourcecode ein. Es endet, wenn die angegebene terminierende Zeichenkette erstmalig vorkommt. Die Grundsyntax lautet:

```
print <<ende_ausgabe
...auszugebender Text
ende_ausgabe
```

Dabei ist `ende_ausgabe` der frei wählbare terminierende String.

Wie bei der Ausgabe von Zeichenketten kann `$variable` durch den Wert der Variablen ersetzt werden – das hängt von den den terminierenden String umgebenden Hochkommata ab:

- `"ende_ausgabe"`
  hier findet eine Ersetzung der Variablen durch den Wert statt;
- `'ende_ausgabe'`
  führt zu einer Ausgabe ohne Ersetzung;
- `ende_ausgabe`
  hat die gleiche Bedeutung wie doppelte Anführungszeichen, also mit Ersetzung.

Damit können wir das einfache CGI-Beispiel aus dem Script `helloperlcgi.pl` sinnvoller formulieren durch:

```
#!"c:/programme/Perl/bin/perl.exe" -w
# Kompendium der Web-Programmierung
#
# Beispiel fuer einfachstes Perl-CGI mit Here-Document

$phrase = "Hello World!<BR>I like Here-Documents " .  \
          " and<BR>\"Kompendium der Web-Programmierung\"";

print <<ende
Content-type: text/html

<HTML>
<HEAD><TITLE>WebKompendium: Hello World als Perl-cgi</TITLE>
<link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
<link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
</HEAD>
<BODY>
<HR><CENTER><H2>$phrase</H2></CENTER><HR>
</BODY>
</HTML>
ende
;
```

**Listing 10.11:** einfaches CGI in Perl mit Here-Document (Script `helloperlcgi2.pl`)

**Auch beim Here-Document muss die print-Anweisung mit einem Semikolon abgeschlossen werden (im Beispiel aus Listing 10.11 die letzte Zeile). Der terminierende String muss am Anfang einer Zeile stehen.**

### 10.4.4 Beispiel: Serverzeit

Mit dem vorgestellten können einfache CGI-Anwendungen in Perl implementiert werden; als weiteres Beispiel betrachten wir, um Dynamik zu erzielen, die Ausgabe der *Zeit des Servers*. Dazu stellt Perl zwei Funktionen zur Verfügung:

- Die Funktion `time` liefert – wie auf Unix üblich – die Anzahl der seit dem 1.1.1970, 0h, vergangenen Sekunden (der Zeitpunkt 0 von Unix, „epoch“).[1]
- `gmtime` hingegen ist eine Funktion im Listenkontext (vgl. 10.3.12.3), die eine Liste mit neun Elementen zurückgibt.

  Tabelle 10.8 gibt eine Übersicht über die Rückgabewerte von `gmtime`.

| Index | Bedeutung |
|---|---|
| 0 | Sekunden |
| 1 | Minuten |
| 2 | Stunden |
| 3 | Tag im Monat |
| 4 | Monat |
| 5 | Jahr |
| 6 | Tag der Woche |
| 7 | Tag im Jahr |
| 8 | isdst |

**Tabelle 10.8:** Rückgabewerte von `gmtime`

[1] am 19.1.2038 sind allerdings $2^{32}$ Sekunden vergangen.

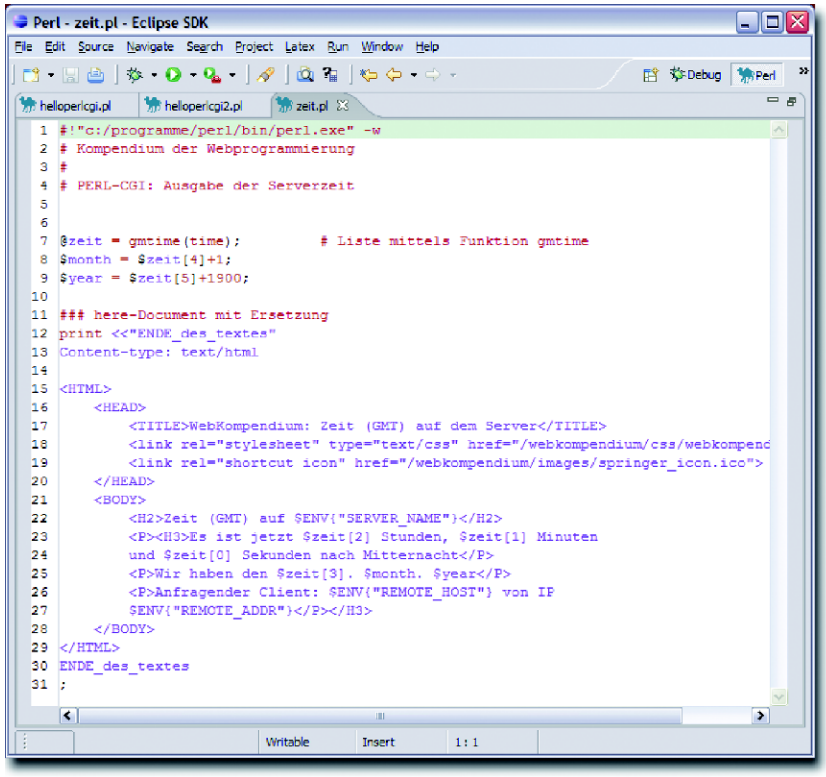



**Abbildung 10.18:**
Das Script `zeit.pl`: CGI für die Serverzeit

Ausführung des Scripts ergibt eine Ausgabe nach Abbildung 10.19.

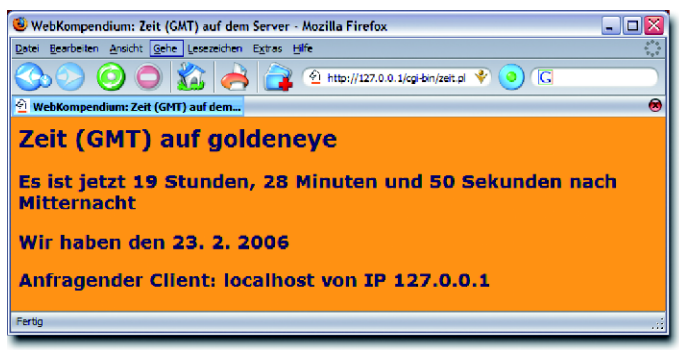



**Abbildung 10.19:**
Ausführen von `zeit.pl`

### 10.4.5 Beispiel: Seitenzähler

Eine „klassische“ CGI-Anwendung ist der *Seitenzähler* auf einer Web-Site. Hier müssen wir uns persistent merken, wieviele Zugriffe es auf die jeweilige Site schon gab, was wir mit Hilfe einer einfachen Textdatei (vgl. 10.3.13) realisieren können. Dazu definieren wir im Perl-Script ein geeignetes Verzeichnis, in welchem wir unsere Daten ablegen, und verwenden dort eine einfache Textdatei `zaehler.txt`, welche lediglich eine Zeile mit dem Ganzzahlwert „Anzahl der Zugriffe“ enthält. Das Script liest nun zuerst diese Datei, erhöht den Wert um eins und schreibt die Datei neu mit dem veränderten Wert.

Abbildung 10.20 zeigt ein einfaches Perl-Script, welches dies leistet.

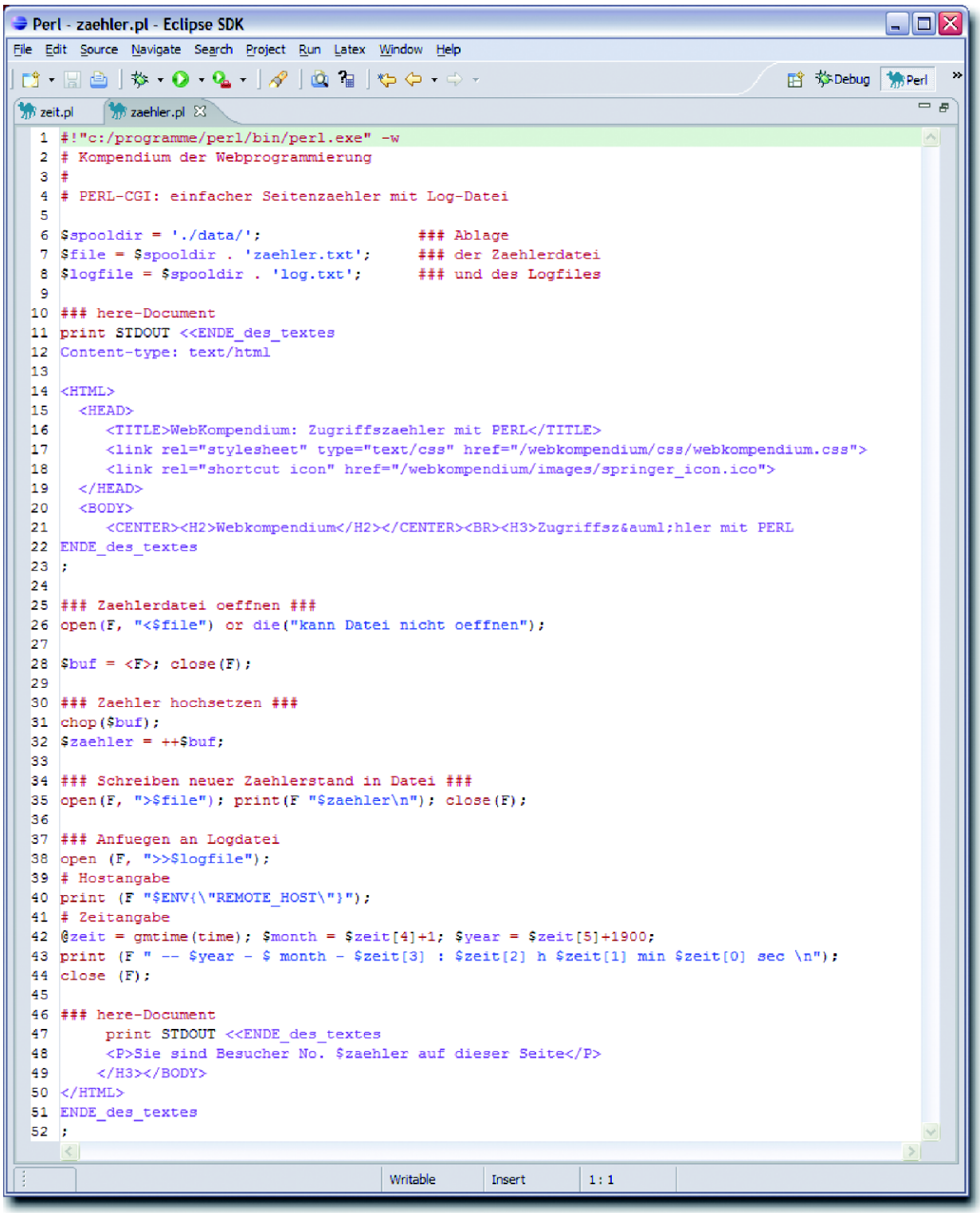



**Abbildung 10.20:** Einfacher Seitenzähler in Perl (Script `zaehler.pl`)

Der Abruf der kanonischen URL `http://<rechnername>/cgi-bin/zaehler.pl` liefert eine Ausgabe wie in Abbildung 10.21.

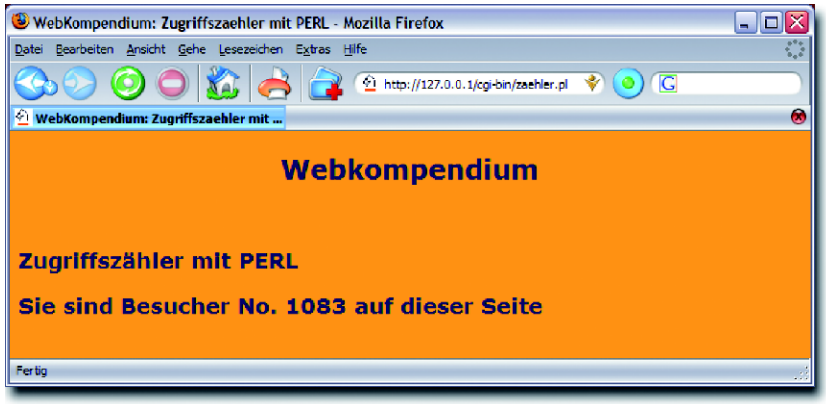



**Abbildung 10.21:**
Abrufen von `zaehler.pl`

Die Frage, warum es bereits derart viele Zugriffe auf diese eher wenig spannende Seite gibt, wird in Abschnitt 10.8 beantwortet.

Das Script `zaehler.pl` leistet aber etwas mehr als nur die Ausgabe und das Hochzählen des Seitenzählers: Es schreibt gleichzeitig ein Logfile: In dem gleichen Unterordner wie die Zählerdatei wird eine Datei `logfile.txt` geführt, welche zu jedem Zugriff die Uhrzeit und die Clientadresse enthält.

#### 10.4.5.1 Sicherheit: Zugriff auf die Dateien

Neben dem Schreiben des Logfiles stellt sich hier auch schön eine typische Frage zur Sicherheit: Ist es einem Client direkt möglich, auf die Zähler- oder Logdatei zuzugreifen? Zunächst kennt der Client weder den Namen noch den Speicherort dieser Dateien. Das ist gut, aber sichert keineswegs diese Dateien. Liegen die Dateien etwa im Ordner `htdocs` eines Standard-Apaches, werden sie natürlich vom Webserver auf einen HTTP-GET-Request an den Client ausgeliefert.

Durch Anpassen des Speicherortes der Dateien können wir die Sicherheit gewährleisten:

- sinnvollerweise wählt man ein Verzeichnis, welches außerhalb der vom Webserver ausgelieferten Dateibäume liegt; damit kann über Webclients nicht mehr direkt auf die Dateien zugegriffen werden;
- lässt man die Dateien im Ordner `cgi-bin` oder einem geeigneten Unterordner liegen, so sollte man die Konfiguration dieses Ordners innerhalb der `httpd.conf`-Datei beachten (vgl. 6.3): Diese Ordner sollten nur ausführbare CGI-Scripte, aber keine normalen Dokumente, die ausgeliefert werden, enthalten;
- umgekehrt aber braucht die Benutzerkennung, die den CGI-Prozess ausführt, lesenden und schreibenden Zugriff auf die zwei Dateien; mittels der Filetests in 10.3.13.3 haben wir dies bereits exemplarisch überprüft.

Mehr zu diesen Fragen wird in Kapitel 30 besprochen.

### 10.4.6 Testen einfacher CGI-Scripte

Ein (einfaches) CGI-Script muss nicht gleich auf dem Webserver betrieben werden – man kann es durchaus zunächst direkt auf der Kommandozeile ausführen und erhält dann den „errechneten“ HTML-Code. Dies ist ein leichter, aber sinnvoller Test, um Fehler schnell zu finden.

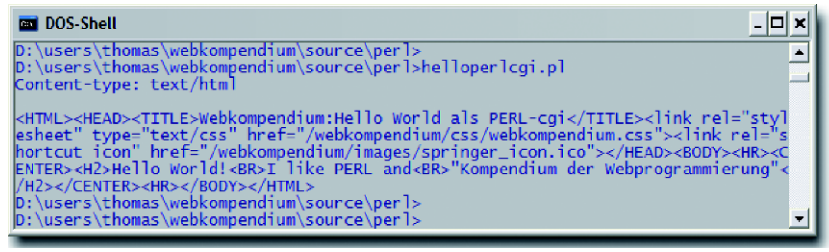



**Abbildung 10.22:** Ausführen von `helloperlcgi.pl` auf der Kommandozeile

#### 10.4.6.1 Typische Fehler

Einige Fehler treten, gerade bei der Einarbeitungsphase, häufig auf:

- Eine falsche Shebang-Zeile: Durch eine falsche Angabe der Shebang-Zeile wird das Script unausführbar; man kann dies einfach ausschließen, indem man das Script ohne expliziten Aufruf des Perl-Interpreters auf der Kommandozeile ausführt.
- Falsche Rechte: Die Perl-Datei muss eine lesbare und ausführbare Datei sein, und zwar für diejenige Benutzerkennung, die den Apache Webserver betreibt.

  Unter Windows sind diese Fehlerquellen entschärft, da hier meistens kein explizites Rechtemanagement verwendet wird.

#### 10.4.6.2 Ausführen einfacher CGI-Scripte mit Eclipse

Über das EPIC-Plugin für Eclipse (vgl. 10.3.4) können nicht nur Kommandozeilenscripte, sondern auch CGI-Programme ausgeführt und getestet werden.

## 10.5 Perl erweitern: Module und mehr

Die Programmiersprache Perl ist, wie wir schon gesehen haben, sehr flexibel und für viele Probleme erfolgreich und schnell einsetzbar. Aus diesem Grund gibt es für Perl eine sehr große Zahl an Erweiterungen: *Perl Module.*

Zunächst ist ein Perl-Modul eine Perl-Datei mit der vorgegebenen Endung `.pm`, in der ein einzelnes Paket enthalten ist.

Ein Perl-Modul endet standardmäßig mit der Anweisung

```
1;
```

welche den Rückgabewert `1` festlegt, der besagt, dass das jeweilige Modul korrekt eingebunden werden konnte (durch die Import-Anweisung `use`, siehe 10.5.3).

### 10.5.1 Namensräume in Perl: package

Ein Namensraum in Perl ist ein *package.* Der Zugriff auf einen Namensraum in Perl ist (seit der Version 5.0) über doppelte Doppelpunkte möglich, etwa:

```
packagename::methodenname()
```

In 10.3.14.2 wurde die Perl-typische Problematik der globalen Variablen bereits angesprochen: Alle Variablen, die nicht über `my` oder `local` eingeschränkt sind, gelten im gesamten package ohne alle Übergabemechanismen.

Perl verfügt auch über ein Standardpaket, in dem wir bisher ausschließlich gearbeitet haben: `main`.

Nützlich kann die Variable

```
__PACKAGE__
```

sein, da sie den Namen des aktuellen Paketes speichert.

#### 10.5.1.1 Deklaration von Paketen in Perl

In Perl werden Pakete über das Schlüsselwort `package` deklariert, etwa

```
package WebKompendium;
```

Eine Perl-Datei kann durchaus mehrere Pakete enthalten. Genauer wird hier in den jeweiligen Namensraum gewechselt, weshalb auch mehrfach innerhalb einer Perl-Datei in das gleiche Paket gewechselt werden kann.

### 10.5.2 Wo findet Perl Perl-Dateien?

Ein Perl-Script kann – wie wir gleich in 10.5.3 sehen werden – natürlich auch andere Perl-Dateien einbinden. Der Pfad, den Perl zum Einbinden durchsucht, wird intern in der List `@INC` abgelegt; Listing 10.12 zeigt das Auslesen dieser Variablen.

```
# Kompendium der Web-Programmierung
#
# Ausgabe von @INC
#
# kuerzer:  print $_ \n" foreach @INC;

foreach $i (@INC) {
print "$i\n";
}
```

**Listing 10.12:** Ausgabe von `@INC`

Bei geschachtelten Paketen der Art

```
paket1::paket2::paket3
```

sucht Perl nach einer Verzeichnishierarchie `paket1/paket2/paket3`.

Innerhalb einer Perl-Anweisung kann aber `@INC` nicht verändert werden; zusätzliche Pfade werden deshalb mittels der Anweisung

```
use lib "pfad_zum_modul"
```

aufgenommen.

**Die Liste `@INC` hat somit eine ähnliche Bedeutung wie der Klassenpfad in Java.**

### 10.5.3 Einbinden von Perl-Code: do, require und use

Zum Importieren von Paketen und Modulen stehen in Perl drei Anweisungen zur Verfügung: `do`, `require` und `use`.

#### 10.5.3.1 do

Mit der Anweisung

```
do("dateiname.pl");
```

wird die Perl-Datei `dateiname.pl` direkt an dieser Stelle ausgeführt. Typischerweise können so Unterprogramme ausgelagert und eingebunden werden.

#### 10.5.3.2 require

`require` ist analog zu `do`.

#### 10.5.3.3 use

Die Perl-Anweisung `use` entspricht den zwei anderen, allerdings werden hier nicht Pakete oder einzelne Scripte, sondern Perl-Module eingebunden.

Bei `use` wird anstelle des Namens einer Perl-Datei mit Endung nur der Name des Moduls als Parameter angegeben, ohne die für Module zwingende Endung `.pm`.

Eine weitere Besonderheit ist die Verbindung von `use` mit einem numerischen Wert: Dieser wird als Versionsnummer gelesen und es wird überprüft, ob diese Version mindestens vorliegt.

### 10.5.4 CPAN: Comprehensive Perl Archive Network

Das *Comprehensive Perl Archive Network* [web] ist eine nahezu grenzenlose Quelle für Perl-Module. Für jede denkbare Aufgabe sind hier gute, erprobte Lösungen zu finden.

**Die effiziente Nutzung von CPAN erleichtert die Arbeit mit Perl drastisch.**

### 10.5.5 Installation von Perl-Modulen

Das Installieren von Perl-Modulen kann mitunter aufwändig sein; es steht ein manueller, ein automatisierter und ein sehr bequemer Weg über das Werkzeug PPM (siehe 10.5.6) hierfür bereit.

#### 10.5.5.1 Manuelle Installation

Die manuelle Installation entspricht dem „Unix-typischen“ Weg der Softwareinstallation: Nach dem Herunterladen und Entpacken wird mittels Perl selbst ein Makefile durch

```
perl Makefile.pl
```

erzeugt, mittels des üblichen

```
make
```

werden die Quelldateien dann übersetzt und mit

```
make install
```

installiert; hierfür sind möglicherweise besondere Systemprivilegien notwendig. Dieses Grundverfahren kann je nach Modul etwas variieren.

Der Weg ist prinzipiell auch auf Windows möglich, allerdings sind in einer Standardinstallation die notwendigen Werkzeuge (C-Compiler, make) nicht enthalten.

#### 10.5.5.2 Automatische Installation

Einfacher ist die Installation unter Nutzung des Moduls CPAN. Mittels

```
perl -MCPAN -e shell
```

wird dieses gestartet, anschließend sind einige wenige manuelle Konfigurationen über die Konsole notwendig. Dieses Modul verwendet zahlreiche Unix-typische Werkzeuge bis hin zum Lynx-Browser.

### 10.5.6 ppm: Perl Package Manager

Mit dem *Perl Package Manager* (ppm) steht ein einfaches und benutzerfreundliches Werkzeug bereit, um Perl-Module zum jeweiligen System zu ergänzen. ppm ist Bestandteil der ActiveState-Distribution von Perl [web] und insbesondere unter Windows der bequemste Weg, Module zu ergänzen.

ppm wird mit der Anweisung

```
ppm-shell
```

auf der Konsole gestartet (oder einfach über `ppm`, teilweise wird auch für die neueste Version `ppm3` verwendet, da die aktuelle Versionsnummer mit „3“ beginnt).

### 10.5.7 Module selbst schreiben

Natürlich ist es sinnvoll, die eigene Software in Modulen übersichtlich zu implementieren.

Nachfolgendes Modul erlaubt etwa eine einfache Versionsüberprüfung mittels

```
Perl -MWebKompendium -le "print(WebKompendium->VERSION)"
```

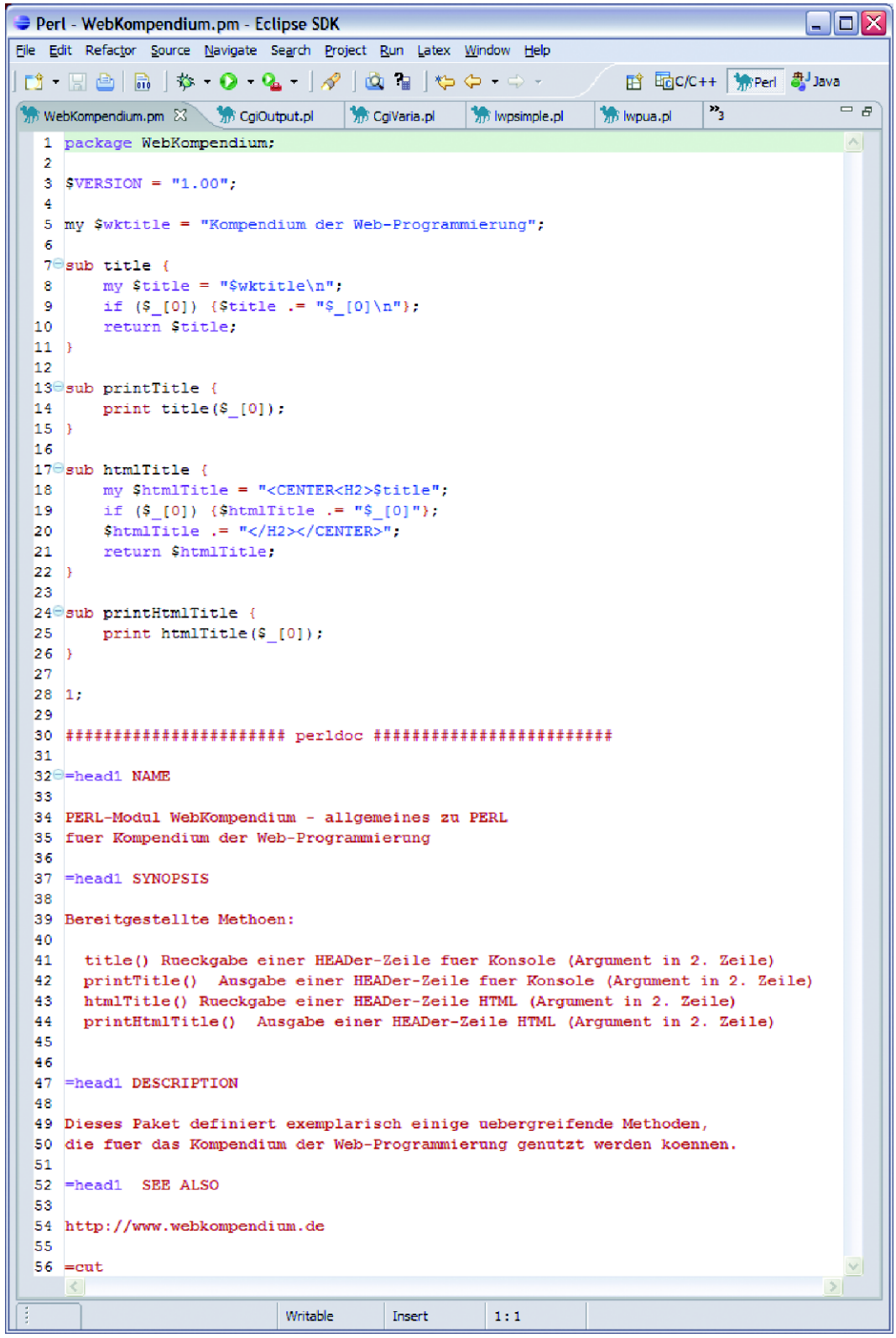



**Abbildung 10.23:**
Ein eigenes Modul:
(`WebKompendium.pm`

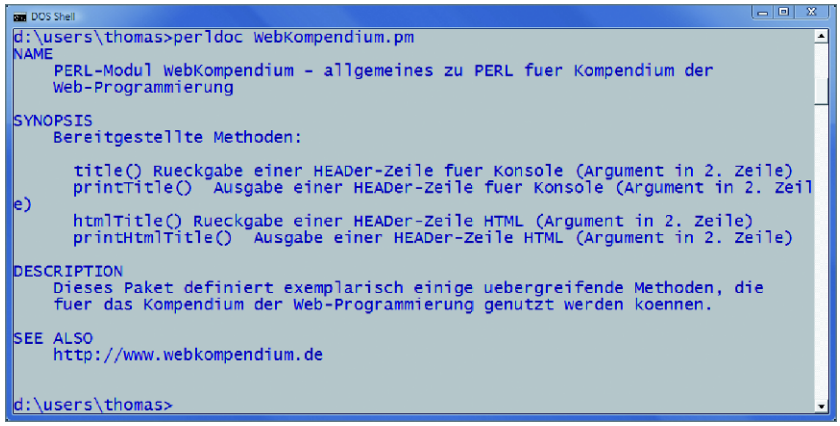



**Abbildung 10.24:**
perldoc von `WebKompendium.pm`

## 10.6 Das Perl-Modul CGI

Eines der wichtigsten Perl-Module ist das von Lincoln Stein geschriebene Modul *CGI*. Dieses Modul erleichtert die CGI-Programmierung mit Perl ungemein. Seine Funktionalitäten lassen sich in drei Hauptbereiche untergliedern:

1. Übernahme von Formulardaten;
2. Erstellung von HTML-Code;
3. weitere Hilfen für CGI.

Der erste Bereich ist für den Einsatz von Perl als CGI-Sprache praktisch unerlässlich (rein theoretisch kann zwar über die CGI-Umgebungsvariablen das gleiche „zu Fuß“ erreicht werden, das ist aber in der Praxis kaum möglich). Der zweite Bereich kann sehr nützlich sein, hier liegt es aber im Ermessen des Entwicklers, ob die Perl-CGI-Funktionen oder direkt (X)HTML genutzt werden. Der dritte Bereich bündelt zahlreiche nützliche Funktionalitäten.

### 10.6.1 Allgemeines zu CGI.pm

Seit der Perl-Version 5.004 ist CGI.pm Bestandteil jeder Perl-Distribution, es liegt also praktisch auf allen Systemen bereit; allerdings muss es vor seiner Verwendung mittels `use CGI;` importiert werden. Mittels des Perl Packet Managers ppm (siehe 10.5.6) oder eines vergleichbaren Tools sollte überprüft werden, ob eine aktuelle Version installiert ist. Im Web [web] sind zahlreiche wichtige Ressourcen zu Perl zu finden, im Anhang A sind besonders nützliche aufgeführt.

Das CGI-Modul kann in einem prozeduralen und im objektorientierten Kontext verwendet werden, allerdings ist heute der prozedurale Einsatz von CGI.pm nicht mehr von Bedeutung.

**Abbildung 10.25:**
Das CGI-Modul bei CPAN

### 10.6.2 Übernahme von Formularfeldern mit CGI.pm

Der Grundmechanismus der Übergabe von Formularfeldern an CGI-Scripte über Umgebungsvariablen ist in Kapitel 9 ausführlich dargestellt. Der Weg des direkten Zugriffs auf die CGI-Umgebungsvariablen steht prinzipiell auch für Perl offen, ist aber keine praktische Umsetzung; hier hilft das CGI-Modul wesentlich.

Wir betrachten das prinzipielle HTML-Formular

```
<FORM action="/cgi-bin/script.pl" method="POST" >
     <INPUT name="feld">
     ...
</FORM>
```

Hier wird ein Eingabefeld mit dem Bezeichner „feld“ mittels der HTTP-Methode POST an das Script `script.pl` geschickt. Genauer betrachtet ist die Situation komplexer: Der Webclient baut eine Verbindung zum Webserver auf, fordert das Script `script.pl` an und überträgt über einen separaten IO-Kanal ein Key-Value-Paar mit dem Key „feld“; der zugehörige Value ist der eingetragene Wert des Eingabefeldes.

Der Webserver legt das Key-Value-Paar in der Umgebungsvariablen `$QUERY_STRING` ab; über diese könnte man umständlich direkt auf die Eingabe zugreifen, es geht aber mittels des CGI-Moduls sehr viel einfacher:

#### 10.6.2.1 Zugriff auf ein einzelnes Formularfeld

Perl-CGI stellt eine einfache Methode bereit, um den Value zu einem Key abzufragen, also den Inhalt des *Formularfeldes;* dabei ist es unerheblich, ob das Formular mittels der HTTP-Methode GET oder POST übertragen wurde.

Diese Funktionalität stellt die Perl-CGI-Methode `param` bereit; hier in dem Beispiel etwa mit der Syntax

```
use CGI;
$cgi = new CGI;
$value = $cgi->param("feld");
```

Zunächst wird – da ja das Modul objektorientiert verwendet wird – durch `new` eine Instanz der Klasse CGI erzeugt, welche über `$cgi` adressiert wird. Anschließend wird auf dieses Objekt die Methode `param` angewendet; Argument der Methode ist der Name des Feldes (Key), dessen Eingabewert (Value) abgefragt werden soll.

Dies ist eine einfache und praktische Methode, um Formularfelder auszulesen, allerdings ist sie für komplexere Formulare zu umständlich; deshalb bietet Perl-CGI eine weitere Funktionalität.

#### 10.6.2.2 Zugriff auf mehrere Formularfelder

Für die Verarbeitung komplexerer Formulare bietet das CGI-Modul die Methode `Vars`; diese liefert eine Referenz auf einen Perl-Hash, welcher in seinen Key-Value-Paaren die gesamten Formularfelder verwaltet. Durch diese Funktionalität kann bequem mit großen Formularen gearbeitet werden, da das ganze Formular über diese Referenz direkt zugänglich ist. In dem exemplarischen Beispiel wäre es:

```
use CGI;
$cgi = new CGI;
$formular = $cgi->Vars();
$value = $formular->{"feld"};
```

Nach Erzeugung einer Instanz der Klasse CGI wird mittels der Methode `Vars` eine Referenz auf einen Perl-Hash mit den Key-Value-Paaren des Formulars erzeugt. Um den Value zum Key `feld` zu erhalten, muss dieses Hash zum entsprechenden Key dereferenziert werden; dies kann in der angegebenen Syntax oder durch

```
$value = $$formular{"feld"};
```

geschehen, vgl. 10.3.5.3 und 10.3.14.4.

#### 10.6.2.3 Beispiel: ein Ratespiel im Web

Die Formularverarbeitung mit Perl soll an einem Beispiel verdeutlicht werden: Eine beliebte Programmierübung ist das „Zahlenraten“, bei welchem eine Zufallszahl zwischen 1 und 100 zu erraten ist. Das Programm sagt dabei jeweils,

ob der gerade geratene Wert zu klein, zu groß oder richtig ist und zählt die Anzahl der Versuche.[2]

Dieses Spiel soll im Web mit Perl implementiert werden. Dazu ist ein einzelnes Script `rate.pl` ausreichend (Abbildungen 10.26 und 10.27): Das Perl-Script überprüft zunächst, ob schon eine Zufallszahl vorhanden ist, ansonsten wird eine neue Zufallszahl errechnet. Ist ein Ratewert vorhanden, wird angegeben, ob dieser zu klein/zu groß oder korrekt ist. Danach wird ein Formular aufgebaut, welches den Ratewert abfragt und die Anzahl der Versuche als nicht-editierbares Feld ausgibt; ein zweites Formular startet ein neues Spiel.

Das Besondere bei der Implementierung im Web ist das „fehlende Gedächtnis“ des CGI-Prozesses: Da dieser mit jedem Rateversuch neu gestartet wird, kennt er zunächst die zu erratende Zahl nicht. Hier wird deshalb diese Zahl als versteckter Parameter im HTML-Formular, als hidden field, weitergereicht. Eine noch elegantere Lösung wäre die Speicherung der Zufallszahl in einer Session, vgl. Kapitel 26. Außerdem nutzt das Spiel noch eine praktische JavaScript-Funktion, welche in Abschnitt 15.9.6 genauer beschrieben ist.

[2] mittels der binären Suche sind bei 100 Zahlen höchstens 10 Versuche notwendig.

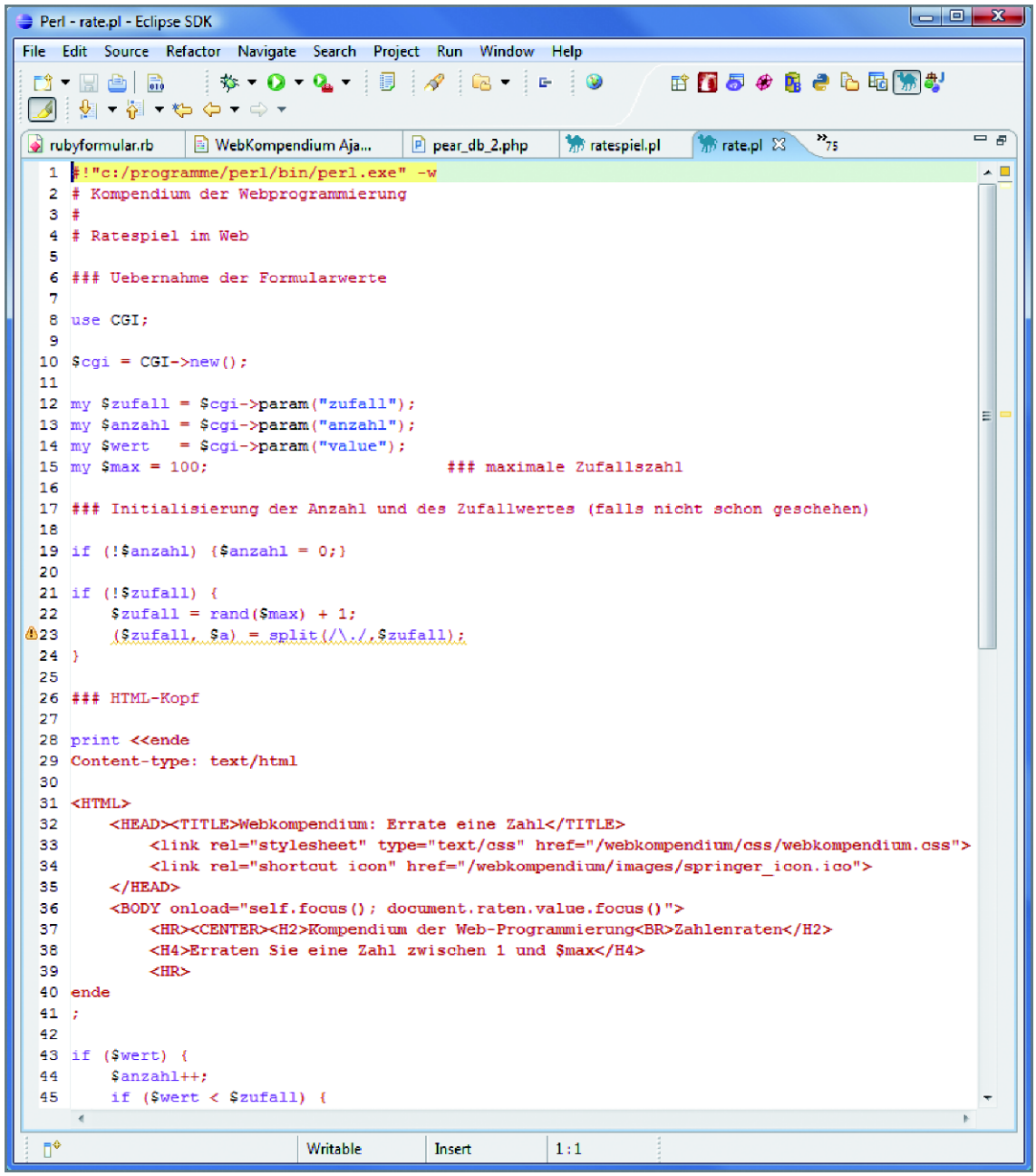
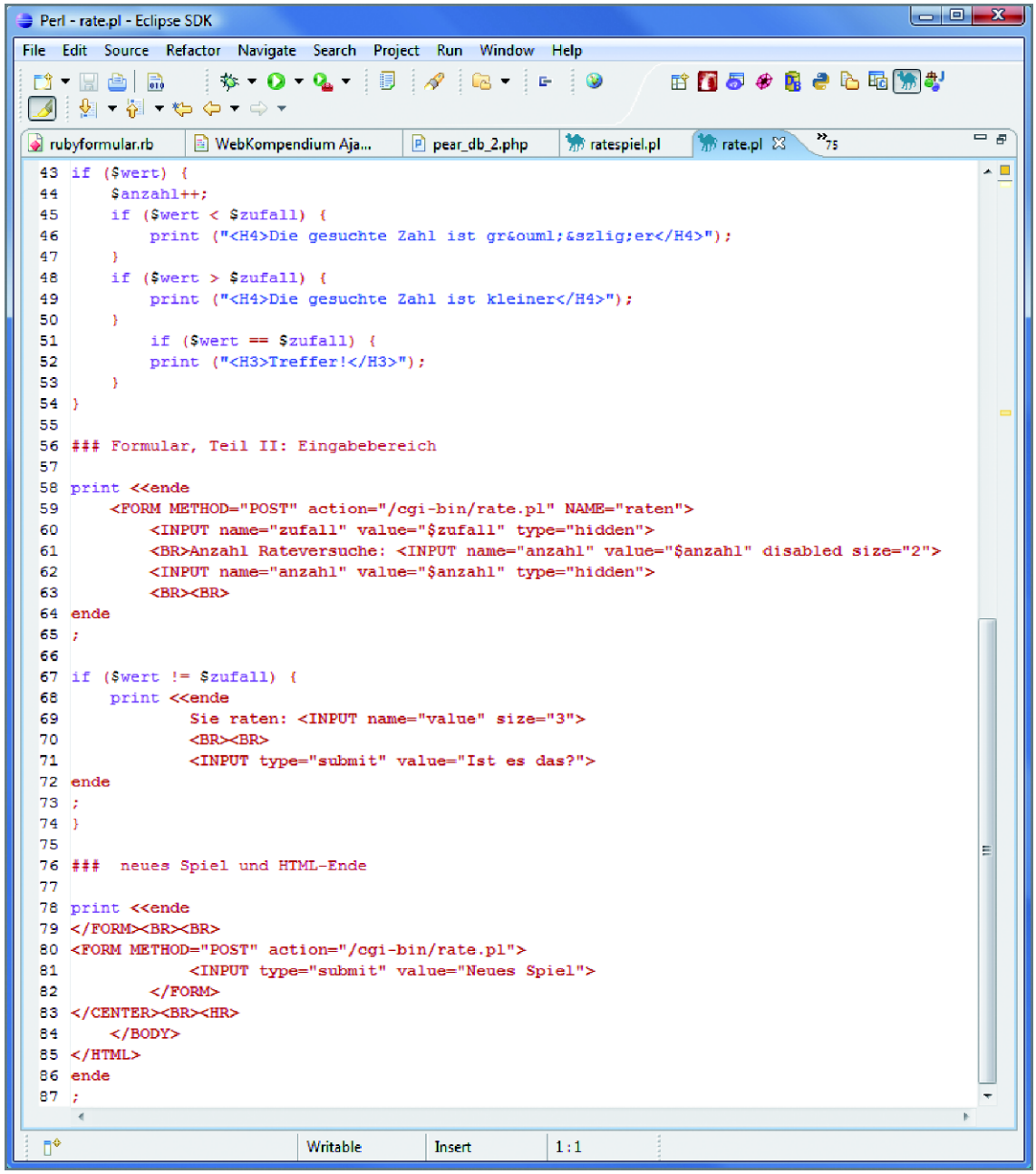



**Abbildung 10.26:**
Script `rate.pl`, Teil I




```
43 if ($wert) {
44     $anzahl++;
45     if ($wert < $zufall) {
46         print ("<H4>Die gesuchte Zahl ist gr&ouml;&szlig;er</H4>");
47     }
48     if ($wert > $zufall) {
49         print ("<H4>Die gesuchte Zahl ist kleiner</H4>");
50     }
51         if ($wert == $zufall) {
52         print ("<H3>Treffer!</H3>");
53     }
54 }
55
56 ### Formular, Teil II: Eingabebereich
57
58 print <<ende
59     <FORM METHOD="POST" action="/cgi-bin/rate.pl" NAME="raten">
60         <INPUT name="zufall" value="$zufall" type="hidden">
61         <BR>Anzahl Rateversuche: <INPUT name="anzahl" value="$anzahl" disabled size="2">
62         <INPUT name="anzahl" value="$anzahl" type="hidden">
63         <BR><BR>
64 ende
65 ;
66
67 if ($wert != $zufall) {
68     print <<ende
69             Sie raten: <INPUT name="value" size="3">
70             <BR><BR>
71             <INPUT type="submit" value="Ist es das?">
72 ende
73 ;
74 }
75
76 ###  neues Spiel und HTML-Ende
77
78 print <<ende
79 </FORM><BR><BR>
80 <FORM METHOD="POST" action="/cgi-bin/rate.pl">
81             <INPUT type="submit" value="Neues Spiel">
82         </FORM>
83 </CENTER><BR><HR>
84     </BODY>
85 </HTML>
86 ende
87 ;
```

**Abbildung 10.27:**
Script `rate.pl`, Teil II

**Abbildung 10.28:**
Ausführen von `rate.pl`

### 10.6.3 Erstellung von HTML-Code mit CGI.pm

Viele Aufgaben in der HTML-Codierung sind wiederholend und eher „lästig“; hierfür hat Lincoln Stein nützliche Funktionalität in CGI.pm integriert.

Das CGI-Modul schreibt damit HTML-Code, standardmäßig wird seit der Version 2.69 sogar direkt XHTML (siehe 2.4.7) erzeugt, es kann aber auf normales HTML umgeschaltet werden. Das Vorgehen zur Erzeugung von HTML-Code mittels CGI.pm ist vergleichbar zur Parameterübernahme: Zunächst muss ein Objekt der Klasse CGI erzeugt werden, dann können die entsprechenden Methoden auf dieses Objekt angewendet werden.

Einige Beispiele für diese Methoden sind:

- `header`: erzeugt einen HTTP-Header für die Ausgabe;
  typischer Parameter ist der Content-type, der mit dem Key `-type` übergeben wird, sowie der HTTP-Responsecode (`-status`).
- `start_html`: Anfang des HTML-Dokuments;
  hier sind zahlreiche Parameter möglich, etwa der HTML-Title des Dokuments sowie die globale CSS-Datei;
- `h1`: Beispiel für eine typische Methode; diese gibt das Argument in `H1`-Tags aus; ein möglicher Parameter ist `-align` für die Zentrierung;
- `startform`: Funktion, um ein Formular zu beginnen; hat sehr zahlreiche Parameter;
- `endform`: Ende eines Formulars;
- `end_html`: Ende des HTML-Dokuments.

Ein Beispiel für ein derartiges Script ist in Abbildung 10.29) wiedergegeben; die damit erzeugte Ausgabe im Quelltext – echtes XHTML – zeigt Abbildung 10.30.

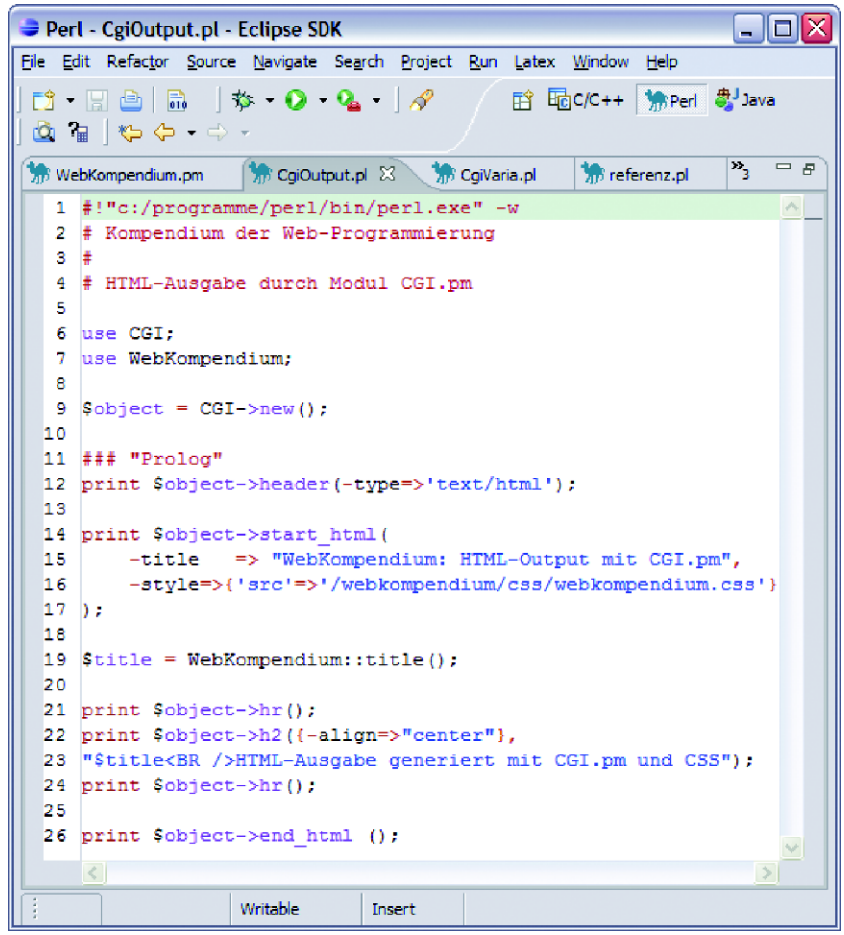



**Abbildung 10.29:** Script für (X)HTML-Ausgabe mittels CGI.pm (`CgiOutput.pl`

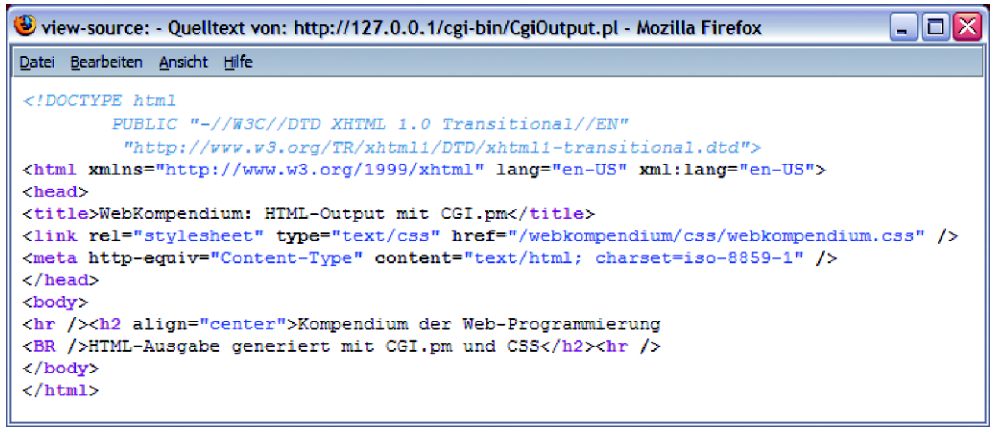



**Abbildung 10.30:** XHTML-Ausgabe von Script `CgiOutput.pl`

Um mit CGI.pm, reinen HTML-Code zu erzeugen, muss beim Import die entsprechende Option gewählt sein:

```
use CGI qw(-no_xhtml);
```

### 10.6.4 Weitere CGI-Hilfen von CGI.pm

Neben diesen zwei Kernbereichen von Funktionalitäten – der Übernahme der Formularwerte und der Erzeugung von HTML-Code – bietet CGI.pm weitere nützliche Methoden, insbesondere den einfachen Zugriff auf die CGI-Umgebungsvariablen; damit ist es beispielsweise möglich, die Übertragungsmethode zu überprüfen.

Exemplarische Beispiele hierfür sind:

- `request_method`: die HTTP-Übertragungsmethode (GET oder POST).
- `server_software`: Informationen zum Webserver.
- `server_name`: Name des Webservers.

Abbildung 10.31 zeigt ein Beispiel.

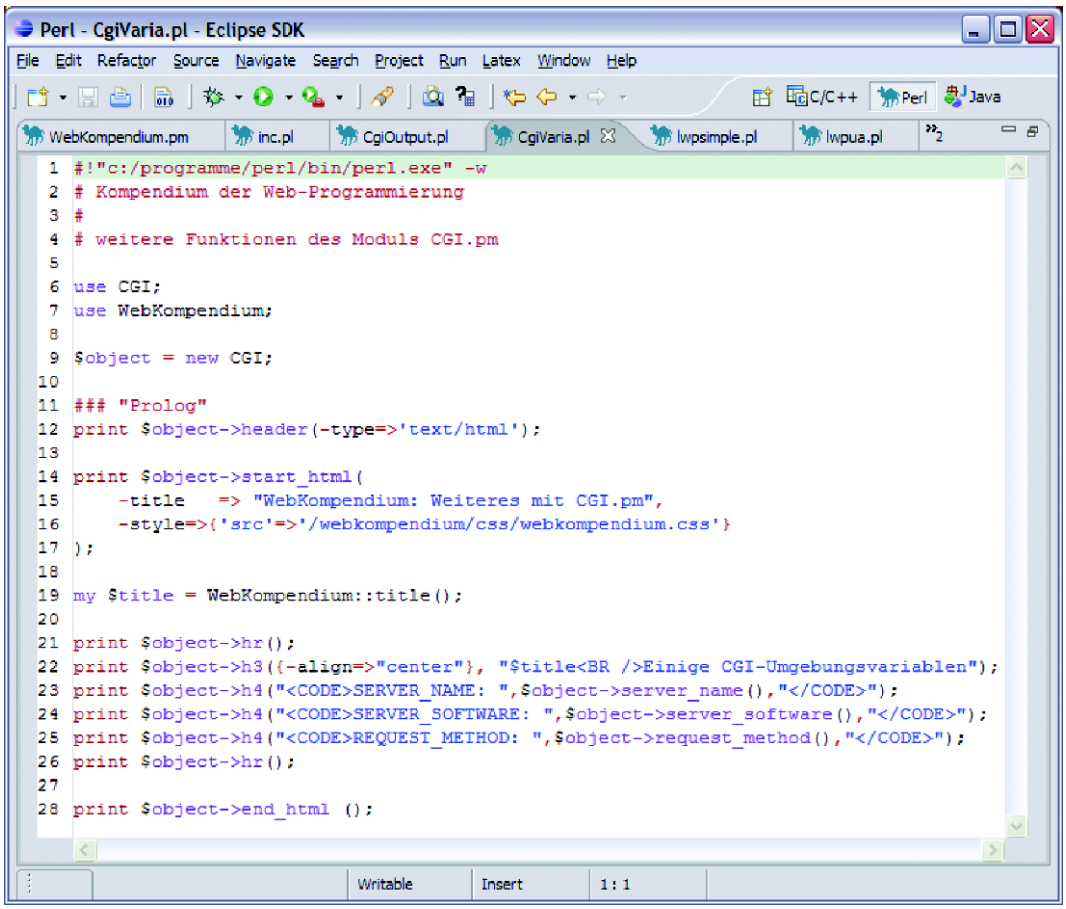



**Abbildung 10.31:**
Script `CgiVaria.pl`

## 10.7 Das Perl-Modul DBI

Die starke Scriptsprache Perl kann – natürlich – auch gut zusammen mit Datenbanken eingesetzt werden. Hier kommt ein besonderes Perl-Modul zum tragen: DBI; das *Database Independent Interface for Perl*. Das Konzept von DBI ist sehr überzeugend, weshalb es inzwischen zunehmend von anderen Sprachen übernommen wird; so gehen die neueren Entwicklungen für PHP mit PEAR und PECL in eine sehr ähnliche Richtung; auch ist der Datenbankzugriff mit Ruby bis hin zur Syntax sehr ähnlich zu DBI, aber momentan noch nicht so

weit entwickelt/implementiert. Gedruckte Literatur gibt es nicht sehr viel – das wichtigste ist [DB00] –, dafür ist im Web [web] zahlreiches zu dem Thema zu finden (vgl. Anhang A).

Die moderne Scriptsprache Ruby hat das Konzept des DBI-Moduls bis hin zur Syntax übernommen (vgl. 13.6.2).

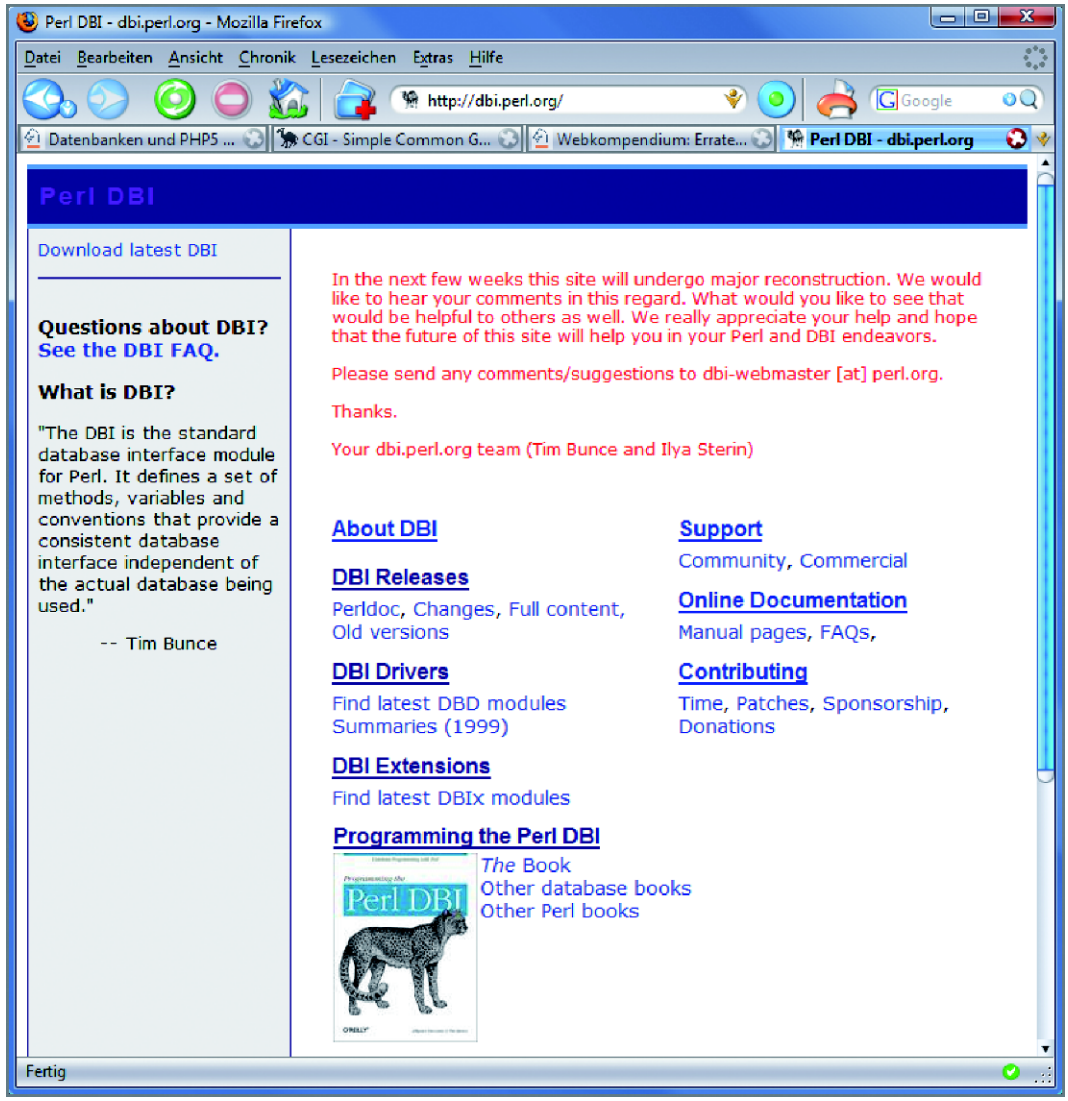



**Abbildung 10.32:** DBI-Site

### 10.7.1 Das Prinzip von DBI

Die Idee von DBI ist naheliegend: Während frühere Datenbank-Anbindungen in Perl genauso wie in PHP mit einer proprietären Syntax für jedes DBMS arbeiten, stellt DBI eine vom jeweiligen DBMS unabhängige Abstraktionsschicht bereit, so dass die Perl-Anwendung nur noch diese unabhängige Schicht verwendet. Die Übersetzung auf die DBMS-spezifischen Calls leistet DBI im Hintergrund, so dass der Entwickler unabhängig davon ist – ein sehr großer Vorteil.

Das bedeutet, dass zunächst das DBI-Modul geladen werden muss: `use DBI;`. Anschließend wird im wesentlichen eine (oder mehrere) Verbindung zu einer

Datenbank auf einem Datenbankmanagementsystem hergestellt. Anhand der Syntax dieses Verbindungsaufbaus erkennt DBI, welche speziellen Treiber benötigt werden – und lädt diese automatisch dazu. Diese DBMS-spezifischen Treiber werden DBD genannt: Database Driver. Ein sogenannter Database Handle, häufig `$dbh` bezeichnet, verwaltet diese Verbindung zu einer Datenbank.

Nun können über diese Verbindung mehrere Statements abgeschickt werden; dabei sind zwei Gruppen zu unterscheiden:

- SELECT-Abfragen: Diese haben als Ergebnis eine komplexere Struktur in Form eines Resultsets (vgl. 3.3.4);
- andere SQL-Anweisungen wie `INSERT`, `DELETE` oder `UPDATE`: hier erhalten wir eine einfache Rückgabe über den Erfolg der Operation.

DBI unterscheidet diese zwei Bereiche deutlich. Es verwaltet jedes SQL-Statement über einen Statement Handle, häufig mit `$sth` bezeichnet.

Dabei kann eine Perl-Anwendung mehrere Database Handles verwenden, und zu jedem Database Handle können mehrere Statement Handles gehören.

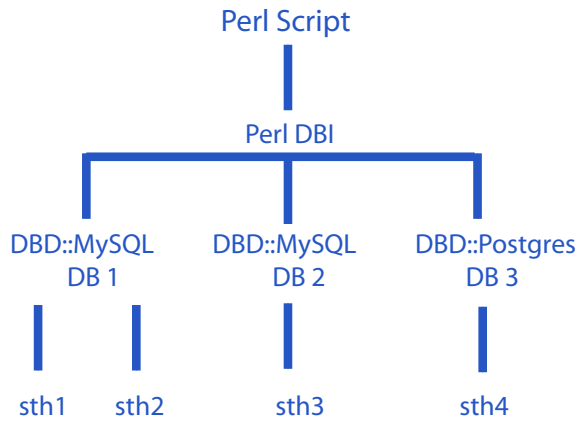



**Abbildung 10.33:**
Prinzip von DBI: Perl-Anwendung mit drei Database-Handles, zwei Database Drivers und vier Statement Handles

### 10.7.2 Schichtmodell

DBI implementiert mit diesem Prinzip ein Schichtmodell; neben dem DBMS als Datenhaltungsschicht gibt es die eigentliche DBI-Schicht, die letztlich das DBMS kapselt, so dass die Perl-Entwicklung unabhängig vom jeweiligen DBMS ist; bei einem Austausch des DBMS sind dann nur noch die Parameter des Verbindungsaufbaus zum DBMS anzupassen.

**Mit DBI entstehen Perl-Anwendungen, die unabhängig vom verwendeten DBMS sind; Voraussetzung hierfür ist aber der Einsatz einer sauberen, vom DBMS unabhängigen SQL-Syntax.**

In der Praxis kommt es häufig vor, dass geschickte proprietäre SQL-Syntax passend für das jeweilige DBMS verwendet wird; das klassische Beispiel hierfür ist `AUTO_INCREMENT` von MySQL; diese praktische Anweisung ist leider kein SQL-Standard. Verwendet man derartige Spezialitäten, kann später trotz DBI das DBMS nicht einfach ausgetauscht werden.

Neben DBI verfolgen etliche andere Konzepte ein ähnliches Ziel; am bekanntesten sind aus der C- und Windows-Welt ODBC und für Java JDBC.

### 10.7.3 Installation

DBI ist im Gegensatz zum CGI-Modul nicht Bestandteil der Standard-Distribution von Perl, es muss – etwa mit dem Paketmanager ppm (vgl. 10.5.6) – zusätzlich installiert werden.

Allerdings ist die Situation im Falle von DBI komplexer: Mit der Installation von DBI steht nur die beschriebene Abstraktionsschicht bereit, aber noch keine Datenbank-Treiber. Die einzelnen DBDs müssen nachträglich installiert werden.

DBI stellt eine nützliche Anweisung bereit, die eine Übersicht über die vorhandenen DBDs gibt: `available_drivers()`. Script `dbi_0.pl` (Abbildung 10.34) zeigt den typischen Einsatz dieser Methode.

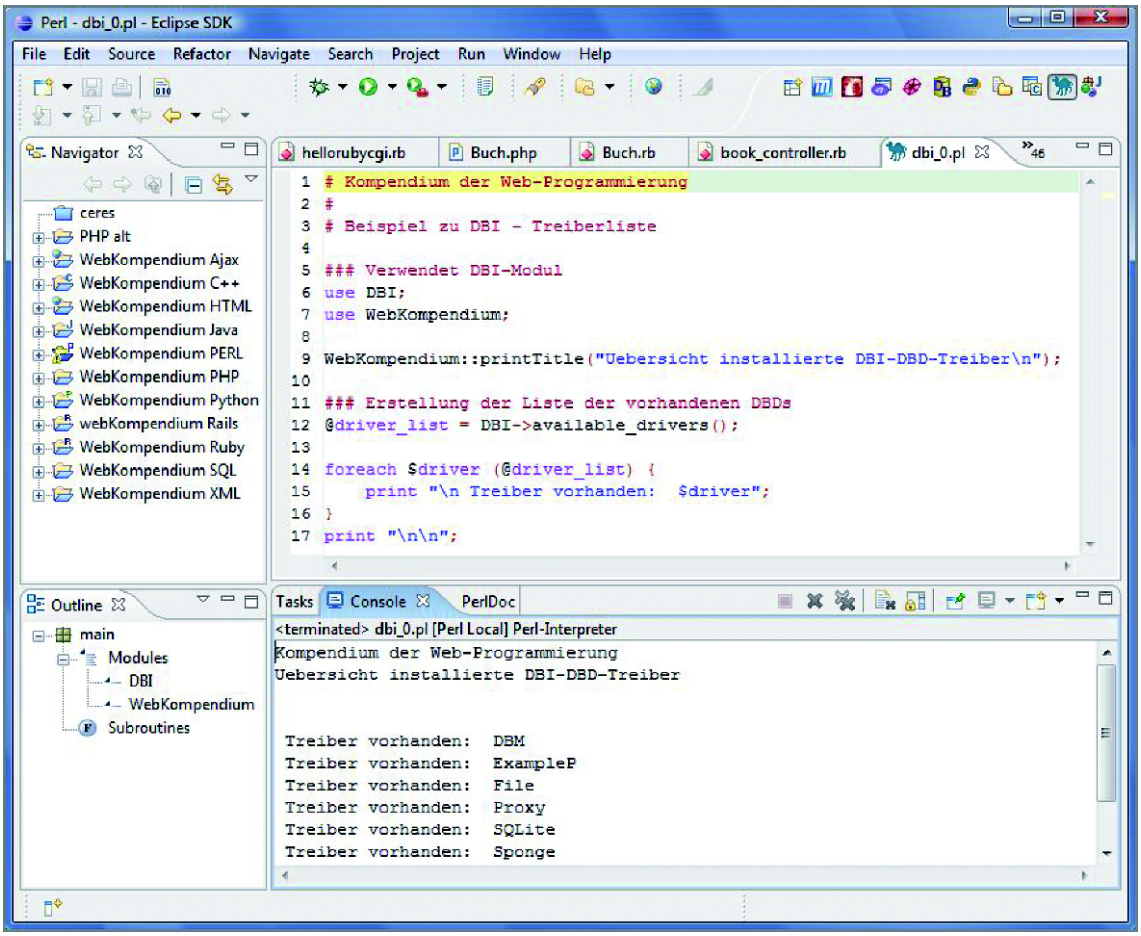



**Abbildung 10.34:** Script `dbi_0.pl`: Abfrage der installierten DBDs

Fehlende DBDs können am einfachsten über ppm ergänzt werden; dabei kann insbesondere die Suchfunktion vom ppm nützlich sein.

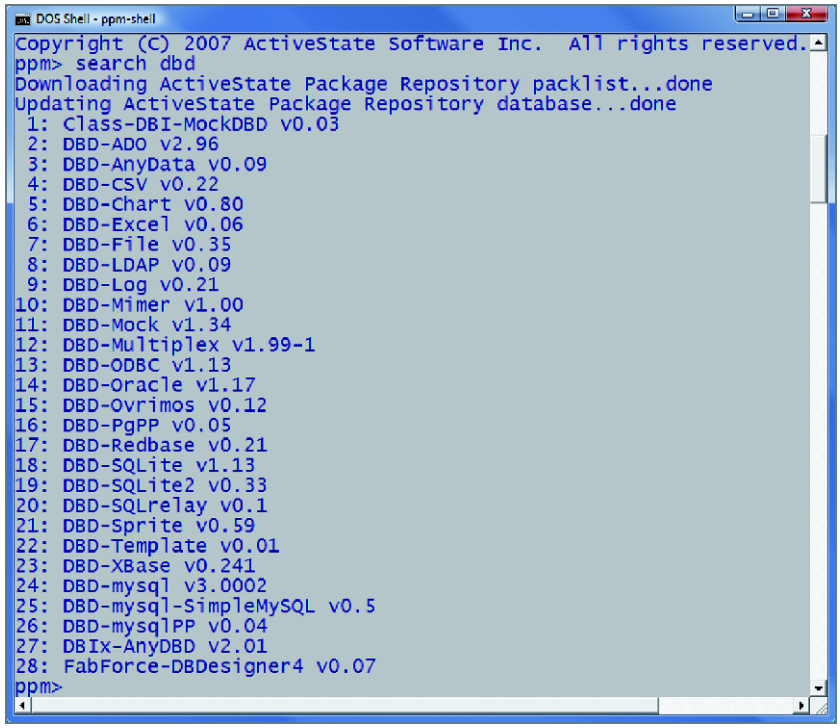



**Abbildung 10.35:**
ppm-Suche nach DBD-Treibern

### 10.7.4 Verbindungsaufbau

Der zentrale Schritt ist der Verbindungsaufbau zum jeweiligen DBMS und dort zu einer Datenbank. Eine derartige Verbindung wird in einem Database Handle verwaltet und benötigt folgende Parameter:

- Name des DBD – hiermit wird festgelegt, welchen Datenbanktreiber DBI automatisch hinzu lädt;
- Name (oder ip-Adresse) des Hosts, der das DBMS betreibt;
- Port, auf dem die Datenbank läuft;
- Name der Datenbank, zu der die Verbindung aufgebaut wird;
- Benutzername und Kennwort.

Die Anmeldung erfolgt dann unter dieser Benutzerkennung; damit stehen dem Perl-Script genau die SQL-Rechte dieses Benutzers zur Verfügung.

**Es ist sinnvoll, diese Paramter übersichtlich zu verwalten – etwa durch Auslagerung in eine externe Datei. Problematisch ist, dass das Kennwort für die Anmeldung im Klartext enthalten ist.**

#### 10.7.4.1 Verbindungsaufbau für MySQL

Die Parameter außer Benutzernamen und Kennwort werden häufig als Data Source Name `$dsn` zusammengefasst; exemplarisch etwa so:

```
$dsn = "DBI:mysql:database=$database;host=$host;port=$port";
$dbh = DBI->connect($dsn,$user,$pw);
```

Damit erhalten wir mit `$dbh` einen Handle zur Datenbank, deren Bezeichner in der Variablen `$database` abgelegt ist. Nach der Arbeitsweise von DBI wird allein durch die Syntax des Arguments der `connect`-Methode der notwendige DBD geladen, hier der MySQL-Treiber.

#### 10.7.4.2 Verbindungsaufbau andere DBMS

Für den *Verbindungsaufbau zu anderen DBMS* ist im Kern nur das Argument `$dsn` anzupassen, etwa für ODBC zu

```
$dsn = "DBI:ODBC:$odbc_dsn";
```

oder für PostgreSQL zu

```
$dsn =
     "DBI:PgPP:database=$database;host=$host;port=$port";
```

#### 10.7.4.3 Verbindungsabbau

Nach „getaner Arbeit“ wird mittels der DBI-Methode `close`, angewendet auf einen Database Handle durch die Syntax

```
$dbh->close();
```

die Verbindung wieder geschlossen, wodurch die allokierten Ressourcen freigesetzt werden. Dies geschieht bei Beenden des Perl-Interpreters zwar automatisch, ist für länger laufende Serveranwendungen aber essentiell wichtig.

### 10.7.5 Durchführen einer SQL-Anweisung

Nach der Anmeldung ist der weitere Prozess unabhängig davon, welches DBMS verwendet wird – eine saubere SQL-Syntax vorausgesetzt. Perl DBI unterscheidet aber syntaktisch zwischen SELECT-Abfragen, die eine komplexere Antwort erwartet, und den übrigen SQL-Anweisungen wie INSERT, UPDATE und DELETE.

#### 10.7.5.1 SELECT-Abfrage

Der Ablauf einer `SELECT`-Anfrage über einen existierenden Database Handle sieht folgendermaßen aus:

- Definition des SQL-Statements;
- Erzeugung des Statement-Handles für die SQL-Anweisung durch die DBI-Methode `prepare`;
- Durchführung der SQL-Anweisung;
- zeilenweises Abarbeiten des erhaltenen ResutSets (vgl. 3.3.4); hierfür stehen die Methoden `fetchrow_hashref` und `fetchrow_arrayref` zur Verfügung;
- Freisetzen des durch das Statement Handle belegten Speicherplatzes durch die DBI-Methode `finish`.

Ein realer Anwendungsfall soll dies verdeutlichen: In Kapitel 7 haben wir unsere Beispieldatenbank mit den zwei Tabellen `buecher` und `autoren` angelegt; nun soll mittels dbi jedes Buch und der Erstautor dazu ausgegeben werden. Script `dbi_buecher_1.pl` leistet dies (Abbildung 10.36). Nach dem Import des DBI-Moduls werden in den Zeilen 11 bis 16 die notwendigen connect-Parameter festgelegt und damit in Zeile 19 die Verbindung definiert. Der eigentliche Verbindungsaufbau geschieht dann in Zeile 22 (in diesem Beispiel ohne Kennwort, da das MySQL-System so konfiguriert wurde – was natürlich nicht für den Echtbetrieb denkbar ist). In Zeile 29ff wird die SQL-Abfrage über beide Tabellen definiert, danach das Statement Handle erzeugt (Zeile 34) und die Abfrage durchgeführt (Zeile 37). Das Statement Handle `$sth` verwaltet nun das erhaltene Resultset, welches zeilenweise abgefragt wird: Die Methode `fetchrow_arrayref` liefert für jede Zeile des Resultsets eine Referenz auf ein Array, dessen Elemente nach den Argumenten der SELECT-Abfrage belegt sind. Alternativ liefert die Methode `fetchrow_hashref` eine Referenz auf ein Perl-Hash, welches als Keys die Attributnamen verwendet; dies ist sehr übersichtlich, versagt aber bei Abfragen über mehrere Tabellen mit gleichen Attributnamen und ist von der Performance etwas schlechter. Die Auswertung über das ganze Resultset erfolgt über eine `while`-Schleife, anschließend wird das Statement Handle gelöscht und die Verbindung abgebaut (Zeilen 46/49 ).

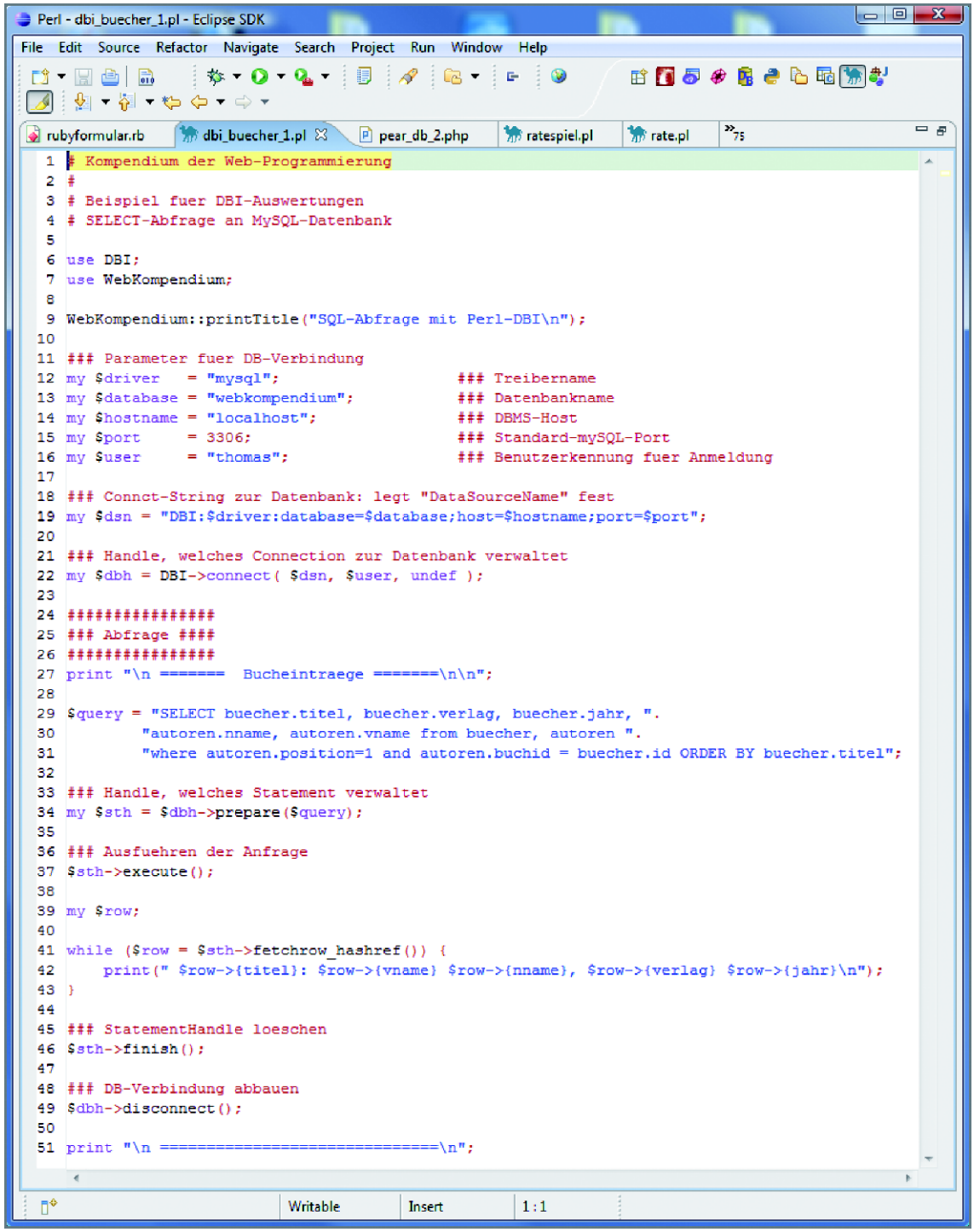



**Abbildung 10.36:**
Script `dbi_buecher_1.pl`: SQL-Abfrage über die Tabellen `buecher` und `autoren`

**Abbildung 10.37:** Ausführen von Script `dbi_buecher_1.pl` auf der Kommandozeile

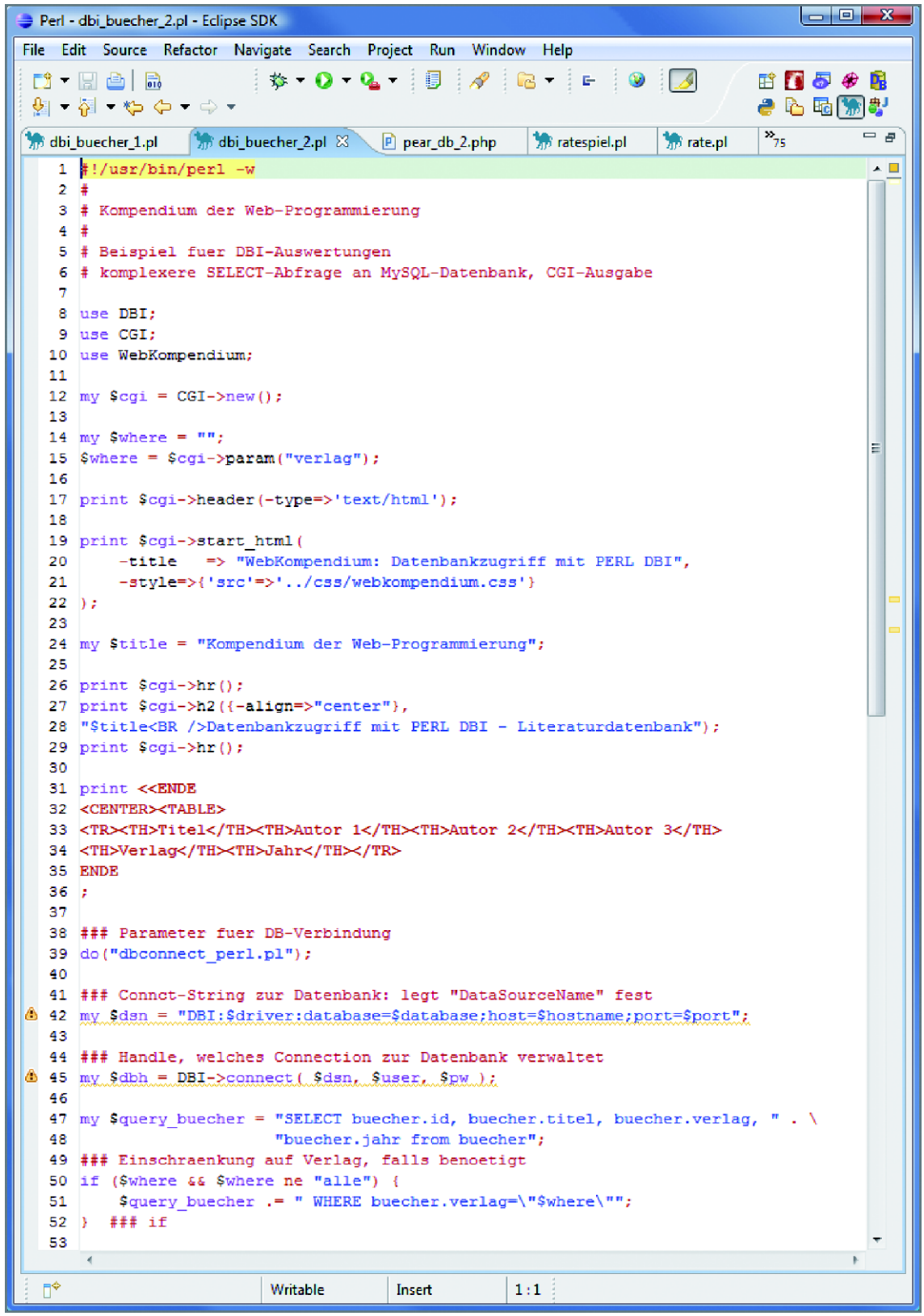
Nützlich kann die für ein Resultset anwendbare Methode `rows` sein: über

```
$anzahl = $sth->rows();
```

erhält man die Anzahl der Zeilen des Resultsets, welches über das Statement Handle `$sth` verwaltet wird; eine Anwendung zeigt das nachfolgende Beispiel.

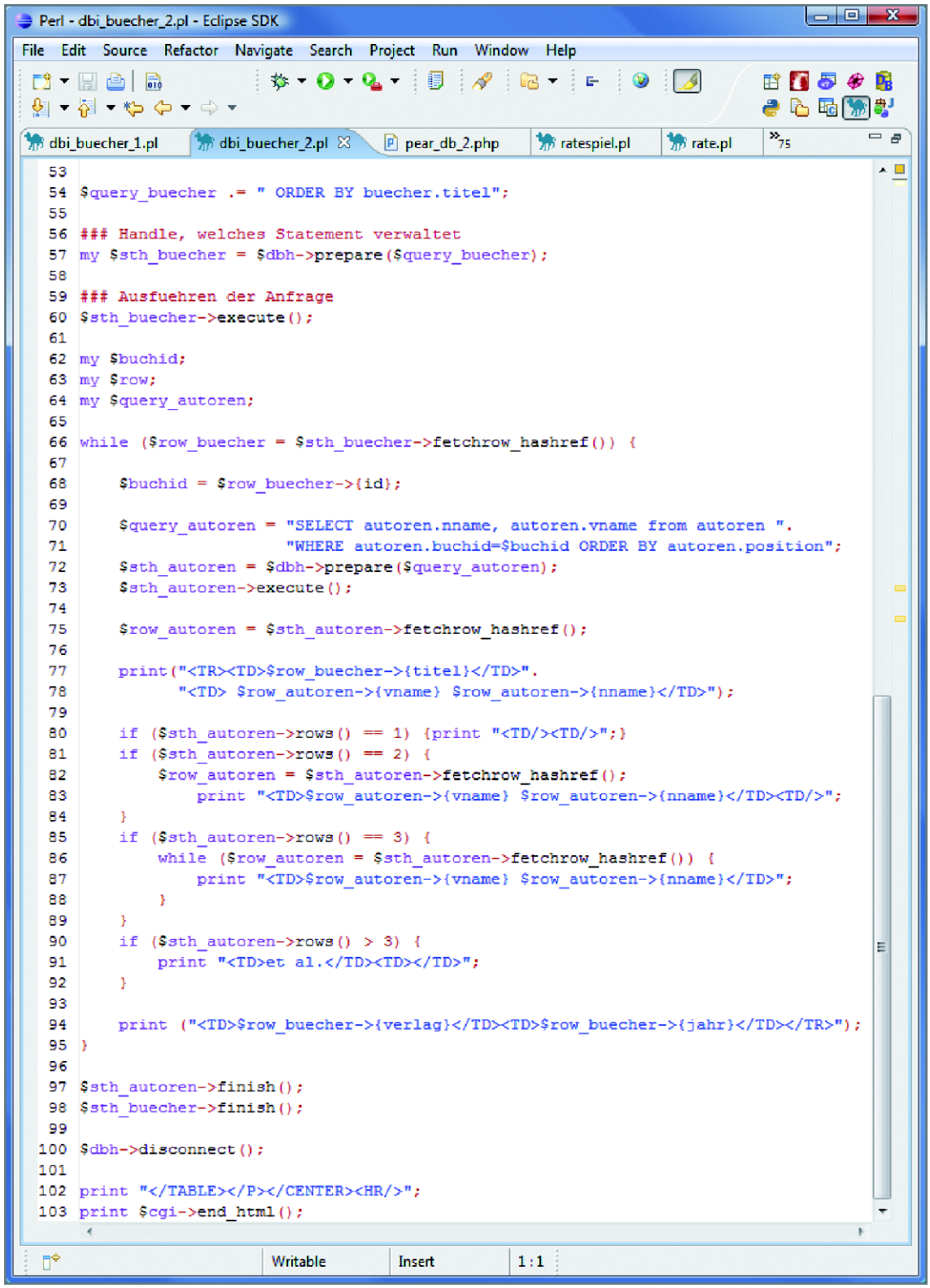
Nun wollen wir noch einen Schritt weiter gehen und die Autoren vollständig nach den „Regeln der alphabetischen Katalogisierung“ ausgeben: Sind es bis zu drei Autoren werden diese in der festgelegten Reihenfolge aufgezählt, bei mehr als drei Autoren wird der Erstautor und die Bemerkung „et al.“ (et alii, und andere) angegeben. Hierfür verwenden wir nun weiterhin einen Database Handle, dazu aber zwei Statement Handles: einen ersten für die Abfrage der Tabelle Bücher, die in einer äußeren Schleife verarbeitet wird, und einen zweiten Statement Handle innerhalb dieser Schleife, der zu dem jeweiligen Buch alle Autoren ausgibt. Zusätzlich kann dem Script ein Parameter für den Verlag übergeben werden, so dass nur die Literatur dieses Verlages ausgegeben wird. Und letztlich soll dieses Script als CGI-Script auf einem Webserver laufen, so dass die Ausgabe CSS-formatiert in einem Browser erfolgt. Dies alles leistet das Beispiel `dbi_buecher_2.pl` (Abbildungen 10.38 und 10.39).

Ebenso sind in diesem Beispiel – wie schon empfohlen – die Parameter für die Datenbankanbindung in eine externe Datei ausgelagert (`dbconnect_Perl.pl`, Abbildung 10.40).




```
#!/usr/bin/perl -w
#
# Kompendium der Web-Programmierung
#
# Beispiel fuer DBI-Auswertungen
# komplexere SELECT-Abfrage an MySQL-Datenbank, CGI-Ausgabe

use DBI;
use CGI;
use WebKompendium;

my $cgi = CGI->new();

my $where = "";
$where = $cgi->param("verlag");

print $cgi->header(-type=>'text/html');

print $cgi->start_html(
    -title   => "WebKompendium: Datenbankzugriff mit PERL DBI",
    -style=>{'src'=>'../css/webkompendium.css'}
);

my $title = "Kompendium der Web-Programmierung";

print $cgi->hr();
print $cgi->h2({-align=>"center"},
"$title<BR />Datenbankzugriff mit PERL DBI - Literaturdatenbank");
print $cgi->hr();

print <<ENDE
<CENTER><TABLE>
<TR><TH>Titel</TH><TH>Autor 1</TH><TH>Autor 2</TH><TH>Autor 3</TH>
<TH>Verlag</TH><TH>Jahr</TH></TR>
ENDE
;

### Parameter fuer DB-Verbindung
do("dbconnect_perl.pl");

### Connct-String zur Datenbank: legt "DataSourceName" fest
my $dsn = "DBI:$driver:database=$database;host=$hostname;port=$port";

### Handle, welches Connection zur Datenbank verwaltet
my $dbh = DBI->connect( $dsn, $user, $pw );

my $query_buecher = "SELECT buecher.id, buecher.titel, buecher.verlag, " . \
                    "buecher.jahr from buecher";
### Einschraenkung auf Verlag, falls benoetigt
if ($where && $where ne "alle") {
    $query_buecher .= " WHERE buecher.verlag=\"$where\"";
}  ### if

```

**Abbildung 10.38:**
Script `dbi_buecher_2.pl`, Teil I




```
53
54 $query_buecher .= " ORDER BY buecher.titel";
55
56 ### Handle, welches Statement verwaltet
57 my $sth_buecher = $dbh->prepare($query_buecher);
58
59 ### Ausfuehren der Anfrage
60 $sth_buecher->execute();
61
62 my $buchid;
63 my $row;
64 my $query_autoren;
65
66 while ($row_buecher = $sth_buecher->fetchrow_hashref()) {
67
68     $buchid = $row_buecher->{id};
69
70     $query_autoren = "SELECT autoren.nname, autoren.vname from autoren ".
71                      "WHERE autoren.buchid=$buchid ORDER BY autoren.position";
72     $sth_autoren = $dbh->prepare($query_autoren);
73     $sth_autoren->execute();
74
75     $row_autoren = $sth_autoren->fetchrow_hashref();
76
77     print("<TR><TD>$row_buecher->{titel}</TD>".
78           "<TD> $row_autoren->{vname} $row_autoren->{nname}</TD>");
79
80     if ($sth_autoren->rows() == 1) {print "<TD/><TD/>";}
81     if ($sth_autoren->rows() == 2) {
82         $row_autoren = $sth_autoren->fetchrow_hashref();
83             print "<TD>$row_autoren->{vname} $row_autoren->{nname}</TD><TD/>";
84     }
85     if ($sth_autoren->rows() == 3) {
86         while ($row_autoren = $sth_autoren->fetchrow_hashref()) {
87             print "<TD>$row_autoren->{vname} $row_autoren->{nname}</TD>";
88         }
89     }
90     if ($sth_autoren->rows() > 3) {
91         print "<TD>et al.</TD><TD></TD>";
92     }
93
94     print ("<TD>$row_buecher->{verlag}</TD><TD>$row_buecher->{jahr}</TD></TR>");
95 }
96
97 $sth_autoren->finish();
98 $sth_buecher->finish();
99
100 $dbh->disconnect();
101
102 print "</TABLE></P></CENTER><HR/>";
103 print $cgi->end_html();
```

**Abbildung 10.39:**
Script `dbi_buecher_2.pl`, Teil II

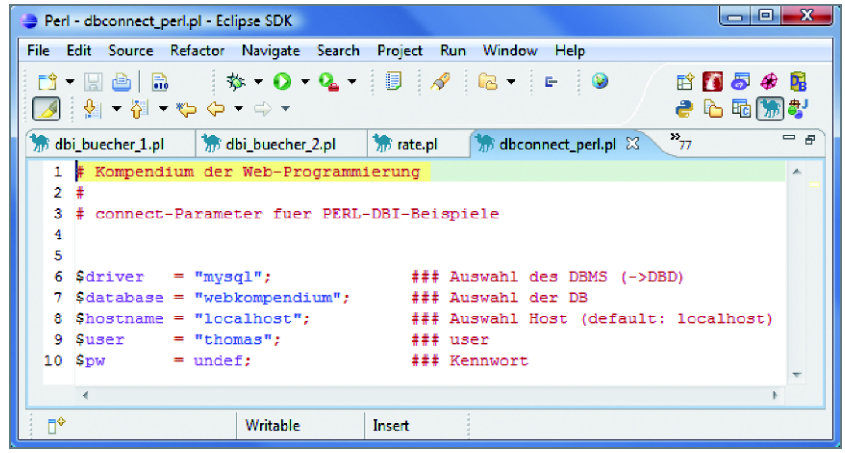



**Abbildung 10.40:** Auslagerung der zentralen connect-Parameter in Datei `dbconnect_perl.pl`

**Abbildung 10.41:** Ausführen von `dbi_buecher_2.pl`

### 10.7.5.2 Andere SQL-Anweisungen

Während die Situation bei einer SELECT-Abfrage vergleichsweise komplex ist, ist es bei den übrigen SQL-Anweisungen wie UPDATE, INSERT oder DELETE einfacher. Die DBI-Methode `do` ermöglicht die Ausführung dieser Anweisungen.

Exemplarisch sieht eine solche Anfrage dann folgendermaßen aus:

```
### Aufbau der Verbindung zu DB ueber Handle $dbh

$sql = "DELETE FROM buecher WHERE verlag!='Springer'";

$dbh->do($sql);
```

## 10.8 Das Perl-Modul LWP

Momentan ist Perl, wie wir es kennen, ideal auf dem Webserver einsetzbar – wir können es aber noch nicht auf dem Webclient verwenden, obwohl es ja so einfach ist, mit Perl Clientscripte zu schreiben. Das liegt daran, dass wir noch keine Möglichkeit haben, mit Perl das HTTP-Protokoll zu bedienen, insbesondere fehlen die hierfür notwendigen Netzwerk-Sockets.

Mit dem *LWP-Modul* (Library for WWW Access in Perl) steht diese Funktionalität bereit.

### 10.8.1 Installation

Im Gegensatz zu CGI ist LWP nicht Bestandteil der Standard-Distribution von Perl und muss zusätzlich installiert werden.

Ist LWP vorganden, so kann mittels

```
perl -MLWP -le "print(LWP->VERSION)"
```

die installierte Version abgefragt werden.

### 10.8.2 Grundfunktionalität

Das „Untermodul“ `LWP::Simple` stellt die wichtigsten Basisfunktionen zur Verfügung. Hierzu zählen:

- `head($url)`: Liefert eine Liste mit fünf Elementen, die die HEADer-Informationen der neueren HTTP-Versionen enthalten:
  - Content-type
  - document-length
  - modified-time
  - expires
  - server
- `get($url)`: Lädt die Adresse `$url` und gibt dieses Dokument als Zeichenkette zurück.
- `getprint($url)`: Lädt die Adresse `$url` und gibt dieses auf `<STDOUT>` aus.

Abbildung 10.8.2 zeigt ein Beispielscript mit diesen Methoden.

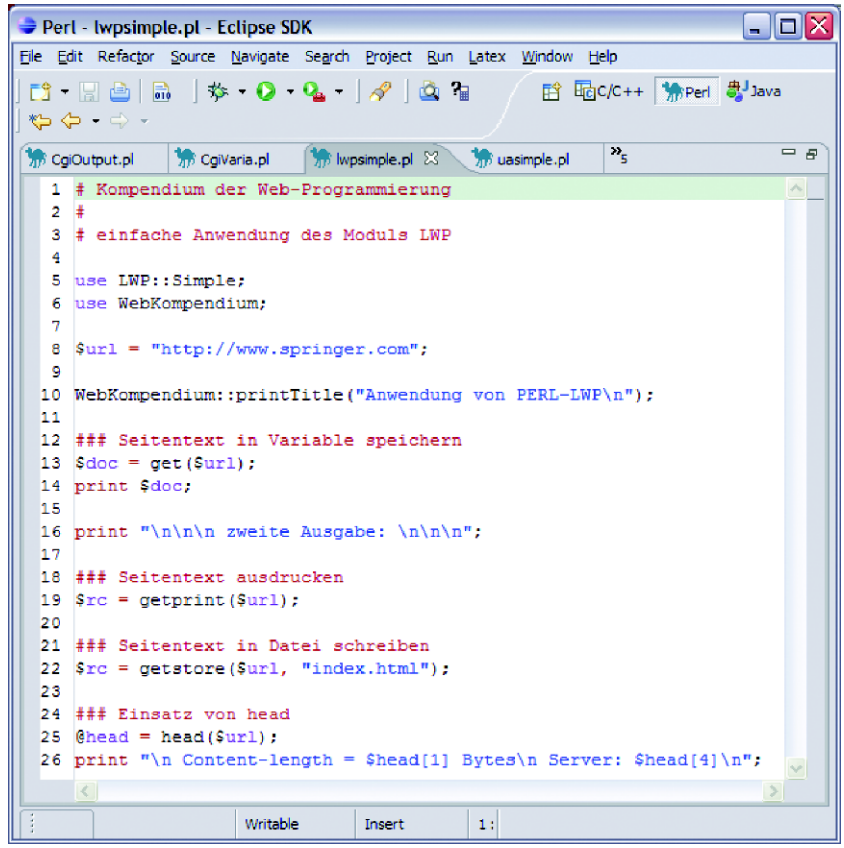



**Abbildung 10.42:**
Script `lwpsimple.pl`

### 10.8.3 LWP UserAgent

Mehr Funktionalität bietet der *UserAgent*. Dieser implementiert zwei zentrale Klassen `HTTP::Request` und `HTTP::Response`, welche die Anfrage des Clients und die Antwort des Servers verwalten.

**Dieses Grundprinzip der Kapselung in Request- und Response-Objekte ist richtungsweisend und typisch für moderne Konzepte der Web-Programmierung; beispielsweise bei den Java-Servlets (siehe 24.2) werden wir es wieder vorfinden.**

Nützlich ist ferner etwa die Klasse `HTTP::Status`. Abbildung 10.43 zeigt ein typisches Beispiel.

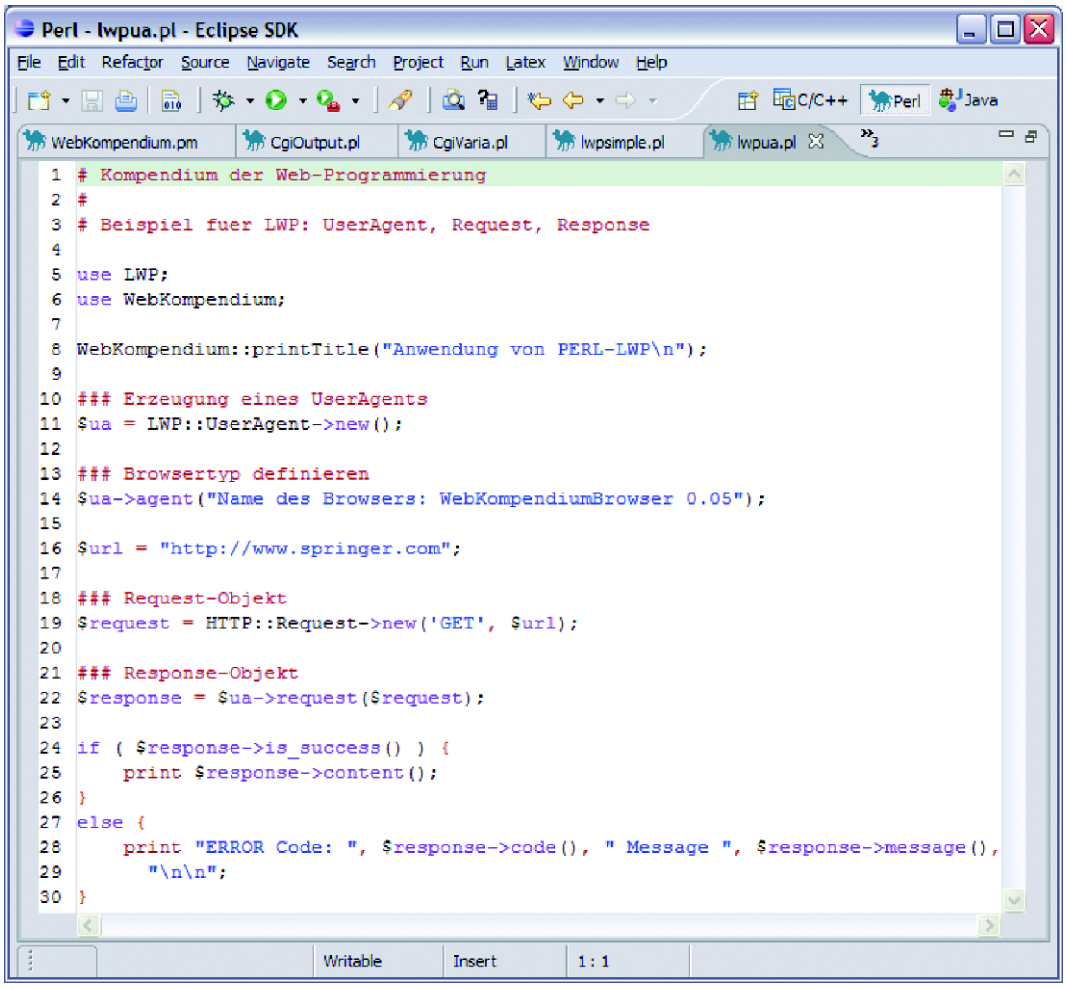



**Abbildung 10.43:** Script `lwpua.pl`

### 10.8.4 Anwendung von LWP

Mit dem Perl-Modul LWP lassen sich für das Web sehr zentrale Dienstprogramme einfach realisieren; Beispiele hierfür sind:

- Es lassen sich damit einfach und effizient Performance-Tests für Web-Anwendungen realisieren: Mittels LWP können einfach HTTP-Anfragen an eine Anwendung geschickt und die Antwortzeit gemessen werden.
- Implementierung von Suchverfahren im Web: Robots, Crawler und Spider.
- Ebenfalls einfach realisierbar ist ein Link-Checker, welcher eine Datei auf fehlerhafte Hyperlinks analysiert.

Für alle diese Beispiele lassen sich im Web leicht zahlreiche Umsetzungen in Perl finden.

## 10.9 Zusammenfassung

Wir haben nun Perl als Klassiker der Scriptsprachen kennengelernt, sowohl für Kommandozeilenapplikationen als auch im Web über den CGI-Mechanismus. Perl ist nach vielen Untersuchungen die weltweit am häufigsten eingesetzte Sprache im Bereich der Business-Entwicklungen (Quelle iX), und viele der

Konzepte von Perl werden wir in den folgenden Kapitel in anderen Sprachen wiederfinden.

Nachteile von Perl sind in der zunächst durch den CGI-Prozess beschränkten Performance (ein Ausweg bieten hier freilich die fastCGI-Techniken, vgl. Kapitel 19) und im „altbackenen“, meistens prozeduralen Programmierparadigma zu sehen. In der Tat bietet Perl keine moderne Objektorientierung wie etwa Java, aber das trifft auf fast alle Scriptsprachen zu.

Die Stärken von Perl liegen in der einfachen Anwendbarkeit und Erlernbarkeit, in der grenzenlosen Erweiterbarkeit über Module, die über CPAN bezogen werden können, und in dem sehr breiten Anwendungsbereich sowie in der großen internationalen Verbreitung – die „Lingua franca“ des modernen Computers.

# 11

# PHP

## 11.1 Die Scriptsprache PHP

Die *Scriptsprache PHP* ist im Gegensatz zu Perl von vornherein auf den Einsatz im Web ausgelegt, erst in jüngerer Zeit kommt auch PHP als Kommandozeilensprache zum Einsatz, das ist aber nach wie vor nebensächlich. International und im deutschsprachigen Raum erfreut sich PHP seit längerem einer großen Beliebtheit (Abbildung 11.1).

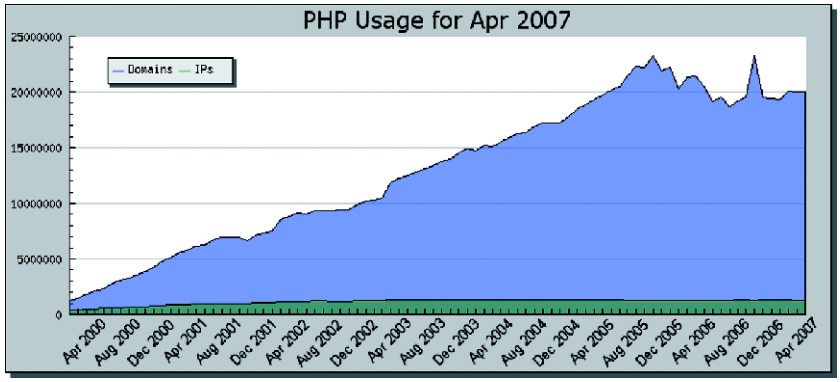



**Abbildung 11.1:** Verbreitung von PHP (Quelle: Netcraft)

Mit PHP ist es ausgesprochen einfach, dynamische Webseiten zu erstellen. Dabei verfolgt PHP vom Vorgehen den umgekehrten Ansatz wie Perl: Während Perl aus dem Programm heraus HTML-Code erzeugt, integriert PHP die dynamischen Inhalte in eine HTML-Datei. Von daher ist der „Abschreckungsgrad“ deutlich kleiner, ebenso der administrative Aufwand: keine Shebang-Zeile mehr, keine komplexe Konfiguration des Apache über ScriptAlias, keine detaillierte Rechtevergabe, keinen HTTP-Code 500: „Internal Server Error“ – alles einfacher in PHP.

Initiator für die große Beliebtheit von PHP in Deutschland war vermutlich das 1999 erschienene Buch „php - dynamische Webauftritte professionell realsieren“ von Egon Schmidt und Christian Cartus, da es die erste verbreite Publikation zu diesem Thema war; hier wurde das damals neue PHP3 vorgestellt. Inzwischen liegen zahlreiche, umfangreiche Publikationen zu dem Thema vor, etwa [Fue05] und [Kra05]. Die wichtigsten Bestandteile von PHP werden hier vorgestellt, allerdings kann nicht die volle Breite der umfangreichen Sprache behandelt werden; mittels der im Web [web] verfügbaren Quellen ist aber das vorhandene sehr leicht zu ergänzen.

**Abbildung 11.2:** PHP-Logo

Die zentrale Ressource für PHP ist `http://www.php.net` (Abbildung 11.3).

Die Entwicklung von PHP könnte aber auch ein Problem dieser Sprache werden: Ausgehend von der einfachen, direkt anwendbaren Web-Scriptsprache ist mit jeder Version PHP mächtiger und den großen Sprachen wie Java ähnlicher geworden – damit verliert in Teilen PHP seine Attraktivität. Das kann die Stagnation der PHP-Ausbreitung in den letzten Jahren (Abbildung 11.1), konkret seit dem Erscheinen der Version 5 von PHP, vielleicht erklären.

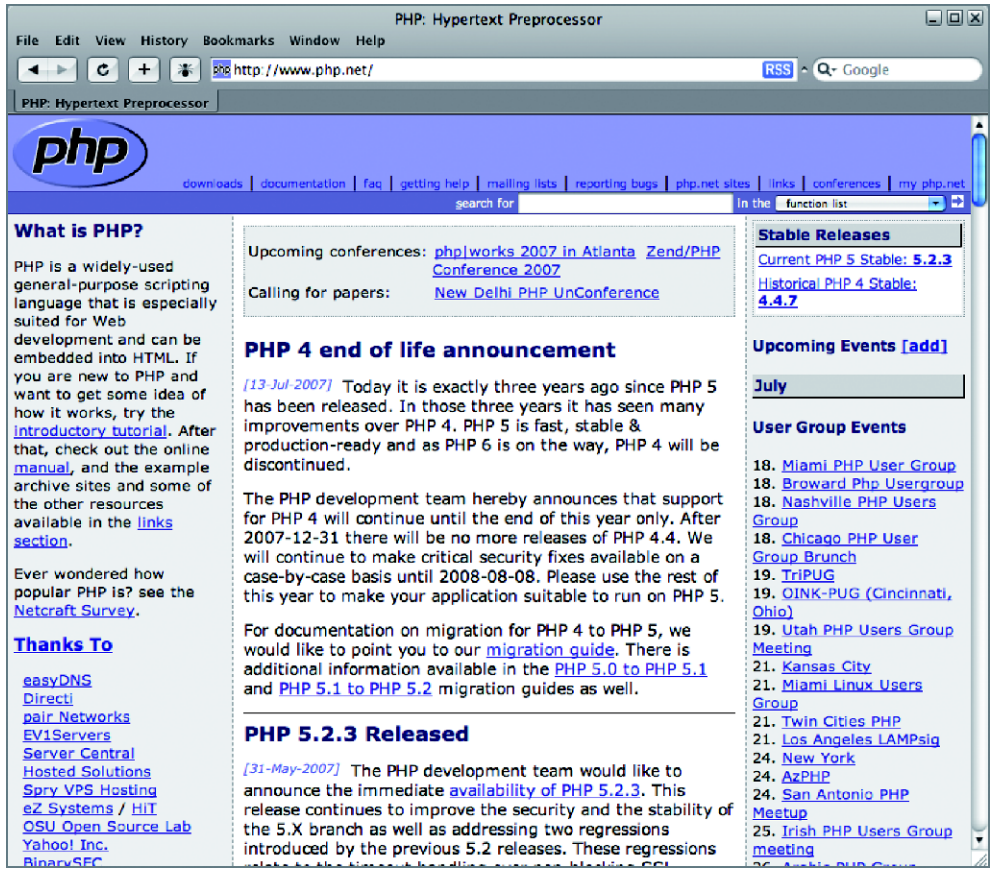



**Abbildung 11.3:** Die offizielle PHP-Site mit der Ankündigung der Einstellung von Version 4

Netcraft gibt aktuell eine Zahl von rund 20 Millionen Domains an, die mittels PHP gehostet werden (Abbildung 11.1); ihre Zahl steigt seit Jahren kontinuierlich, allerdings gibt es seit der Einführung der Version 5 die eingangs beschriebene Stagnation.

### 11.1.1 Genese von PHP

Der Vater von PHP ist Rasmus Lerdorf. Er stellte 1994 den ersten Vorläufer vor, gemeinsam mit einem weiteren Projekt mit dem Namen FI. 1995 verschmolzen beide zu „PHP/FI Version 2“. In der Folge musste er seine Entwicklergruppe vergrößern, insbesondere kamen Zmievski, Suraski und Gutmans dazu. Gemeinsam stellten sie 1997 die durchbrechende Version 3 von PHP vor.

Da PHP direkt für den Einsatz im Web entwickelt wurde, ist es an einen Webserver gebunden; die frühen Versionen haben nur den Apache unterstützt, die neueren können auch mit anderen Webservern betrieben werden.

Die neue Hauptversion 5 von PHP ist gut abwärtskompatibel, es können die alten Scripte meistens weiter genutzt werden. Eine wesentliche Ausnahme stellen jeweils neu hinzugekommene Schlüsselworte dar, etwa `parent` in PHP 5.

### 11.1.2 LAMP

Ein geläufiger Begriff in Zusammenhang mit PHP ist *„LAMP“,* eine Abkürzung für **L**inux als Betriebssystem, **A**pache für den Webserver, **M**ySQL als DBMS und **P**HP als Scriptsprache. Hinzu kommen einige Variationen, etwa LAMPP für das zusätzliche **P**erl oder WAMP mit **W**indows als Serverbetriebssystem, was im Prinzip genauso möglich ist (siehe 3.2)

## 11.2 Installation als Apache-Modul

Es gibt zwei Möglichkeiten der Installation von PHP: als CGI (vgl. 9) oder als Apache-Modul (vgl.6.3.7). Zwar ist die Installation als CGI einfacher, wegen der deutlich schlechteren Performance muss aber von dieser deutlich abgeraten werden: Als CGI wird bei jedem Request an eine PHP-Seite der PHP-Interpreter neu in einem eigenen Prozess gestartet, während er bei einer Installation als Servermodul Bestandteil des Webservers ist und somit immer mitläuft, ein deutlicher Performancegewinn.

**PHP sollte immer als Server-Modul betrieben werden.**

Bei der Installation als Apache-Mopdul stehen aber immer noch zwei Wege zur Auswahl (vgl. 6.3.7): statisch und dynamisch. Da beim statischen Compilieren PHP fester Teil des Webservers wird, die PHP-Releasezyklen aber völlig losgelöst von Apache sind, kommt man in die Lage häufiger Neuinstallationen des kompletten Apaches, was nicht sinnvoll ist. Deshalb bietet sich der Weg über die dynamische Installation – wie üblich für Fremdmodule des Apache – an, da hier nur entweder PHP oder der Webserver neu aufzusetzen sind.

### 11.2.1 Installation auf Unix/Linux

Unter Unix stehen wieder die üblichen zwei Wege der Installation zur Verfügung: über einen Paketmanager des Betriebssystems wie `aptitude` und `pkgadd` oder die Installation von Hand.

Der erste Weg ist vergleichsweise einfach, beim zweiten liegen zunächst die üblichen Schritte an: herunterladen, entpacken, konfigurieren, compilieren und installieren. Wichtig ist hierbei das Konfigurieren von PHP, da hier festzulegen ist, welche Datenbanktreiber in PHP integriert werden sollen (Default ist je nach Version MySQL, ODBC oder SQLite).

### 11.2.2 Installation auf Windows

Unter Windows liegt wieder eine msi-Installer-Datei vor, die zunächst die üblichen Dinge abfragt, insbesondere für welchen Webserver PHP installiert werden soll (Abbildung 11.4).

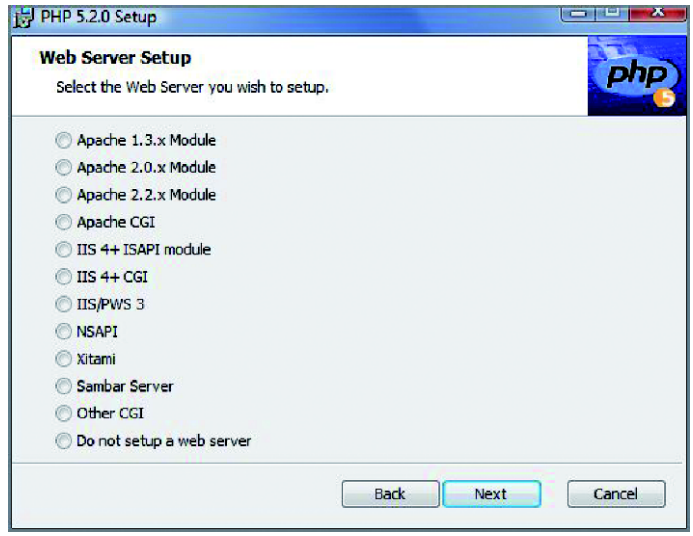



**Abbildung 11.4:**
PHP-Installation auf Vista

Danach werden verschiedene Komponenten von PHP angeboten; hier sollten nicht einfach die Default-Werte übernommen werden (dann fehlen etwa der MySQL-Support und die PEAR-Bibliothek), sondern es sollten wohlüberlegt die benötigten Komponenten ausgewählt werden (Abbildung 11.5).

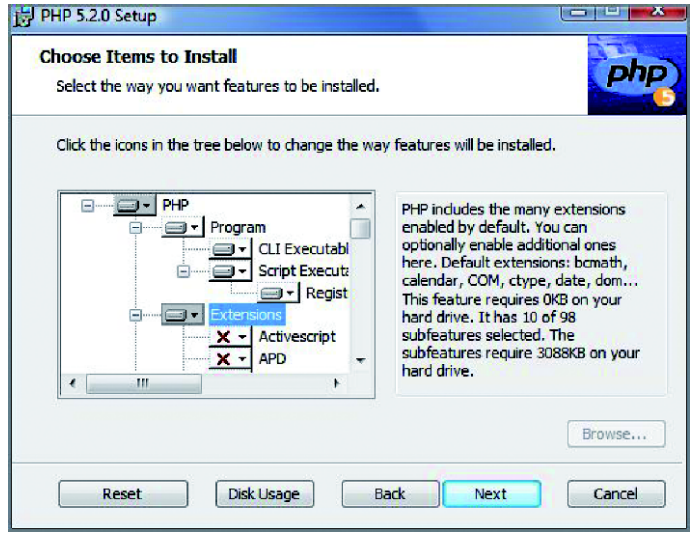



**Abbildung 11.5:**
Auswahl der PHP-Komponenten

### 11.2.2.1 Alternative unter Windows: XAMPP

Für Windows-Benutzer mit wenig Zeit steht das bereits vorgestellte Gesamtpaket XAMPP zur Verfügung (vgl. 3.4). Der Vorteil ist die schnelle Installation, der Nachteil, dass man wenig über die Konfiguration und das Zusammenspiel der einzelnen Komponenten lernt, weshalb der Griff dazu sehr genau überlegt sein muss.

### 11.2.3 Konfiguration des Servers

Abschließend muss noch – im Kern gleich für Unix und Windows – der Apache-Webserver für die Verwendung von PHP konfiguriert werden; hierzu sind wenige Einträge in der Datei `httpd.conf` notwendig: Das dynamische Laden des Moduls (bei Apache 1.3 ist hier neben der `LoadModule`-Anweisung noch ein entsprechendes `AddModule` für den Neuaufbau der Modulliste nach `clearmodule` notwendig) und die Verknüpfung von Dateien mit der Endung .php mit dem PHP-Interpreter; Abbildung 11.6 zeigt ein Beispiel.

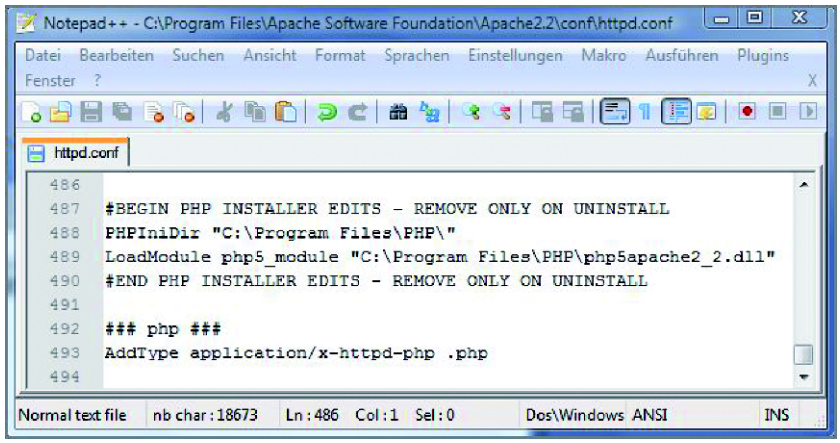



**Abbildung 11.6:**
Anpassung der `httpd.conf`

### 11.2.4 Testen der Installation: phpinfo

Eine ausgesprochen nützliche PHP-Funktion ist `phpinfo`; eine einfache PHP-Datei mit folgendem Inhalt

```
<?php  phpinfo();  ?>
```

gibt damit eine umfassende Information über das Gesamtsystem aus (Abbildung 11.7).

### 11.2.5 Konfiguration von PHP: die Datei `php.ini`

Nach der Installation muss die zentrale Konfigurations-Datei `php.ini` grundlegend angepasst werden; ähnlich zur Apache-Konfiguration sind zahlreiche Kommentare vorhanden, die dies erleichtern, es sollten aber nie unbedacht alle Standardwerte übernommen werden.

Einen Überblick über die aktuelle Konfiguration von PHP gibt die Funktion `phpinfo`, vgl. 11.2.4.

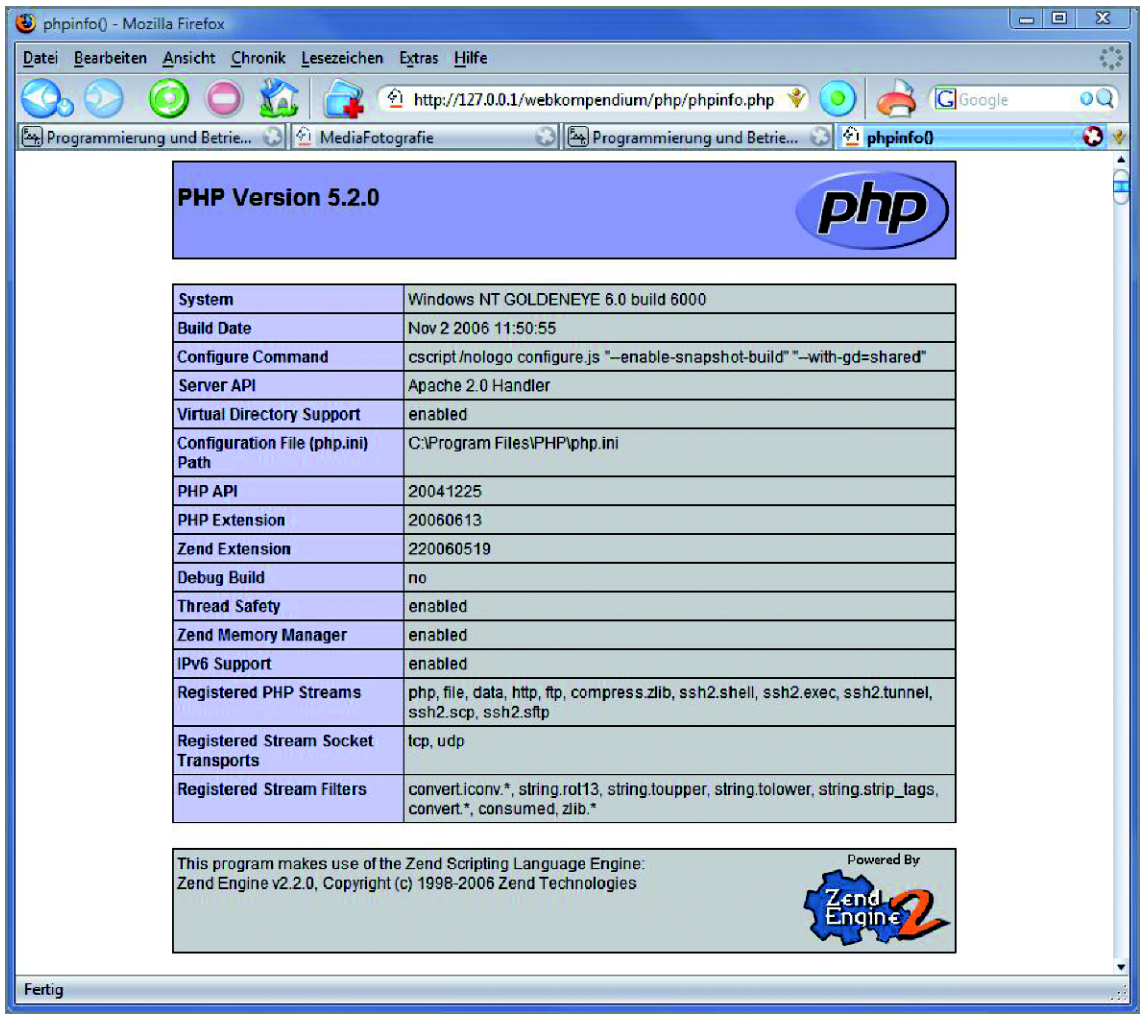



**Abbildung 11.7:**
Ein kleiner Ausschnitt der Ausgabe von `phpinfo`

### 11.2.6 Besondere Sicherheit bei PHP: safe_mode

PHP verfügt über die Möglichkeit, gewisse sicherheitskritische Operationen zentral auszuschalten: der *safe_mode*. Ziel ist es, insbesondere auf einem von mehreren Benutzern genutzten Webserver einen sichereren Betrieb zu gewährleisten.

Der safe_mode wird über die zentrale Konfigurationsdatei `php.ini` ein- oder ausgeschaltet (vgl. 11.2.5): Die Direktive

```
safe_mode = Off | On
```

steuert dies.

In der `php.ini` stehen für den safe_mode zahlreiche Parameter bereit, über welche das Verhalten des PHP-Systems im safe_mode geregelt werden kann. Beispielsweise kann über

```
safe_mode_include_dir
```

die Ausführung von PHP auf spezielle Verzeichnisse (und deren Unterverzeichnisse) beschränkt werden.

## 11.3 Grundlegende Syntax

In diesem Abschnitt wird die grundlegende Arbeitsweise und Syntax von PHP vorgestellt. Vieles ist ähnlich zu Perl, es bestehen aber auch große Unterschiede.

### 11.3.1 Ein HelloWorld in PHP

Nach erfolgreicher Installation des Server-Moduls soll hier das erste PHP-Script erzeugt und ausgeführt werden. Dazu wird eine Datei mit dem Namen `HelloWorld.php` erzeugt (Listing 11.1). Diese enthält ein HTML-Gerüst und dazu wenige PHP-Elemente, die, wie bereits beschrieben, in den HTML-Code integriert sind.

Die Datei kann etwa folgenden Inhalt haben:

```
<?php
/*
 * Kompendium der Web-Programmierung
 * Web-Applikationen mit PHP
 *
 * HelloWorld in PHP
 */
?>
<HTML>
<HEAD>
<TITLE>WebKompendium: HelloWorld mit PHP</TITLE>
<link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
<link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
</HEAD>
<BODY>
<H2><HR><CENTER>

<!-- Script Anfang -->

<SCRIPT language="php">
    // Ausgabe im php-Script
    echo "Kompendium der Web-Programmierung<BR>I like PHP";
</SCRIPT>

<!-- Script Ende -->

<HR><IMG src="/webkompendium/images/php.gif"><HR></CENTER></H2>
</BODY>
</HTML>
```

**Listing 11.1:** HelloWorld in PHP (`Helloworld.php`)

Hier sind zwei PHP-Blöcke integriert, am Anfang ein reiner Kommentarbereich und in der Mitte eine Ausgabe. Diese Datei versehen wir mit den normalen Rechten einer HTML-Datei (in Unix normalerweise 644: Jeder und die Gruppe dürfen lesen, der Besitzer zusätzlich schreiben, ausführen keiner) und kopieren diese Datei in den Verzeichnisbaum unterhalb des Dokument-Roots des Webservers (vgl. 6.3.5.2), also zu den normalen HTML-Webdokumenten. Der Aufruf über einen Browser ergibt dann eine Ausgabe nach Abbildung 11.8.

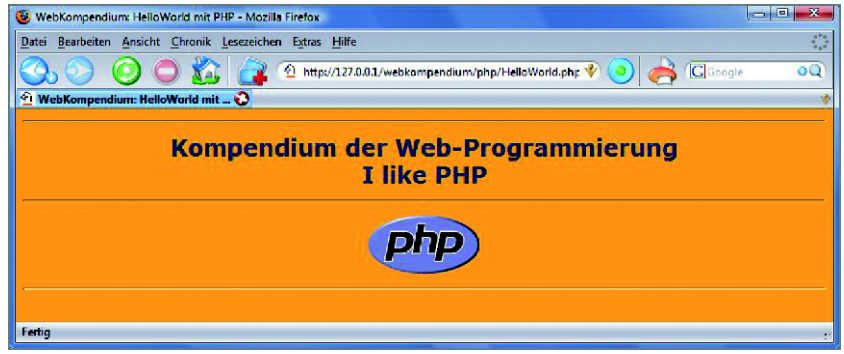



**Abbildung 11.8:**
Ausführen von `Helloworld.php`

### 11.3.2 Begrenzung der PHP-Anteile

Das Beispiel `HelloWorld.php` zeigt bereits, dass die PHP-Codeanteile durch spezielle Tags begrenzt werden müssen; hierfür stehen mehrere Möglichkeiten zur Verfügung:

- Sehr verbreitet ist die Begrenzung der Art

```
<?php
   // PHP-Code
?>
```

  wobei „php“ sowohl in Klein- als auch in Großschreibweise benutzt werden kann.
- Eine kürzere Schreibweise ist

```
<?
   // PHP-Code
?>
```

  Diese ist aber nur möglich, wenn in der `php.ini`-Datei (vgl. 11.2.5) die entsprechende Direktive gesetzt ist:

```
short_open_tag = On | Off
```

  Inzwischen ist der Default-Wert „Off“ (Verbot der kurzen Schreibweise), was insoweit von Vorteil ist, weil dann der PHP-Code ohne Konflikte auf andere PHP-Server übertragen werden kann.
- An die Syntax von JavaScript angelehnt ist die Schreibweise

```
<script language="php">
   // PHP-Code
</script>
```

  mit dem `<script>`-Tag; allerdings muss dann das Attribut `language` belegt werden, da als Default-Spache für dieses Script JavaScript gesetzt ist (vgl. 15).

- Sehr ähnlich zu PHP, aber proprietär gebunden, ist die Microsoft-Technik Active Server Pages (ASP). Diese verwendet spezielle Tags, welche auch für PHP genutzt werden können, wenn in der Konfigurationsdatei `php.ini` die entsprechende Direktive gesetzt ist. Die Syntax lautet dann

```
<%
   // PHP-Code
%>
```

  Entscheidend über den Einsatz dieser Syntax ist die php.ini-Datei, konkret die Setzung (in diesem Beispiel – der Default-Fall – werden die ASP-Tags nicht unterstützt):

```
; Allow ASP-style <% %> tags.
asp_tags = Off
```

- Soll lediglich der Wert einer PHP-Variablen ausgegeben werden, steht eine weitere praktische Syntax zur Verfügung:

```
<?= $auszugebende_variable ?>
```

### 11.3.3 Entwicklungsumgebung für PHP

**Abbildung 11.9:** PHPeclipse

Für PHP stehen aufgrund seiner großen Popularität zahlreiche Entwicklungsumgebungen zur Verfügung. Hierzu zählen auch die fortgeschrittenen Web-Editoren wie Dreamweaver und GoLive. Empfehlenswert ist wieder der Einsatz eines Plugins für die Open Source-Entwicklungsumgebung Eclipse (vgl.8.1). Auch hier stehen mehrere zur Verfügung, eine gute Möglichkeit ist PHPeclipse, das auch eine umfassende HTML-Unterstützung enthält [web].

Das Plugin bindet auch einen lokalen Webserver ein und kann eine MySQL-Datenbank verwalten.

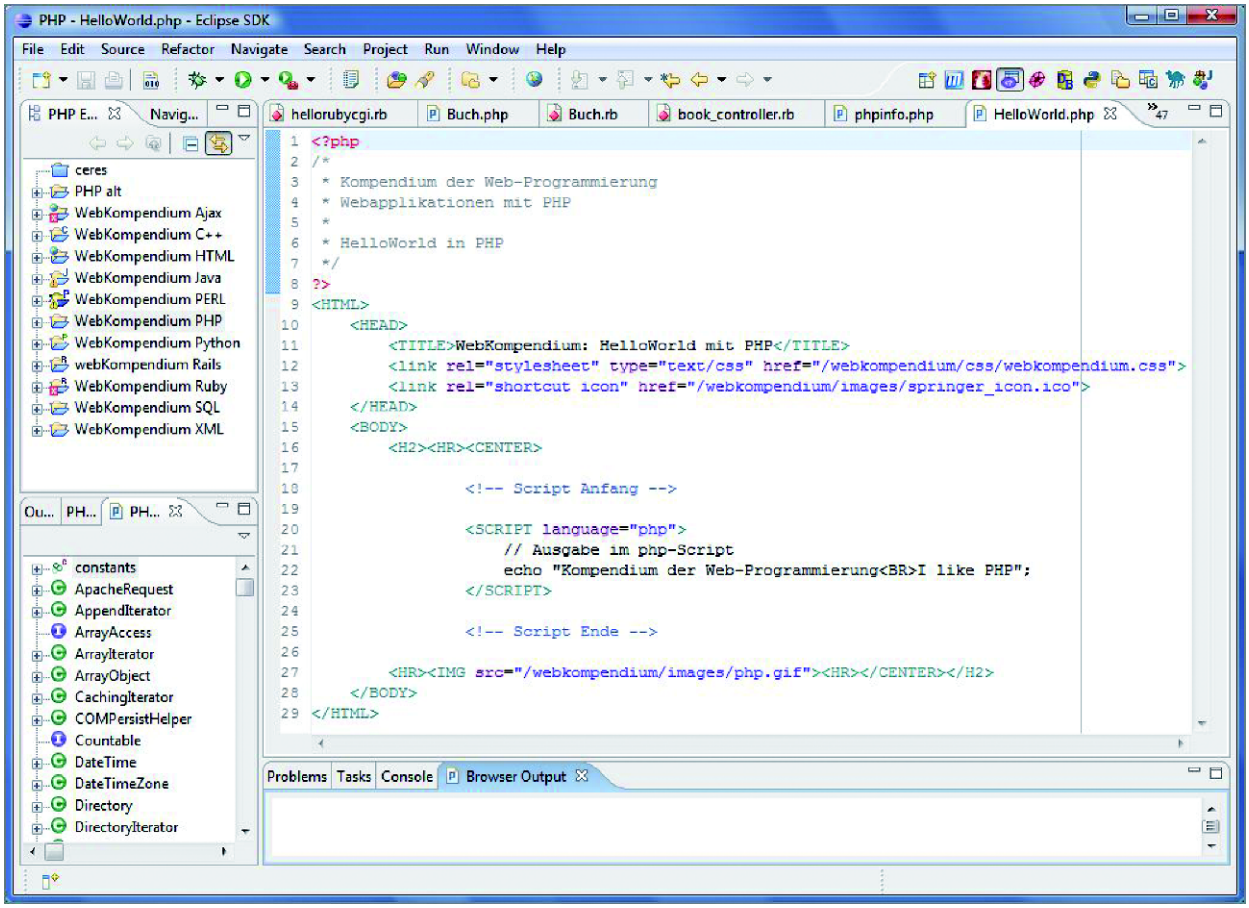



**Abbildung 11.10:**
Die Entwicklungsumgebung Eclipse mit dem Plugin PHPeclipse

### 11.3.3.1 Unterbrochenes „HelloWorld“

Das Script `HelloWorld.php` kann auch durch HTML-Code „unterbrochen“ werden; Abbildung 11.11 zeigt ein entsprechendes Script, welches zunächst eine Variable definiert, PHP verlässt und später in einem zweiten PHP-Block die Variable ausliest.

Dieses Beispiel zeigt auch die Arbeitsweise von PHP: Durch den HTTP-Request der PHP-Datei wird diese vor Auslieferung auf dem Server verarbeitet, was bedeutet, dass alle PHP-Scriptanteile gelesen und ausgeführt werden. Dadurch gelangt kein PHP-Code zum Client – die Web-Entwicklung bleibt vollständig geschützt.

Als Beispiel soll die „Ausgabe“ des Scriptes `HelloWorld.php` dienen; der Webclient hat die Codesequenz nach Abbildung 11.12 erhalten, also keinerlei Codeanteile – dies wäre auch sinnlos, da der Webclient sowieso kein PHP ausführen kann.

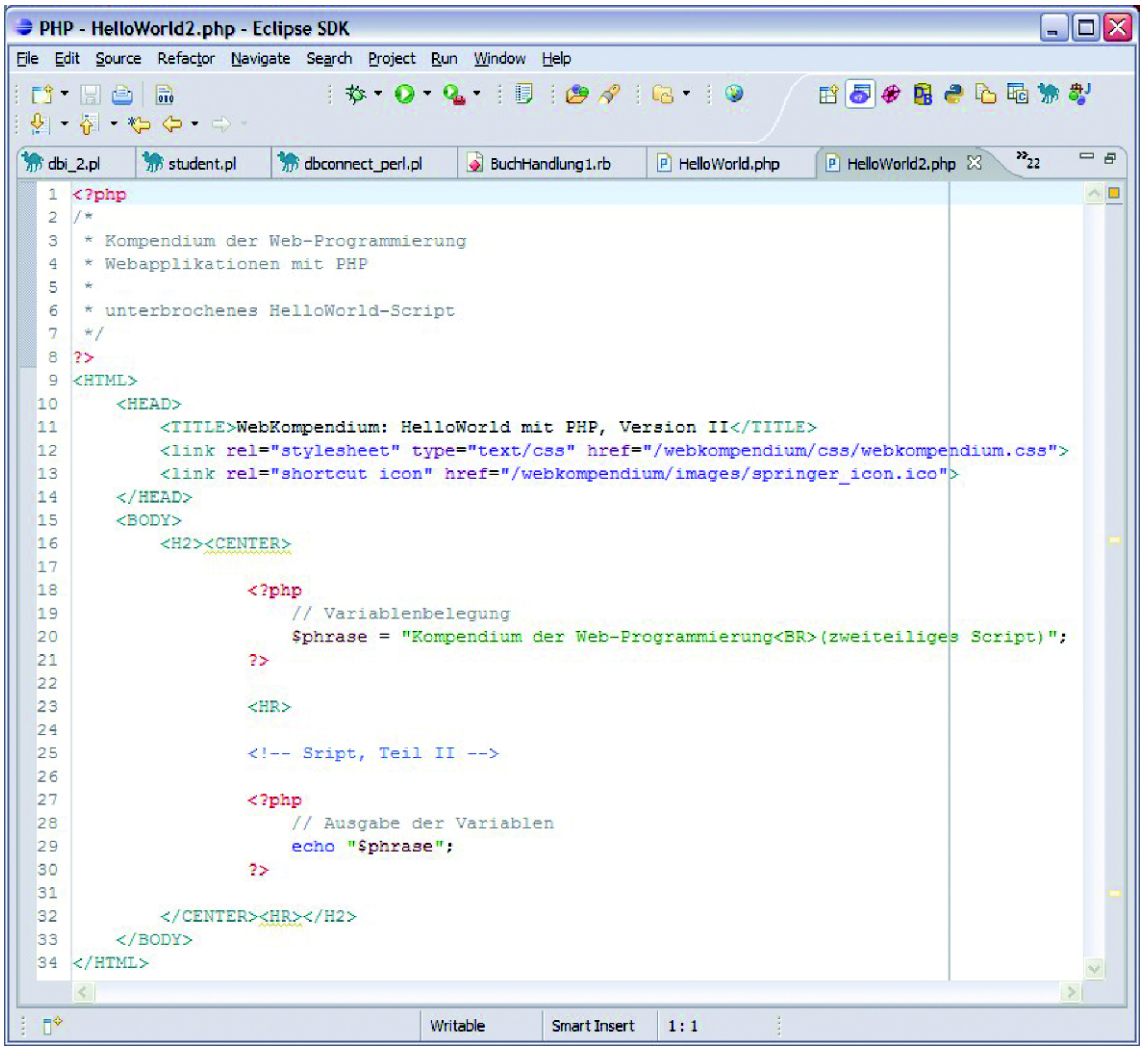



**Abbildung 11.11:**
Unterbrochenes HelloWorld-Script (Script `HelloWorld2.php`)

### 11.3.4 Kommentare in PHP

Eine nachhaltige Programmierung bedarf zwangsweise einer guten und vollständigen Kommentierung, denn nur mit Kommentierung hat der Programmcode eine Chance, weiter verwendet zu werden. Für PHP stehen zwei Möglichkeiten der Kommentierung bereit.

**Softwareentwicklung ist nur mit ausführlicher Kommentierung sinnvoll, so dass die Programmentwicklungen zu einem späteren Zeitpunkt weiter verwendet werden können. Ein Kommentar-Anteil von 50 % ist nicht unüblich. Zur Erstellung einer guten Kommentierung stehen unterschiedliche Werkzeuge zur Verfügung, siehe Abschnitt 4.5**

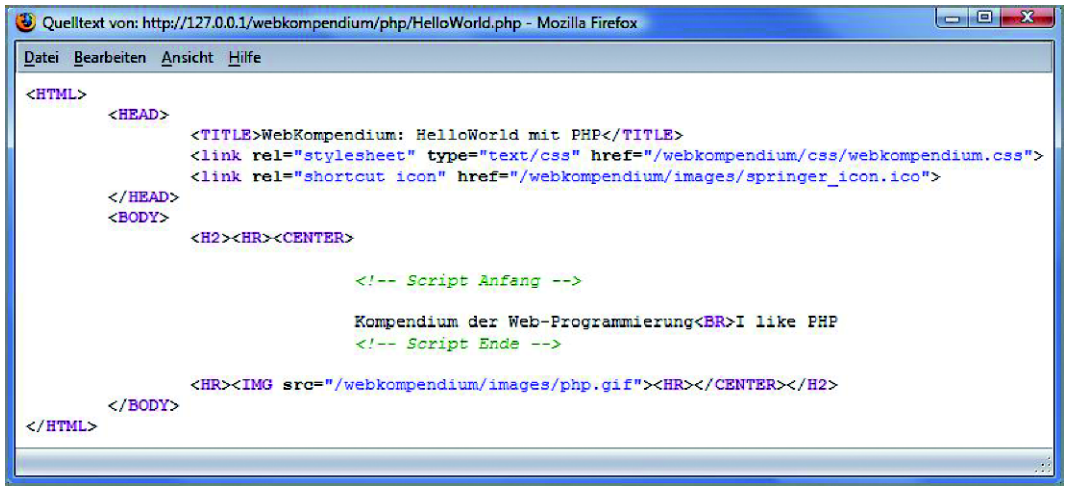



**Abbildung 11.12:** Ausgabe von Script `HelloWorld.php` im Quellcode

### 11.3.4.1 Allgemeine Kommentare

Für die an moderne Programmiersprachen Gewöhnten ist es erfreulich, dass PHP mehr – und vorallem die üblichen – Arten der Kommentierung unterstützt als Perl.

Es gibt den „Block-Kommentar“ über mehrere Zeilen

```
/*
 * hier ein Kommentar
 * im PHP-Code
 * ueber mehrere Zeilen
 */
```

Wichtig für diese Art der Kommentierung ist die Anfangscodierung `/*` und die Endmarkierung `*/`, dazwischen ist man völlig frei.

Demgegenüber beginnt der Zeilenkommentar an einer Stelle einer Anweisungszeile durch die Zeichenfolge `//` und geht bis zum Zeilenende:

```
echo "PHP muss kommentiert werden";   // wichtige Ausgabe
```

PHP unterstützt an dieser Stelle aber auch die klassische Kommentierung durch `#`, wie sie etwa bei der Shell-Programmierung oder bei Perl verwendet wird.

### 11.3.4.2 phpdoc

Die Programmiersprache Java brachte mit ihrem Erscheinen zahlreiche moderne Konzepte erfolgreich in die Programmierung ein, darunter eine stark unterstützte Generierung von Dokumentation im HTML-Format: javadoc. Dieses Konzept haben viele Sprachen inzwischen übernommen, und so steht auch phpdoc als Ergänzung zu PHP zur Verfügung [web], muss also zusätzlich installiert werden.

Ein phpdoc-Kommentar ist syntaktisch ähnlich zum Mehrzeilenkommentar, beginnt allerdings mit der Zeichenfolge `/**` mit doppeltem Stern:

```
/**
* Short description - used in indexlists
*
* Multiple line detailed description.
* The handling of line breaks and HTML is up to the renderer.
* Order: short description - detailed description - doc tags.
*
* @param string Target directory for the generated HTML Files
*/
```

**Die syntaktische Nähe zu javadoc ist sehr groß.**

phpdoc ist – ebenso wie javadoc – ideal für eine objektorientierte Sichtweise.

Einen weiterführenden, universelleren Ansatz verfolgt das Dokumentationssystem Doxygen.

### 11.3.5 Datenstrukturen in PHP

#### 11.3.5.1 Bezeichner in PHP

Bezeichner – wählbare Namen für Variablen und Methoden – können in PHP frei gewählt werden, vorausgesetzt das erste Zeichen ist ein Buchstabe oder das Zeichen `_`; Variablenbezeichner beginnen wie in Perl mit einem führenden `$`.

Eine Ausnahme bilden noch die rund 50 Schlüsselwörter der Scriptsprache; natürlich kann keine Funktion die Bezeichnung `while` bekommen.

#### 11.3.5.2 Skalare Variable

Variablen, die nur einen Wert ablegen – „skalare“ Variablen – sind in PHP ähnlich einfach wie in Perl zu behandeln, etwa:

```
$pi    = 3.1415926;              // Gleitkommazahl
$wert = 42;                      // ganze Zahl
$text = "Web-Kompendium";        // Zeichenkette

$a = $b = $c = 313;              // dreifache Zuweisung
```

Insgesamt verwendet PHP nur vier Datentypen für skalare Variablen:

- `boolean` für Wahrheitswerte;
- `integer` für ganze Zahlen (in der internen Darstellung als Zweierkomplement mit 32 bit Größe);
- `float` für Gleitkommazahlen nach IEEE 754 in 64 bit Darstellung;
- `string` für Zeichenketten.

#### 11.3.5.3 Typisierung

PHP ist – typisch für eine Scriptsprache – keine typisierte Sprache: Variablen müssen nicht deklariert werden, haben also keinen festgelegten Typ wie integer oder float. Eine Typumwandlung (Casting) erfolgt direkt durch die Scriptsprache, ohne dass der Entwickler hier spezielle Operationen benötigt. Dies macht die Entwicklung auf der einen Seite sehr bequem – auf der anderen Seite ist es jedoch dadurch auch fehleranfälliger, da der Entwickler die Verwendung einer Variablen im falschen Kontext nicht direkt bemerkt.

Am vorherigen Beispiel kann man etwa

```
$wert = 42;                    // ganze Zahl
$wert = $wert + 3.1415;        // Gleitkommazahl
```

syntaktisch korrekt verwenden; dabei wird die Variable `$wert` automatisch von integer nach float umgewandelt.

#### 11.3.5.4 Typabfrage und Initialisierung

PHP bietet zwei nützliche Funktionen für skalare Variablen:

- `gettype($wert)` liefert den Typ der Variablen `$wert`;
- die Methode `isset($wert)` überprüft, ob die Variable `$wert` einen Wert hat; als „kein Wert“ gilt dabei nur `undef`, auch eine Belegung mit dem numerischen Wert `0` oder einer leeren Zeichenkette ist wahr.

#### 11.3.5.5 Gültigkeitsbereich

Während in Perl normale Variablen noch global sind, verfolgt PHP hier den sichereren und in modernen Sprachen verbreiteten Ansatz der lokalen Variablen:

**Variablen in PHP sind zunächst lokal, also nur in ihrem jeweiligen Codesegment gültig.**

Eine Deklaration der Variablen benötigt PHP nicht, die Variable muss einfach verwendet werden.

PHP stellt das Schlüsselwort `global` zur Verfügung: mit diesem kann einer Variablen eine globale Natur zugewiesen werden, d.h. sie ist über ihren direkten Bereich gültig. Das Skipt `globalevariable.php` (Abbildung 11.14) zeigt ein typisches Beispiel: In der Methode `globalisierung` wird die Variable `$variable` als global definiert (Zeile 20), weshalb sie dann im gesamten PHP-Code und nicht nur in ihrer Methode gültig ist.

Einen wesentlichen Schritt weiter gehen die „Superglobals“, die wir in 11.3.5.10 kennenlernen werden.

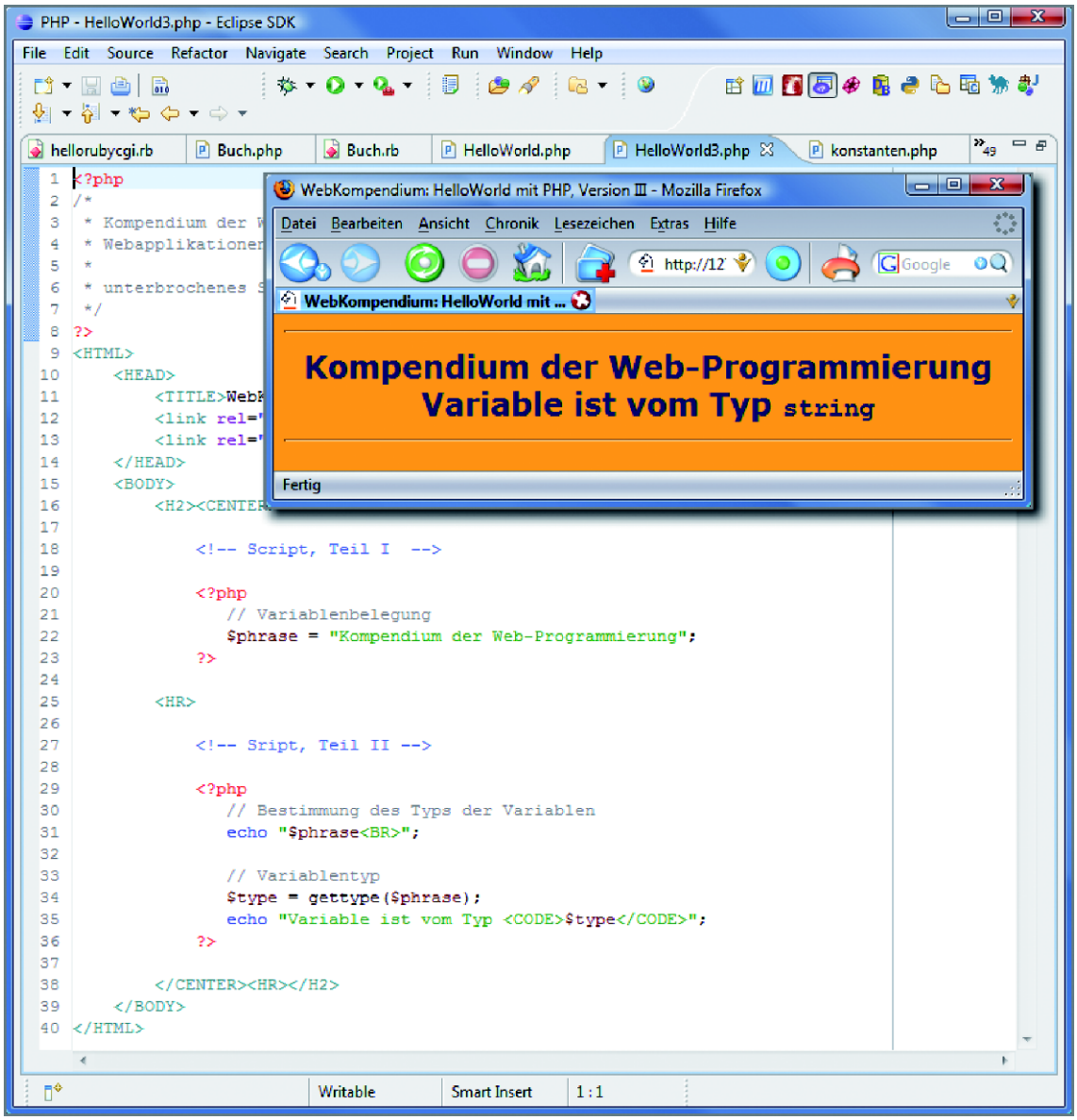



**Abbildung 11.13:**
Typabfrage in PHP (Script `HelloWorld3.php`)

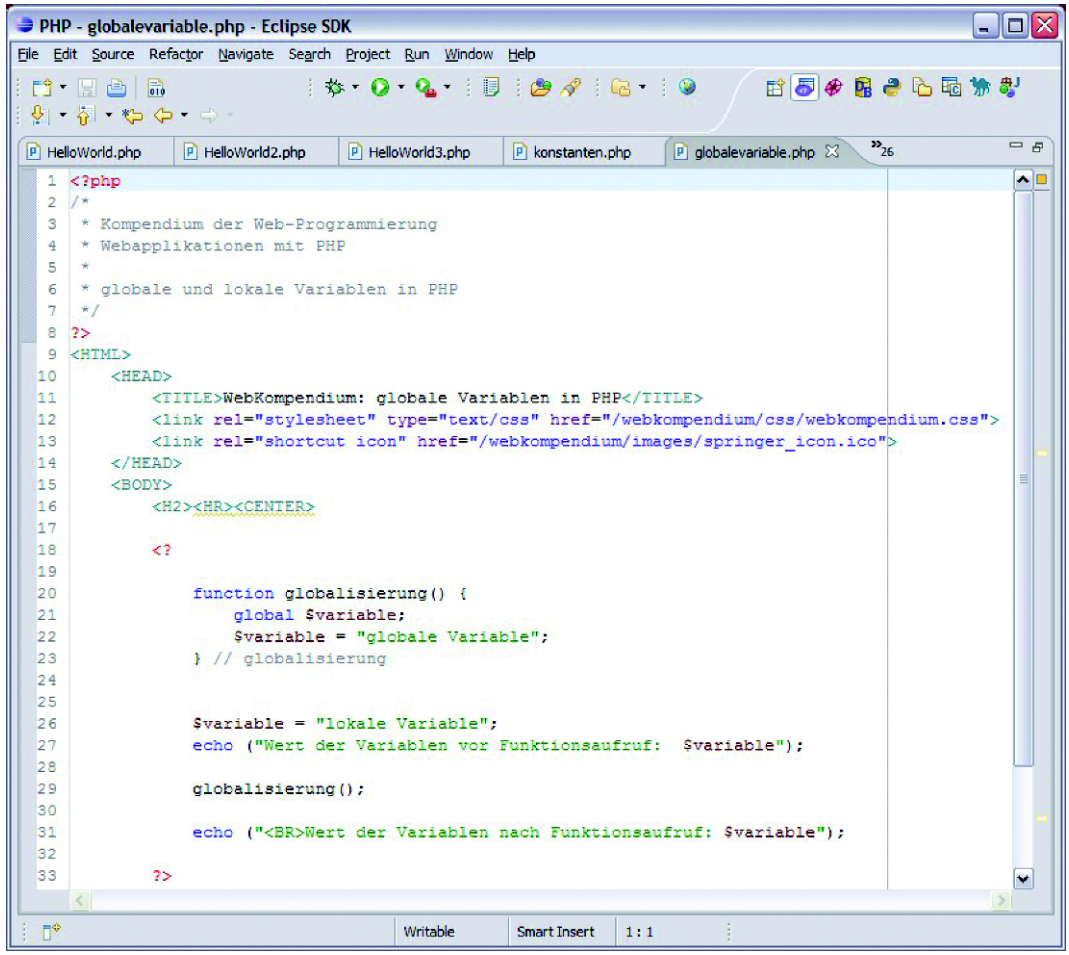



**Abbildung 11.14:** Konstanten in PHP (Script `globalevariable.php`)

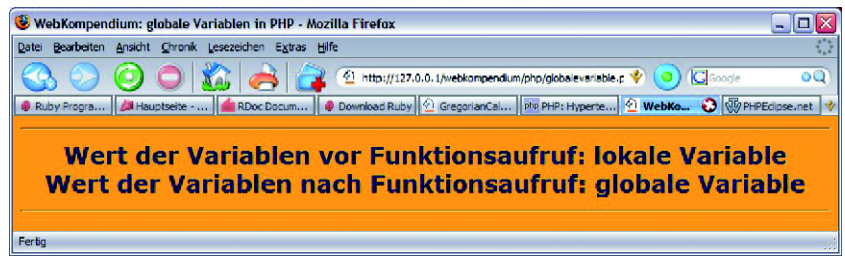



**Abbildung 11.15:** Ausführen von Script `globalevariable.php`

### 11.3.5.6 Konstanten in PHP

In PHP können *Konstanten* definiert werden – Variablen, denen nur einmalig ein Wert zugewiesen werden kann. Diese beginnen nicht mit einem führenden $, sondern werden mit einem Bezeichner – typischerweise in Großbuchstaben – mittels der Methode `define` erklärt:

```
define(PI,3.1415926);
```

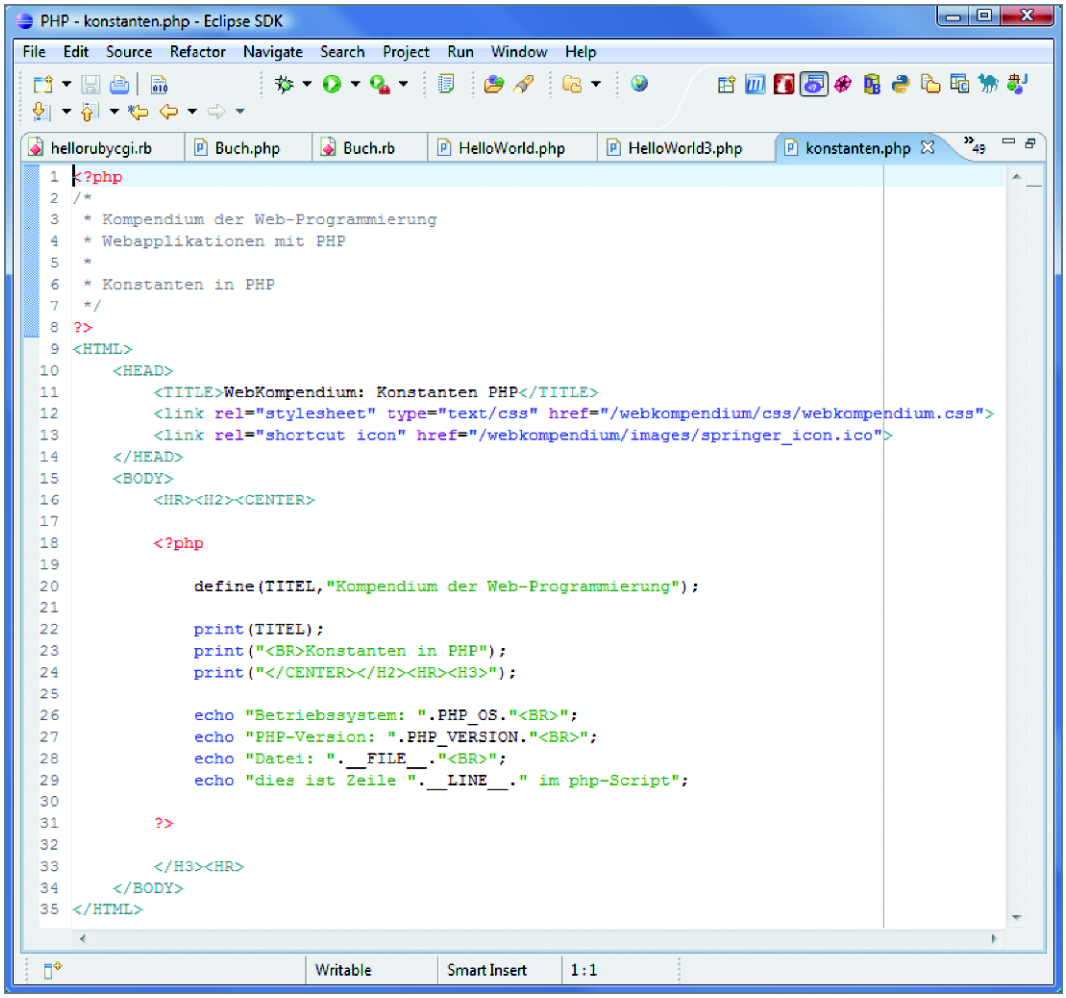
definiert die Konstante `PI`. Auf diese kann dann normal zugegriffen werden, etwa

```
echo PI;
```

Neben der Möglichkeit, selbst Konstanten zu bestimmen, verfügt PHP über etliche vordefinierte Konstanten. Hierzu zählen:

- `PHP_VERSION`: die Version des aktuellen PHP-Systems;
- `PHP_OS`: das Betriebssystem des Servers, der PHP betreibt;
- `__FILE__`: Dateiname des PHP-Scriptes;
- `__LINE__`: aktuelle Zeilennummer des PHP-Scriptes;
- `FALSE` und `TRUE`: boolesche Konstanten für Falsch und Wahr.

Das Script `konstanten.php` (Abbildung 11.16) zeigt ein exemplarisches Beispiel für Konstanten in PHP.




```
<?php
/*
 * Kompendium der Web-Programmierung
 * Webapplikationen mit PHP
 *
 * Konstanten in PHP
 */
?>
<HTML>
    <HEAD>
        <TITLE>WebKompendium: Konstanten PHP</TITLE>
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
    </HEAD>
    <BODY>
        <HR><H2><CENTER>

        <?php

            define(TITEL,"Kompendium der Web-Programmierung");

            print(TITEL);
            print("<BR>Konstanten in PHP");
            print("</CENTER></H2><HR><H3>");

            echo "Betriebssystem: ".PHP_OS."<BR>";
            echo "PHP-Version: ".PHP_VERSION."<BR>";
            echo "Datei: ".__FILE__."<BR>";
            echo "dies ist Zeile ".__LINE__." im php-Script";

        ?>

        </H3><HR>
    </BODY>
</HTML>
```

**Abbildung 11.16:**
Konstanten in PHP (Script `konstanten.php`)

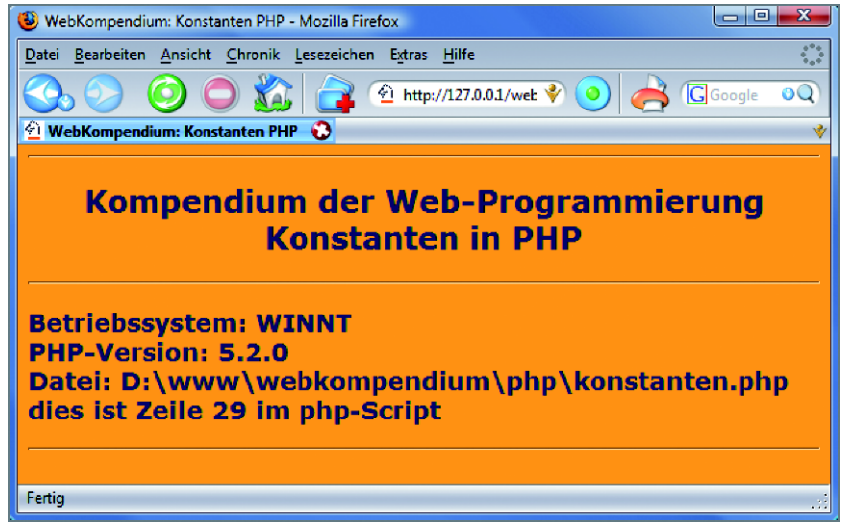



**Abbildung 11.17:** Ausführen von Script `konstanten.php`

#### 11.3.5.7 Arrays und Listen in PHP

Das klassische *Array* – eine Variable, welche mittels eines Index mehrere Werte des gleichen Typs abspeichern kann und dabei von vornherein eine feste Größe besitzt – kennt PHP genauso wie die anderen Scriptsprachen nicht, sondern wir verfügen in PHP direkt über die Datenstruktur der *linearen Liste:* eine Verkettung von einzelnen Werten – in PHP sogar von unterschiedlichem Typ –, die einfach erweitert und auch verkleinert werden kann.

**Das Java-Array ist in PHP nicht vorhanden, es gibt direkt Strukturen der Art** `java.util.Vector`

Vor diesem Hintergund ist im Folgenden - wie allgemein bei Scriptsprachen – der Begriff „Array“ als lineare, einfach verkettete Liste zu verstehen, auf deren Elemente direkt über einen Index, stets beginnend bei `0` für das erste Element, zugegriffen werden kann.

Arrays in PHP werden mit einem „normalen Bezeichner“ beginnend mit `$` adressiert (auch hier weicht PHP von Perl mit dem Sonderbezeichner `@` ab). Man kann ein Array dann einfach dadurch erzeugen, dass man auf einen Index zugreift:

```
$liste[0] = 42;
```

Wichtig für PHP ist in diesem Zusammenhang aber die Funktion `array`; über

```
$liste = array();
```

wird ein leeres Array angelegt, über

```
$liste = array(42);
```

das gleiche Array mit nur einem Element wie im ersten Fall. Ein gemischts Array kann etwa mittels

```
$liste = array(42,"WebKompendium",3.1415926);
```

erzeugt werden (mit den Indices 0, 1 und 2).

PHP bietet natürlich eine Menge nützlicher Funktionen für Arrays, etwa:

- `unset($liste)` löscht das Array `$liste`;
- `count($liste)` zählt die Anuahl der Elemente im Array `$liste`;
- `asort($liste)` sortiert die Values eines Arrays `$liste` aufsteigend;
- `rsort($liste)` sortiert die Values eines Arrays `$liste` absteigend.

**Listenoperationen für PHP-Arrays**

Arrays in PHP sind wie beschrieben direkt als lineare Liste implementiert. Wie in Perl stellt PHP alle vier Listen-Methoden bereit, allerdings mit abweichender Syntax:

- `array_push($liste,$wert)` legt einen Wert (oder auch gleich mehrere Werte) am Ende der Liste ab;
- `array_pop($liste)` entfernt das oberste Element der Liste, also das am Ende abgelegte Element;
- `array_unshift($liste,$wert)` fügt einen Wert (oder mehrere) am Anfang der Liste ein;
- `array_shift($liste)` entfernt einen Wert vom Anfang der Liste.

### 11.3.5.8 Assoziative Arrays in PHP

Genauso wie Perl (vgl. 10.3.5.3) bietet PHP nicht nur die Adressierung über einen bei 0 beginnenden Index, sondern über beliebige Keys; damit steht die wichtige Datenstruktur zur Verwaltung von Key-Value-Paaren zur Verfügung: assoziative Arrays, Hash Tables oder Dictionaries. Auch diese Strukturen können über die Methode `array` erzeugt werden.

Der direkte Weg zur Erzeugung eines solchen assoziativen Arrays besteht wieder in dem direkten Zugriff auf einen entsprechenden Key, etwa

```
$buecher['neu'] = "Kompendium der Web-Programmierung"';
```

Das gleiche kann eleganter über die `array`-Methode erreicht werden:

```
$buecher = array('neu' => "Kompendium der Web-Programmierung");
```

Das Script `liste.php` (Abbildung 11.18) zeigt ein Beispiel.

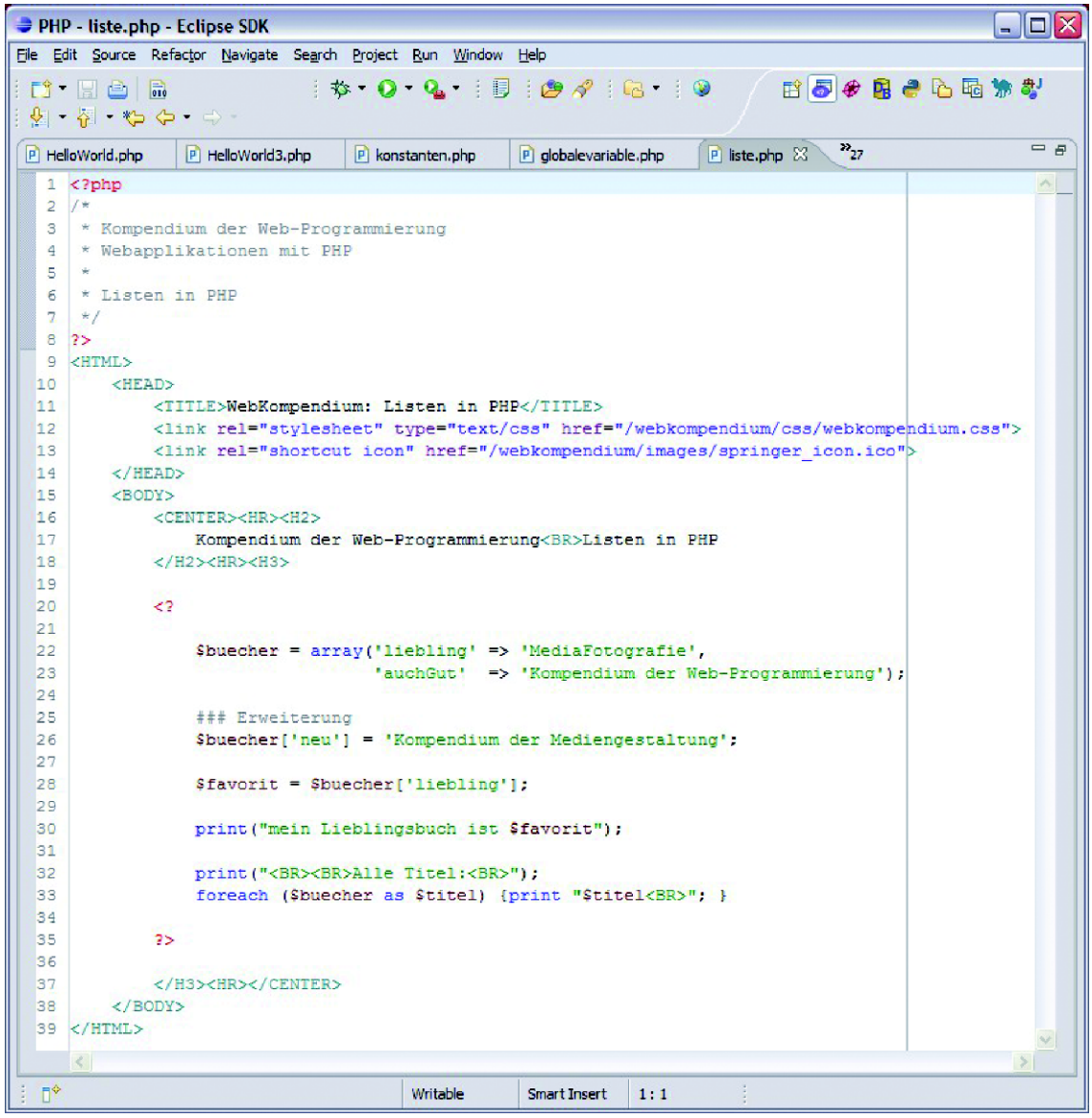



**Abbildung 11.18:**
Arrays in PHP (Script `liste.php`)

**Abbildung 11.19:**
Ausführen von Script `liste.php`

Neben den schon vorgestellten Funktionen für Arrays in PHP, die auch für assoziative Arrays anwendbar sind, ist zusätzlich noch

- `ksort($liste)` sortiert das Array `$liste` nach den Keys

zu erwähnen.

#### 11.3.5.9 Listenzeiger

Für den Umgang mit Arrays stellt PHP einen ausgesprochen nützlichen *Listenzeiger* zur Verfügung: Ein Zeiger, der nach Initialisierung auf das erste Element zeigt und die Liste durchschreiten kann. Auch hierfür steht eine ganze Reihe von PHP-Funktionen bereit:

- `current($array)` gibt dasjenige Element zurück, auf welches der Zeiger aktuell zeigt;
- `reset($array)` stellt den Zeiger auf das erste Element;
- `end($array)` stellt den Zeiger auf das letzte Element;
- `next($array)` rückt Zeiger eine Position weiter;
- `prev($array)` rückt Zeiger eine Position zurück.

#### 11.3.5.10 Superglobals in PHP

*Superglobals* sind eine Reihe von assoziativen Arrays, die in PHP vom System angelegt und automatisch als globale Variablen deklariert werden, also globale Gültigkeit innerhalb der PHP-Instanz und damit über ein einzelnes Script hinaus haben.

Wir werden mit diesen noch häufig zu tun haben, der Vollständigkeit halber sollen hier exemplarisch einige Superglobals vorgestellt werden:

- `$_POST` assoziatives Array, welches die Key-Value-Paare eines Formulars speichert, das mit der HTTP-Methode POST übertragen wurde;
- `$_SERVER` assoziatives Array, welches die Umgebungsvariablen des Webservers verwaltet;
- `$_SESSION` assoziatives Array für die Sessionvariablen.

Eine vollständige Übersicht über alle neun Superglobals ist in 11.4.6 aufgeführt.

#### 11.3.5.11 Operatoren in PHP

*Operatoren in PHP* verhalten sich „wie erwartet“; es gibt die üblichen Operatoren `+`, `-` und `*`, hinzu kommen `/` für geteilt (nicht div, da es keine Typisierung gibt) und `%` für mod im Ganzzahlbereich. Strings werden durch `.` zusammengefügt.

Entsprechend sind die Zuweisungsoperatoren erklärt (Tabelle 11.1).

| Operatoren | Bedeutung |
|---|---|
| `$x += $y` | `$x = $x + $y` |
| `$x -= $y` | `$x = $x - $y` |
| `$x *= $y` | `$x = $x * $y` |
| `$x /= $y` | `$x = $x / $y` |
| `$x %= $y` | `$x = $x` modulo `$y` |
| `$x++` | `$x = $x + 1` |
| `$x-` | `$x = $x - 1` |
| `$x ** $y` | `$x` hoch `$y` |

**Tabelle 11.1:** Operatoren in PHP

#### 11.3.6 Kontrollstrukturen in PHP

Die *Kontrollstrukturen in PHP* unterscheiden sich syntaktisch nicht wesentlich von Perl; wir sehen im Folgenden die typischen Fälle.

#### 11.3.6.1 Logik in PHP

Da Variablen in PHP nicht typisiert sind, ist der Datentyp „boolean“, wie ihn viele Programmiersprachen kennen, nicht vorgesehen – es gibt aber die Schlüsselwörter `FALSE` und `TRUE`. PHP geht hier den Weg, den viele Scriptsprachen verwenden: Variablen mit dem numerischen Wert `0` sind falsch, ebenso die leere Zeichenkette; zusätzlich können mit `===` Variablen ohne Typumwandlung verglichen werden:

```
$a = 42; $b = "42";
$a == $b   // TRUE
$a === $b  // FALSE
```

Vergleiche werden über die übliche Syntax `==`, `!=`, `<=` usw. durchgeführt.

Einzelne logische Werte können mit „nicht“, „oder“ und „und“ verknüpft werden; auch hierfür verwendet PHP die heute übliche Syntax.

#### 11.3.6.2 Der Block – die sequenzielle Komposition

PHP verwendet für die Bündelung von Anweisungen die viel verbreitete Schreibweise der geschweiften Klammer:

```
{
   Anweisung_1;
   Anweisung_2;
   ...
   Anweisung_n;
}
```

#### 11.3.6.3 Wenn-dann-sonst-Verzweigung

Der klassische Fall der wenn-dann-sonst-Verzweigung ist in PHP ebenfalls in der üblichen Form moderner Programmiersprachen – etwa Java – implementiert:

```
if ($boole)
   wahr_anweisung;
else
   falsch_anweisung;
```

Dabei ist der gesamte else-Block optional, kann also auch entfallen.

Komplexere Situationen können über die `elsif`-Klausel behandelt werden: Ein else-Block wird wiederum mit einer booleschen Abfrage verbunden, so können beliebig viele Blöcke verbunden werden:

```
if ($boole1)
   wahr_anweisung;
elseif ($boole2)
   falsch_wahr_anweisung;
elseif ($boole3)
```

```
    falsch_falsch_wahr_anweisung;
else
    falsch_anweisung;
```

Es wird jeweils maximal ein Anweisungsblock ausgeführt; ist der optionale else-Block vorhanden, wird immer genau ein Anweisungsblock ausgeführt.

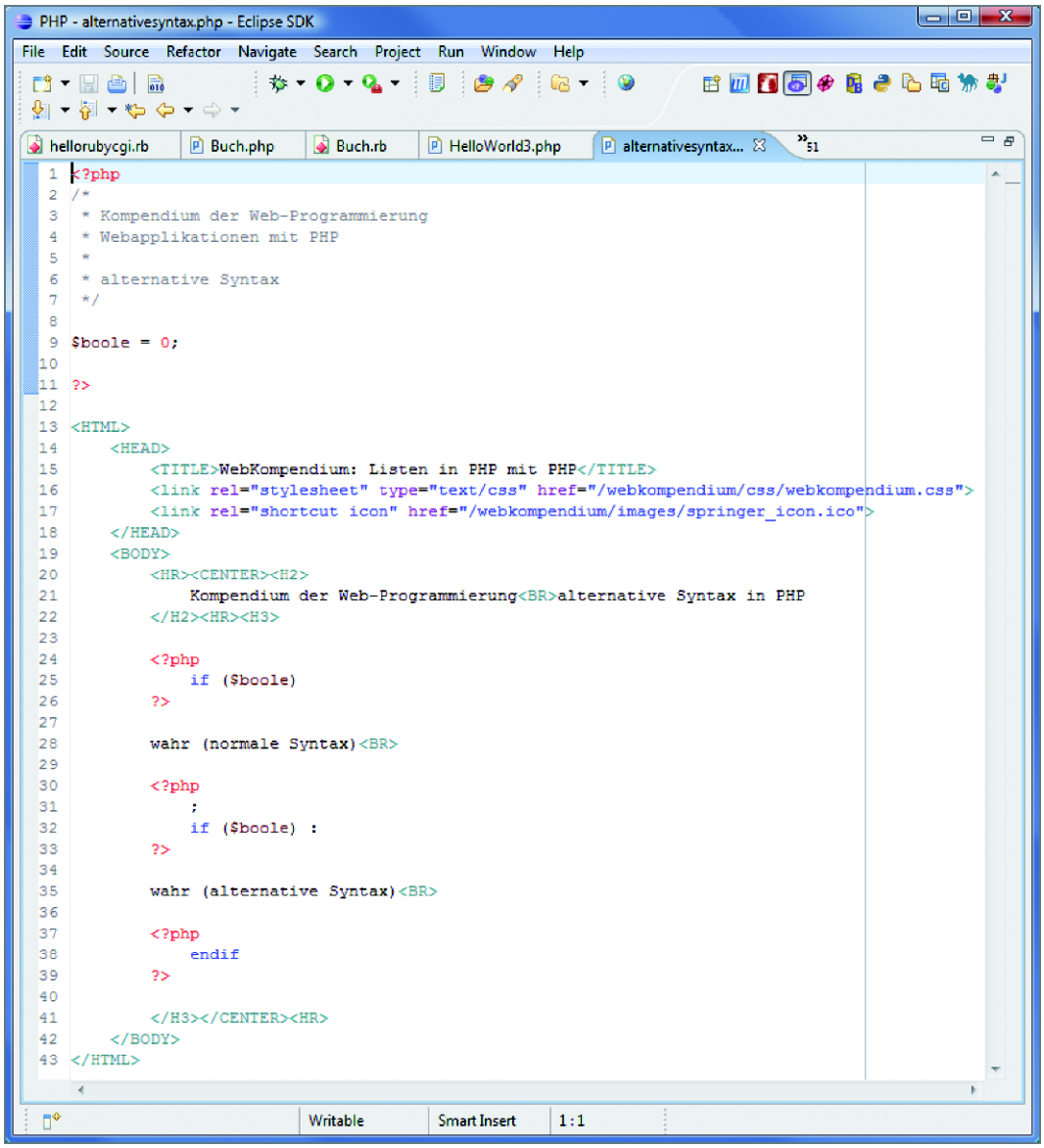



**Abbildung 11.20:**
Alternative Syntax in PHP (Script `alternativesyntax.php`)

### 11.3.6.4 Alternative Syntax

Die bisher vorgestellte Verzweigung in PHP wird wieder wie üblich innerhalb des PHP-Blocks ausgeführt; wird dieser Block unterbrochen durch HTML-Code, wird dieser stets zur Anzeige gebracht. Mittels der *alternativen Syntax* bietet PHP die Möglichkeit, auch die außerhalb des PHP-Codes befindliche HTML-Ausgabe zu steuern. Diese Syntax gibt es für die if-Verzweigung, aber auch für die Schleifen.

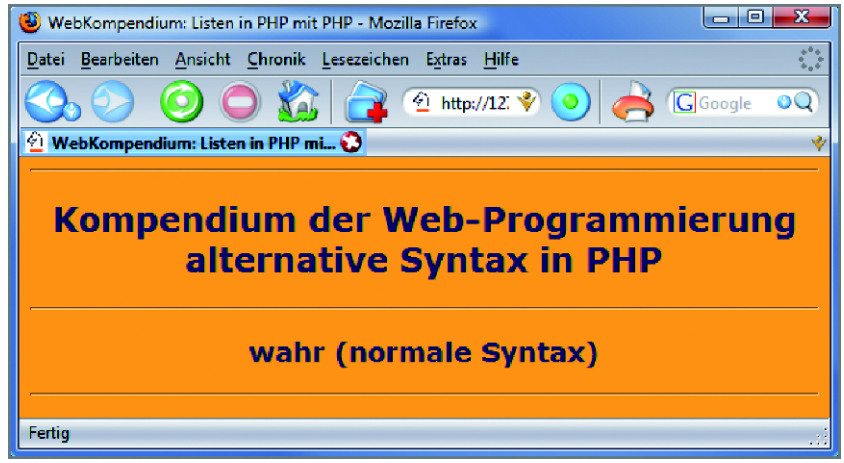



**Abbildung 11.21:** Alternative Syntax in PHP (Script `alternativesyntax.php`)

#### 11.3.6.5 Die Schalter-Struktur switch-case

Obwohl PHP über die elseif-Konstruktion verfügt, steht zusätzlich die klassische *Schalterstruktur mit switch-case* zur Verfügung. Hierbei wird in Abhängigkeit des Werts einer „Schaltervariablen“ zu verschiedenen Fällen (Labels) gesprungen.

Prinzipiell hat diese Kontrollstruktur folgende Syntax:

```
switch($schalter) {

   case 1: anweisung1;       break;
   case 2: anweisung2;       break;
   case 42: anweisung42;     break;
   default: sonst_anweisung; break;

}
```

Dies bedeutet, wenn die Variable `$schalter` den Wert `1` hat, wird zum Label „case 1“ gesprungen und die Anweisung `anweisung1` ausgeführt; anschließend wird wegen des `break`s die Kontrollstruktur verlassen.

Der `default`-Teil ist optional und wird nur dann ausgeführt, wenn kein case zutreffend war. seine Position ist beliebig, es darf aber nur einen `default`-Block geben.

Das letzte `break` kann auch entfallen.

**Diese Kontrollstruktur entspricht der Gleichen in Java.**

#### 11.3.6.6 Schleifen in PHP

PHP verfügt über vier Schleifen; drei davon (while, do-while und for) sind die bekannten Standardschleifen, die vierte (foreach) dient dem Abarbeiten von Listen.

### while-Schleife

Den Standardtyp der Schleifen in PHP bildet wie meistens die `while`-Schleife: solange das Argument nicht als boolescher Wert false interpretiert wird (vgl. 11.3.6.1) wird die Schleife ausgeführt.

Das Script `schleife.php` zeigt die Anwendung der Schleifen wieder an dem Beispiel der Beziehung (10.1) $\sum_{i=1}^{100} i$.

```
$n = 100; $i = 0; $summe = 0;

while($i <= $n) {
   $summe += $i;
   $i++;
}

print("Summation mit while-Summierung:  $summe<BR>");
```

**Listing 11.2:** Beispiel der while-Schleife in PHP (schleifen.php, Teil 1)

### do-while-Schleife

Während die while-Schleife bereits vor ihrem ersten Durchlauf die Bedingung überprüft (und so der Schleifenkörper nicht notwendig mindestens einmal durchlaufen wird – eine kopfgesteuerte Schleife) überprüft die `do-while`-Schleife die Bedingung erst nach dem Durchlauf (fußgesteuert), so dass der Schleifenkörper mindestens einmal durchlaufen wird.

```
$n = 100; $i = 0; $summe = 0;

do {
   $summe += $i;
   $i++;
} while($i <= $n);

print("Summation mit do-while-Summierung:  $summe<BR>");
```

**Listing 11.3:** Beispiel der do-while-Schleife in PHP (schleifen.php, Teil 2)

### for-Schleife

Nicht grundlegend neu, aber als häufiger Spezialfall von großer Bedeutung ist die `for`-Schleife; sie dient dazu, den „Wiederholungsfall“ einfacher zu behandeln.

Die Grundsyntax dieser Schleife in PHP ist gleich zu Perl und vielen anderen Sprachen:

```
for(init; bedingung; inkrement)  {
   schleifenanweisung;
}
```

Dabei ist `init` eine Anweisung, die einmalig vor dem ersten Schleifendurchlauf ausgeführt wird; die Schleife wird so lange ausgeführt, wie `bedingung` den logischen Wert true hat, und nach jedem Durchlauf wird die `inkrement`-Anweisung einmal ausgeführt, um die Schleifenvariable zu verändern.

Das Beispiel der Implementierung der Beziehung (10.1) mit diesem Schleifentyp in PHP lautet:

```
$summe = 0;
for ($i = 0; $i <= $n; $i++)
{
   $summe += $i;
}
print("Summation mit for-Summierung: $summe<BR>");
```

**Listing 11.4:** Beispiel der for-Schleife in PHP (schleifen.php, Teil 3)

**foreach-Schleife**

Ein spezieller Schleifentyp von PHP ist die `foreach`-Schleife. Hier wird eine Liste systematisch abgearbeitet und die Schleifenvariable nimmt bei jedem Durchlauf einen weiteren Werte der Liste an.

Die Grundsyntax dieser Kontrollstruktur lautet

```
foreach ($array as $value)
anweisung;
```

Hier nimmt die Variable `$value` alle Werte der Liste `$array` an.

Das Beispiel für Berechnung von (10.1) lautet mit dieser Schleife:

```
$liste = range(1,100); $summe = 0;
foreach ($liste as $i) {
   $summe += $i;
}
print("Summation mittels foreach ergibt: $summe <BR>");
```

**Listing 11.5:** Beispiel der foreach-Schleife in PHP (schleifen.php, Teil 4)

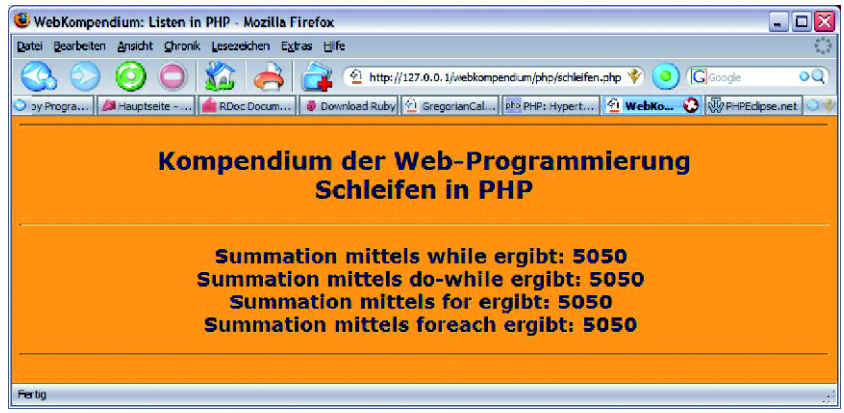



**Abbildung 11.22:** Ausführen von Script `schleifen.php`

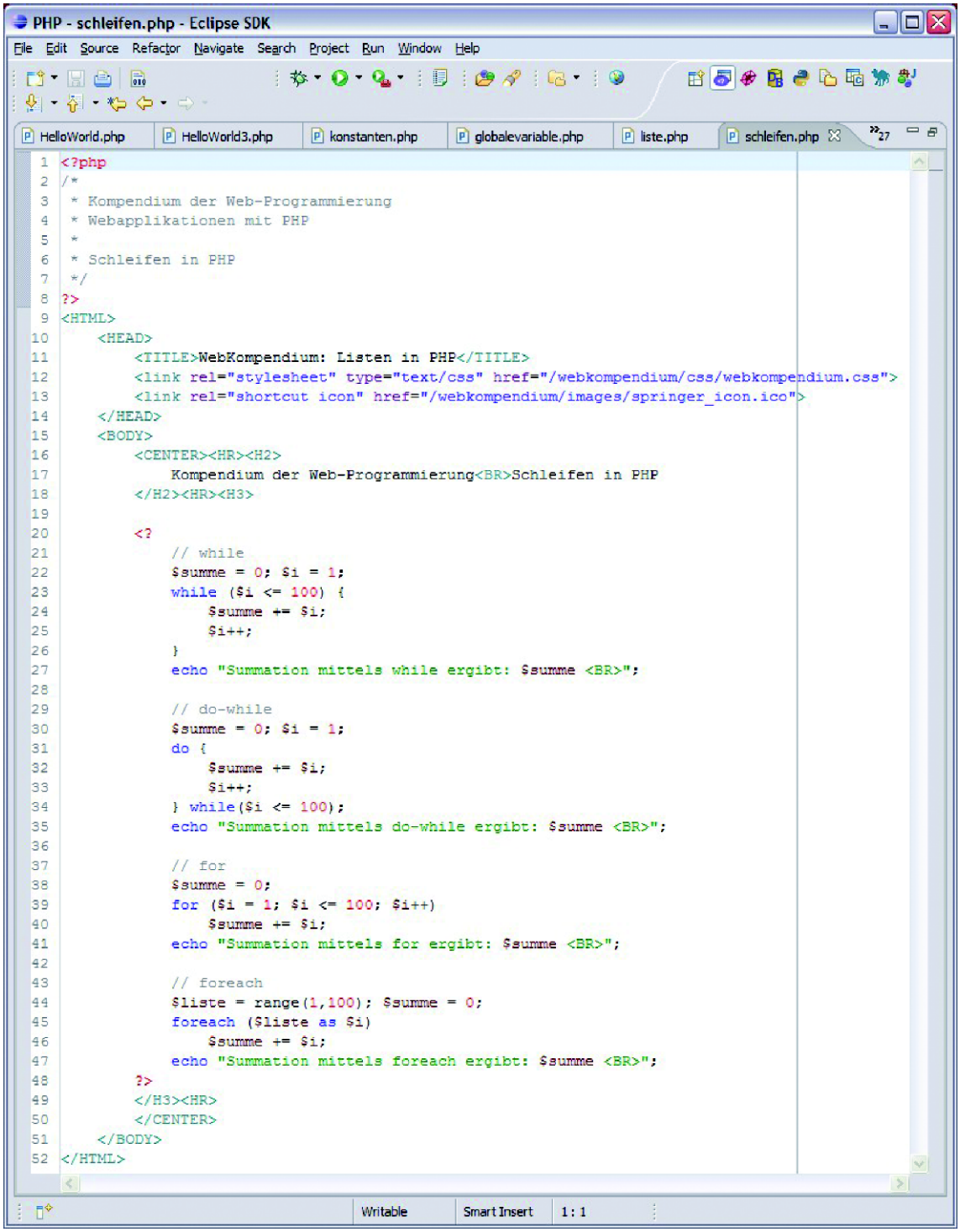



**Abbildung 11.23:** Schleifentypen in PHP (Script `schleifen.php`)

### Alternative Syntax und Schleifen

Natürlich können auch effizient Schleifen in der alternativen Syntax nach Abschnitt 11.3.6.4 verbunden werden. Hiermit können etwa HTML-Sequenzen einfach wiederholt ausgegeben werden:

```
<?php
   while (boole) :
?>
HTML-Code
<?php
   endwhile
?>
```

#### 11.3.6.7 Beenden von Kontrollstrukturen

Bereits im Zusammenhang mit der switch-case-Struktur haben wir gesehen, dass PHP Möglichkeiten zum *Beenden von Kontrollstrukturen* bereitstellt. Insgesamt bietet PHP hier zwei Anweisungen, die in ihrer Syntax und Wirkungsweise gleich zu Java sind:

- `break` beendet die Kontrollstruktur und fährt mit der ersten Anweisung nach dem Ende der Struktur fort.
- `continue` kann nur bei Schleifen verwendet werden; es veranlasst, dass der aktuelle Durchlauf beendet und mit dem nächsten Durchlauf begonnen wird. Für die for-Schleife bedeutet das, dass die `inkrement`-Anweisung ausgeführt und die Bedingung überprüft wird.

#### 11.3.6.8 Beenden von PHP

Natürlich kann PHP auch ganz verlassen, also die Ausführung von PHP beendet werden. Die PHP-Anweisung hierfür ist `exit`, wobei der Anweisung auch als Argument eine Zeichenkette übergeben werden kann, die dann ausgegeben wird.

Als Alias für exit steht die Methode `die` zur Verfügung, die genauso verwendet wird:

```
die("File not found");
```

### 11.3.7 Dateizugriff mit PHP

PHP erlaubt einen einfachen, an C angelehnten *Dateizugriff*. Zunächst sehen wir das Importieren von externem PHP-Code, danach den Zugriff auf das File-System.

#### Importieren von HTML-/PHP-Dateien

Für den Import von externem PHP- oder HTML-Code stellt PHP zwei (bzw. vier) Methoden zur Verfügung: `require` und `include`, die es auch in den Varianten `require_once` und `include_once` gibt.

- `require($file)` ersetzt den Methodenaufruf durch den Inhalt der Datei `$file`;
- `include($file)` lädt vergleichbar bei jedem Aufruf die Datei `$file`, ist aber auch in Schleifen einsetzbar.

Im Falle eines IO-Fehlers beim Zugriff auf die einzufügende Datei gibt `require` eine Fehlermeldung, während `include` nur eine Warnnmeldung ausgibt.

Diese zwei einfachen Funktionen sind von großer praktischer Bedeutung, da beliebige Dateien eingebunden werden, was konkret bedeutet: modularer HTML-Code durch Auslagerung typischer footer-/header-Dateien und Modularisierung der Programmierung durch Einbinden von externem PHP-Code.

**Mit `require` und `include` können HTML-Code und PHP-Programmierungen importiert werden – somit können modulare Web-Sites einfach aufgebaut werden.**

Für den ersten Fall – das Importieren von HTML-Code – lernen wir in Kapitel 14 noch eine einfache Alternative kennen, die ohne PHP auskommt.

Die hierbei verwendeten Dateinamen sind beliebig – es ist aber häufig üblich, die Dateiendung `.inc` für die „include-Dateien“ zu verwenden. Eine zentrale Ablage für diese zu importierenden Dateien ist häufig sinnvoll.

Die PHP-Konfigurationsdatei `php.ini` stellt einige sehr zentrale Parameter bereit. Im Abschnitt „Paths and Directories“ kann etwa der `include_path` definiert werden, in welchem PHP nach den zu importierenden Dateien sucht.

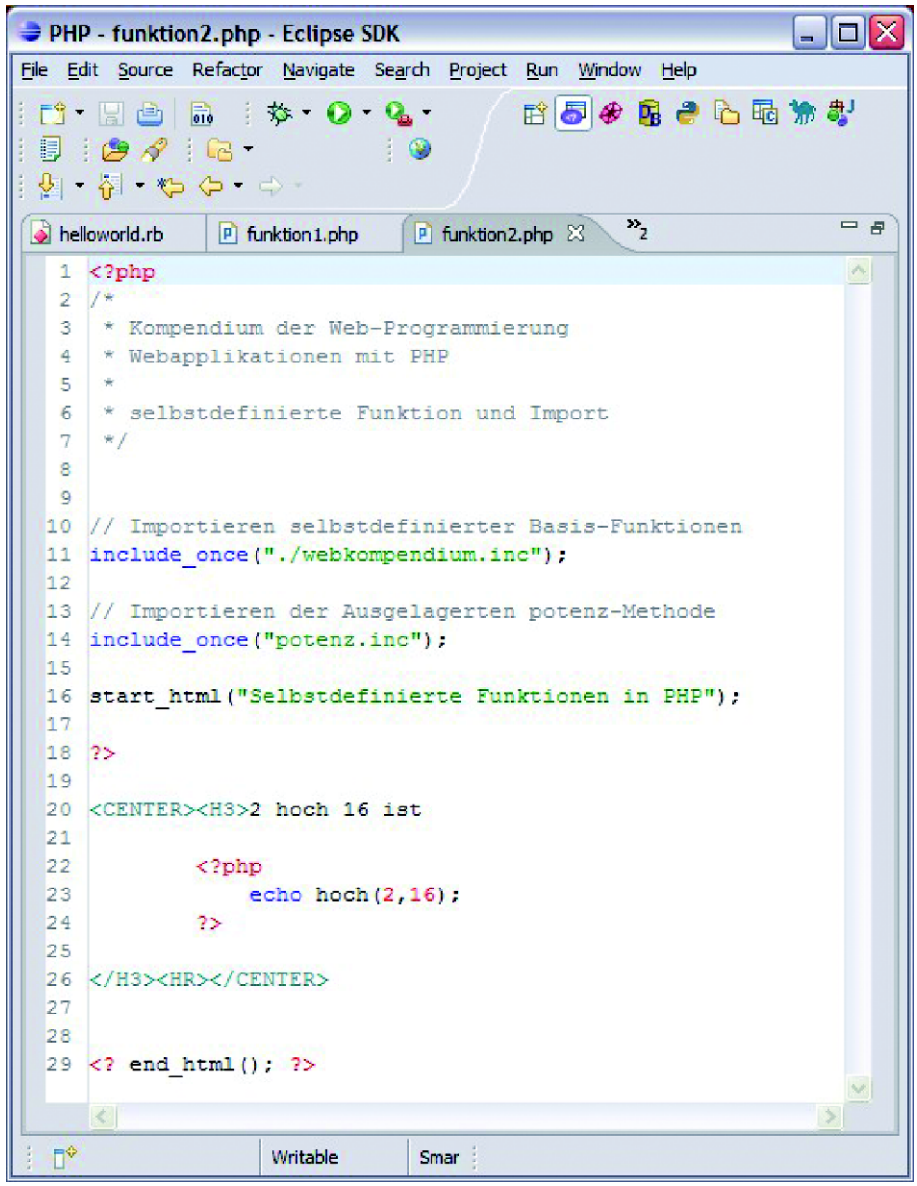



**Abbildung 11.24:**
Importieren von PHP-Dateien (Script `funktion2.php`) am Beispiel einer selbstdefinierten Funktion (`potenz.inc` in Abschnitt 11.4.1, Abbildung 11.27) und allgemeiner Methoden (`webkompendium.inc`, Abbildung 11.25)

```
<?php
/*
 * Kompendium der Web-Programmierung
 * Webapplikationen mit PHP
 *
 * include-Datei fuer allgemeine Setzungen
 */


function settings() {

   global $titel_wk;
   global $style;
   global $favicon;

   $titel_wk = "Kompendium der Web-Programmierung";
   $style    = "/webkompendium/css/webkompendium.css";
   $favicon  = "/webkompendium/images/springer_icon.ico";

}


/*
 * Ereugt HTML-Anfangsformatierung
 */
function start_html($titel="") {

   global $titel_wk;
   global $style;
   global $favicon;

   settings();

   print ("<HTML>
   <HEAD>
       <TITLE>$titel</TITLE>
       <link rel=\"stylesheet\" type=\"text/css\" href=\"$style\">
       <link rel=\"shortcut icon\" href=\"$favicon\">
   </HEAD>
   <BODY>
       <CENTER><HR><H2>$titel_wk</H2><H3>$titel</H3></CENTER><HR>");


}

/*
 * HTML-Abschluss
 */
function end_html() {
   print("</BODY></HTML>");
}
?>
```

**Abbildung 11.25:**
Auslagerung nützlicher, wiederverwertbarer PHP-Codes (Script `webkompendium.inc`)

### Importieren und Sicherheit

Das Importieren von „fremdem“ Code ist in jedem Fall nicht ungefährlich; im `safe_mode` (vgl. 11.2.6) ist das Importieren von PHP-Code deshalb eingeschränkt.

Ein früher häufig aufgetretenes Problem war das Nachladen von PHP-Code aus einer fremden Internet-Ressource, was bei ungeprüfter Verwendung dieser Methoden eintreten kann (bei Angabe einer URL anstelle von `$file`).

**Lesen und Schreiben von Dateien im File-System**

Der Zugriff auf das File-System ist in PHP sehr ähnlich zu C; ein IO-Handle verwaltet den Zugriff auf eine Datei, bei seiner Erzeugung muss festgelegt werden, ob die Datei gelesen oder geschrieben werden soll:

- `$fh = fopen($file,$mode)` erzeugt einen IO-Handle auf die Datei `$file` im Modus `$mode`; die folgenden Modi sind dabei möglich:
  - `r` zum Lesen;
  - `w` zum Schreiben;
  - `a` zum Anfügen;
  - `r+` zum Lesen und Schreiben;
  - `w+` zum Lesen und Schreiben.
- `$string = fgets($fh,$length)` liest eine Zeile über den IO-Handle `$fh`, jedoch maximal `$length`-viele Zeichen;
- `$i = fputs($fh,$string)` schreibt die Zeile `$string` zum IO-Handle `$fh`; der Rückgabewert `$i` kann als boolescher Wert interpretiert werden und gibt an, ob das Schreiben erfolgreich war;
- `$i = fclose($fh)` schließt den IO-Handle `$fh`.

## 11.4 Mehr PHP

### 11.4.1 Selbstdefinierte Methoden

Funktionen in PHP werden mittels des Schlüsselworts `function` definiert, wobei – natürlich nicht typisierte – Argumente in einer Argumentenliste angegeben werden:

```
function beispielFunktion($arg1,$arg2)  {
   ...
}
```

definiert eine Methode mit dem Bezeichner `beispielFunktion`, welche zwei Argumente erwartet; diese werden innerhalb der Methode als lokale Variablen mit den Bezeichnern `$arg1` und `$arg2` bereitgestellt; das Beispiel `funktion1.php` (Abbildung 11.26) zeigt eine einfache Anwendung `potenz.inc` (Abbildung 11.27) geht einen Schritt weiter und lagert die selbstdefinierte Methode aus.

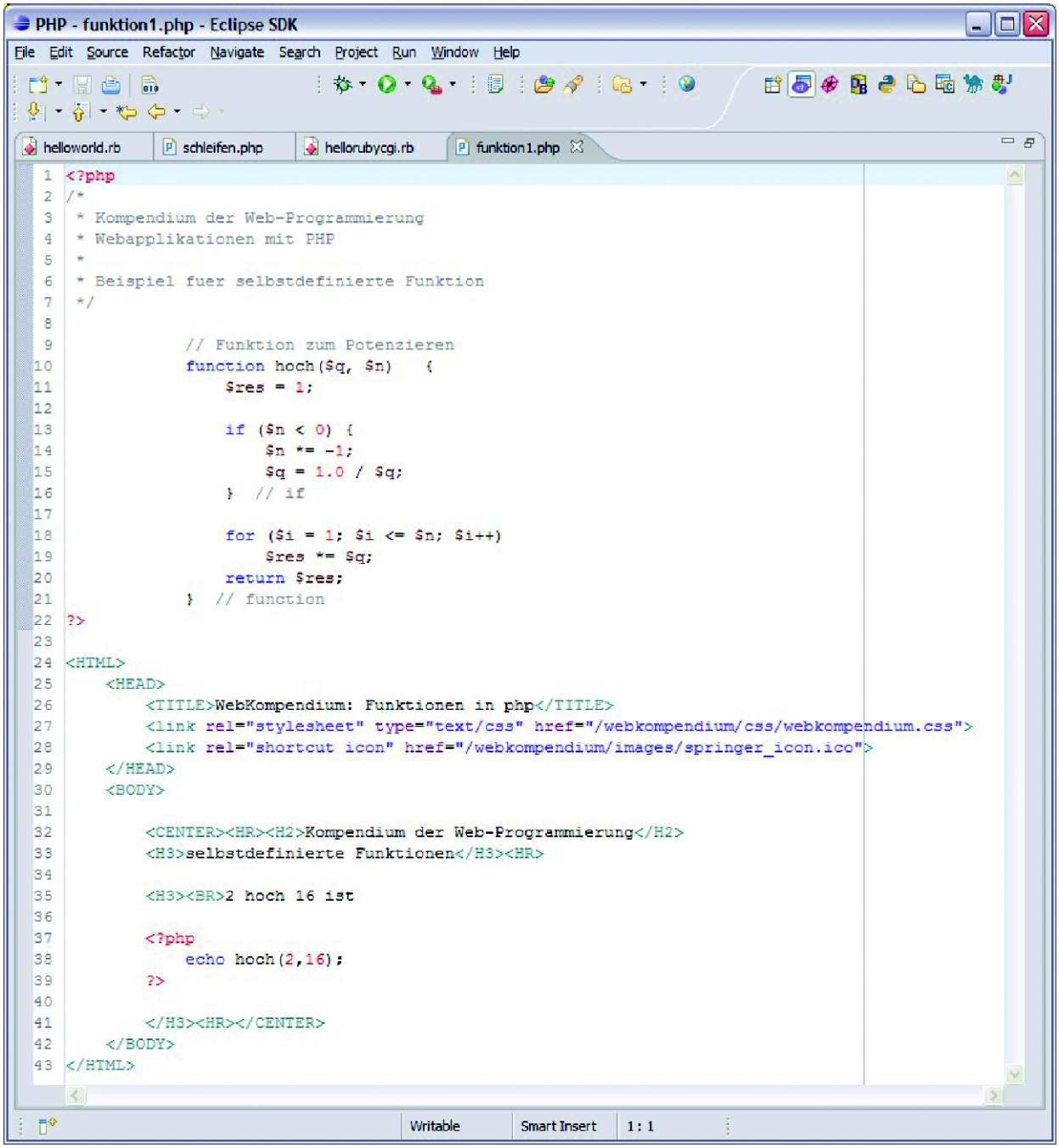



**Abbildung 11.26:**
Selbstdefinierte Methoden in PHP (Script `funktion1.php`)

Die so definierten Methoden können für Rückgabewerte das übliche `return` verwenden, wobei auch PHP-Arrays zurückgegeben werden können. Rekursion ist natürlich ebenfalls möglich, was das klassische Beispiel zeigt:

```
function fakultaet($n) {
  if ($n < 0) return 0;
  if ($n == 0) return 1;
  return $n * fakultaet($n - 1);
}
```

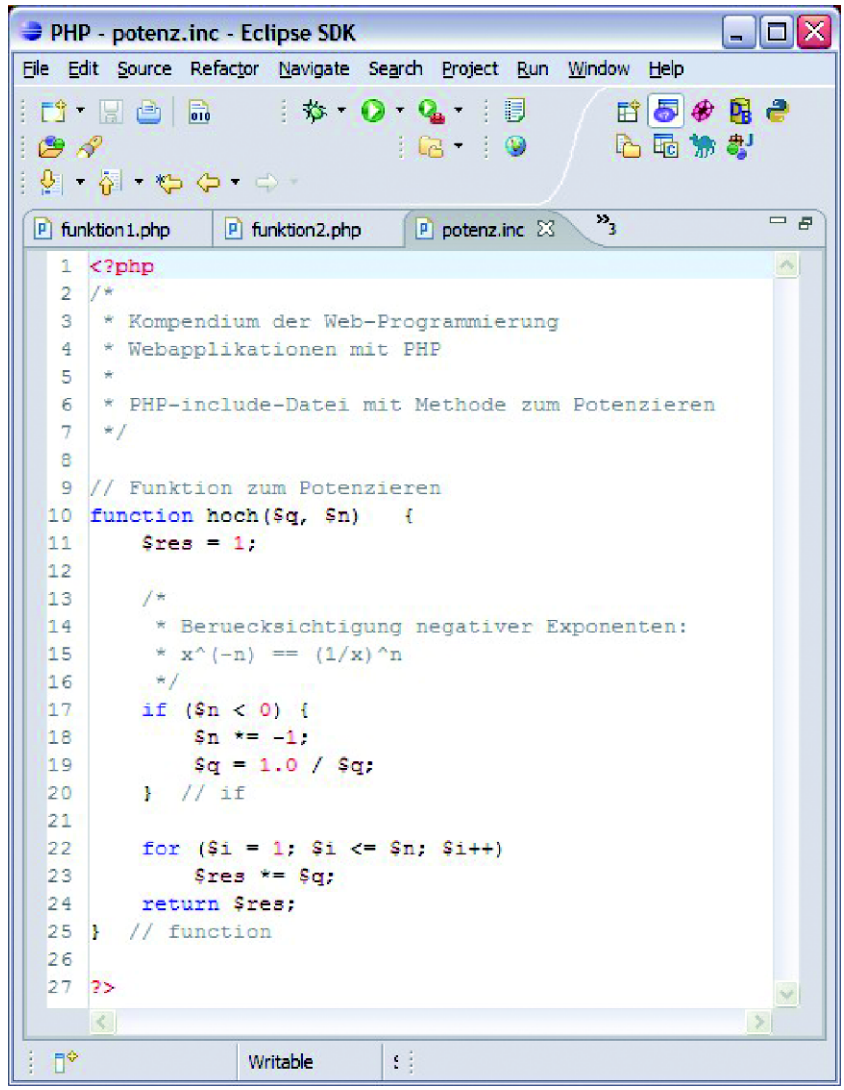



**Abbildung 11.27:** Selbstdefinierte Methoden in separater Datei für den Import (Script `potenz.inc`)

### Defaultwerte und überladen

Da die Scriptsprache PHP keine typisierten Variablen verwendet, ist ein klassisches Überladen nicht möglich, es gibt aber einen möglichen Ersatz: Defaultwerte. Hierbei werden Standardwerte für einzelne Parameter definiert, die dann verwendet werden, wenn beim Methodenaufruf weniger Werte übergeben werden:

```
function beispielFunktion($arg1,$arg2=42)  {
   ...
}
```

Nun kann die Methode mittels `beispielFunktion(1,2)`, aber auch mittels `beispielFunktion(1)` aufgerufen werden; im zweiten Fall hat dann die lokale Variable `$arg2` den Wert 42.

### 11.4.2 Variable Variablen und variable Methoden

Ein besonderes Sprachkonstrukt von PHP sind variable Variablen und variable Methoden. Gemeint ist damit, dass die Bezeichner für Variablen oder Methoden selbst in einer Variablen abgelegt werden:

```
$variable = "hello";
$$variable = "world";
// gleichwertig zu: $hello = ...;
// klarere Syntax:  ${$variable} = ...;

$methode = "beispielFunktion";
$methode(1);
// gleichwertig zu: beispielFunktion(1);
```

### 11.4.3 Referenzen in PHP

PHP bietet einen – sehr ähnlich zu C – Referenzierungsoperator `&`:

```
$ref = &$variable;
```

bedeutet, dass `$ref` eine Referenz auf den Speicherplatz der Variablen `$variable` ist.

Referenzen sind vielfältig von Nutzen, etwa kann mit ihnen die Natur von globalen Variablen erzeugt werden:

```
function erhoehe(&$arg)  {
   $arg++;
}

$x = 4711;
erhoehe($x);
echo($x);                  // Ausgabe 4712
```

#### Referenzen als Rückgabetyp

Methoden können auch Referenzen als Rückgabetyp haben; dann ist der Methodenbezeichner ebenfalls mit dem Referenzierungsoperator `&` zu versehen:

```
def &beispielFunktion(...) {
   ...
   return $eine_referenz;
}
```

### 11.4.4 HTML-Formulare und PHP

Eine noch fehlende Grundfunktionalität von PHP ist die einfache Art der Verarbeitung von *HTML-Formularen* (vgl. 2.3.3).

Betrachten wir ein Formular der Art

```
<FORM ACTION="script.php" METHOD="(GET|POST)">
<INPUT NAME="name">
</FORM>
```

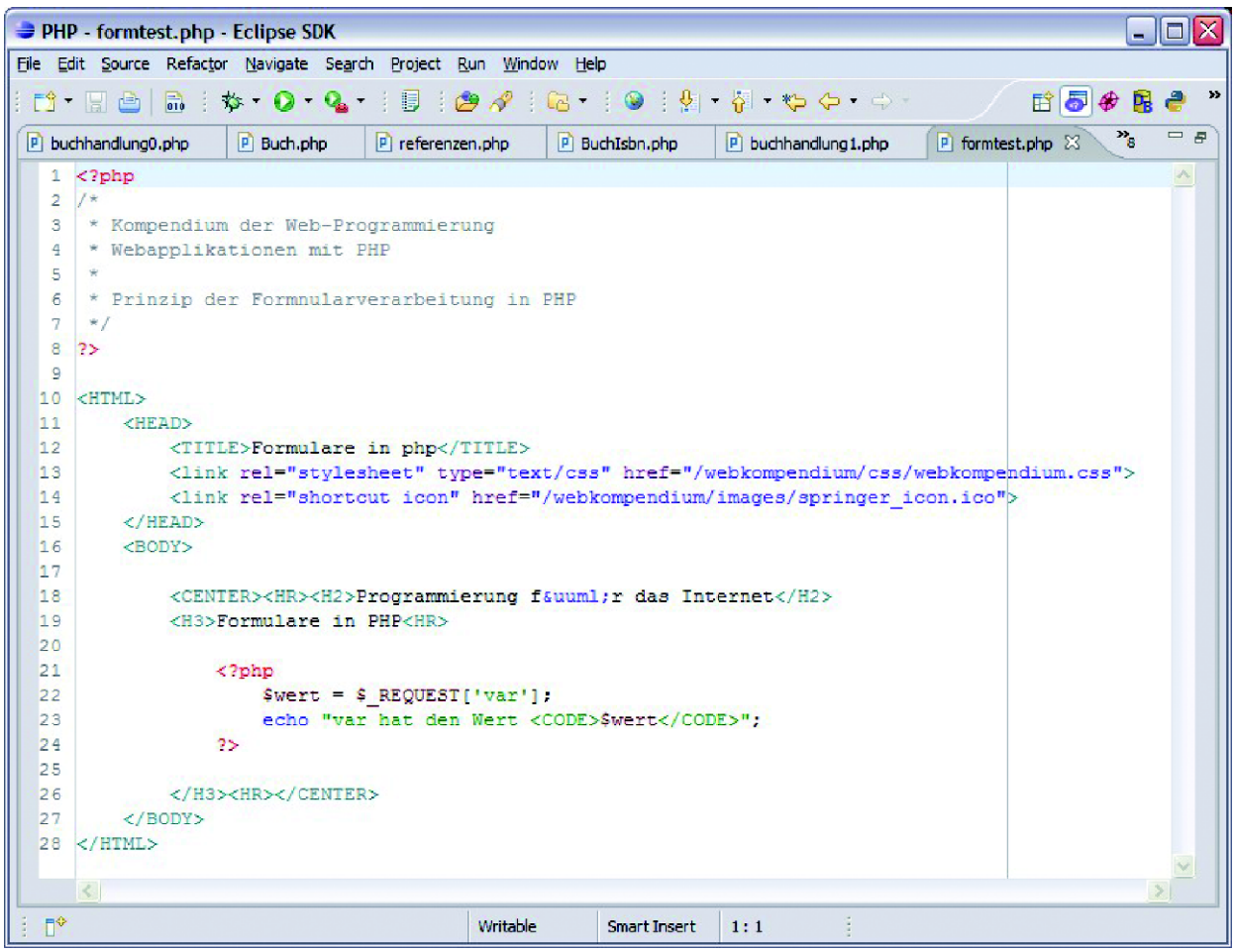



**Abbildung 11.28:** Auswertung von Formularen mit PHP (Script `formtest.php`)

Dieses Formular wird mit der im `METHOD`-Attribut angegebenen HTTP-Methode an das Script `script.php` übertragen und enthält ein Key-Value-Paar mit dem Key `name`.

Innerhalb des Scriptes `script.php` steht ein Superglobal bereit (vgl. 11.3.5.10), welches zum Key `name` den entsprechenden eingetragenen Wert als Value enthält. Insgesamt stehen hierfür drei Superglobals bereit:

- `$_POST` für die mit der HTTP-Methode POST übertragenen Formularfelder;
- `$_GET` für die mit der HTTP-Methode GET übertragenen Formularfelder;
- `$_REQUEST` für die mit einer beliebigen HTTP-Methode übertragenen Formularfelder.

Das Script `formtest.php` zeigt ein prinzipielles Beispiel; hier wird ein übertragenes Formular über das Superglobal `$_REQUEST` (Abbildung 11.28) ausge-

wertet, also unabhängig von der HTTP-Methode; ein typischer Aufruf dieses Scriptes zum Ausprobieren lautet dann etwa

```
http://<server>/formtest.php?var=WebKompendium
```

für die Übertragung des Key-Value-Paares (var,WebKompendium) mit der HTTP-Methode GET; Abbildung 11.29 zeigt das Ergebnis.

**Abbildung 11.29:** Ausführung von `formtest.php` mit der URL `http://<server>/formtest.php?var=WebKompendium`

Abbildung 11.30 zeigt ein etwas komplexeres Beispiel, welches an die selbstdefinierten Funktionen von Abschnitt 11.4.1 anknüpft, und ein mittels GET übertragenes Formular nach den Parametern `q` und `n` auswertet und damit $q^n$ berechnet.

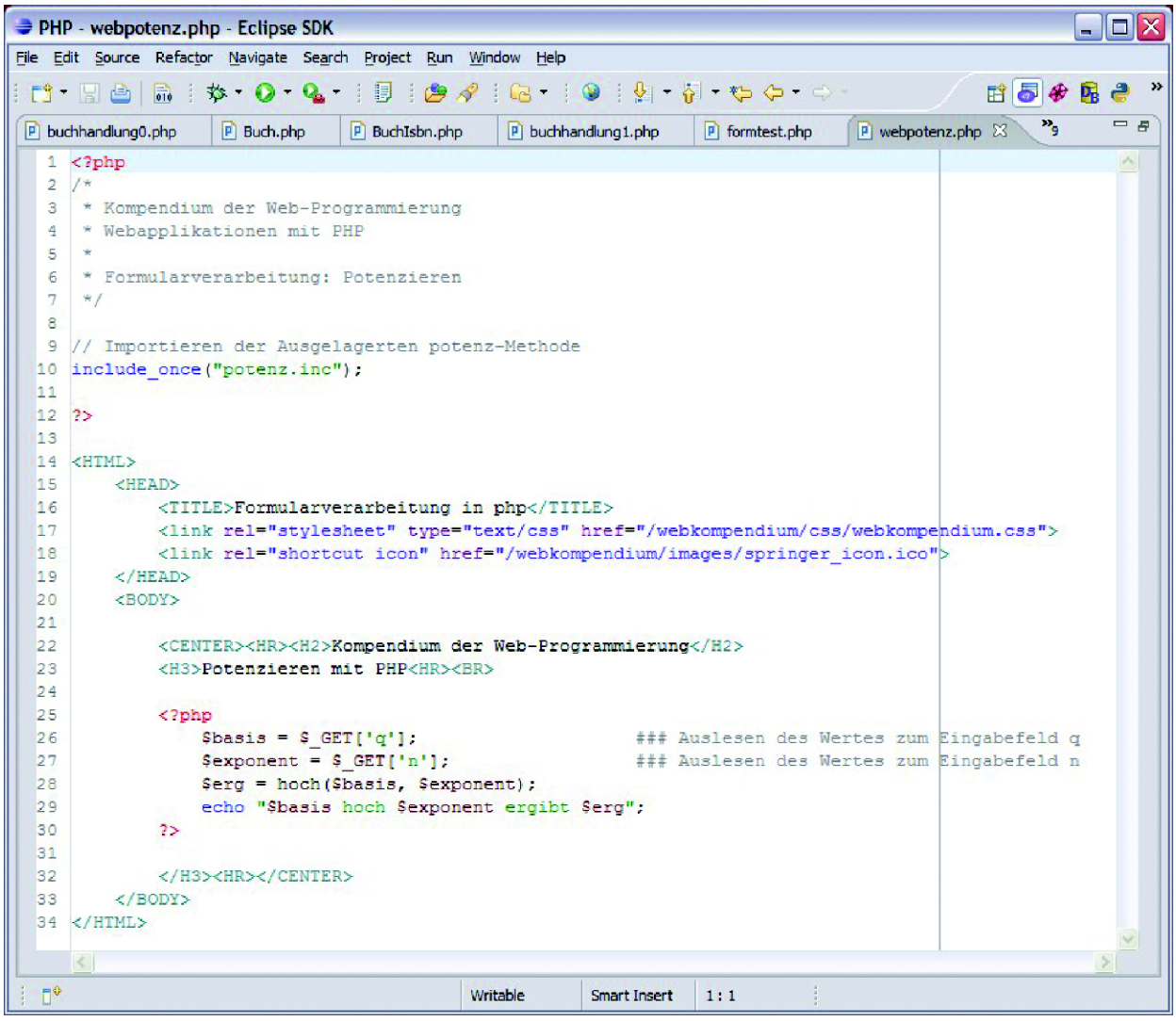
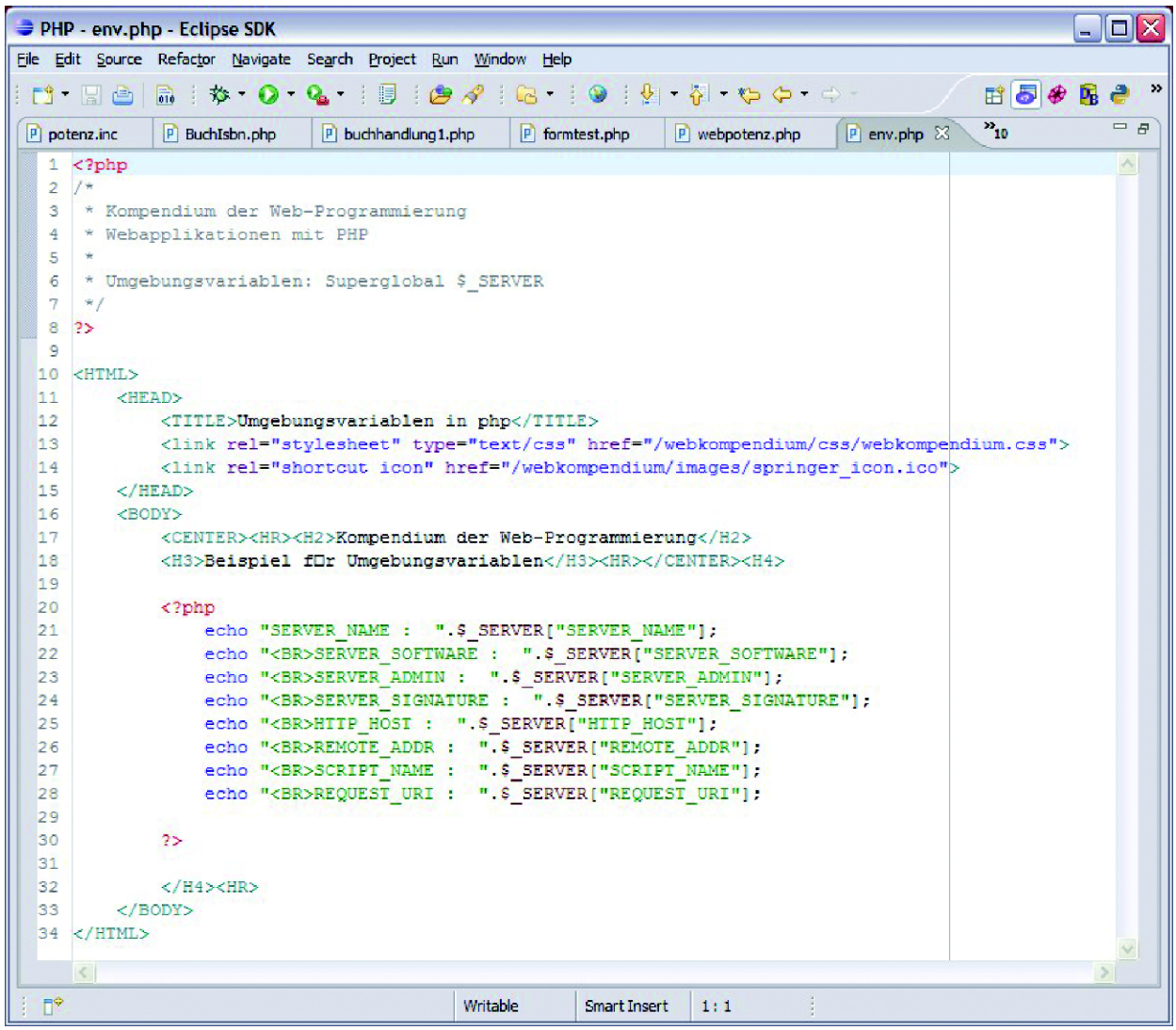



**Abbildung 11.30:** PHP-Script `webpotenz.php`)

### 11.4.5 Umgebungsvariablen und Superglobals

Die Bedeutung der Superglobals `$_POST`, `$_GET` und `$_REQUEST` haben wir gesehen; ein weiteres, wichtiges Superglobal ist `$_SERVER`, welches die Umgebungsvariablen – insbesondere die CGI-Variablen – speichert. Das Script `env.php` zeigt eine beispielhafte Anwendung.




```php
<?php
/*
 * Kompendium der Web-Programmierung
 * Webapplikationen mit PHP
 *
 * Umgebungsvariablen: Superglobal $_SERVER
 */
?>

<HTML>
    <HEAD>
        <TITLE>Umgebungsvariablen in php</TITLE>
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
    </HEAD>
    <BODY>
        <CENTER><HR><H2>Kompendium der Web-Programmierung</H2>
        <H3>Beispiel f□r Umgebungsvariablen</H3><HR></CENTER><H4>

        <?php
            echo "SERVER_NAME :  ".$_SERVER["SERVER_NAME"];
            echo "<BR>SERVER_SOFTWARE :  ".$_SERVER["SERVER_SOFTWARE"];
            echo "<BR>SERVER_ADMIN :  ".$_SERVER["SERVER_ADMIN"];
            echo "<BR>SERVER_SIGNATURE :  ".$_SERVER["SERVER_SIGNATURE"];
            echo "<BR>HTTP_HOST :  ".$_SERVER["HTTP_HOST"];
            echo "<BR>REMOTE_ADDR :  ".$_SERVER["REMOTE_ADDR"];
            echo "<BR>SCRIPT_NAME :  ".$_SERVER["SCRIPT_NAME"];
            echo "<BR>REQUEST_URI :  ".$_SERVER["REQUEST_URI"];

        ?>

        </H4><HR>
    </BODY>
</HTML>
```

**Abbildung 11.31:**
Anwendung des Superglobals `$_SERVER` (Script `env.php`)

**Abbildung 11.32:** Ausführen des Scripts `env.php` auf einem Unix-Server

### 11.4.6 Übersicht über alle Superglobals

PHP bietet insgesamt neun Superglobals:

- `$GLOBALS`: alle globalen Variablen;
- `$_SERVER`: Umgebungsvariablen des PHP-Servers;
- `$_GET`, `$_POST` und `$_REQUEST` für Formularverarbeitung;
- `$_COOKIE` Superglobal mit den Key-Value-Paaren der gespeicherten Cookies (vgl. 25);
- `$_SESSION` Superglobal mit den Sessionvariablen (vgl. 26.2);
- `$_FILES` für das File-Upload;
- `$_ENV` Umgebungsvariablen des Servers.

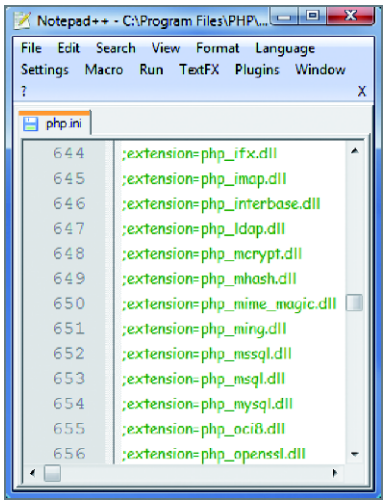



**Abbildung 11.33:** Konfiguration der PHP-Erweiterungen unter Windows

#### 11.4.6.1 Die Bedeutung von `register_globals` und die Superglobals

Die Formularverarbeitung mit PHP kann sogar noch einfacher geschehen: Wird in der `php.ini`-Konfigurationsdatei der Schalter `register_globals` auf „on“ gesetzt – bis zur Version 4.2 von PHP war dies sogar die Standardsetzung – dann steht zu jedem übertragenen Key-Value-Paar eine Variable bereit, welche als Bezeichner den Key und als Wert den Value des übertragenen Datenwertes unabhängig von der verwendeten HTTP-Methode hat. Am Beispiel des skizzierten Formulars bedeutet das, dass eine Variable `$name` angelegt wird, die mit dem übertragenen Wert belegt wird.

Insbesondere aus Sicherheitsgründen ist aber inzwischen die Vorbelegung des Schalters `register_globals` „off“, weshalb gleich die Superglobals verwendet werden sollten, damit die PHP-Programmierung auch übertragbar auf andere Server ist.

Die PHP-Originaldokumentation führt dazu aus: „In PHP 4.2.0 and later, the default value for the PHP directive `register_globals` is off. This is a major change in PHP. Having `register_globals` off affects the set of predefined variables available in the global scope. For example, to get `DOCUMENT_ROOT` you'll use `$_SERVER['DOCUMENT_ROOT']` instead of `$DOCUMENT_ROOT`, or `$_GET['id']` from the `URL` `http://www.example.com/test.php?id=3` instead of `$id`, or `$_ENV['HOME']` instead of `$HOME`."

## 11.5 Datenbankzugriff mit PHP

### 11.5.1 PHP und Datenbanken – ein Erfolg mit kleinen Umwegen

Eine der ganz großen Stärken von PHP liegt in der vergleichsweise einfachen Datenbankanbindung. Dabei ist diese insgesamt durchaus komplex: Der „direkte Weg" der frühen PHP-Versionen ist einfach, aber unflexibel – eine vom DBMS unabhängige, universelle Syntax, wie es etwa mittels Perl-DBI (vgl. 10.7) ermöglicht wird, bietet PHP zunächst nicht.

Auf dieser „nativen" Datenbankanbindung bauen dann aber mehrere ähnliche, konkurrierende Konzepte auf, die eine universellere Lösung ermöglichen. Diese werden in diesem Buch etwas später vorgestellt (etwa 20.5 und 11.8). In diesem Abschnitt wird die ursprüngliche (und bis heute am meisten verwendete) direkte DB-Unterstützung von PHP vorgestellt, wie sie seit der Version 3 in PHP genutzt wird.

**PHP bietet mehrere Möglichkeiten der Datenbankanbindung; hier wird zunächst die ursprüngliche, DBMS-abhängige vorgestellt. Weiterführende Konzepte sind in 20.5 und 11.8 zu finden.**

#### 11.5.1.1 Unterstützte Datenbankmanagementsysteme

PHP ist in der Lage, eine große Vielzahl von DBMS zu unterstützen, allerdings ist hier einiges zu beachten.

Insbesondere auf Unix-Systemen ist bei der Konfiguration des PHP-Moduls, also noch vor dem Compilieren, zu entscheiden, welche DBMS unterstützt werden sollen, da die entsprechenden Treiber statisch in das Modul gelinkt werden, also direkt mit compiliert werden müssen. Um dies zu ermöglichen, werden in vielen Fällen (praktisch alle außer MySQL) zusätzliche Libraries des DBMS benötigt, weshalb bei der Konfiguration anzugeben ist, wo diese zu finden sind. Ein typischer Fehler, der das Compilieren scheitern läßt, sind fehlende DBMS-Pakete. So wird etwa für die Einbindung der Informix-Unterstützung auf dem Webserver auch das Informix Client-Paket benötigt, damit dieser eine Netzwerkverbindung zu einem Informix-Server aufbauen kann (u.a. aus lizenzrechtlichen Problemen sind diese Pakete nicht bereits im PHP-Paket enthalten).

**Beim Konfigurieren `configure` ist bereits zu entscheiden, welche DBMS unterstützt werden sollen; dabei ist meistens auch die Information notwendig, wo diese Datenbanken installiert sind (Pfadangabe), damit die entsprechenden Libraries eingebunden werden können.**

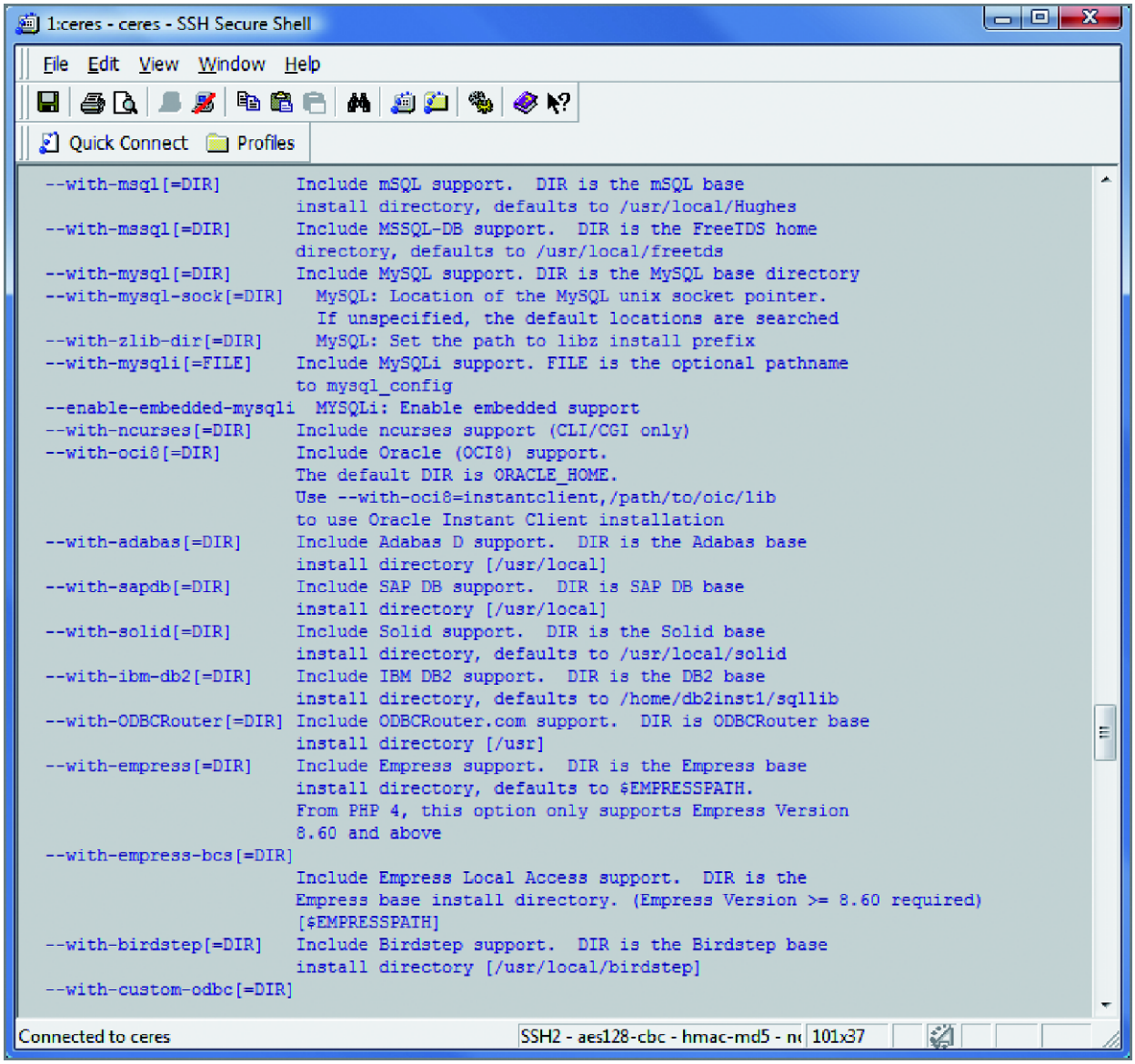



**Abbildung 11.34:** Kleiner Auszug von `configure -help`

Im Falle von Windows ist dies typischerweise einfacher, da hier die Datenbanktreiber dynamisch eingebunden werden können; die Konfiguration erfolgt in der Datei `php.ini` (Abbildung 11.33.

#### 11.5.1.2 Das Standard-DBMS

PHP wurde in der Version 4 stets zuerst mit dem DBMS MySQL verbunden; die Default-Installation war für MySQL (vgl. 3.3.1) ausgelegt und damit direkt einsetzbar (unter Windows wird auch häufig die universelle ODBC-Schnittstelle unterstützt).

Diese „Traumehe“ hat mit der Version 5 etwas zu leiden begonnen; die aktuelle Version von PHP ist für ein einfacheres DBMS vorkonfiguriert: SQLite (vgl. 3.3.3).

Unabhängig von der Voreinstellung kann aber durch entsprechende Konfiguration PHP wie schon beschrieben mit fast allen DBMS betrieben werden.

### 11.5.1.3 Konfiguration

Die Konfigurationsdatei `php.ini` enthält eine Vielzahl wichtiger Konfigurationsparameter für jedes DBMS; dazu zählt etwa der verwendete Standardport und der Datenbankserver; ein typisches Beispiel für den MySQL-Treiber zeigt Abbildung 11.35.

Eine Übersicht über die verwendeten Konfigurationen gibt am einfachsten die Methode `phpinfo` (vgl. 11.2.4) wieder.

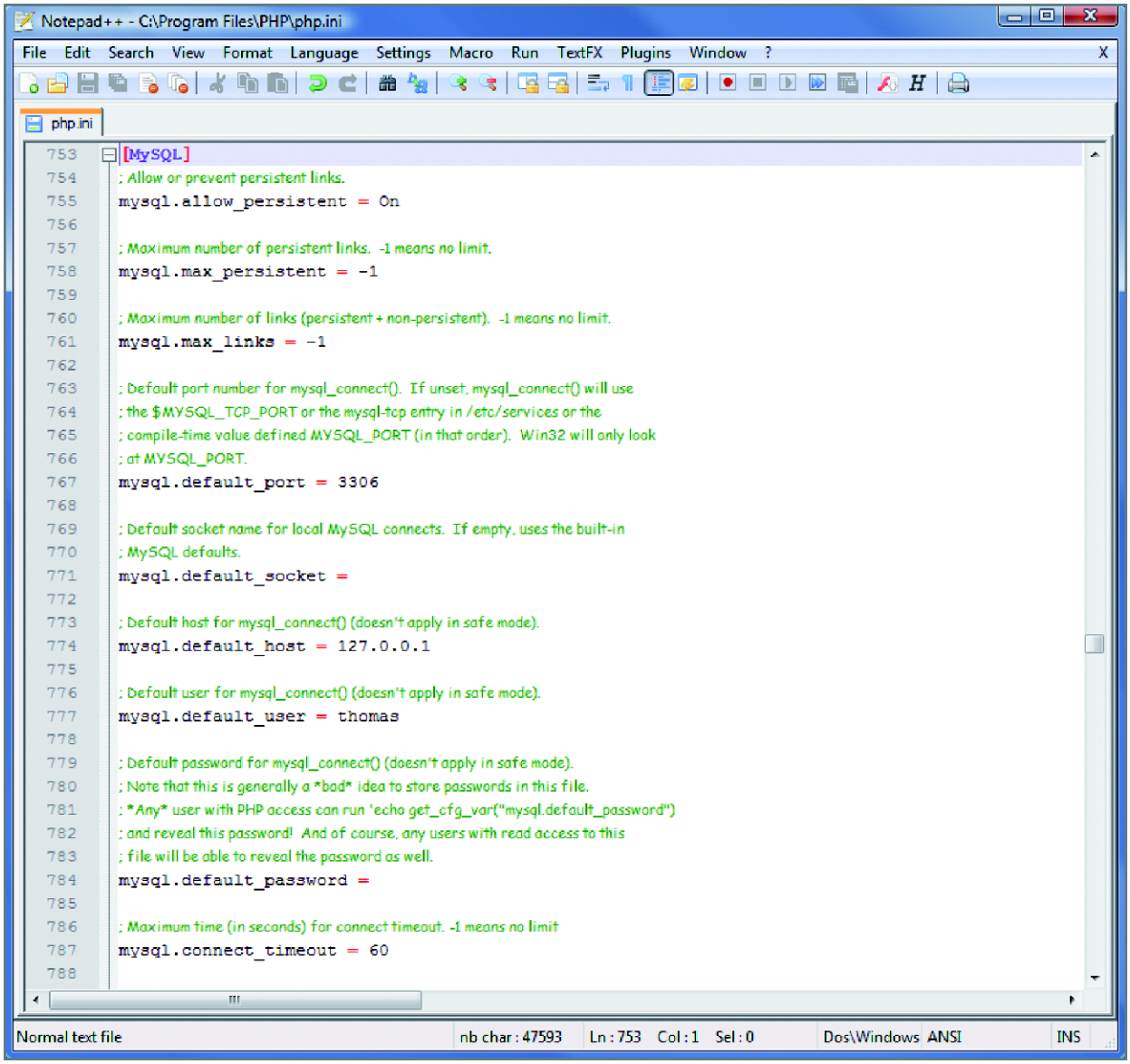



**Abbildung 11.35:** Konfiguration der Datenbankanbindung für MySQL in der `php.ini`

### 11.5.2 Verbindungsaufbau zum DBMS

Der erste Schritt für die Verwendung eines DBMS ist der Verbindungsaufbau; für den klassischen MySQL-Treiber lautet die Syntax hierfür

```
$dbh = mysql_connect($host:$port,$user,$password);
```

Die Parameter `$port`, `$user` und `$password` können entfallen, dann werden die in der Konfigurationsdatei `php.ini` festgelegten Werte verwendet.

Der Rückgabewert der connect-Methode ist 0 – also im booleschen Sinne false –, falls der Verbindungsaufbau fehlschlägt; im Erfolgsfall ist der Rückgabewert > 0 und kennzeichnet die jeweilige Verbindung, da PHP mit mehreren parallel arbeiten kann.

PHP bietet auch die Möglichkeit von persistenten, also dauerhaften Datenbankverbindungen. Voraussetzung hierfür ist natürlich der Einsatz von PHP als Servermodul und nicht als CGI, da als CGI der Prozess nach jedem Request sowieso beendet wird, sowie die entsprechende Konfiguration in der `php.ini`. Der Vorteil der persistenten Verbindungen ist eine größere Performance. Schwierig ist im Umgang mit diesen Verbindungen die Tatsache, dass alle Verbindungen in der Kennung der ersten Verbindung – also mit deren Rechten im DBMS – ablaufen. Insgesamt haben deshalb persistente Datenbankverbindungen heute keine Bedeutung.

#### Auswahl der Datenbank

Nachdem die Verbindung zum DBMS aufgebaut ist, ist eine spezielle Datenbank auszuwählen. Geschieht dies nicht, muss bei jeder Datenbankanweisung jeweils die Datenbank als Parameter übergeben werden, was zu sehr umständlichem Code führt. Diese Auswahl geschieht mittels

```
mysql_select_db($dbname,$dbh);
```

Entfällt der zweite Parameter `$dbh`, so wird automatisch die letzte verwendete DBMS-Verbindung benutzt.

### 11.5.3 Datenbank-Abfragen

SQL-Operationen fallen wieder wie bei Perl in zwei prinzipielle Klassen: `SELECT`-Abfragen und Datenbank-Operationen, die die Datenbank verändern wie `INSERT` und `UPDATE`.

Für `SELECT`-Abfragen liegt die gewohnte Situation vor: Die Abfrage ist in SQL zu formulieren, anschließend wird sie durchgeführt und liefert ein zeilenweise zu verarbeitendes Resultset. Die Durchführung der SQL-Abfrage `$sql` erfolgt mit der Methode

```
$sth = mysql_query($sql,$dbh);
```

wobei der zweite Parameter `$dbh` wieder entfallen kann, wenn die zuletzt aktive DBMS-Verbindung (mit ausgewählter Datenbank) verwendet wird.

Das somit erhaltene Resultset ist wieder zeilenweise zu verarbeiten: mittels der Methode `mysql_fetch_array` kann eine Trefferzeile gewonnen werden.

```
$zeile = mysql_fect_array($sth,$typ);
```

liefert eine Trefferzeile für die Abfrage `$sth`, wobei für den Typ `$typ` drei Möglichkeiten bestehen: Für die Abfrage „`SELECT titel, verlag, jahr FROM buecher;`“

- `MYSQL_ASSOC` liefert ein assoziatives Array, also eine Sammlung von Key-Value-Paaren mit, in diesem Beispiel den Keys `titel`, `verlag` und `jahr`;
- `MYSQL_NUM` liefert ein normales, numerisch indiziertes Array mit den Indices 0 (für `titel`), 1 (für `verlag`) und 2 (für `jahr`);
- `MYSQL_BOTH` für eine Mischform, die beide Möglichkeiten bietet.

Das Beispiel `buecher_1.php` ist exemplarisch eine einfache SQL-Select-Abfrage über die Datenbank nach dem Beispiel von Kapitel 7.

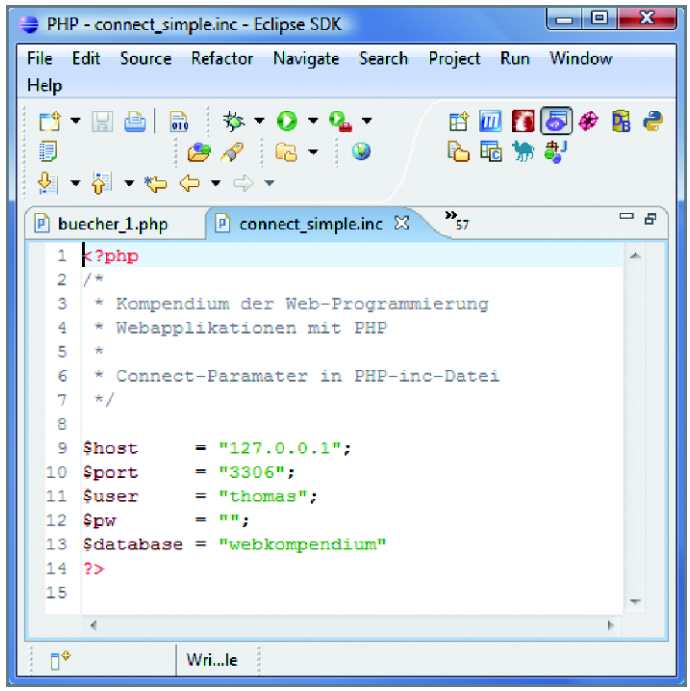
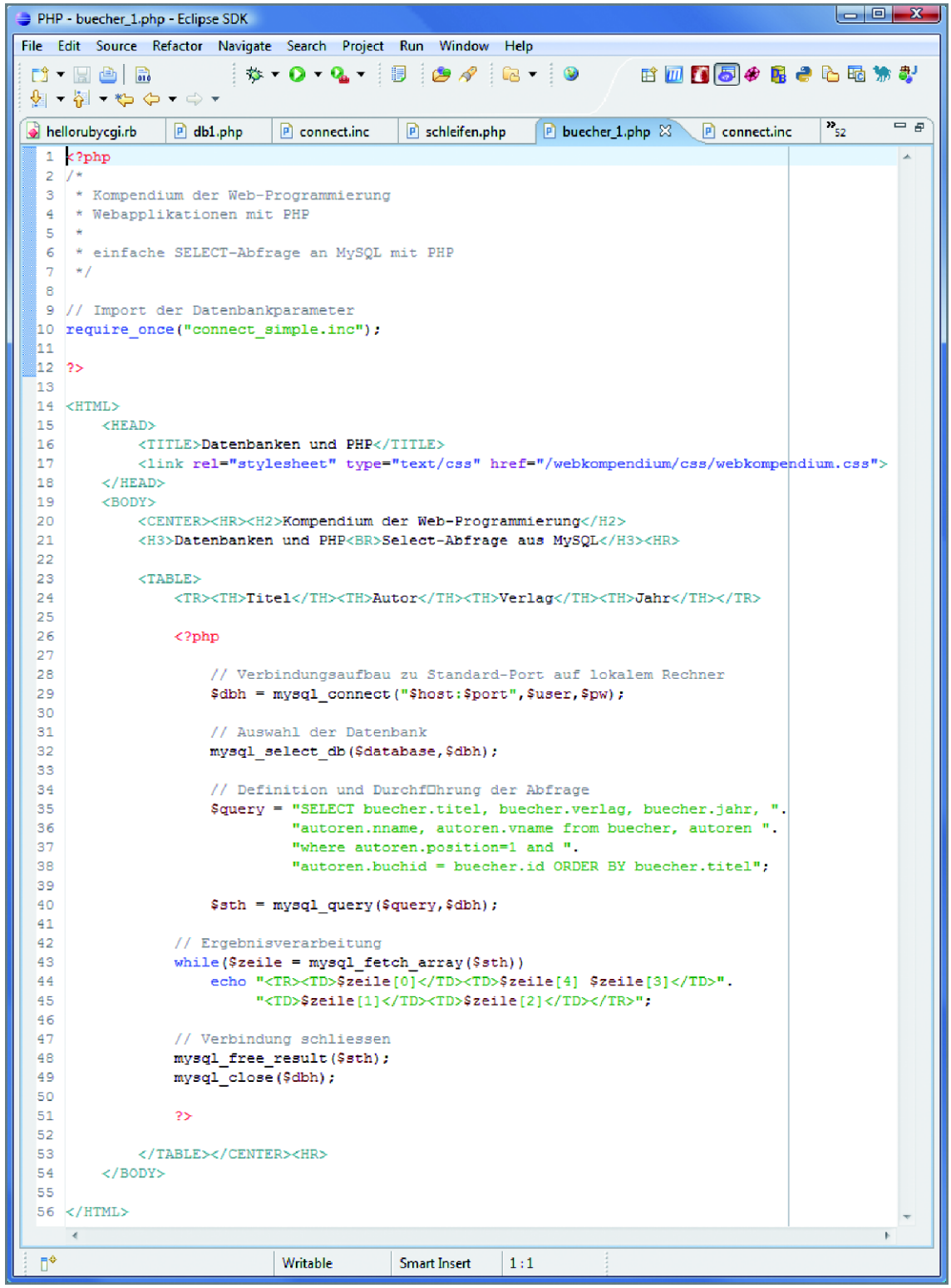



**Abbildung 11.36:**
Ausgelagerte Connect-Parameter `connect_simple.php`




```php
<?php
/*
 * Kompendium der Web-Programmierung
 * Webapplikationen mit PHP
 *
 * einfache SELECT-Abfrage an MySQL mit PHP
 */

// Import der Datenbankparameter
require_once("connect_simple.inc");

?>

<HTML>
    <HEAD>
        <TITLE>Datenbanken und PHP</TITLE>
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
    </HEAD>
    <BODY>
        <CENTER><HR><H2>Kompendium der Web-Programmierung</H2>
        <H3>Datenbanken und PHP<BR>Select-Abfrage aus MySQL</H3><HR>

        <TABLE>
            <TR><TH>Titel</TH><TH>Autor</TH><TH>Verlag</TH><TH>Jahr</TH></TR>

            <?php

                // Verbindungsaufbau zu Standard-Port auf lokalem Rechner
                $dbh = mysql_connect("$host:$port",$user,$pw);

                // Auswahl der Datenbank
                mysql_select_db($database,$dbh);

                // Definition und Durchf□hrung der Abfrage
                $query = "SELECT buecher.titel, buecher.verlag, buecher.jahr, ".
                         "autoren.nname, autoren.vname from buecher, autoren ".
                         "where autoren.position=1 and ".
                         "autoren.buchid = buecher.id ORDER BY buecher.titel";

                $sth = mysql_query($query,$dbh);

            // Ergebnisverarbeitung
            while($zeile = mysql_fetch_array($sth))
                echo "<TR><TD>$zeile[0]</TD><TD>$zeile[4] $zeile[3]</TD>".
                     "<TD>$zeile[1]</TD><TD>$zeile[2]</TD></TR>";

            // Verbindung schliessen
            mysql_free_result($sth);
            mysql_close($dbh);

            ?>

        </TABLE></CENTER><HR>
    </BODY>

</HTML>
```

**Abbildung 11.37:**
SQL-Select-Abfrage der Literaturdatenbank `buecher_1.php`

**Abbildung 11.38:**
Ausführen von `buecher_1.php`

Eine weitere, in der Praxis häufig nützliche PHP-Anweisung ist

```
mysql_num_rows($sth)
```

welche die Anzahl der Trefferzeilen im Resultset zur Abfrage `$sth` ermittelt.

### 11.5.4 Andere SQL-Anweisungen

Der einfache MySQL-Treiber verwendet die gleiche Syntax auch für andere SQL-Anweisungen wie `INSERT` oder `UPDATE`:

```
$sql = "INSERT INTO buecher ..."';

mysql_query($sql,$dbh);
```

Die PHP-Methode

```
mysql_affected_rows($dbh);
```

gibt dann an, wie viele Zeilen der Tabelle verändert wurden.

### 11.5.5 Schließen der Verbindung

Ist PHP als CGI, und nicht als richiges Webserver-Modul, installiert, sollte die Datenbankverbindung bei Beenden des Scriptes geschlossen werden – läuft PHP aber als Webserver-Modul, ist das ordentliche Beenden der Datenbankverbindung absolut unerläßlich, da sonst der Webserver im Laufe der Zeit immer mehr Ressourcen allokiert.

Zunächst sollte der durch ein Resultset `$sth` belegte Speicherbereich freigegeben werden; dies geschieht durch die Anweisung

```
mysql_free_result($sth);
```

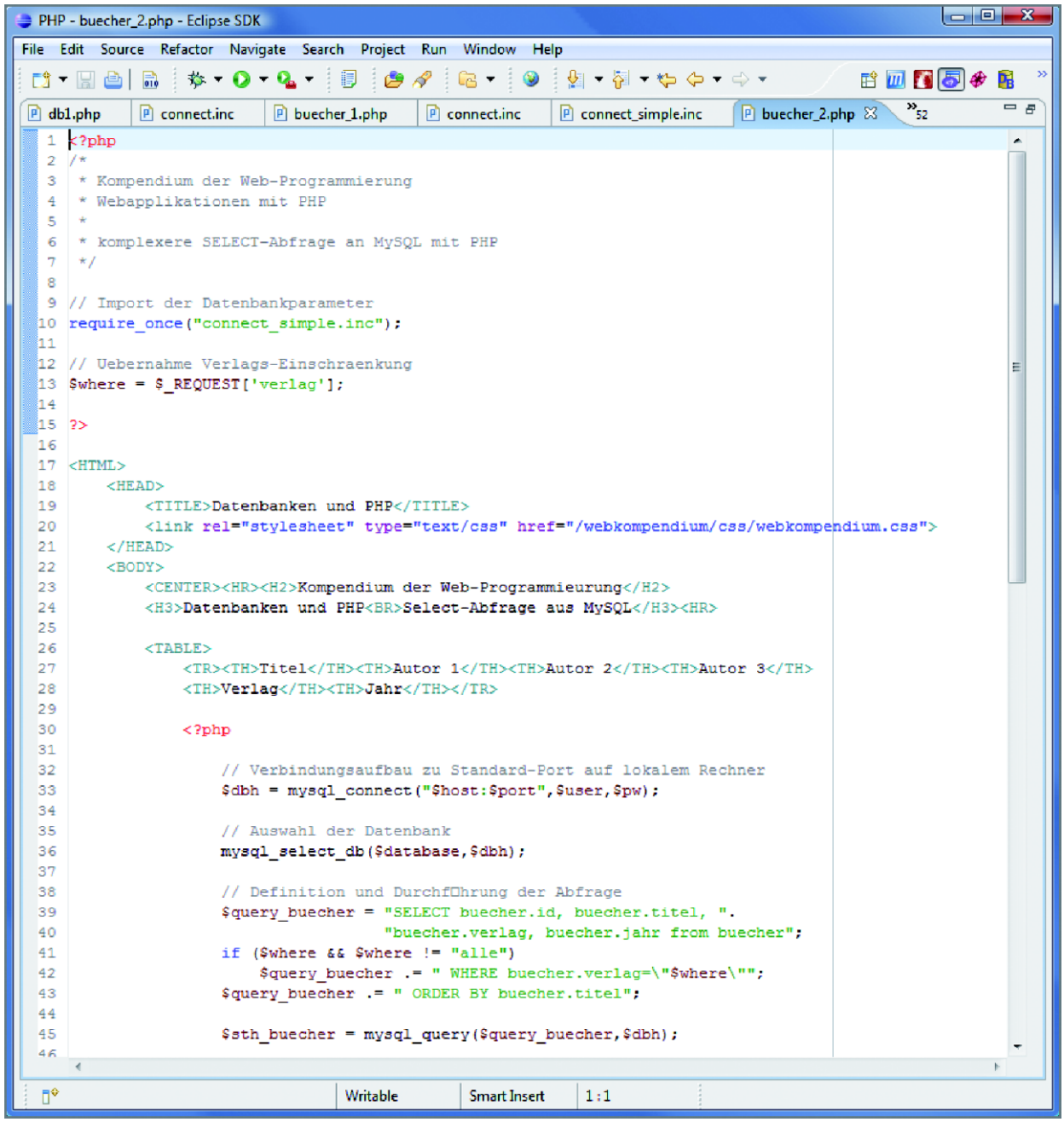
Die Verbindung `$dbh` selbst wird dann durch

```
mysql_close(\dbh);
```

geschlossen.

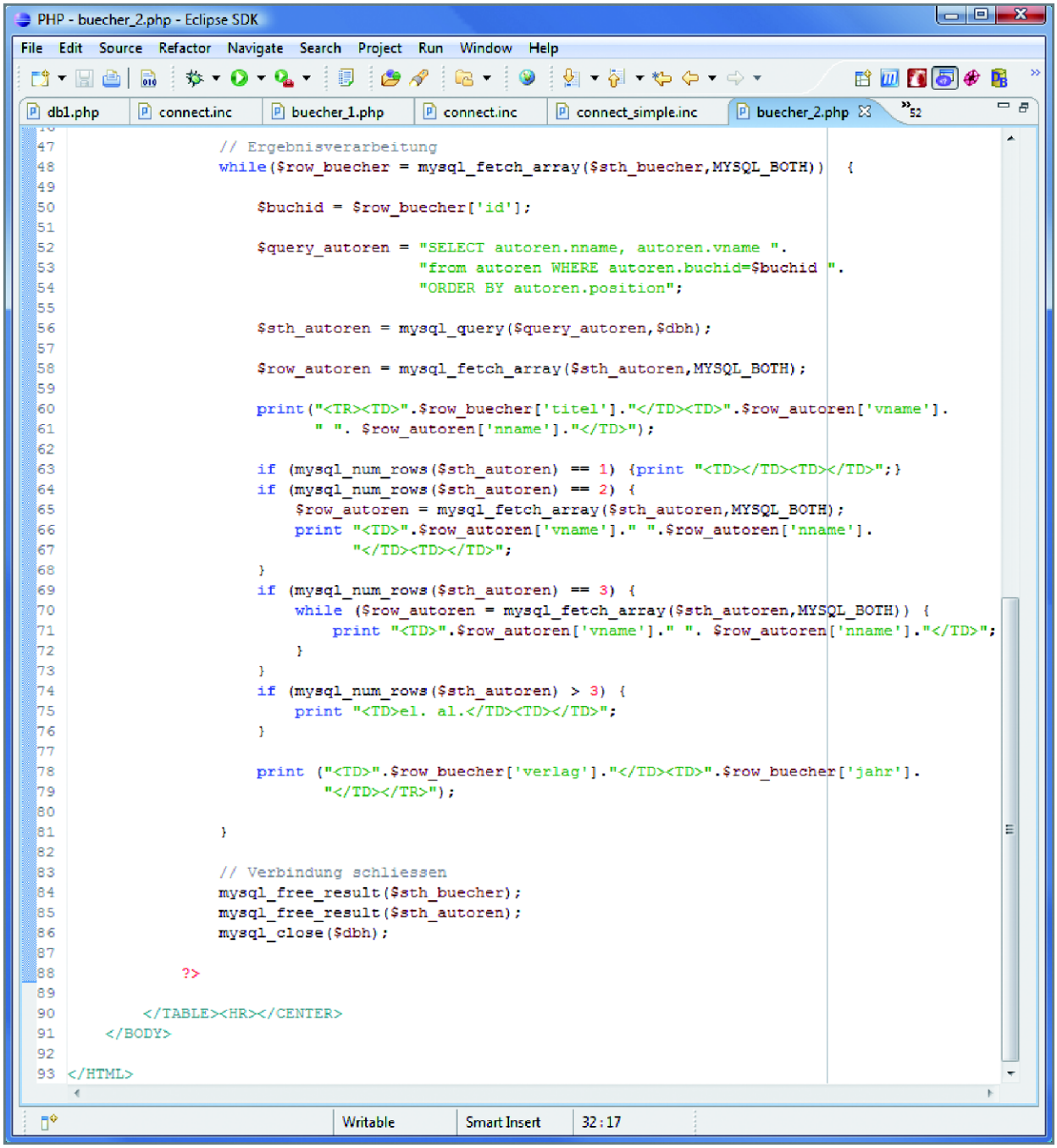
**Komplexeres Beispiel**

Als komplexeres Beispiel sollen nun im Beispiel aus Kapitel 7 zu jedem Buch die Autoren nach den „Regeln der alphabetischen Katalogisierung“ ausgegeben werden: bis zu drei Autoren in ihrer Reihenfolge, bei vier und mehr Autoren der erste und der Hinweis „et al.“. Das PHP-Script `buecher_2.php` (Abbildungen 11.39 und 11.40) leistet dies dadurch, dass zwei Statement Handles verwendet werden. Eine Einschränkung nach dem Verlagsnamen ist zusätzlich möglich, wenn der gesuchte Verlagsnamen als Key zum Value „verlag“ mit übergeben wird.



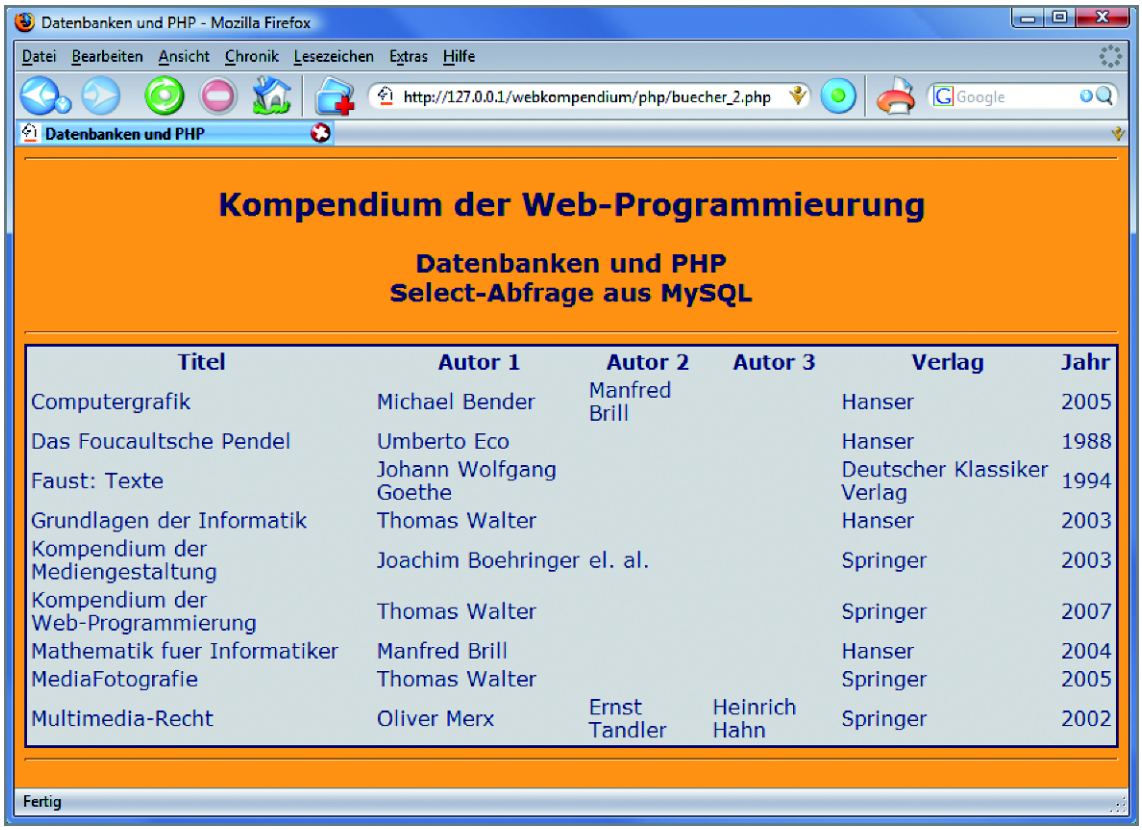

```php
<?php
/*
 * Kompendium der Web-Programmierung
 * Webapplikationen mit PHP
 *
 * komplexere SELECT-Abfrage an MySQL mit PHP
 */

// Import der Datenbankparameter
require_once("connect_simple.inc");

// Uebernahme Verlags-Einschraenkung
$where = $_REQUEST['verlag'];

?>

<HTML>
    <HEAD>
        <TITLE>Datenbanken und PHP</TITLE>
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
    </HEAD>
    <BODY>
        <CENTER><HR><H2>Kompendium der Web-Programmieurung</H2>
        <H3>Datenbanken und PHP<BR>Select-Abfrage aus MySQL</H3><HR>

        <TABLE>
            <TR><TH>Titel</TH><TH>Autor 1</TH><TH>Autor 2</TH><TH>Autor 3</TH>
            <TH>Verlag</TH><TH>Jahr</TH></TR>

            <?php

                // Verbindungsaufbau zu Standard-Port auf lokalem Rechner
                $dbh = mysql_connect("$host:$port",$user,$pw);

                // Auswahl der Datenbank
                mysql_select_db($database,$dbh);

                // Definition und Durchführung der Abfrage
                $query_buecher = "SELECT buecher.id, buecher.titel, ".
                                 "buecher.verlag, buecher.jahr from buecher";
                if ($where && $where != "alle")
                    $query_buecher .= " WHERE buecher.verlag=\"$where\"";
                $query_buecher .= " ORDER BY buecher.titel";

                $sth_buecher = mysql_query($query_buecher,$dbh);
```

**Abbildung 11.39:**
Komplexere SQL-Select-Abfrage im Beispiel der Literaturdatenbank (`buecher_2.php`, Teil I)




```
                // Ergebnisverarbeitung
                while($row_buecher = mysql_fetch_array($sth_buecher,MYSQL_BOTH))  {

                    $buchid = $row_buecher['id'];

                    $query_autoren = "SELECT autoren.nname, autoren.vname ".
                                     "from autoren WHERE autoren.buchid=$buchid ".
                                     "ORDER BY autoren.position";

                    $sth_autoren = mysql_query($query_autoren,$dbh);

                    $row_autoren = mysql_fetch_array($sth_autoren,MYSQL_BOTH);

                    print("<TR><TD>".$row_buecher['titel']."</TD><TD>".$row_autoren['vname'].
                          " ". $row_autoren['nname']."</TD>");

                    if (mysql_num_rows($sth_autoren) == 1) {print "<TD></TD><TD></TD>";}
                    if (mysql_num_rows($sth_autoren) == 2) {
                        $row_autoren = mysql_fetch_array($sth_autoren,MYSQL_BOTH);
                        print "<TD>".$row_autoren['vname']." ".$row_autoren['nname'].
                              "</TD><TD></TD>";
                    }
                    if (mysql_num_rows($sth_autoren) == 3) {
                        while ($row_autoren = mysql_fetch_array($sth_autoren,MYSQL_BOTH)) {
                            print "<TD>".$row_autoren['vname']." ". $row_autoren['nname']."</TD>";
                        }
                    }
                    if (mysql_num_rows($sth_autoren) > 3) {
                        print "<TD>el. al.</TD><TD></TD>";
                    }

                    print ("<TD>".$row_buecher['verlag']."</TD><TD>".$row_buecher['jahr'].
                           "</TD></TR>");

                }

                // Verbindung schliessen
                mysql_free_result($sth_buecher);
                mysql_free_result($sth_autoren);
                mysql_close($dbh);

            ?>

        </TABLE><HR></CENTER>
    </BODY>

</HTML>
```

**Abbildung 11.40:**
Kompelxere SQL-Select-Abfrage in dem Beispiel der Literaturdatenbank (`buecher_2.php`, Teil II)




| Titel | Autor 1 | Autor 2 | Autor 3 | Verlag | Jahr |
|---|---|---|---|---|---|
| Computergrafik | Michael Bender | Manfred Brill | | Hanser | 2005 |
| Das Foucaultsche Pendel | Umberto Eco | | | Hanser | 1988 |
| Faust: Texte | Johann Wolfgang Goethe | | | Deutscher Klassiker Verlag | 1994 |
| Grundlagen der Informatik | Thomas Walter | | | Hanser | 2003 |
| Kompendium der Mediengestaltung | Joachim Boehringer | el. al. | | Springer | 2003 |
| Kompendium der Web-Programmierung | Thomas Walter | | | Springer | 2007 |
| Mathematik fuer Informatiker | Manfred Brill | | | Hanser | 2004 |
| MediaFotografie | Thomas Walter | | | Springer | 2005 |
| Multimedia-Recht | Oliver Merx | Ernst Tandler | Heinrich Hahn | Springer | 2002 |

**Abbildung 11.41:**
Ausführen von `buecher_2.php`

### 11.5.6 Weitere native Datenbanktreiber für PHP

Der hier vorgestellte Datenbanktreiber bezieht sich auf den ursprünglichen MySQL-Treiber, wie er schon in PHP 3 verwendet wurde. Andere Treiber haben durchgängig eine abweichende Syntax, was den Austausch des DBMS sehr erschwert. Einige typische Treiber werden hier vorgestellt.

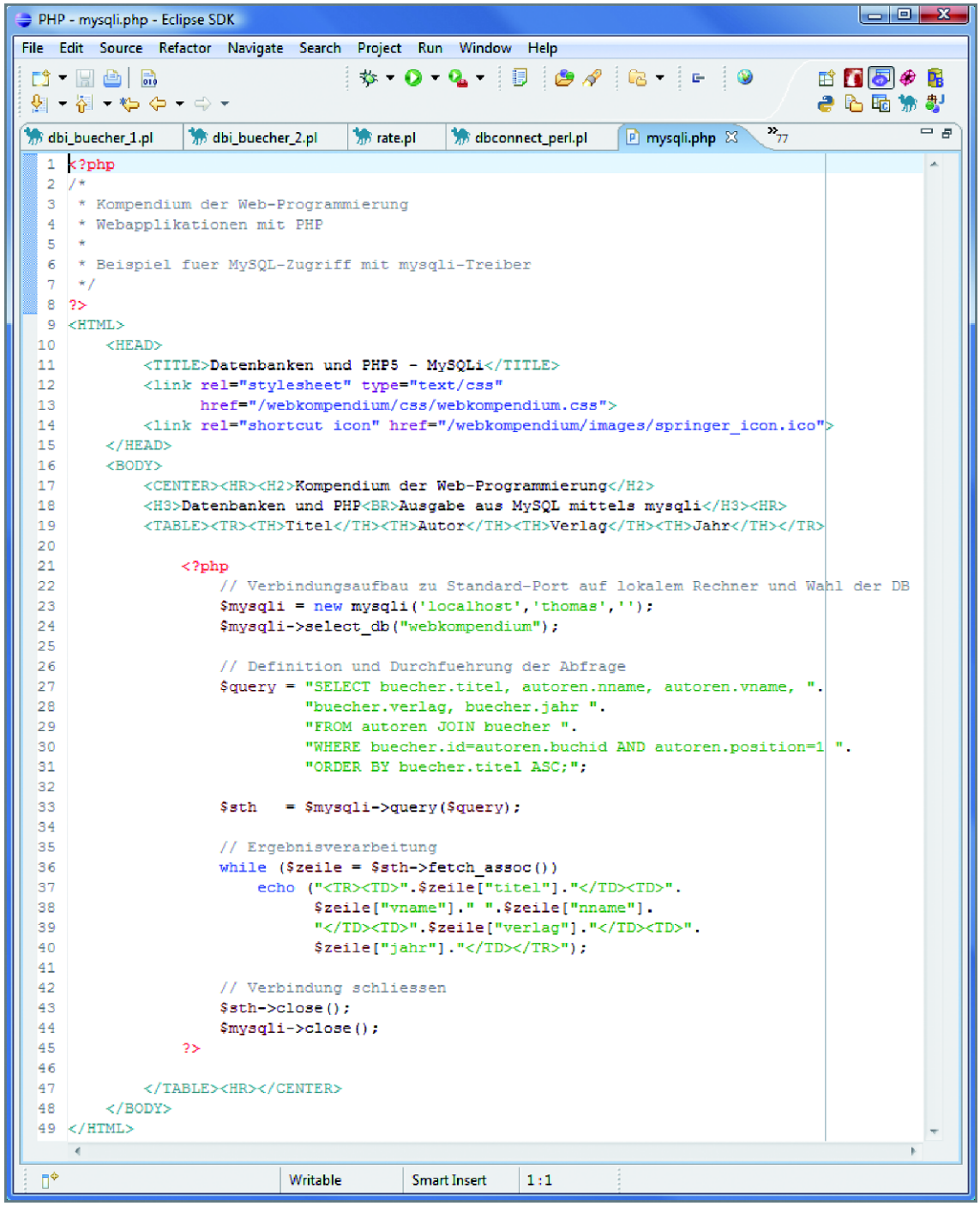
#### 11.5.6.1 mysqli

Der bisher vorgestellte „klassische“ MySQL-Treiber von PHP ist bereits älteren Datums und noch mit Versionen 3 von MySQL einsetzbar. Mit PHP5 steht ein verbesserter Treiber zur Verfügung: *mysqli* (das „i“ steht dabei für „improved“).

Der mysqli-Treiber ist für MySQL ab der Version 4.1 einsetzbar. Er ist performanter gegenüber dem klassischen Treiber[1] und ist auch objektorientiert einsetzbar (vgl. Abbildung 11.42). Aber sein wichtigster Vorteil: mysqli unterstützt die neueren Eigenschaften der aktuellen MySQL-Versionen, etwa Transaktionsmanagement.

Das Script `mysqli.php` (Abbildung 11.42) zeigt die exemplarische Auswertung der Literatur-Datenbanktabellen nach Kapitel 7.

[1] interne Messungen geben hier aber kein eindeutiges Bild, ob und wenn wieviel performanter mysqli ist.




```
<?php
/*
 * Kompendium der Web-Programmierung
 * Webapplikationen mit PHP
 *
 * Beispiel fuer MySQL-Zugriff mit mysqli-Treiber
 */
?>
<HTML>
    <HEAD>
        <TITLE>Datenbanken und PHP5 - MySQLi</TITLE>
        <link rel="stylesheet" type="text/css"
              href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
    </HEAD>
    <BODY>
        <CENTER><HR><H2>Kompendium der Web-Programmierung</H2>
        <H3>Datenbanken und PHP<BR>Ausgabe aus MySQL mittels mysqli</H3><HR>
        <TABLE><TR><TH>Titel</TH><TH>Autor</TH><TH>Verlag</TH><TH>Jahr</TH></TR>

            <?php
                // Verbindungsaufbau zu Standard-Port auf lokalem Rechner und Wahl der DB
                $mysqli = new mysqli('localhost','thomas','');
                $mysqli->select_db("webkompendium");

                // Definition und Durchfuehrung der Abfrage
                $query = "SELECT buecher.titel, autoren.nname, autoren.vname, ".
                         "buecher.verlag, buecher.jahr ".
                         "FROM autoren JOIN buecher ".
                         "WHERE buecher.id=autoren.buchid AND autoren.position=1 ".
                         "ORDER BY buecher.titel ASC;";

                $sth    = $mysqli->query($query);

                // Ergebnisverarbeitung
                while ($zeile = $sth->fetch_assoc())
                    echo ("<TR><TD>".$zeile["titel"]."</TD><TD>".
                          $zeile["vname"]." ".$zeile["nname"].
                          "</TD><TD>".$zeile["verlag"]."</TD><TD>".
                          $zeile["jahr"]."</TD></TR>");

                // Verbindung schliessen
                $sth->close();
                $mysqli->close();
            ?>

        </TABLE><HR></CENTER>
    </BODY>
</HTML>
```

**Abbildung 11.42:** MySQL-Datenbankabfrage mittels mysqli-Treiber (Script `mysqli.php`)

Der mysqli-Treiber ist standardmäßig nicht installiert. Er kann einfach durch dynamisches Laden des Treibers verwendet werden, wozu die zentrale Konfigurationsdatei `php.ini` abzuändern ist:

```
extension=php_mysqli.so       ; Unix
extension=php_mysqli.dll      ; Windows
```

oder bei Compilation von PHP durch statisches Einbinden durch eine Konfiguration wie

```
--with-mysqli=mysql_config_path/mysql_config,
```

Für den mysqli-Treiber können in der `php.ini` zahlreiche Default-Parameter festgelegt werden.

Natürlich ist es möglich, mysqli und den klassischen mysql-Treiber parallel zu verwenden, also beide in der PHP-Instanz verfügbar zu haben.

**Abbildung 11.43:**
Ausführen von `mysqli.php`

### 11.5.6.2 ODBC

Universell auf Windows-Systemen ist der *ODBC-Treiber:* Hier wird eine weitere Zwischenschicht verwendet, so dass PHP die ODBC-Schicht anspricht und diese dann auf das jeweilige DBMS zugreift. Die entsprechende ODBC-Verbindung muss auf dem jeweiligen Serversystem konfiguriert sein, wobei zentral der frei wählbare Name dieser Verbindung ist.

Typischerweise sind unter Systemsteuerung|Verwaltung|Datenquellen (ODBC) die ODBC-Einstellungen zu finden; Abbildung 11.45 zeigt die beispielhafte Konfiguration einer ODBC-Verbindung an ein MySQL-DBMS; Voraussetzung hierfür ist die zusätzliche Installation des entsprechenden ODBC-Treibers, da Windows zunächst nur die Datenbanksysteme des Hauses Microsoft unterstützt. Auf diesem Weg kann beispielsweise Microsoft Access als DBMS mit PHP eingesetzt werden.

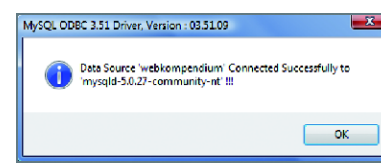



**Abbildung 11.44:**
Testen der ODBC-Verbindung

Die Syntax für den Verbindungsaufbau mittels ODBC lautet nun

```
$dbh=odbc_connect($dsn,$user,$pw);
```

wobei sich hinter `$dsn` einfach der Name der ODBC-Verbindung verbirgt (vgl. Abbildung 11.45).

Die Datenbankabfrage `$query` wird dann mittels

```
$sth = odbc_exec($sth,$query);
```

durchgeführt.
Das zeilenweise Abarbeiten des derart gewonnenen Resultsets kann etwa mittels der Methode

```
odbc_fetch_into($sth,$row)
```

geschehen, und mittels

```
odbc_close($dbh)
```

wird die Verbindung zum DBMS über ODBC wieder geschlossen.

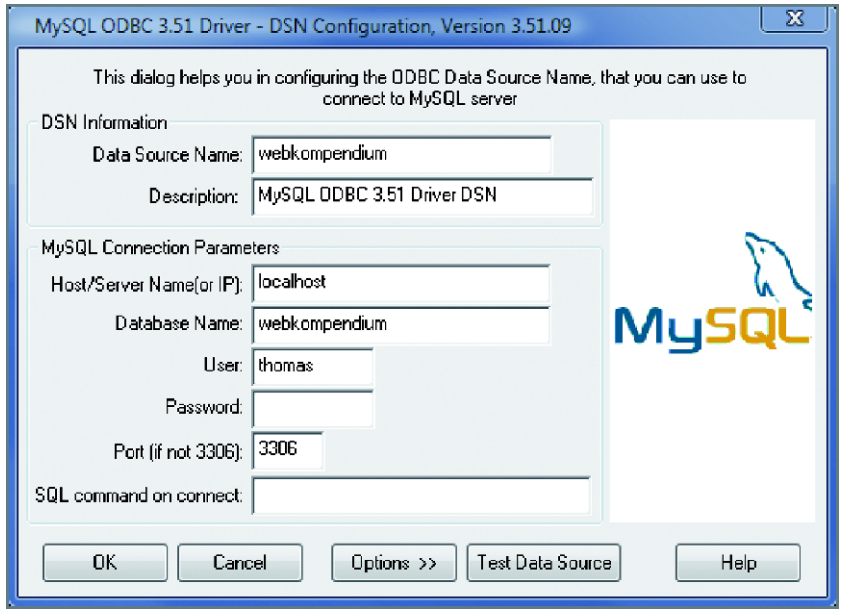
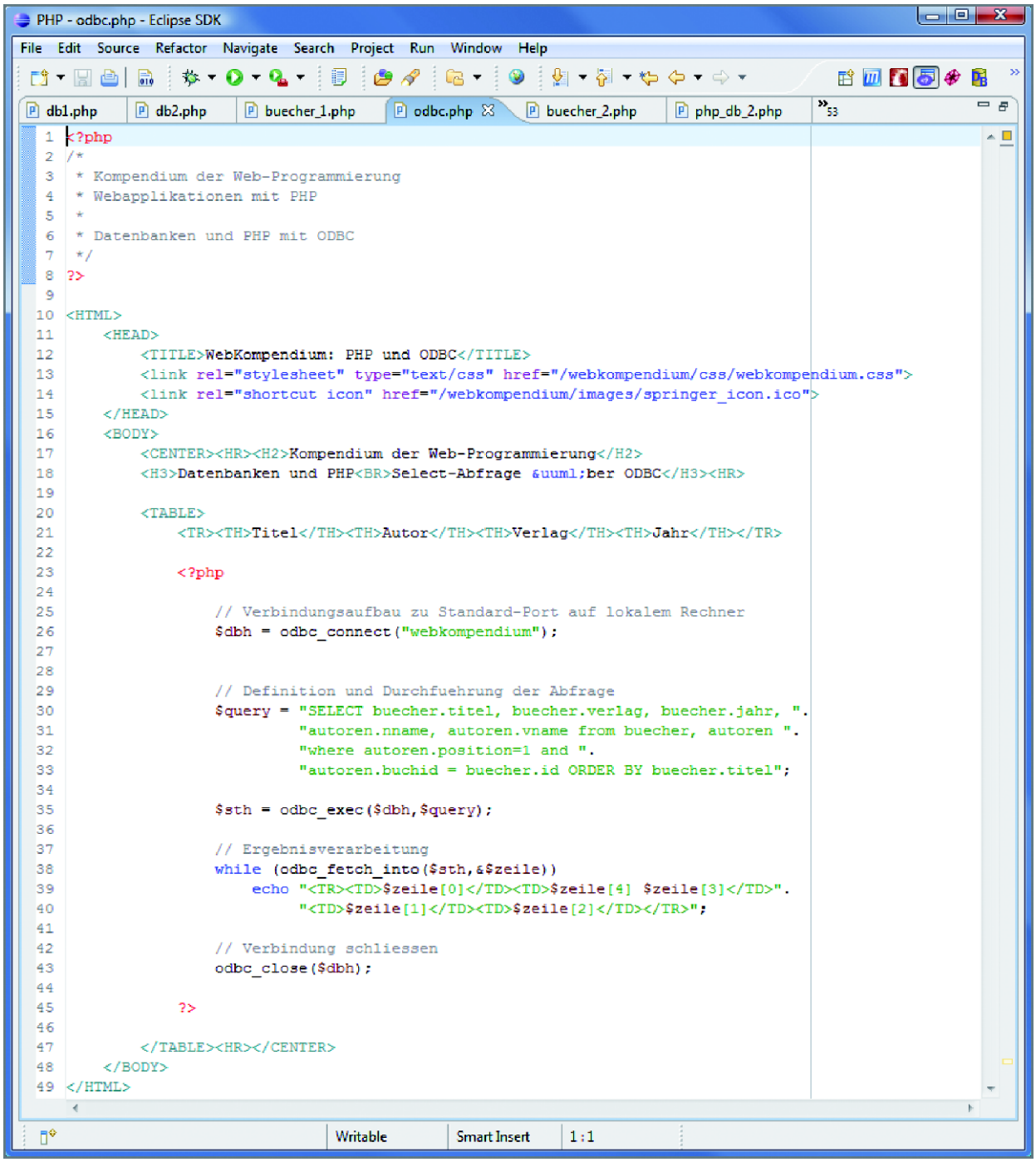



**Abbildung 11.45:** Konfiguration einer ODBC-Verbindung unter Windows (hier für MySQL)

Abbildung 11.46 zeigt die Implementierung der SQL-Abfrage von Script `buecher_1.php` nun mittels ODBC im Script `odbc.php`.




```php
<?php
/*
 * Kompendium der Web-Programmierung
 * Webapplikationen mit PHP
 *
 * Datenbanken und PHP mit ODBC
 */
?>

<HTML>
    <HEAD>
        <TITLE>WebKompendium: PHP und ODBC</TITLE>
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
    </HEAD>
    <BODY>
        <CENTER><HR><H2>Kompendium der Web-Programmierung</H2>
        <H3>Datenbanken und PHP<BR>Select-Abfrage &uuml;ber ODBC</H3><HR>

        <TABLE>
            <TR><TH>Titel</TH><TH>Autor</TH><TH>Verlag</TH><TH>Jahr</TH></TR>

            <?php

                // Verbindungsaufbau zu Standard-Port auf lokalem Rechner
                $dbh = odbc_connect("webkompendium");


                // Definition und Durchfuehrung der Abfrage
                $query = "SELECT buecher.titel, buecher.verlag, buecher.jahr, ".
                         "autoren.nname, autoren.vname from buecher, autoren ".
                         "where autoren.position=1 and ".
                         "autoren.buchid = buecher.id ORDER BY buecher.titel";

                $sth = odbc_exec($dbh,$query);

                // Ergebnisverarbeitung
                while (odbc_fetch_into($sth,&$zeile))
                    echo "<TR><TD>$zeile[0]</TD><TD>$zeile[4] $zeile[3]</TD>".
                         "<TD>$zeile[1]</TD><TD>$zeile[2]</TD></TR>";

                // Verbindung schliessen
                odbc_close($dbh);

            ?>

        </TABLE><HR></CENTER>
    </BODY>
</HTML>
```

**Abbildung 11.46:**
Script `odbc.php`

## 11.6 Strukturierte Softwareentwicklung mit PHP

Die Entwicklung umfangreicherer PHP-Applikationen setzt eine *strukturierte Softwareentwicklung* voraus. Der Weg dazu führt letztlich zu einem komplexen Vorgehen wie in Kapitel 4 skizziert. Ein wesentlicher Bestandteil der modernen Softwareentwicklung ist die Objektorientierung. Diese ist in PHP inzwischen sehr stark ausgeprägt.

### 11.6.1 Objektorientierung in PHP

Die *Objektorientierung in PHP* (vgl. 4.2.2) hat sich mit jeder neuen Version erweitert; seit PHP5 liegt ein sehr umfassendes Konzept vor, welches in der Syntax und auch in der Semantik deutlich an Java orientiert ist.

#### 11.6.1.1 Die Klasse und der Konstruktor

Zentral ist das Schlüsselwort `class` mit der Syntax

```
class  {
    ...
}
```

für die Definition einer *Klasse*.

Die Syntax des *Konstruktors* hat sich von der Version 4 auf die Version 5 von PHP grundlegend geändert, wobei allerdings PHP5 weitgehend abwärtskompatibel ist,[2] im Normalfall also die PHP4-Syntax versteht. In PHP5 ist ein Konstruktor definiert über die Methode

```
__construct(...) {
   ...
}
```

In PHP4 ist es hingegen – wie auch in Java – eine Methode mit dem gleichen Namen wie die Klasse.

Es steht auch ein Destruktor zur Verfügung, also eine Methode, welche beim Vernichten eines Objektes ausgeführt wird. Diese hat die Syntax

```
__destruct() {
   ...
}
```

PHP zählt intern, wie viele Referenzen auf ein Objekt verweisen; wenn keine Referenzen mehr ein Objekt adressieren, ist dies „unerreichbar“ (unreachable) und kann zerstört werden, die sogenannte Speicherbereinigung (Garbage Collection).

#### 11.6.1.2 Datenkapselung in PHP

Seit der Version 5 verfügt PHP auch über eine *Datenkapselung*, also die Möglichkeit, den Zugriff auf Klassenattribute für fremde Klassen einzuschränken.

Es stehen drei Attribute zur Verfügung:

- `public` ist der default-Fall und entspricht der Deklaration von Klassenattributen in PHP4.[3] Hier besteht globaler Zugriff, alle anderen Klassen können auf dieses Attribut zugreifen.

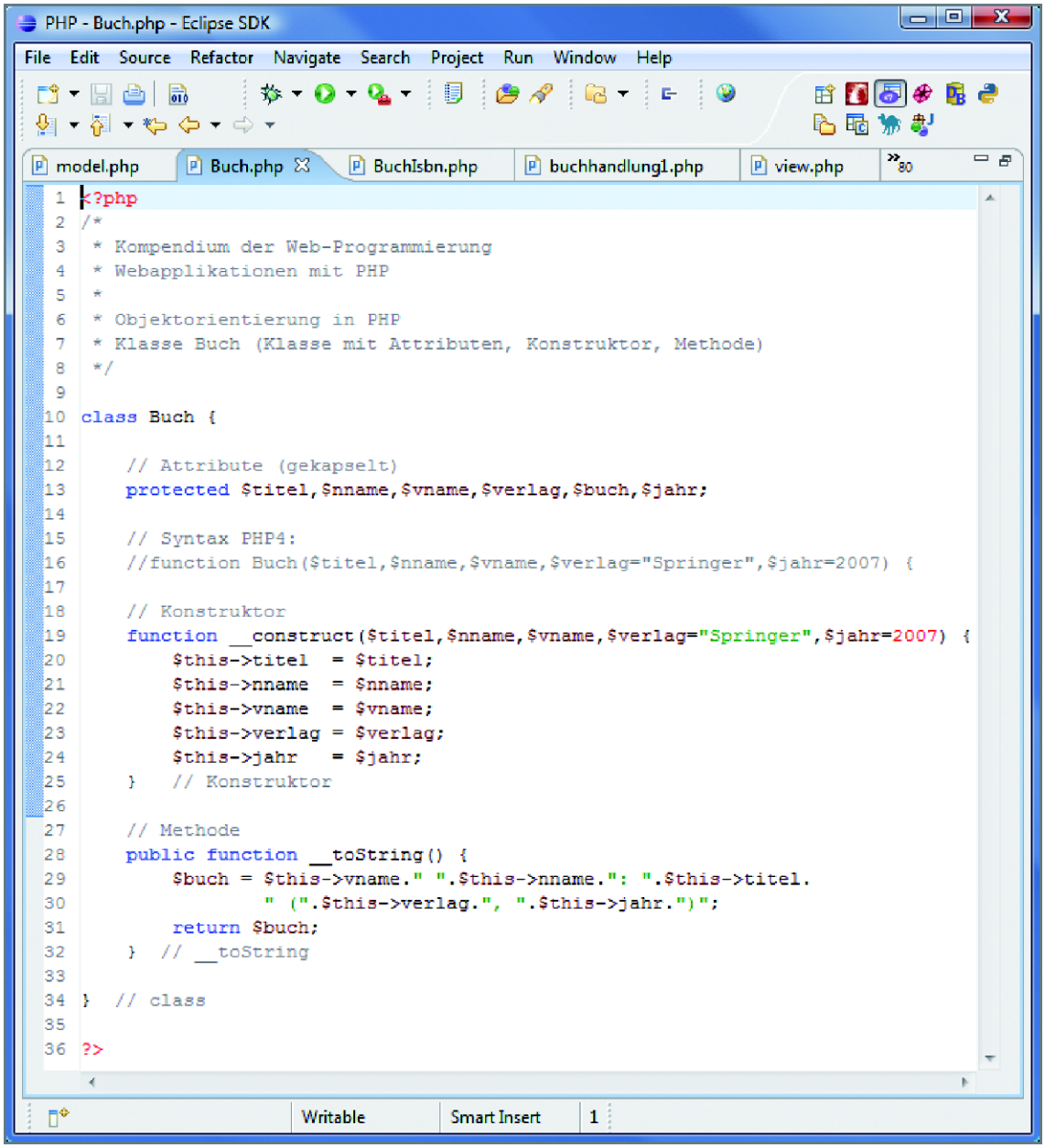
[2] genaue Details zur Migration von Version 4 zu Version 5 sind unter `http://de2.php.net/manual/de/migration5.php` zu finden.

[3] in PHP4 mit dem Schlüsselwort `var`.

- `protected` schränkt den Zugriff ein, die eigene Klasse und alle abgeleiteten Klassen haben Zugriff.
- `private` ist die stärkste Einschränkung, nur die Klasse selbst hat Zugriff.

Bei gekapselten Attributen ist es üblich, den Zugriff über set- und get-Methoden zu steuern.

Die Klasse `Buch` in Abbildung 11.47 zeigt eine einfache Klasse vergleichbar zu der Java-Klasse aus Abbildung 4.1. Sie definiert eine, hier als `protected` definierte Membervariable, enthält einen Konstruktor und ergänzt eine Methode `__toString()` für eine sinnvolle Zeichenkettendarstellung des Objektes.




```php
<?php
/*
 * Kompendium der Web-Programmierung
 * Webapplikationen mit PHP
 *
 * Objektorientierung in PHP
 * Klasse Buch (Klasse mit Attributen, Konstruktor, Methode)
 */

class Buch {

    // Attribute (gekapselt)
    protected $titel,$nname,$vname,$verlag,$buch,$jahr;

    // Syntax PHP4:
    //function Buch($titel,$nname,$vname,$verlag="Springer",$jahr=2007) {

    // Konstruktor
    function __construct($titel,$nname,$vname,$verlag="Springer",$jahr=2007) {
        $this->titel  = $titel;
        $this->nname  = $nname;
        $this->vname  = $vname;
        $this->verlag = $verlag;
        $this->jahr   = $jahr;
    }   // Konstruktor

    // Methode
    public function __toString() {
        $buch = $this->vname." ".$this->nname.": ".$this->titel.
                " (".$this->verlag.", ".$this->jahr.")";
        return $buch;
    }  // __toString

}  // class

?>
```

Die Zugriffsattribute `public`, `protected` und `private` können nicht nur für Attribute, sondern auch für die Methoden einer Klasse verwendet werden.

**Abbildung 11.47:**
Die Klasse `Buch` im Script `Buch.php`

### 11.6.1.3 Instanziierung in PHP

Mittels des Operators `new` werden auch in PHP Objekt-Instanzen gebildet, wobei stets der Konstruktor der Klasse aufgerufen wird.

Das Script `buchhandlung1.php` (Abbildung 11.48) erzeugt eine Instanz der Klasse `Buch` (Abbildung 11.47) und gibt eine Zeichenkettendarstellung dieses Objektes aus; dazu ruft PHP5 die Methode `__toString()` automatisch auf (Abbildung 11.49).

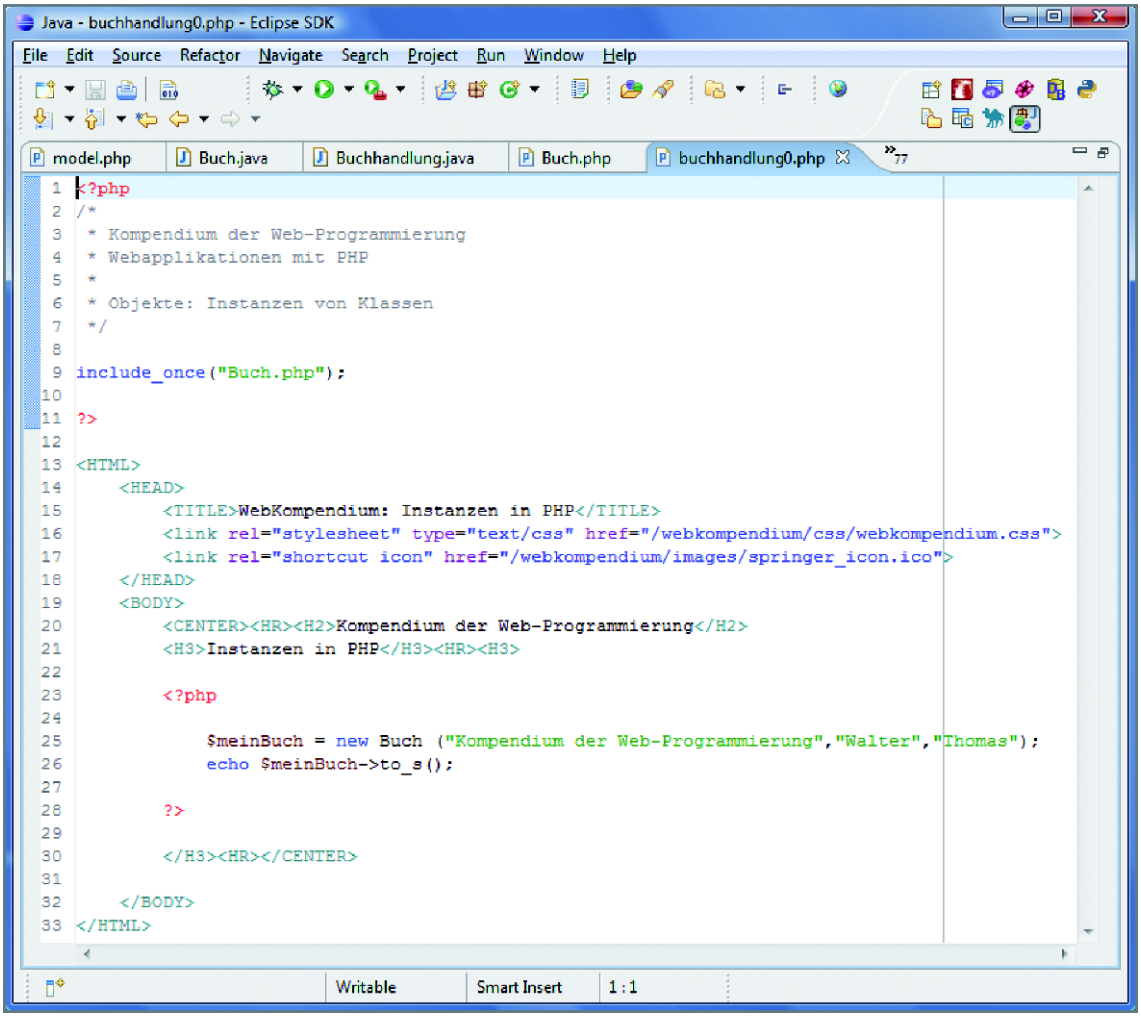



**Abbildung 11.48:**
Erzeugen einer Instanz der Klasse `Buch` (Script `buchhandlung1.php`)

**Abbildung 11.49:**
Ausführen von `buchhandlung1.php`

#### 11.6.1.4 Vererbung in PHP

Die *Vererbung* ist ein zentraler Mechanismus der Objektorientierung; durch Ableiten (Vererben) übernehmen die Kindklassen alle Eigenschaften und Methoden der Elternklasse, redundanter Programmcode kann entfallen.

Natürlich bietet auch PHP die Vererbung, wie in Java über das Schlüsselwort `extends`:

```
class B extends A {
   ...
}
```

Als Beispiel soll die Klasse `Buch` aus Abbildung 11.47 abgeleitet werden zur Klasse `BuchIsbn` (Abbildung 11.50), welche ein weiteres Attribut für die ISBN-Nummer ergänzt; entsprechend muss die Methode `__toString()` neu implementiert werden, damit auch das weitere Attribut ausgegeben wird – die Methode `to_s()` wird in der Kindklasse überschrieben.

Über das Schlüsselwort `parent` kann die Kindklasse dabei geschickt auf die Elternklasse zugreifen; dies geschieht sowohl beim Konstruktor als auch beim Überschreiben der Methode `__toString()`. `parent` hat damit eine vergleichbare Bedeutung wie `super` für Java.

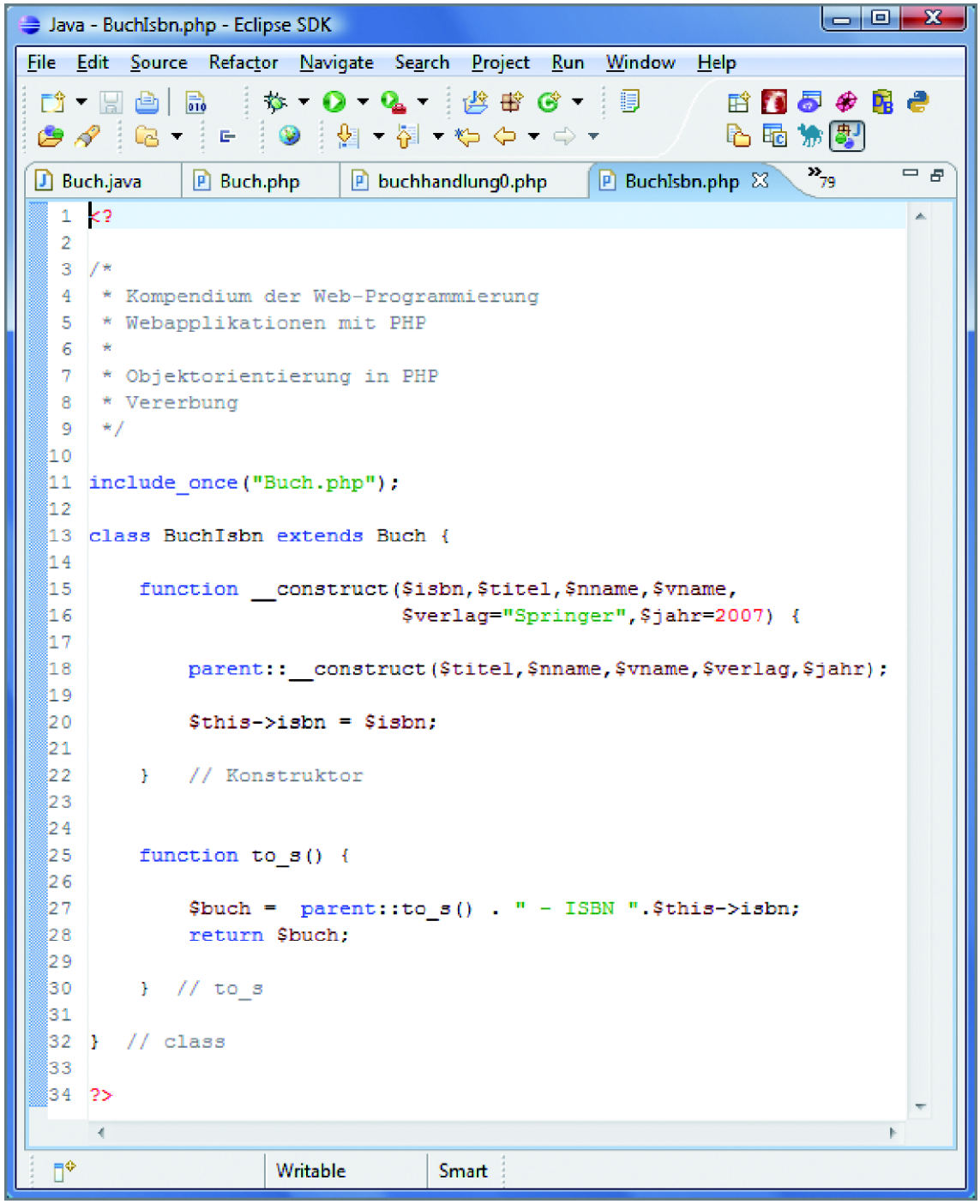



**Abbildung 11.50:**
Die Klasse `BuchIsbn`, abgeleitet von der Klasse `Buch` (`BuchIsbn.php`)

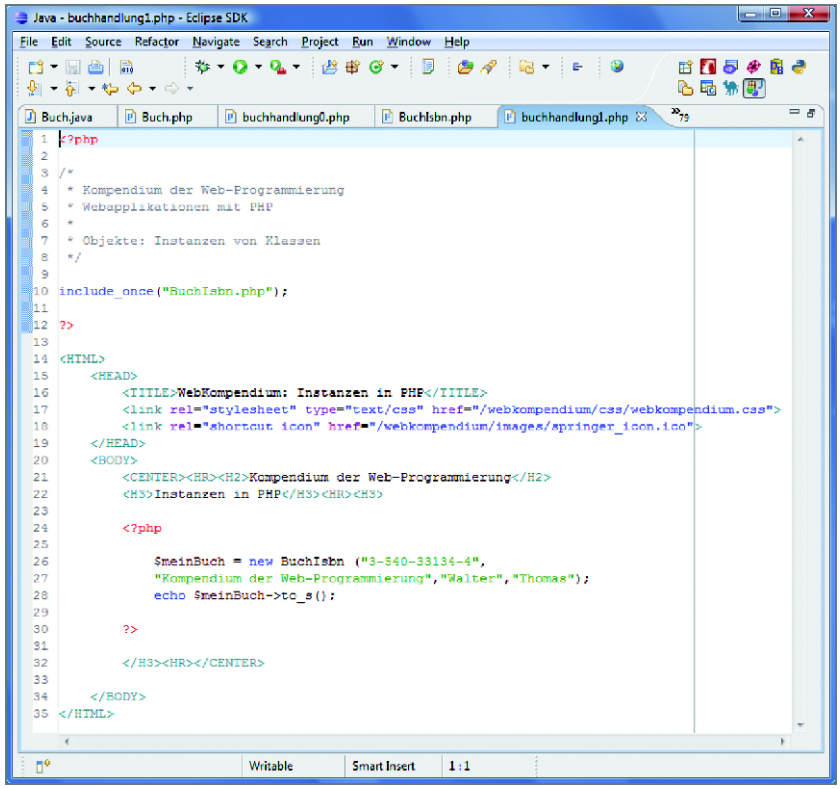



**Abbildung 11.51:**
Erzeugen einer Instanz der Klasse `BuchIsbn` (Script `buchhandlung2.php`)

**Abbildung 11.52:**
Ausführen von `buchhandlung2.php`

### 11.6.1.5 Weitere Mechanismen der Objektorientierung in PHP

PHP in der Version 5 bietet zahlreiche weitere Mechanismen der Objektorientierung; viele sind syntaktisch und semantisch stark an Java angelehnt:

- Das Schlüsselwort `abstract` bezeichnet eine Klasse, welche nicht instanziiert werden kann, von welcher also kein Objekt gebildet werden kann; sie dient als Wurzel einer Vererbungshierarchie.
- `final` bedeutet bei Methoden, dass diese beim Vererben nicht überschrieben werden können, also in Kindklassen nicht neu implementiert werden können.

- Neben der Klasse bietet PHP das Konzept des `interface`. Ein Interface ist ähnlich wie eine abstrakte Klasse, nur für die Vererbungshierarchie anwendbar; allerdings kann in PHP5 mittels des Interface eine schwache Mehrfachvererbung, also das Ableiten von mehreren Wurzeln gleichzeitig, implementiert werden: Während eine Klasse nur von einer Klasse abgeleitet werden kann, kann sie von beliebig vielen Interfaces „abgeleitet“ werden, also diese Interfaces „implementieren“:

  ```
  class B extends A implements InterfaceA,
                               InterfaceB, InterfaceC {
     ...
  }
  ```

  Wie in Java kann ein Interface in PHP5 nur Konstanten definieren und Methoden deklarieren, die dann in den implementierenden Klassen auszuformulieren sind.
- Neben Eigenschaften und Methoden, welche sich auf ein Objekt, also eine Klasseninstanz beziehen, gibt es auch Klasseneigenschaften, also Variablen, welche für alle Instanzen der gleichen Klasse identisch sind. Diese werden – wie in Java – durch das Schlüsselwort `static` gekennzeichnet. Auch für Methoden gibt es dieses Schlüsselwort; es soll dann verwendet werden, wenn die Methode sich nicht auf ein konkretes Objekt bezieht, sondern für alle Instanzen gleich ist.
- PHP5 bietet spezielle Methoden, die die Objektorientierung effizienter nutzbar machen:
  - Eine besondere Rolle nimmt die Methode

    ```
    __toString()
    ```

    ein (vgl. Abbildung 11.47); sie wird automatisch bei jeder Ausgabe einer Objektinstanz der jeweiligen Klasse ausgeführt, ganz analog zur Methode `java-lang.Object.toString()` in Java.
  - die Methode `get_class` liefert den Klassennamen des Objekts.
  - `get_parent_class` liefert entsprechend den Namen der Elternklasse.
  - `get_class_methods` ist ein Array mit den Methoden einer Klasse.
  - `get_class_vars` ist entsprechend ein Array mit den Namen der Attribute der Klasse.

#### 11.6.1.6 Ausnahmebehandlung in PHP

Auch beim Umgang mit Ausnahmen berücksichtigt PHP5 das Java-Konzept: Es gibt einen `try`-Block, Ausnahmen werden im `catch`-Block abgefangen und ein optionaler `finally`-Block wird in beiden Fällen ausgeführt:

```
try {
   ...
}
catch(...) {
   ...
}
fianlly {
   ...
}
```

### 11.6.2 Design-Patterns: MVC und mehr

Die bisherige Auseinandersetzung mit PHP lief auf eine weitgehend unstrukturierte Softwareentwicklung hinaus; dies ist kein Ansatz für eine professionelle Arbeit.

In Abschnitt 4.3 wurde bereits die verbreitete Softwarearchitektur nach *Model-View-Controller* (MVC) vorgestellt; um dies zu verdeutlichen, soll hier ein einfaches PHP-Beispiel nach MVC gezeigt werden. Als exemplarische Anwendung dient wieder das Beispiel nach Abschnitt 7, die Ausgabe der Literaturdaten.

Im MVC-Paradigma werden hierfür drei Pakete, im einfachsten Fall drei einfache PHP-Klassen/-Dateien benötigt:

- Die Model-Klasse: die „Datenhaltungsschicht“ des Models verwaltet alle Daten unserer Anwendung und sorgt für deren persistente Abspeicherung in einer Datenbank.

  Im Beispiel ist dies die Klasse `Model` in der Datei `model.php` (Abbildung 11.53), welche in ihrem Konstruktor in der PHP5-Syntax eine Liste aller Datenbankeinträge als Attribut `buecher` erzeugt. Dabei ist jeder einzelne Eintrag eine Instanz der Klasse `Buch` nach Abbildung 11.47.

  Diese Model-Klasse enthält auch sämtliche Datenbank-Operationen, hier mit dem klassischen mysql-Treiber. Im nächsten Schritt für eine strukturierte Softwareentwicklung könnten diese noch gekapselt und ausgelagert werden. Ein Vorteil des MVC-Patterns sind aber auch so erkennbar: Ändert sich das Datenbanksystem, so ist nur diese Model-Klasse anzupassen.
- Der Controller: Im Beispiel liegt der einfachste Fall eines Controllers vor, welcher nur eine View aufruft, also keine Parameter des HTTP-Requests verarbeitet (Script `controller.php` in Abbildung 11.54). Hier wird folgendes Vorgehen gewählt: Der Controller erzeugt eine Instanz der Model-Klasse; dabei wird der Konstruktor von Model aufgerufen und somit die Datenbank ausgelesen. Anschließend erzeugt der Controller auch eine Instanz der View-Klasse und ruft deren Methode `zeige()` auf und übergibt die Instanz der Model-Klasse an diese.
- Die View: Die Klasse `View` (Abbildung 11.55) bringt nun die übergebene Model-Instanz zur Anzeige; dies geschieht durch Abarbeiten der übergebenen Liste und Ausgabe jeweils in einer Tabellenzeile. Für die Ausgabe eines Elementes der Liste `buecher` des Models wird die Methode `to_s()` der Klasse `Buecher` (Abbildung 11.47) verwendet, die HTML-Formatierung liefert die View-Klasse – natürlich unter Verwendung des Basislayouts des Stylesheets.

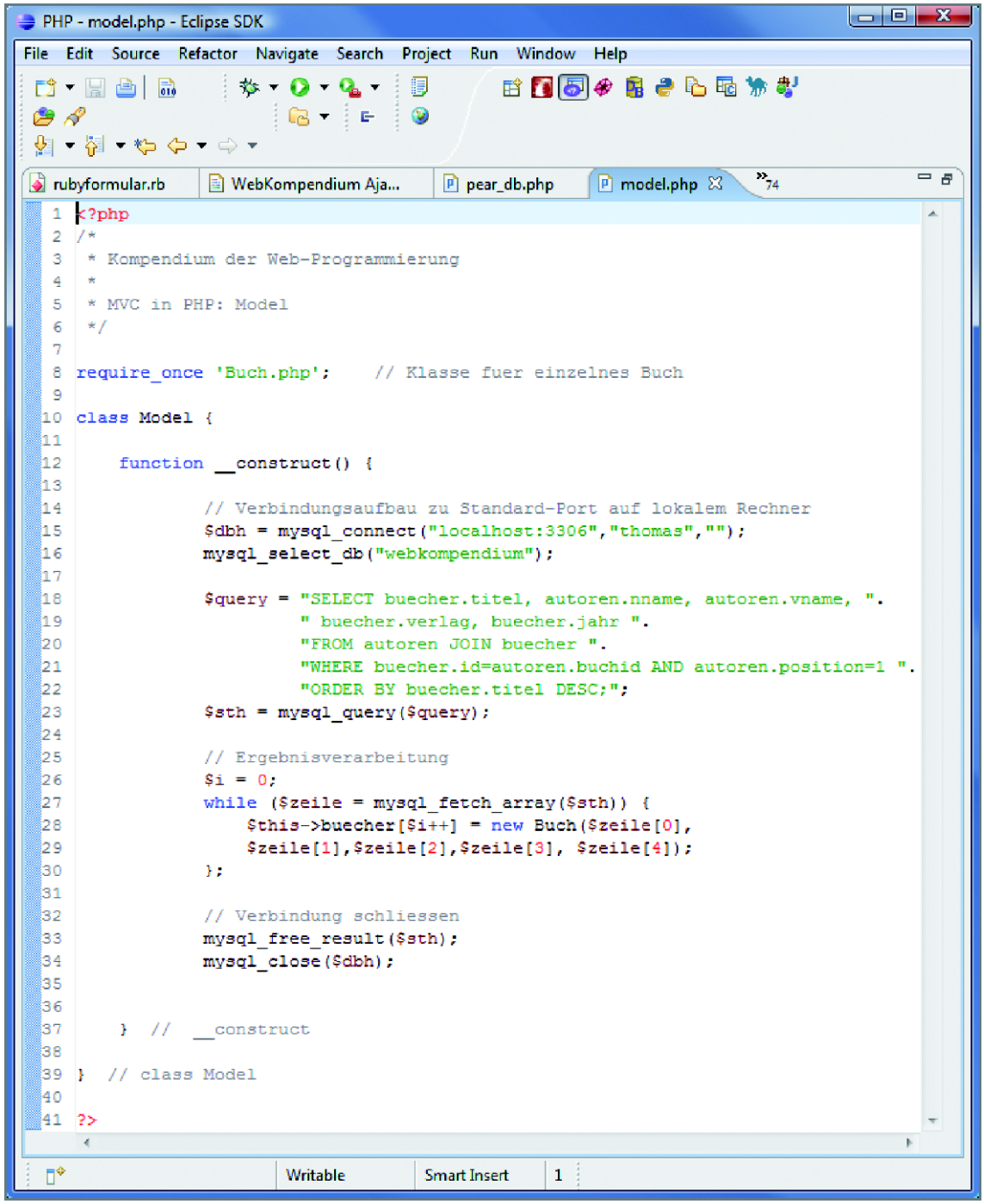
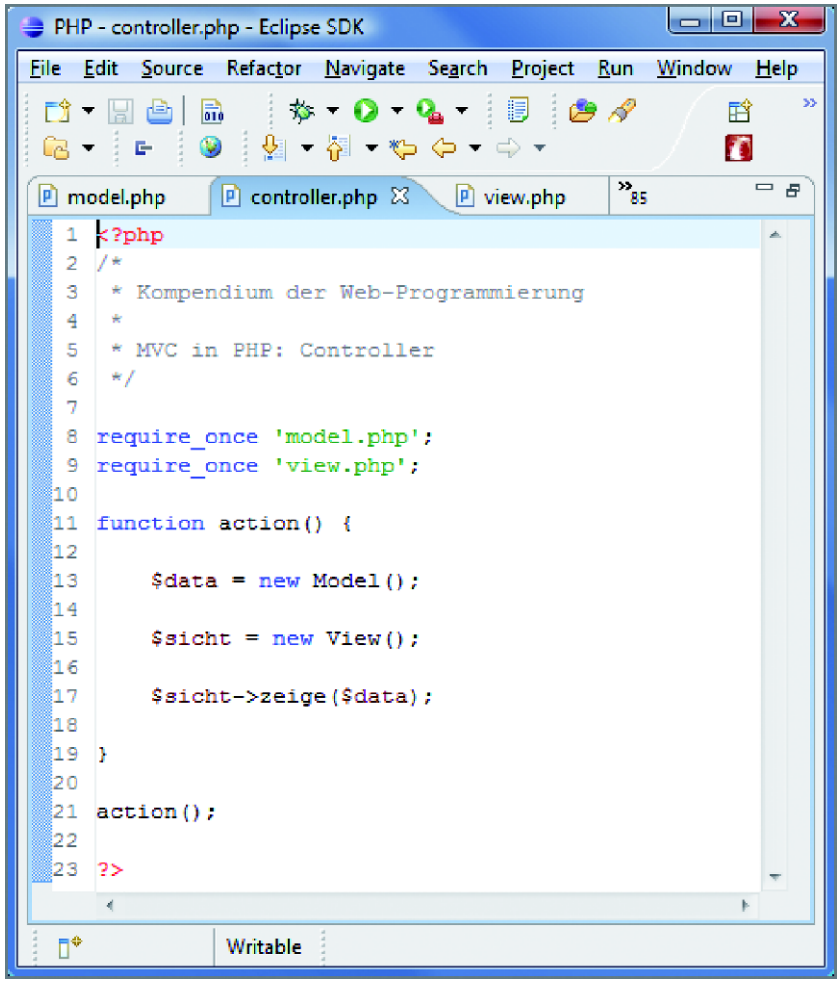



**Abbildung 11.53:**
Die Klasse `Model` im Script `model.php`




```php
<?php
/*
 * Kompendium der Web-Programmierung
 *
 * MVC in PHP: Controller
 */

require_once 'model.php';
require_once 'view.php';

function action() {

    $data = new Model();

    $sicht = new View();

    $sicht->zeige($data);

}

action();

?>
```

**Abbildung 11.54:**
Das Script `controller.php`

Das Resultat der MVC-Anwendung erhält man nun durch einen einfachen GET-Request an den Controller

```
http://<host>/controller.php
```

und ist in Abbildung 11.56 zu sehen; im Kern entspricht dies der Abbildung 11.37 der unstrukturierten PHP-Anwendung `buecher_1.php`.

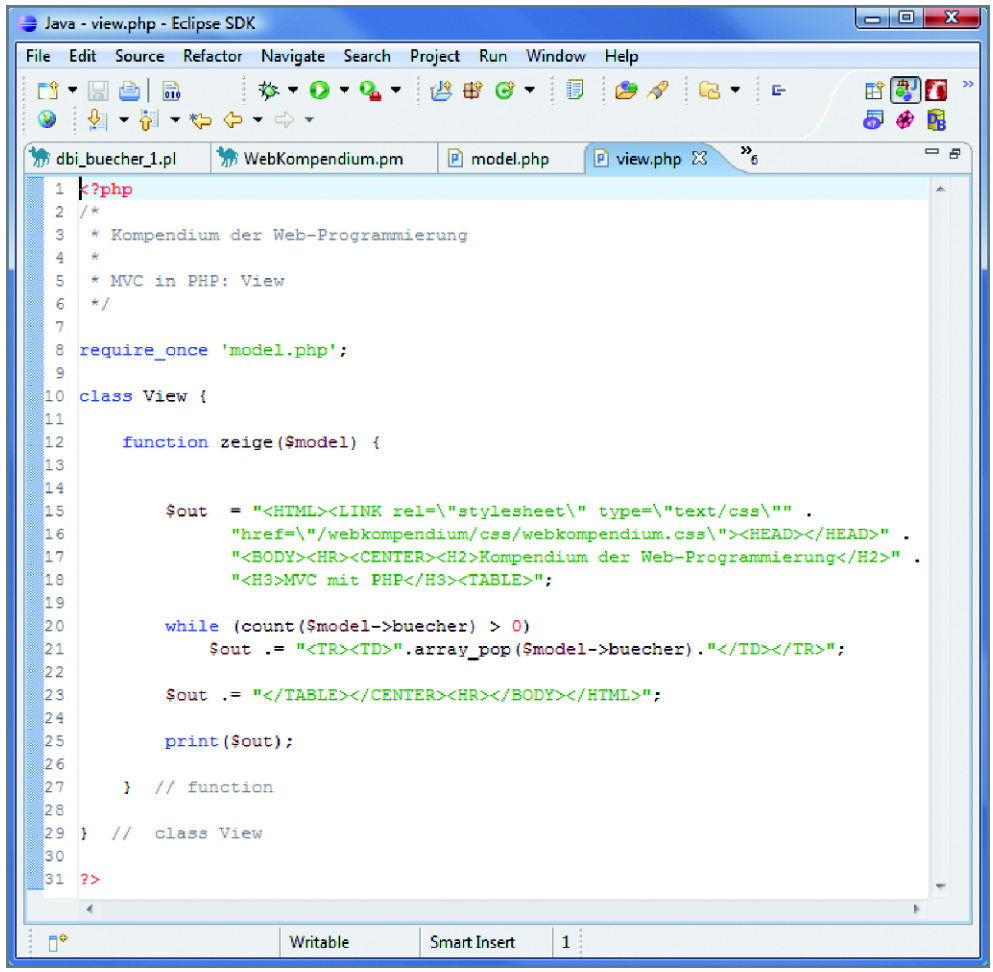



**Abbildung 11.55:**
Die Klasse `View` im Script `view.php`

**Abbildung 11.56:**
HTML-Response der View nach Request an den Controller

Einen Schritt weiter in der Entwicklung strukturierter Software für das Web führen nun die Template-Engines: Hier wird die View durch ein HTML-Gerüst, das Template, generiert, welchem wenige aktive Inhalte wie die anzuzeigende Variable übergeben werden. Für PHP ist Smarty hierfür ein typisches Beispiel, vgl. 21.

## 11.7 Erweiterungen von PHP: PEAR und PECL

Die wesentlichen Sprachbestandteile von PHP wurden hier vorgestellt, es fehlt bisher aber eine Möglichkeit, PHP durch Zusatzmodule zu erweitern und anzupassen. Hierfür gibt es das PEAR-Framework, welches einen Paketmanager ähnlich wie `ppm` für Perl (vgl. 10.5.6) bereitstellt. Auf PEAR wird in Kapitel 20 genauer eingegangen.

#### 11.7.0.1 Universeller Datenbankzugriff: PEAR::DB und PEAR:MDB

Eines der vielen PEAR-Module ist `PEAR::DB`, welches anders als hier bisher vorgestellt einen Zugriff auf ein Datenbankmanagementsystem erlaubt, welcher unabhängig vom jeweiligen DBMS ist. Dieses Modul und seine Fortentwicklung `PEAR:MDB` wird in Abschnitt 20.5 behandelt.

## 11.8 Universeller Datenbankzugriff: PHP Data Objects PDO

*PHP Data Objects* (PDO) sind neben PEAR::DB ein weiterer neuer Weg, wie PHP mittels einer Abstraktionsschicht unabhängig vom jeweiligen Datenbankmanagementsystem mit Datenbanken arbeiten kann (PDO gehört zum in C implementierten PECL-Framework). Aktuell sind PDO-Treiber für die folgenden DBMS verfügbar:

- Informix;
- MySQL
- Oracle
- ODBC;
- PostgreSQL;
- SQLite;
- Firebird;
- FreeTDS/Microsoft SQL Server/Sybase (gleicher Treiber).

Vor Verwendung von PDO ist dieses sowie die DBMS-spezifischen PDO-Treiber zu laden, was wieder statisch und dynamisch geschehen kann, etwa dynamisch in der zentralen PHP-Konfiguration `php.ini` durch die Konfiguration

```
[PHP_PDO]
extension=php_pdo.so
extension=php_pdo_mysql.so
```

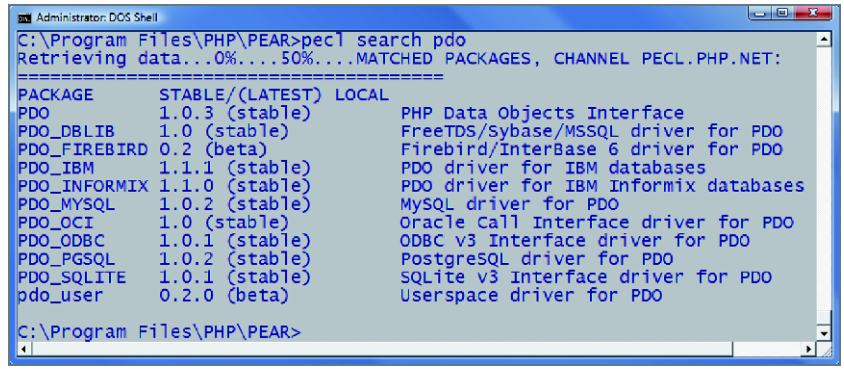



**Abbildung 11.57:**
Suche nach PDO-Bibliotheken

Der Verbindungsaufbau mittels PDO ist wie bei DBI für Perl die einzige Stelle, die auf das jeweilige DBMS Bezug nimmt, etwa:

```
try {
   $dsn  = 'mysql:host=localhost;dbname="webkompendium"';
   $user = "thomas";
   $pw   = "";
   $dbh = new PDO($dsn,$user,$pw);
} catch (PDOException $e) {
   print "PDO-Fehler: " . $e->getMessage() . "<HR>";
   die();
}
```

Für eine bestehende Verbindung wird, ähnlich zu DBI, entweder mittels `query` eine SQL-SELECT-Abfrage (Rückgabewert: Resultset) oder mittels `prepare` und `exec` (Rückgabewert: Anzahl der veränderten Zeilen) eine andere SQL-Anweisung ausgeführt.

PDO ist ein sehr leistungsstarkes Instrument, welches etwa auch Transaktionen unterstützt. Es bleibt abzuwarten, welcher Weg der Datenbankabstraktion sich für PHP in Zukunft etablieren wird, mit PDO und PEAR:DB stehen jedenfalls zwei gute Möglichkeiten der DBMS-Abstraktion für PHP zur Verfügung.

# 12

# Python

## 12.1 Die Scriptsprache Python

Perl ist eine sehr verbreitete Scriptsprache – sie hat aber deutliche Grenzen, etwa durch eine vergleichsweise alte Syntax und eine nicht gelebte Objektorientierung. Es gibt einige modernere Scriptsprachen, die diese Lücken schließen; hierzu zählt *Python.*

**Abbildung 12.1:**
neues Python-Logo

Die Scriptsprache Python wurde um 1989 von Guido van Rossum in den Niederlanden am Centrum voor Wiskunde en Informatica (CWI) in Amsterdam entwickelt. Van Rossum ist ein Anhänger der britischen Komikergruppe Monty Python[1] (Abbildung 12.2), woher der Name der Programmiersprache stammt. Dennoch wurde lange eine Schlange als Icon für die Sprache verwendet.

Der Python-Interpreter ist frei für alle relevanten Betriebssysteme verfügbar. Im Web [web] sind alle für Python benötigten Ressourcen sowie eine umfassende Online-Dokumentation zu finden, eine umfassende gedruckte Dokumentation zu Python ist etwa [Lut06].

Heute ist Python noch in einem weiteren Kontext von Bedeutung: Das umfassende Web-Framework ZOPE, auf welchem etwa das Content-Management-System PLONE (vgl. 28.1.3) basiert, ist in Python geschrieben und die Entwicklung beider hängt eng zusammen (vergleichbar etwa zu Unix und C in den Jahren um 1970). Neben ZOPE/PLONE gibt es einige weitere aktuelle Python-basierte Web-Frameworks, etwa django (vgl. Kapitel 22).

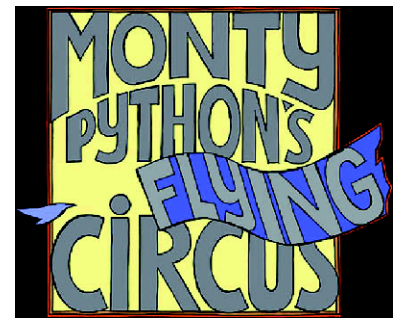



**Abbildung 12.2:**
Monty Python

### 12.1.1 Vorteile von Python

Python verfügt über zahlreiche Vorteile, die es auch gegenüber Perl absetzen; zu den wichtigsten zählen:

- Python ist eine moderne Scriptsprache („Perl++“);
- die Sprache ist plattformunabhängig, da die Python-Codes interpretiert werden;
- im Gegensatz zu Perl ist Python in der Lage, vorcompilierten Code für den Interpreter direkt abzuspeichern, was die Performance deutlich erhöht;
- Python ist objektorientiert;
- Python ist netzwerkzentriert;
- Python ist für CGI geeignet – aber auch für mehr;
- ähnlich zu Perl sind für Python sehr viele Module verfügbar;
- Python ist frei unter der GPL.

[1] bekannt etwa durch den Film „Das Leben des Brian“.

## 12.2 Installation und Entwicklungsumgebungen

### 12.2.1 Installation von Python

Python selbst wird über die Python-Site [web] angeboten; dort ist für Unix/Linux eine Source-Distribution zu finden, für Windows steht dank der Unterstützung von Mark Hammond eine msi-Installerdatei bereit.

Für die gängigen Linux-Distributionen steht natürlich auch die einfache Möglichkeit der Paketinstallation mit Tools wie `pkgadd` oder `aptitude` bereit. Zudem bietet ActiveState ähnlich wie für Perl ein freies Gesamtpaket mit Dokumentation an.

Ebenfalls verfügbar ist die Dokumentation zum lokalen Download, was die Arbeit erleichtert (Abbildung 12.3).

**Abbildung 12.3:** Lokale Python-Dokumentation im Browser

### 12.2.2 Entwicklungsumgebungen für Python

Für Python sind einige *Entwicklungsumgebungen* verfügbar; einfache gehören sogar direkt zur Distribution. Empfehlenswert ist wieder ein für die Entwicklungsumgebung Eclipse (vgl.8.1) verfügbares Plugin: pydev [web].

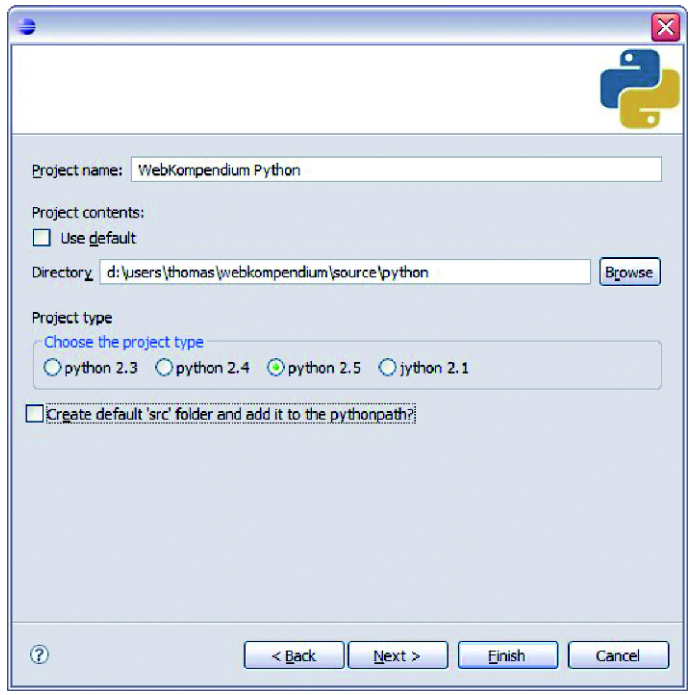



**Abbildung 12.4:**
Anlegen eines Python-Projektes in Eclipse mit pydev

pydev kann über den Eclipse-Update-Manager einfach installiert werden; es stehen damit Python-Projekte und eine spezielle Python-Perspektive zur Verfügung. Der Python-Interpreter muss unter Options|Python vorgegeben werden. pydev verwendet zur Kennzeichnung von Python – etwa für die zugehörige Perspektive – das neue Logo von Abbildung 12.1.

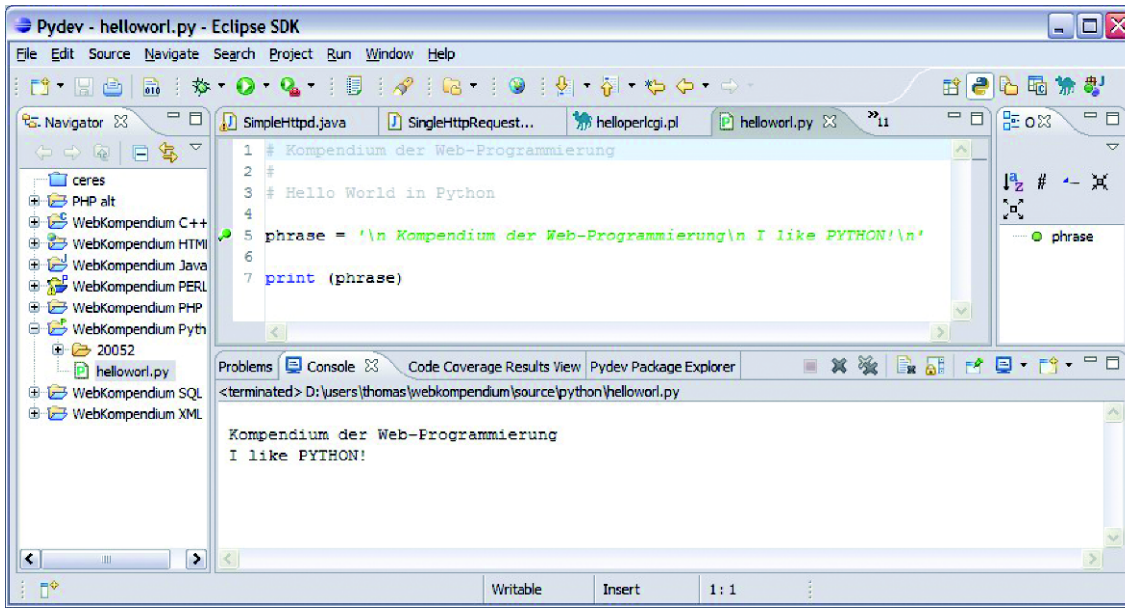



**Abbildung 12.5:**
Das Eclipse-Plugin pydev

## 12.3 Grundlegende Syntax der Scriptsprache Python

Einige der Sprachbestandteile von Python sind allgemein typisch für Scriptsprachen wie in Perl und können hier mit den entsprechenden Verweisen kürzer behandelt werden; anderes – insbesondere die einfache Blockstruktur – ist grundlegend verschieden und eine Spezialität dieser Sprache.

### 12.3.1 Ausführen von Python-Scripten

Der Python-Interpreter kann direkt ein Python-Script ausführen – so wie es der Perl-Interpreter auch kann. Allerdings bietet Python als weitere Besonderheit das „zeilenweise“ Ausführen in einem interaktiven Modus (Abbildung 12.6). Dieser wird mit der Tastenkombination <STRG>-Z oder <STRG>-D verlassen. Komfortabler ist der zur Python-Distribution gehörende Python-GUI mit dem schönen Namen IDLE.

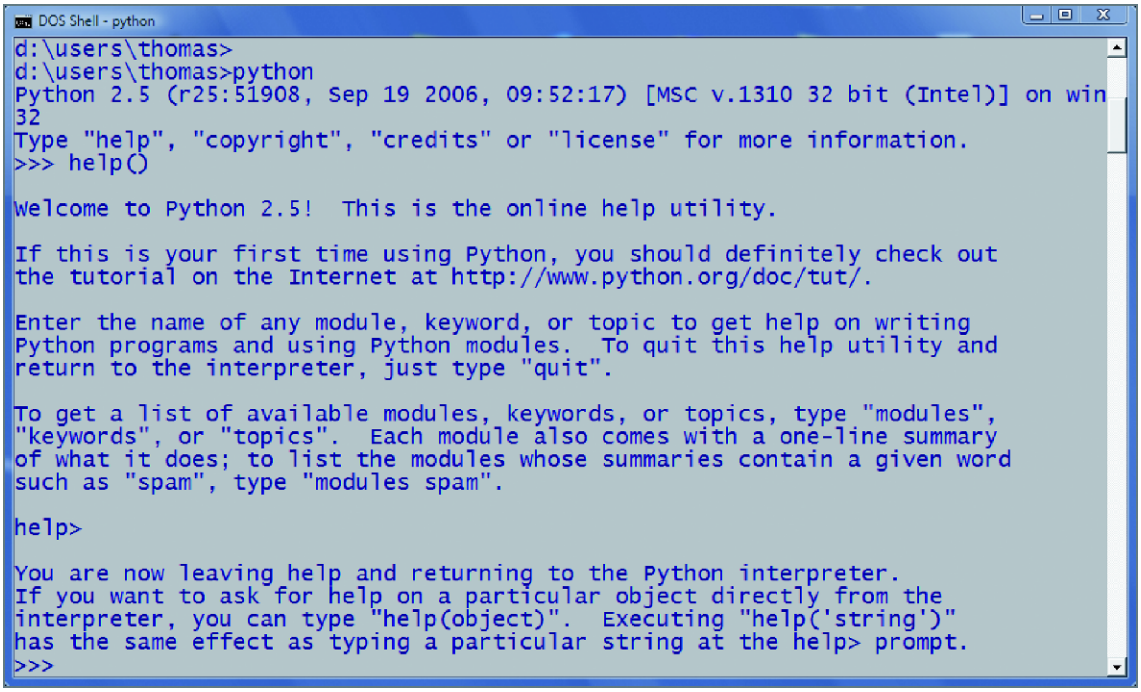



**Abbildung 12.6:** Der interaktive Python-Interpreter

Auch die Entwicklungsumgebung pydev erlaubt natürlich das einfache und direkte Ausführen von Python-Scripten.

### 12.3.2 HelloWorld mit Python

Ein einfaches *HelloWorld-Programm* mit Python sieht folgendermaßen aus:

```
# Kompendium der Web-Programmierung
#
# Hello World in Python

phrase = "\n Kompendium der Web-Programmierung\n " + \
            "I like PYTHON!\n"
```

```
print(phrase)
```

Dieser Code in einer Datei `helloworld.py` kann direkt mit dem Interpreter ausgeführt werden (Abbildung 12.7).

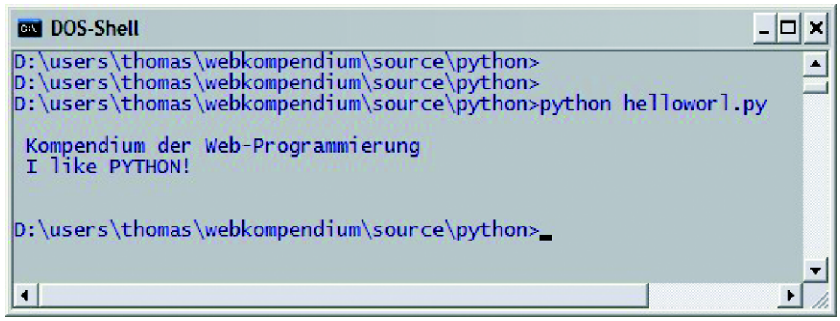



**Abbildung 12.7:**
Ausführen von `helloworld.py`

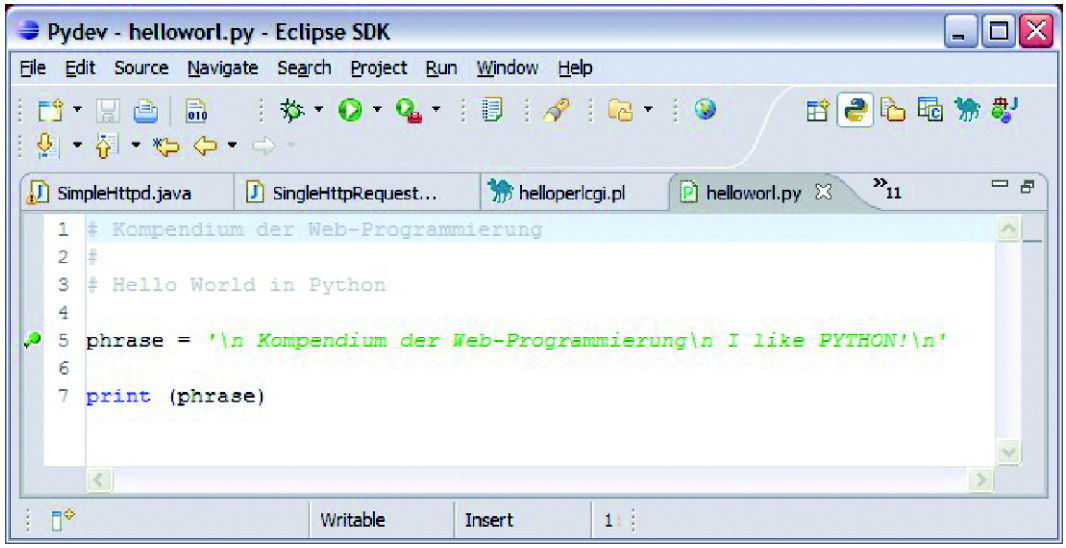



**Abbildung 12.8:**
`helloworld.py` in pydev

Auffällig gegenüber den gewohnten Scriptsprachen Perl und PHP sind schon an diesem Beispiel mehrere Punkte:

- Anweisungen werden nicht mehr mit ; beendet;
- Variablenbezeichner – hier `phrase` – beginnen nicht mehr mit dem üblichen $;
- ...aber die Kommentare werden vertraut mit # eingeleitet.

Im nächsten Schritt erfolgt eine einfache Python-Anwendung zur Kreisberechnung, vergleichbar zum Perl-Beispiel `kreis.pl`. Hierfür ist die Tastatureingabe notwendig, die in Python mittels `input` oder `raw_input`

```
input(prompt)
raw_input(prompt)
```

geschieht; dabei ist `prompt` Bezeichner einer Variable, deren Wert als Eingabeaufforderung ausgegeben wird.

Damit können im Python-Script `kreis.py` (Abbildung 12.9) nach Eingabe des Radius die entsprechenden Werte berechnet werden (Abbildung 12.10).

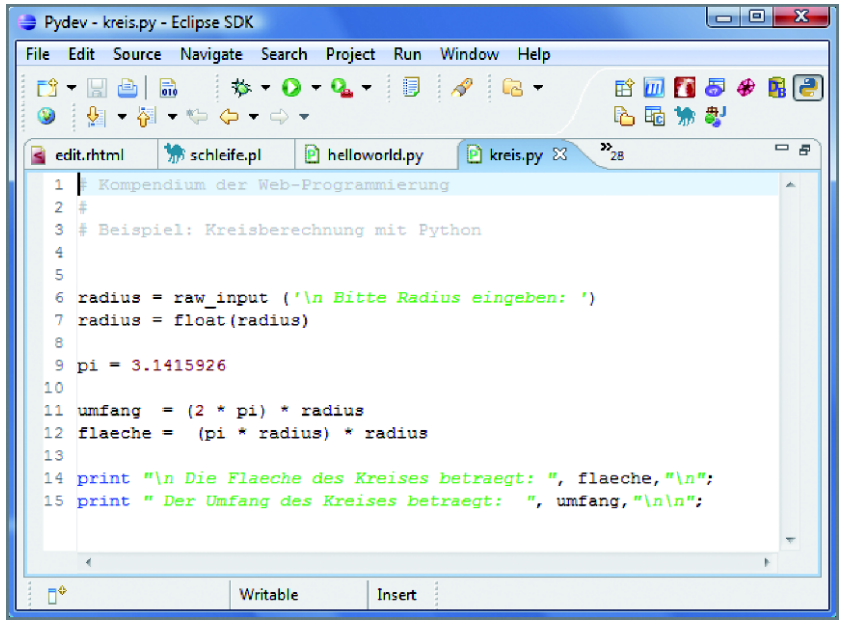



**Abbildung 12.9:** Kreisberechnung mit Python (Script `kreis.py`)

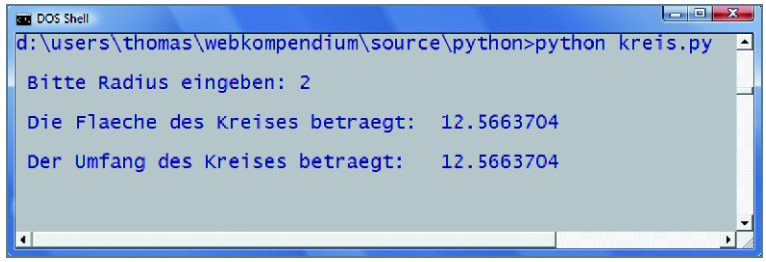



**Abbildung 12.10:** Ausführen von `kreis.py`

### 12.3.3 Python-Interpreter und Bytecode

Der *Python-Interpreter* arbeitet beim Ausführen eines Python-Scriptes zunächst klassisch:

1. Einlesen der Datei (bzw. der interaktiven Eingabe) als 7-bit-ASCII-Strom;
2. lexikalische Analyse;
3. syntaktische Analyse durch Parser, Übersetzung in Bytecode;
4. Interpretation des Bytecodes.

Der Python-Interpreter erlaubt eine Vielzahl von Parametern, eine Übersicht gibt Abbildung 12.11; viele, etwa `-w`, arbeiten gleich zu Perl.

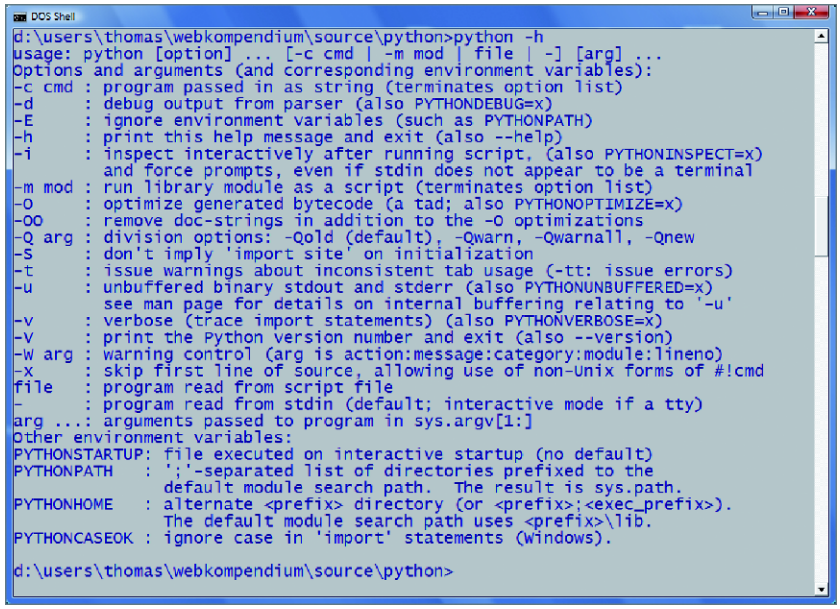



**Abbildung 12.11:**
Parameter des Python-Interpreters

Soweit arbeitet Python gleich wie Perl, als Konsequenz treten niemals zur Laufzeit Syntax-Fehler auf, da bereits vor dem Starten des Programmes der gesamte Code übersetzt ist.

Eine wesentliche Besonderheit von Python ist aber die Möglichkeit, den im dritten Schritt gewonnen Bytecode separat abzuspeichern. Dieser Bytecode wird üblicherweise in Dateien mit der endung `.pyc` abgelegt und ist vollständig plattformunabhängig. Er wird auf der jeweiligen Plattform dann nur noch im jeweiligen Interpreter ausgeführt.

**Dieses Vorgehen von Python entspricht genau dem Java-Prinzip: Der Python-Bytecode pyc entspricht der class-Datei von Java, der Interpreter hat dann die Rolle der Java Virtual Machine JVM.**

Der compilierte pyc-Code wird etwa für das Verteilen von Python-Modulen genutzt.

### 12.3.4 Schlüsselwörter, Semikolon und Kommentare

Python verfügt über vergleichsweise wenige Schlüsselwörter (25 Stück). Das Semikolon wird in Python in der Regel nicht verwendet, um das Ende einer Anweisung zu kennzeichnen, es kann aber verwendet werden; eine Anweisung endet einfach am Zeilenende. Um diesen Automatismus für lange Anweisungen zu umgehen, können Zeilen mittels \ umgebrochen werden.

Kommentare in Python werden wie in Perl durch # eingeleitet; die beliebten Möglichkeiten mittels `//` oder `/* ... */` bestehen in Python nicht.

### 12.3.5 Elementare Datentypen und Literale in Python

#### 12.3.5.1 Numerische Werte

Python verwendet intern vier Typen *numerischer Daten:*

- Ganze Zahlen (plain integer): diese werden wie üblich als `-42` oder `313` geschrieben; oktale Codierungen beginnen mit führender „0“, etwa `042`, hexadezimale Darstellungen werden mit „0x“ eingeleitet wie `0xCAFE`.

- Lange ganze Zahlen (long integer) werden intern mit 64 statt 32 bit dargestellt; entsprechende Literale enden mit „L“: `4711L`.
- Gleitkommazahlen (float) werden intern nach IEEE 754 dargestellt: `3.1415926` oder `-1.602E-19`.
- Komplexe Zahlen (imaginary): Da van Rossum aus einer elektrotechnischen Einrichtung stammt, kommt den komplexen Zahlen eine besondere Bedeutung zu.[2] Deshalb sind die komplexen Zahlen direkt in Python als elementarer Datentyp implementiert, und wie bei Ingenieuren üblich wird `j` als Kennzeichnung der komplexen Einheit verwendet: `42 - 313j`.

#### 12.3.5.2 Zeichenketten

*Zeichenketten* werden durch ´ oder durch ¨ begrenzt; sollen sie über mehrere Zeilen reichen, werden die Begrenzer zu Beginn und am Ende dreifach verwendet.

Eine Unterscheidung zwischen diesen beiden Möglichkeiten – so wie in Perl und PHP – ist hier nicht notwendig, da keine Variablenersetzung möglich ist, weil das $-Symbol nicht verwendet wird.

Es sind die üblichen Escape-Sequenzen möglich, etwa `\n` für eine neue Zeile. Wird eine Zeichenkette vor ihrem Anfang durch `r` gekennzeichnet, so ist es eine raw-Zeichenkette, und die Escape-Sequenzen werden nicht ersetzt, etwa bei:

```
r'Neue Zeile wird mit \n erzeugt'
```

Zeichenketten werden wie in Java durch + zusammengefügt und können auch „multipliziert“ werden:

```
print 5 * 'Web-Kompendium'
```

gibt die Zeichenfolge „Web-Kompendium“ fünf mal aus.

Die neueren Python-Versionen sind Unicode-fähig, allerdings ist eine Unicode-Zeichenfolge vor ihrem Anfang durch `u` zu kennzeichnen. Ein Unicode-Zeichen wird wie in Java etwa durch

```
\u20AC
```

für das Euro-Symbol „€“ erzeugt.

Für Zeichenketten steht die nützliche Funktion `len` zur Längenbestimmung bereit, auf die einzelnen Zeichen kann über den Index im Sinne eines Arrays zugegriffen werden:

```
s[i]     ### Zeichen mit Index i, wobei erstes Zeichen Index 0 hat
s[i:j]   ### Teilzeichenkette von Zeichen i (inklusive) bis j (exklusive)
s[:i]    ### Teilzeichenkette bis Position i-1
s[i:]    ### Teilzeichenkette ab Position i
```

#### 12.3.5.3 Einfache Arithmetik in Python

Python verfügt über die übliche *Arithmetik* für die elementaren Datentypen; ein paar Besonderheiten bestehen aber doch:

[2] einfache Darstellung des Wechselstroms durch die komplexe Exponentialfunktion.

- `a+b`, `a-b`, `a*b`, `a/b` und `a%b` sind wie gewohnt und üblich;
- `a**b` steht wie in Fortran und Perl für die Potenz $a^b$;
- `abs(a)` für den Absolut-Betrag $|a|$;
- `int(a)`, `float(a)` und `long(a)` sind explizite Typumwandlungen;
- `complex(a, b)` erzeugt eine komplexe Zahl mit der Belegung `a + j b`;
- `hex(i)`, `oct(i)` und `str(i)` erzeugen die hexadezimale, die oktale und die Zeichenketten-Darstellung von `i`.

#### 12.3.5.4 Typüberprüfung in Python

Mittels der Methode `type` kann Python einfach den aktuellen Typ einer Variablen ermitteln. Abbildung 12.12 zeigt dies im interaktiven Modus.

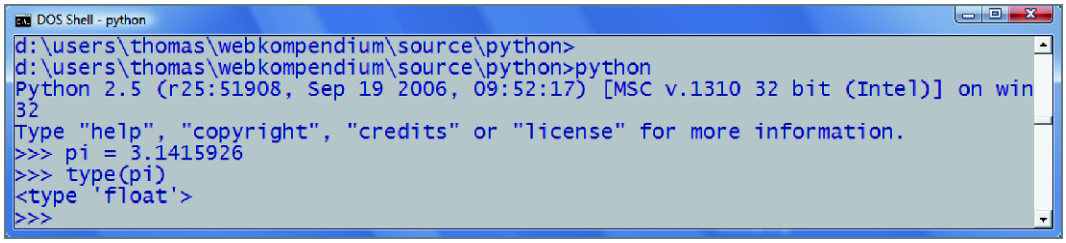



**Abbildung 12.12:**
Die Python-Methode `type`

### 12.3.6 Logik in Python

Python verfügt als klassische Scriptsprache nicht über einen expliziten Typ boolean; alle Variablenwerte können im booleschen Sinne verwendet werden: 0 und die leere Zeichenkette sowie die Referenz nach none sind false, alles andere wird als true interpretiert.

Es gibt die üblichen logischen Verknüpfungen `or`, `and` und `not` sowie Vergleichsoperationen wie == und <=.

Beim Vergleich ist wieder der Datentyp zu beachten; so ist natürlich

```
42 < 313
```

wahr, während

```
"42" < "313"
```

falsch ist, da hier ein lexikalischer Vergleich erfolgt.

### 12.3.7 Listen in Python

Python verfügt wie üblich für Scriptsprachen über einen lässigen Umgang mit Arrays, die direkt als *lineare Listen* implementiert sind; diese Listen können Daten unterschiedlichen Typs aufnehmen.

Eine Liste wird durch die Syntax

```
liste = [1,2,3,"vier"]
```

erzeugt; über die Klammer-Syntax `[]` kann auf das jeweilige Element direkt zugegriffen werden.

Diese Listen können direkt erweitert werden: Die Python-Methode `append` entspricht dem Ablegen auf einer Liste wie push. Zum Entnehmen von der

Liste steht die Methode `pop` direkt zur Verfügung, somit verfügt Python über die klassische LIFO-Struktur.

Das Script `liste.py` (Abbildung 12.13) zeigt eine exemplarische Anwendung für Listen in Python.

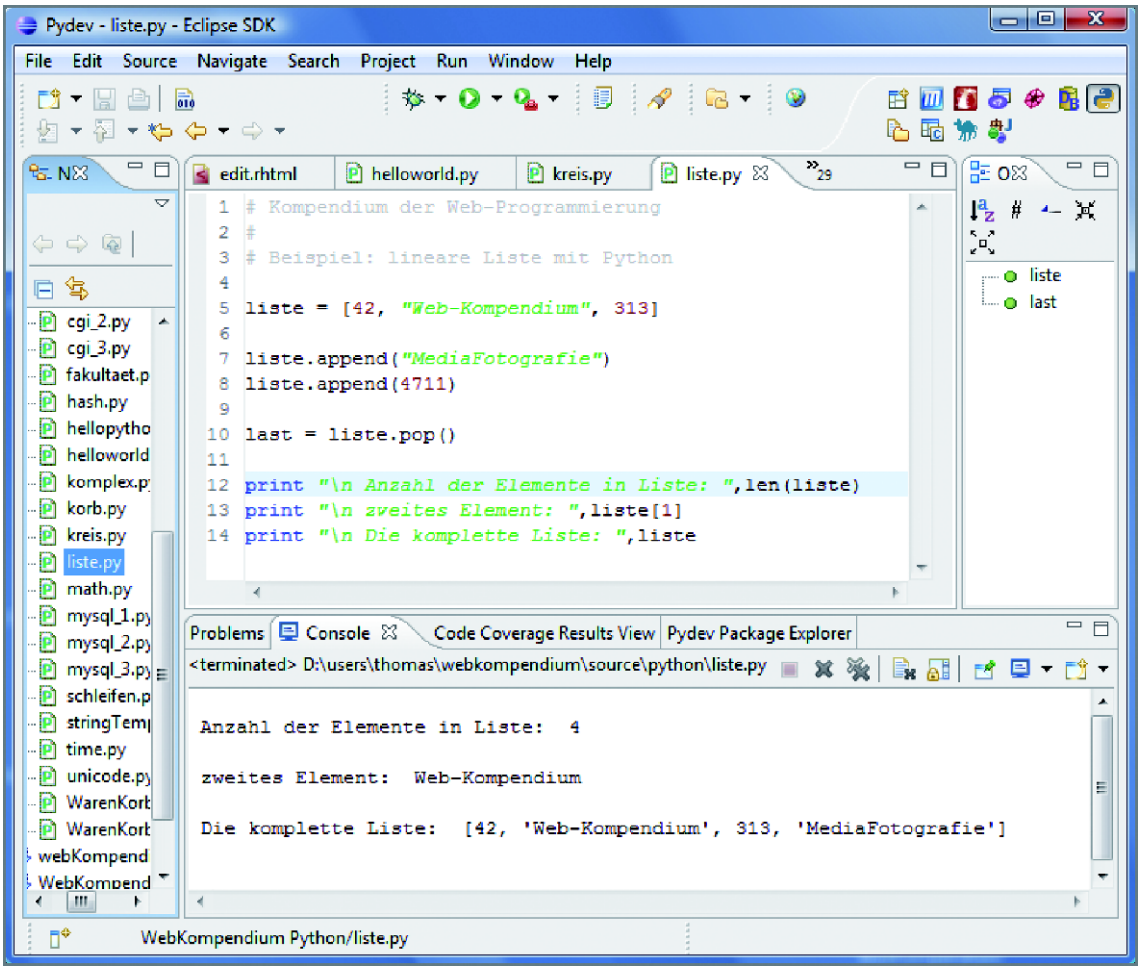



**Abbildung 12.13:** Das Script `liste.py`

Für Listen stellt Python auch spezielle Methoden bereit, etwa `sort` zum Sortieren oder `reverse` zur umgekehrten Sortierung.

### 12.3.8 Assoziative Listen in Python

Python verfügt natürlich auch über eine starke Syntax für den Umgang mit *assoziativen Listen:* eine Sammlung von Key-Value-Paaren läßt sich mittels einer Struktur

```
hash = { key1:value1, key2:value2, ... }
```

verwalten; der value zu einem key ist dann über

```
hash[key]
```

zugreifbar. Das Script `assozListe.py` (Abbildung 12.14) zeigt schematisch den Einsatz solcher Listen.

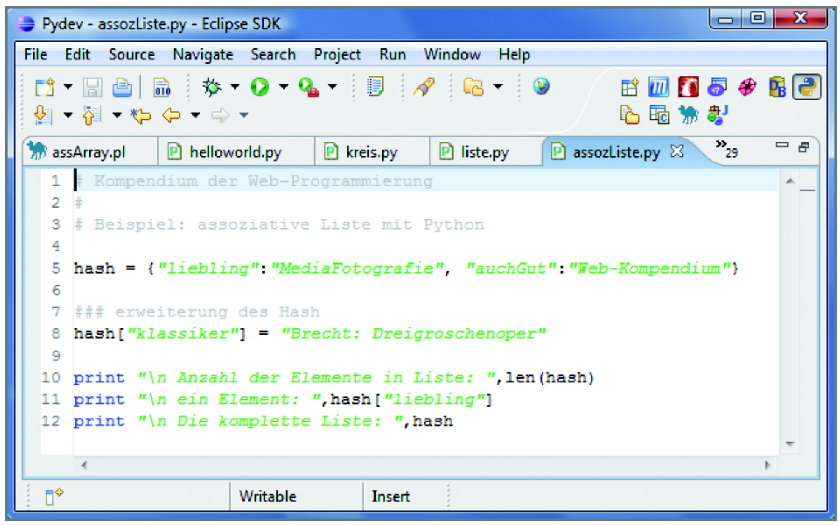



**Abbildung 12.14:**
Das Script `assozListe.py`

Für assoziative Listen stellt Python eine Reihe nützlicher Methoden bereit, etwa

- `items()` erzeugt eine lineare Liste mit 2-Tupeln der Key-Value-Paare;
- `keys()` erzeugt eine Liste nur mit den Keys;
- `values()` erzeugt eine Liste nur mit den Values;
- `has_key(k)` überprüft, ob der Key `k` vorhanden ist;
- `update(hash2)` fügt die assoziative Liste `hash2` an.

### 12.3.9 Kontrollstrukturen in Python

Die *Kontrollstrukturen* in Python enthalten eine zentrale Besonderheit der Sprache, die schon beim einfachen Block auftritt:

#### 12.3.9.1 Der Block in Python

Blöcke werden in Python nicht wie üblich durch eine Klammersyntax oder mit begin-end gebildet: Blöcke werden durch einfaches Einrücken gebildet!

Damit unterscheidet sich Python wesentlich von allen anderen Sprachen; der Hintergrund, der Guido van Rossum zu dieser Regelung geführt hat, ist ein einfacher und einleuchtender: Während in den anderen Sprachen der Programmierer zwar aufgefordert ist, gut leserlichen Sourcecode durch Einrücken zu schreiben, wird er dazu in Python direkt gezwungen.

#### 12.3.9.2 Verzweigung in Python

Die einfachste Kontrollstruktur nach dem Block ist die Verzweigung, die in Python die einfache Syntax

```
if (boole):
    wahr_anweisungen
else:
    falsch_anweisungen
```

hat; der else-Teil ist natürlich wieder optional; `boole` ist dabei ein Ausdruck oder eine Variable, die im booleschen Sinne interpretiert wird. Komplexere Strukturen werden mittels weiterer, mit weiteren Bedingungen versehenen *elseif*-Teilen durch die Syntax

```
elif (boole2)
```

ermöglicht.

#### 12.3.9.3 Die while-Schleife in Python

Die *while-Schleife* in Python entspricht völlig dem Normalfall:

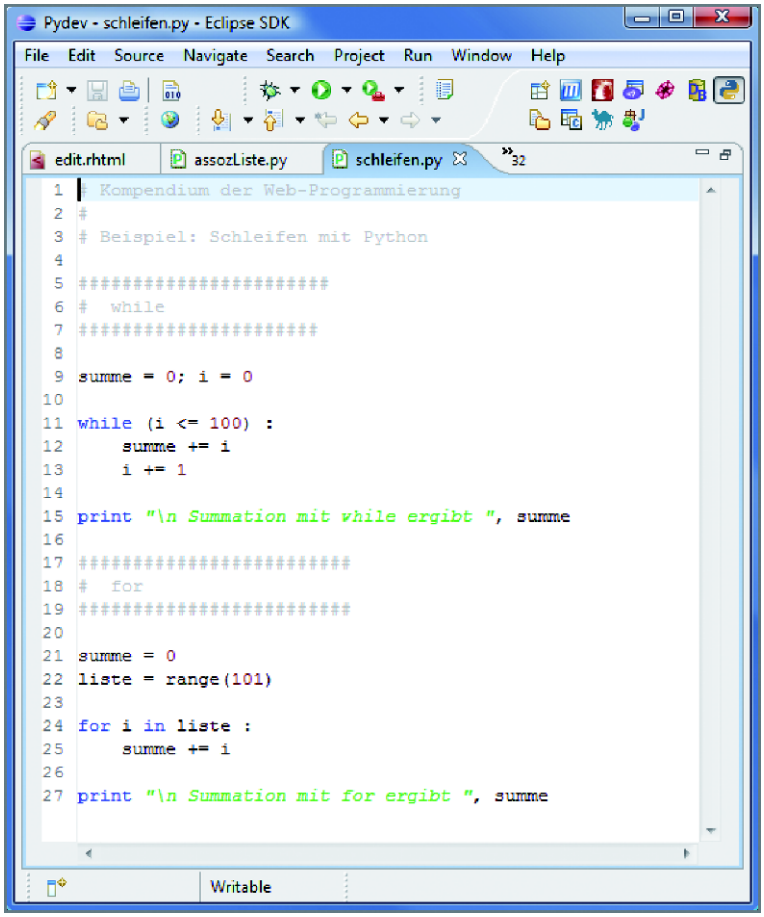
```
while boole :
      schleifen_anweisungen
```

#### 12.3.9.4 Die for-Schleife in Python

Im Gegensatz zur while-Schleife hat die *for-Schleife* in Python eine eigene Syntax, die eher der foreach-Schleife vieler anderer Scriptsprachen entspricht:

```
for variable in sammlung :
     schleifen_anweisungen
```

Dabei ist `sammlung` etwa eine lineare Liste nach Abschnitt 12.3.7, welche dann nach aufsteigendem Index durchlaufen wird; das Script `schleifen.py` (Abbildung 12.15) zeigt die Anwendung der `while`- und `for`-Schleife.




```
1  # Kompendium der Web-Programmierung
2  #
3  # Beispiel: Schleifen mit Python
4
5  ########################
6  #  while
7  ########################
8
9  summe = 0; i = 0
10
11 while (i <= 100) :
12     summe += i
13     i += 1
14
15 print "\n Summation mit while ergibt ", summe
16
17 ##########################
18 #  for
19 ##########################
20
21 summe = 0
22 liste = range(101)
23
24 for i in liste :
25     summe += i
26
27 print "\n Summation mit for ergibt ", summe
```

**Abbildung 12.15:**
Das Script `schleifen.py`: Summation $\sum_{i=0}^{100} i$ mittels `while` und `for`

#### 12.3.9.5 Beenden von Schleifen: break und continue

Mittels `break` wird die Ausführung einer Schleife direkt beendet, während `continue` lediglich mit dem nächsten Schleifendurchlauf fortfährt.

**`break` und `continue` von Python entsprechen den Anweisungen in Java.**

### 12.3.10 Methoden in Python

*Methoden* in Python werden durch das Schlüsselworf `def` definiert:

```
def funktionsName(argument1, ...) :
    anweisungen
```

Methoden in Python müssen allerdings vor ihrer ersten Verwendung bereits definiert sein. Das Schript `fakultaet.py` definiert so zwei Funktionen zur Berechnung der Fakultät iterativ und rekursiv (Abbildung 12.16).

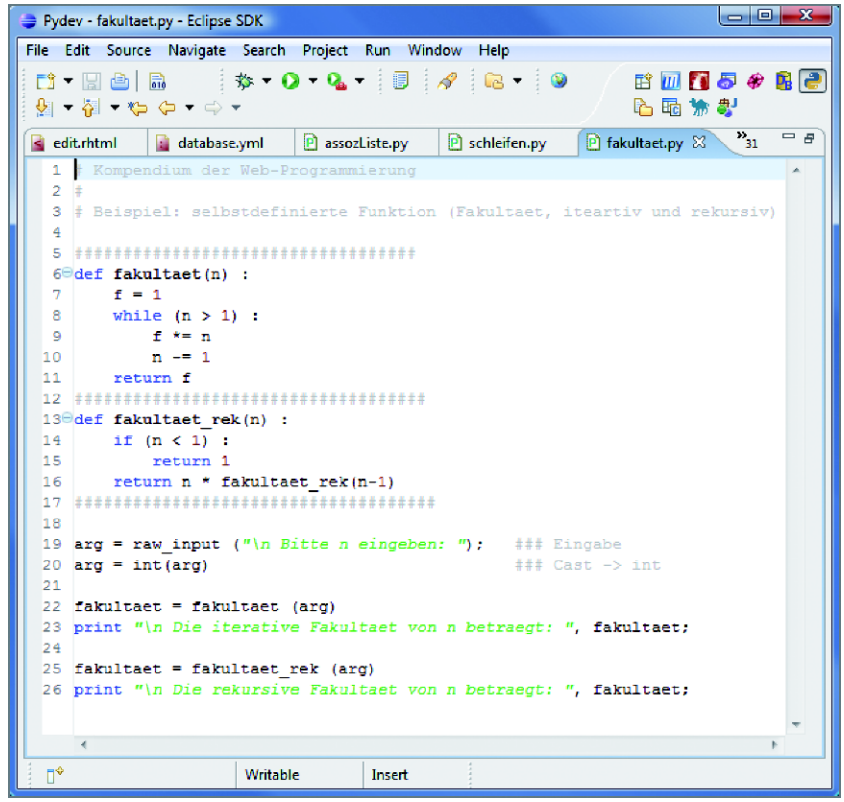



**Abbildung 12.16:** Das Script `fakultaet.py` definiert zwei Funktionen zur Berechnung der Fakultät iterativ und rekursiv

### 12.3.10.1 Argumentübergabe

Wie in PHP können auch in Python bei der Definition von Methoden den Argumenten Default-Werte zugeordnet werden:

```
def funktionsName (argument = "Web-Kompendium") :
      anweisungen
```

belegt die lokale Variable `argument` mit dem Wert „Web-Kompendium“, falls kein Wert übergeben wird.

Python kann auch mit variablen Argumentlisten arbeiten, also alle Argumente in eine Liste aufnehmen:

```
def gruss (*leute) :
     for member in leute :
               print "Hallo " + member
```

### 12.3.11 Namensräume in Python

Python verwendet drei unterschiedliche *Namensräume:*

- lokal: der Normalfall, Gültigkeit nur innerhalb des lokalen Blocks;

- modulweit: Gültigkeit im gesamten Python-Modul;
- Python-Systemvariablen haben globale Gültigkeit.

Durch das Schlüsselwort `global` wird eine Variable über ihren lokalen Block hinaus bekannt gemacht; die Methode `globals()` gibt alle globalen Variablen aus. Abbildung 12.17 zeigt die Bedeutung von beiden im interaktiven Modus.

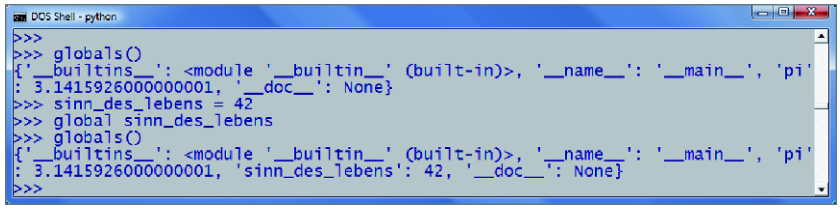



**Abbildung 12.17:** `globals()` und `global` im interaktiven Modus

### 12.3.12 IO in Python

Eine Eingabe über die Standardeingabe in Python lässt sich, wie bereits vorgestellt, einfach realisieren; Python stellt hierfür die Methoden

- `input('Prompt')`
- `raw_input('Prompt')`

bereit; beide können mit einem Argument für die Eingabeaufforderung versehen werden. Im Allgemeinen ist der Weg über `raw_input` zusammen mit einer expliziten Typzuordnung der sinnvollere.

#### 12.3.12.1 Dateizugriff mit Python

Mit Python steht wieder ein – im Vergleich zu Java – einfacher Weg des *Dateizugriffs* bereit. Mittels der Methode

```
open(datei,attribute)
```

wird die Datei, deren Name/Pfad in der Variablen `datei` abgelegt ist, in einem Modus geöffnet, der in der Variablen `attribute` hinterlegt ist. Für den Modus stehen zunächst die klassischen drei Möglichkeiten `r` für das Lesen, `w` für das Schreiben und `a` zum Anfügen bereit; mehr Flexibilität erlauben die Mischformen `r+`, `w+` und `a+`. Diese open-Methode hat als Rückgabewert einen File-Handle, welcher die geöffnete Verbindung verwaltet.

Zum Schließen eines Handles `f` muss die `close`-Methode auf diesen Handle angewendet werden:

```
f.close()
```

Ist eine Datei über den Handle `f` zum Lesen geöffnet, wird ähnlich wie in Perl zeilenweise gelesen. Hierfür gibt es die Methoden:

- `f.readline()`: liest eine Zeile;
- `f.readlines()`: liest alle Zeilen der Datei als Liste;
- `f.read(n)`: liest n Zeichen (ohne das Argument n wird die ganze Datei gelesen).

Das Schreiben in Python geschieht ebenfalls zeilenweise: über die Methoden

- `f.write(String)` wird eine Zeile geschrieben;
- `f.writelines(ListeVonStrings)` werden mehrere Zeilen geschrieben.

Zum Schreiben verwendet Python stets buffered streams; dies erhöht die Performance wesentlich, allerdings muss darauf geachtet werden, dass der Puffer geleert wird. Hierfür steht die Methode

```
f.flush()
```

bereit; beim Schließen des Handles `f` mittels `f.close()` wird der Puffer ebenfalls geleert.

### 12.3.13 Ausnahmebehandlung in Python: Exceptions

Python verwendet für das Fehlermanagement den Weg über Ausnahmen, die ausgelöst und abgefangen werden können.

**Exceptions in Python sind von Prinzip und Syntax gut mit Exceptions in Java vergleichbar.**

Python stellt zunächst ebenfalls eine Kontrollstruktur mit einem `try`- und einem `catch`-Block zur Verfügung: Der try-Block enthält die Anweisungen, welche eine Ausnahme auslösen können; tritt dieser Fall ein, wird der Block verlassen und in den catch-Block gesprungen. In beiden Fällen – mit und ohne ausgelöste Ausnahmen – kann ein weiterer Block folgen, der immer ausgeführt wird: `finally`. Es ist sogar ein `else`-Block möglich, der nur dann ausgeführt wird, wenn keine Ausnahme ausgelöst wurde, also das Gegenteil zum catch-Block.

Ausgelöst wird eine Exception in Python mit dem Schlüsselwort

```
raise
```

Der ausgelösten Exception können auch Argumente übergeben werden.

### 12.3.14 Objektorientierung in Python

Als moderne Scriptsprache bietet Python eine im Vergleich zu Perl weit entwickelte Objektorientierung. Hierzu zählen etwa:

- die Datenkapselung für geschützte Attribute ist vorhanden;
- eine Vererbung ist möglich (sogar Mehrfachvererbung);
- Konstruktoren und Destruktoren sind vorhanden.

#### 12.3.14.1 Definition einer Klasse in Python

Klassen in Python werden durch das Schlüsselwort

```
class
```

definiert; wie üblich in Python ist dann der Klassenrumpf eingerückt.

Die Klasse `Buch` ist im Script `Buch.py` enthalten (Abbildung 12.18); sie greift das Beispiel aus Kapitel 7 auf und definiert eine Klasse, welche ein Buch mit fünf Attributen beschreibt. Hier wird bereits das Modul `string` importiert, was in 12.4.1 genauer vorgestellt wird.

Die Umgebungsvariable `PYTHONPATH` legt fest, in welchen Ordnern Python nach Klassen sucht; sie entspricht damit der Umgebungsvariablen `CLASSPATH` von Java. Wird eine Klasse von einer anderen verwendet, erzeugt Python automatisch den benötigten pyc-Code.

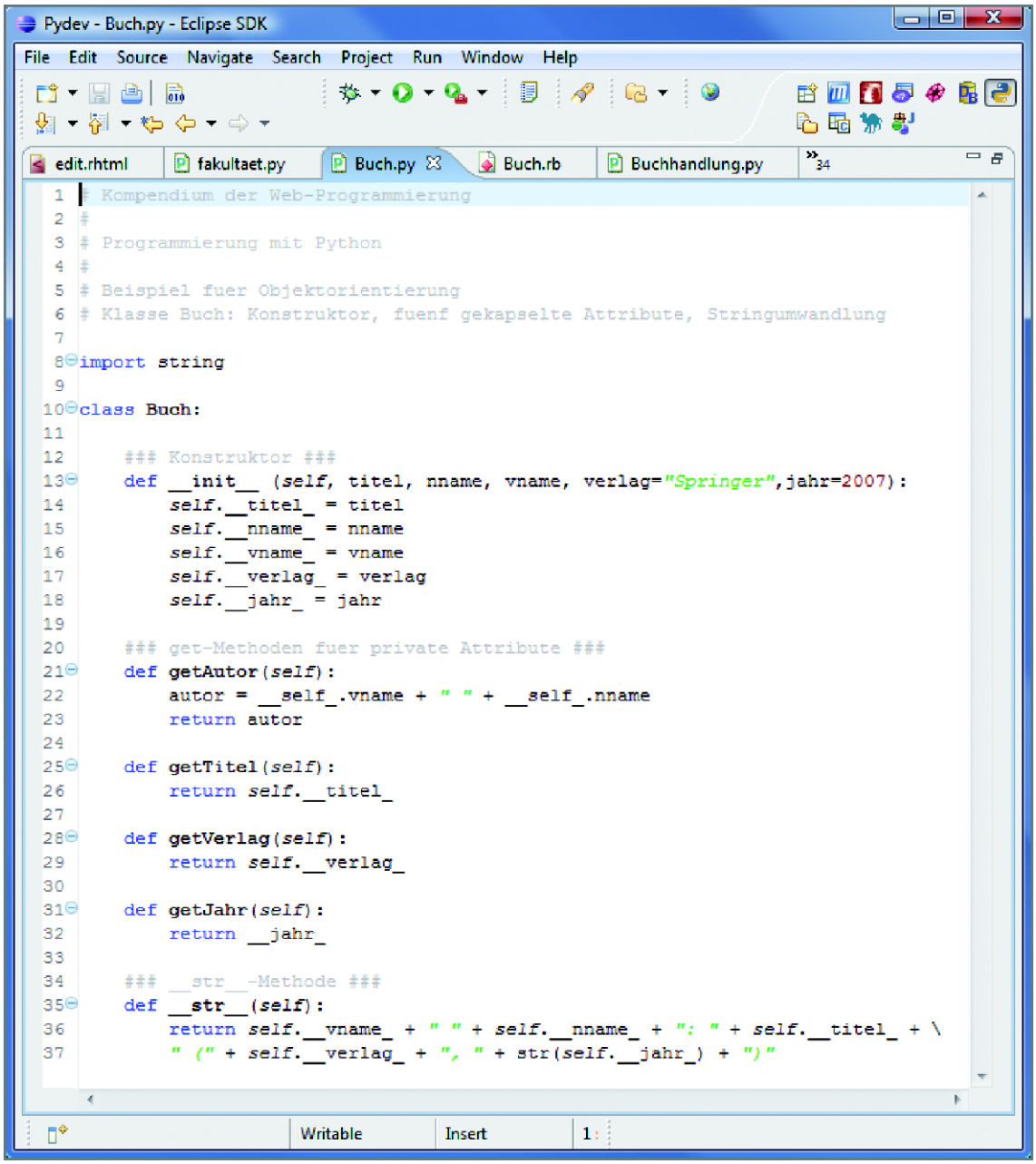



**Abbildung 12.18:**
Die Python-Klasse `Buch` im Script `Buch.py`

### 12.3.14.2 Instanziierung und die Referenz auf das Objekt selbst

In Python wird über den Klassennamen – ggf. mit Argumenten – eine Instanz einer Klasse erzeugt und die Referenz auf diese Instanz zurückgegeben. Im Script `buchhandlung.py` (Abbildung 12.19) wird so eine Instanz der Klasse `Buch` erzeugt (Abbildung 12.19).

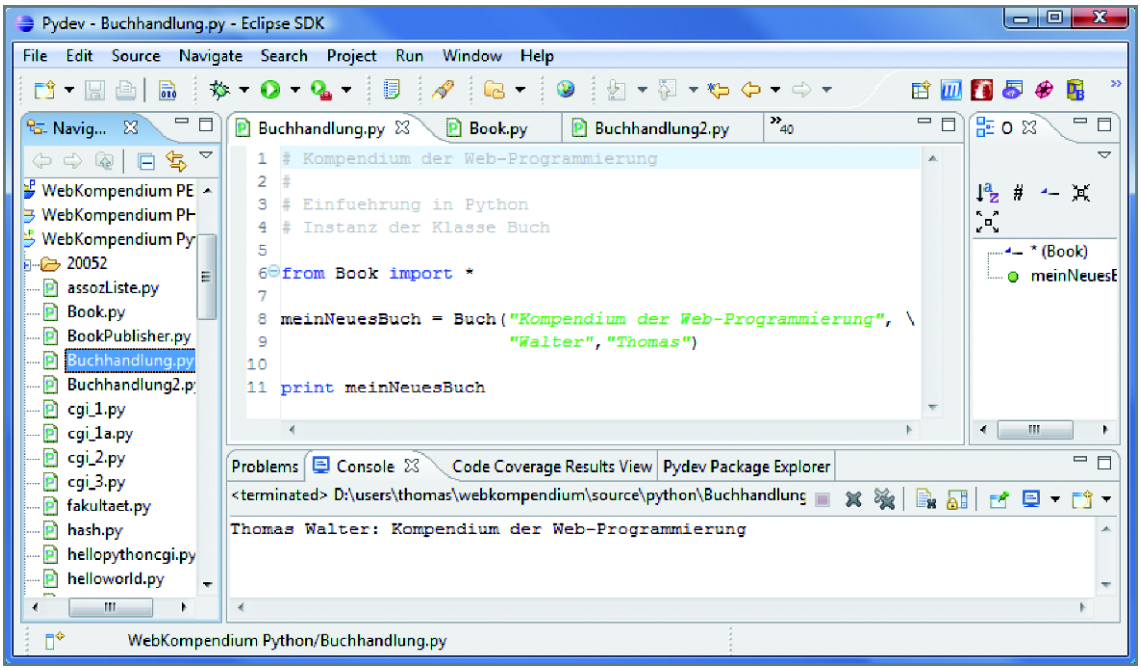



**Abbildung 12.19:** Erzeugung einer Instanz der Klasse `Buch` im Script `buchhandlung.py` und Ausführen in Eclipse mit pydev

Eine besondere Bedeutung hat die Referenz

```
self
```

welche mit dem `this` von Java verglichen werden kann: eine Referenz auf das aktuelle Objekt.

**Die Python-Referenz `self` liefert einen Bezug auf das eigene Objekt; sie entspricht dem `this` von Java oder C++.**

Viele Python-Methoden benötigen das `self`, damit die Methode auf das jeweilige Objekt wirkt, also nicht-statisch wird.

### 12.3.14.3 Konstruktor und Destruktor

Der *Konstruktor* in Python hat den Bezeichner `__self__` (jeweils 2 × der Underscore _ am Anfang und am Ende) und wird durch die feste Syntax

```
def __init__(self, argumentliste)
```

definiert. Das erste Argument – `self` – wird benötigt, um den Bezug zum neuen Objekt herzustellen. Beim Aufruf des Konstruktors können Argumente übergeben werden; bei der Argumentliste sind auch Default-Werte möglich, um eine Art Überladen zu ermöglichen (siehe Beispiel im Script `Buch.py`, Zeile 13).

Der Destruktur ist die Methode, die beim Speicherbereinigen des Objektes ausgeführt wird; er hat in Python die Syntax

```
__del__
```

#### 12.3.14.4 Datenkapselung

Python bietet die Möglichkeit, Attribute zu *kapseln,* also vom Zugriff von anderen Klassen zu schützen. Eine solche private Variable wird durch die Syntax

```
__variablenName_
```

definiert; hier kommt 2 × das Zeichen _ vor und 1 × nach dem Variablenbezeichner vor.

Die Klasse `Buch` verfügt über fünf solcher privaten Variablen.

#### 12.3.14.5 Vererbung in Python

Python bietet eine einfache *Vererbung* durch die Syntax

```
class KindKlasse (ElternKlasse)
```

Die Klasse `Buch` soll beispielsweise abgeleitet werden zur Klasse `BuchPublisher`, welche zusätzlich die gekapselten Attribute `__verlag_` und `__jahr_` enthält. Der Konstruktor der Kindklasse ist so geschrieben, dass er den Konstruktor der Elternklasse aufruft; dies geschieht über die Syntax

```
NameDerElternklasse.__init__(argumente)
```

Das Python-Script `BookPublisher.py` enthält die entsprechende Klasse (Abbildung 12.20), Instanziierung und Ausführung ist in Abbildung 12.21 (Script `buchhandlung2.py`) enthalten.

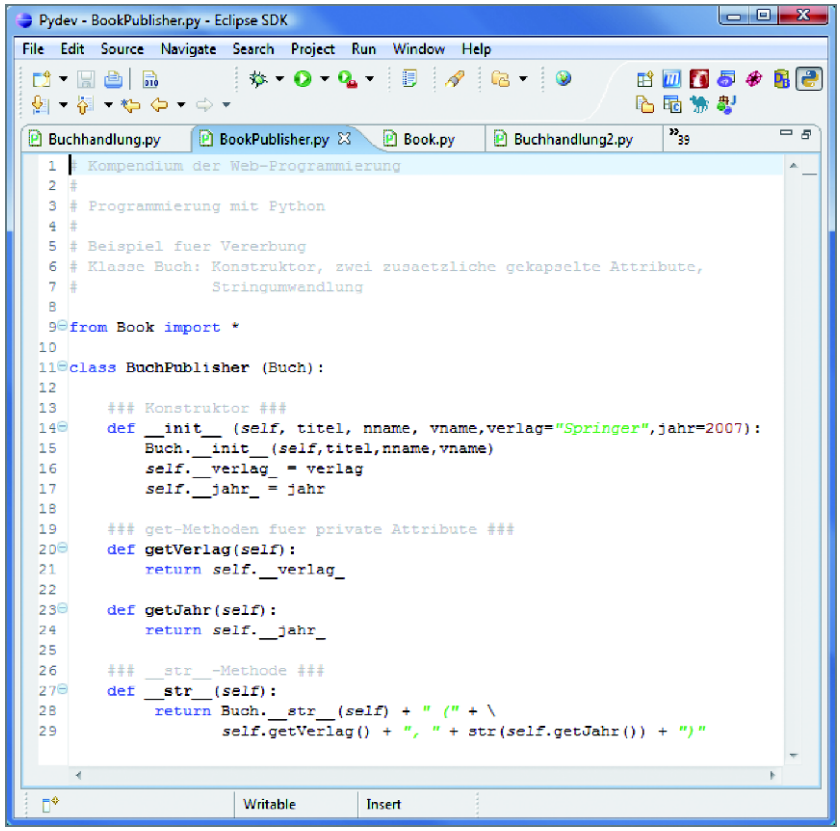



**Abbildung 12.20:**
Die Klasse `BuchPublisher` ist von `Buch` abgeleitet (Script `BookPublisher.py`)

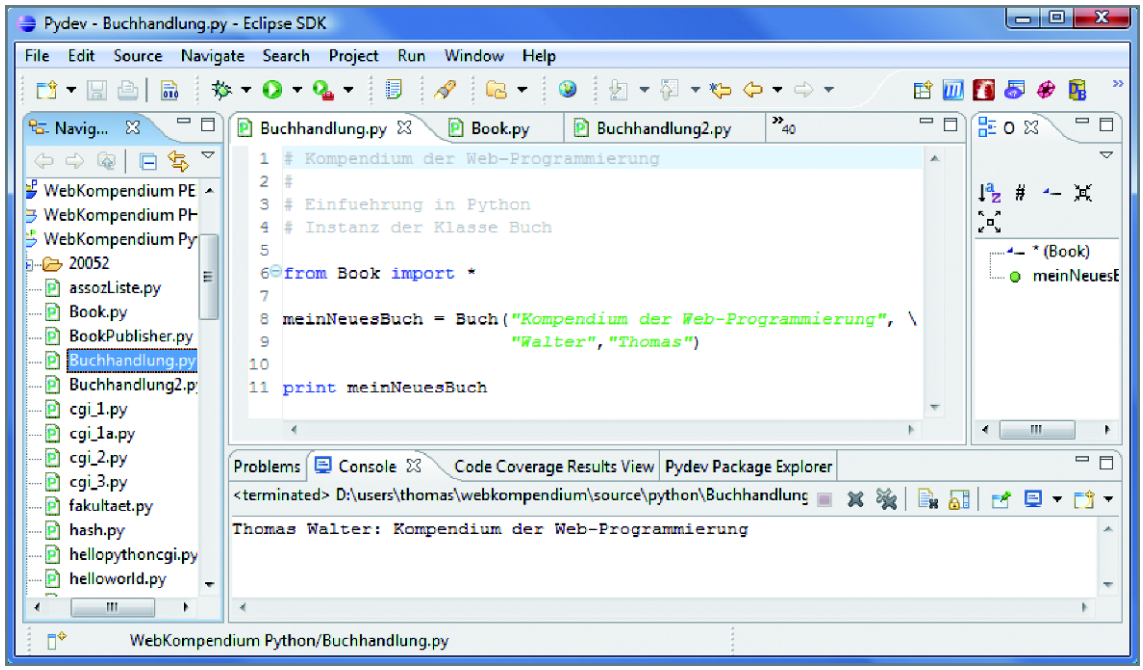



**Abbildung 12.21:** Erzeugung einer Instanz der Klasse `BuchPublisher` im Script `buchhandlung2.py` und Ausführen

Python bietet neben der üblichen Vererbung auch die Mehrfachvererbung: Eine Python-Klasse kann mehrere Elternklassen haben. Die Syntax

```
class KindKlasse (ElternKlasse1, ElternKlasse2, ...)
```

erlaubt durch Aufzählung aller Elternklassen diese Mehrfachvererbung.

#### 12.3.14.6 Spezielle OO-Methoden

Die *Objektorientierung* (vgl. 4.2.2) in Python verwendet eine eigene Syntax; wichtig sind dabei besonders einige Methoden mit fester Bedeutung, wie sie in den Beispielen schon verwendet wurden:

- `__init__(self,args)` für den Konstruktor;
- `__del__(self)` für den Destruktor;
- `__str__(self)` erzeugt eine Zeichenkettendarstellung des Objektes, wenn etwa mittels

  ```
  print objektReferenz
  ```

  das Objekt direkt ausgegeben werden soll; dies entspricht genau der Java-Methode `toString`: ´

**`__str__` in Python entspricht der Methode `toString` in Java aus der Klasse `java.lang.Object`, die in den Kindklassen meistens überschrieben wird.**

## 12.4 Python im Web

### 12.4.1 Python-Module

Python ist über ein Modulkonzept ähnlich zu den Java Packages einfach erweiterbar, und es sind zahlreiche Module verfügbar. Viele gehören direkt zum Python-Paket und werden mit diesem ausgeliefert.

Ein Python-Modul ist zunächst eine einfache Python-Datei, welche Definitionen und Anweisungen enthält; es wird dann eine Datei mit dem Namen des Moduls und angehängter Endung `.py` importiert. Enthält diese Datei Anweisungen, sind diese für die Initialisierung gedacht und werden beim ersten Import einmalig ausgeführt.

Python-Module werden üblicherweise in der compilierten Form als `pyc`-Dateien ausgeliefert. Die Umgebungsvariable `PYTHONPATH` steuert, wo Python nach Modulen sucht.

Zum Einbinden von Modulen dient das Schlüsselwort `import`; es kann auf zwei Arten angewendet werden:

- `import modulname` importiert das Modul „modulname“ und erweitert den Namensraum entsprechend (modulname ist dann bekannt);
- `from modulname import obj1, obj2, obj3` importiert nur die Klassen obj1, obj2 und obj3 vom Modul „modulname“; der Namensraum wird nur um diese Klassen erweitert.

Im Web [web] ist eine Übersicht über die Module zu finden, die standardmäßig mit Python ausgeliefert werden.

Einige dieser Standard-Module sind für diesen Abschnitt grundlegend, da sie CGI- und Datenbankfunktionalitäten liefern; andere wie etwa `string` sind allgemein wichtig.

### 12.4.2 Das Modul `string`

Das Modul `string` liefert zahlreiche Methoden zur Zeichenkettenverarbeitung. Es ist Ziel der Python-Entwicklung, ab der Version 3 der Programmiersprache viele dieser Methoden direkt in der Kernsprache zu implementieren, so dass dieses Modul an Bedeutung verlieren wird.

Zu den momentan im Modul `string` implementierten Methoden gehören:

- `count (string, muster)`
- `find (string, muster)`
- `rfind (string, muster)`
- `split (string, pattern)`
- `replace (string,old,new)`

Zum Modul `string` gehört auch die Klasse `Template`; hiermit gibt es Möglichkeiten der Ersetzung von Variablen durch ihre Werte, ähnlich, wie es Perl und PHP bieten. Dies geschieht durch eine Syntax der Form von Script `stringTemplate.py` (Abbildung 12.22).

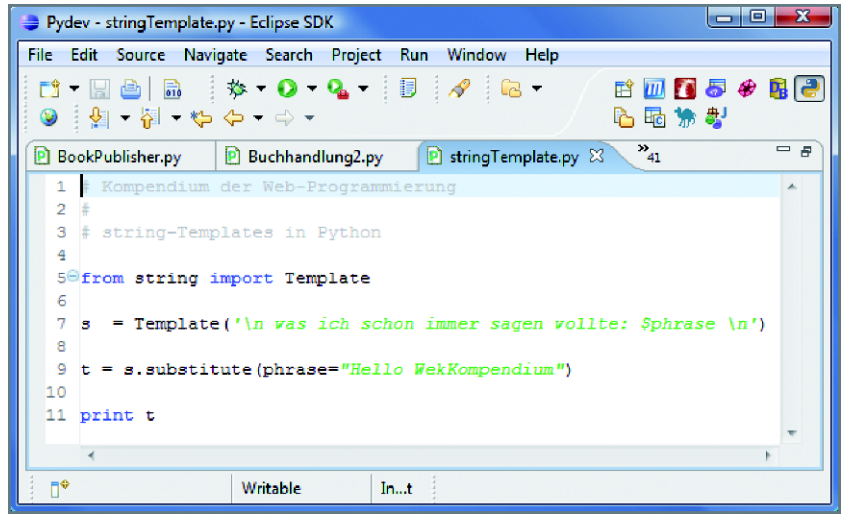



**Abbildung 12.22:**
Das Script `stringTemplate.py`

### 12.4.3 Das Modul `math`

Das Modul `math` liefert mathematische Funktionen in großem Umfang; hierzu gehören etwa

- das Attribut `pi` für die Konstante $\pi$;
- das Attribut `e` für die Konstante der Eulerschen Zahl $e$;
- sämtliche mathematischen Funktionen wie `sqrt(x)`, `pow(x,y)`, `sin(x)`, `sinh(x)`, `cos(x)` und `cosh(x)`.

Das Modul `cmath` liefert vergleichbare Funktionen zusätzlich für komplexe Argumente mit dem Datentyp `complex`, welcher in Python ja direkt integriert ist.

Das Script `math.py` beinhaltet ein typisches Beispiel der Anwendung beider Module (Abbildung 12.23).

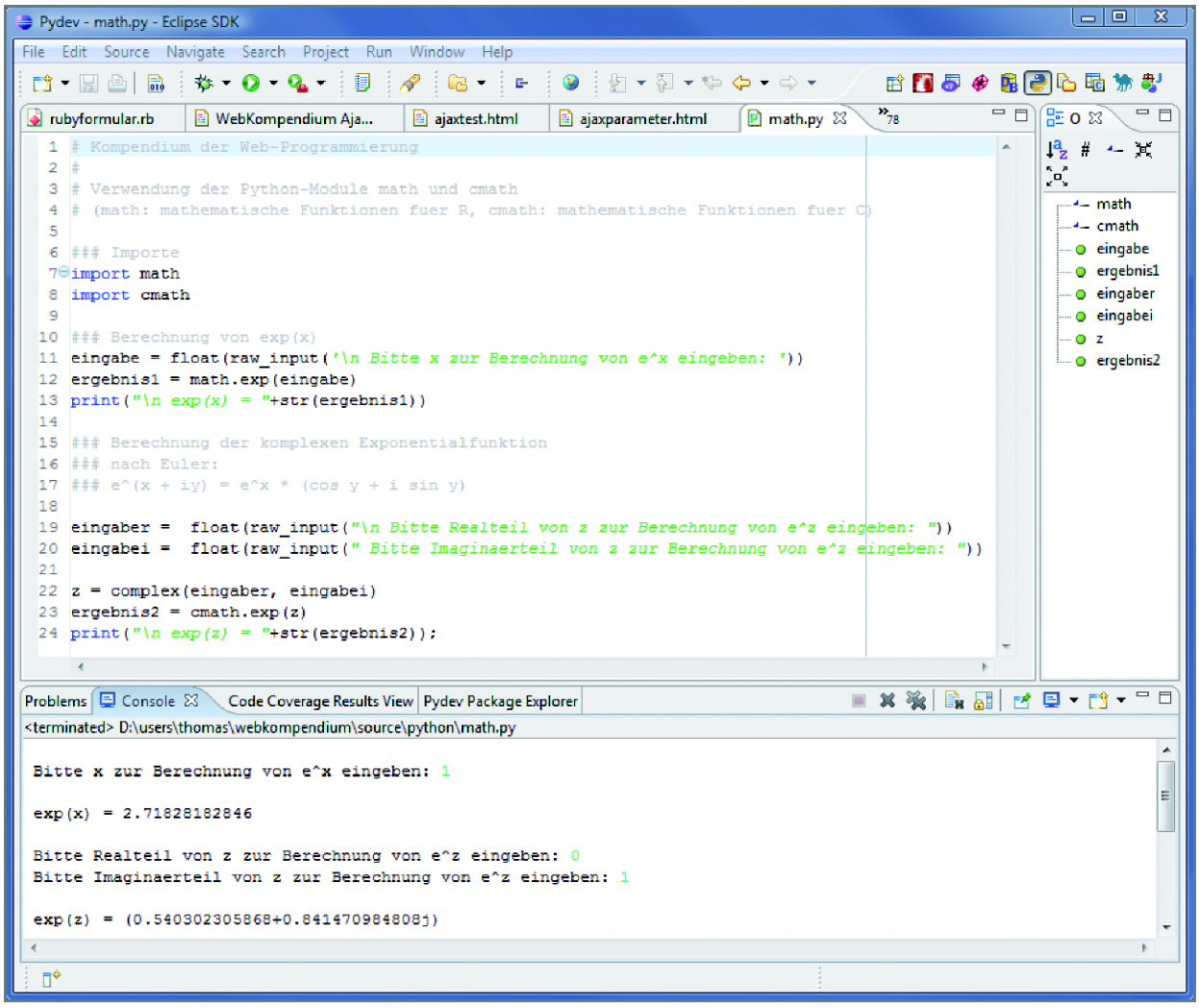



**Abbildung 12.23:** Anwendung der Mathematik-Module im Script `math.py`

### 12.4.4 Das Modul `time`

Zeitausgaben in Python werden über das Modul `time` ermöglicht. Dieses stellt ähnlich zu `gmtime` bei Perl ein Array mit der Zeit und dem Datum bereit.

Abbildung 12.26 zeigt eine Anwendung (Script `time.py`) in einer CGI-Implementierung.

### 12.4.5 Weitere Python-Module

Python verfügt über eine sehr große Zahl weiterer Module, die vielfältige Funktionalität bereitstellen. Hierzu gehören etwa das Modul `os` mit speziellen Betriebssystem-Funktionen und das Modul `freeze` für die Überführung von Python-Scripten nach C; beide sind wesentlich für Unix/Linux ausgelegt.

**Für Unix/Linux stehen besonders umfassende Python-Module bereit.**

### 12.4.6 Python und CGI

Zuerst soll ein einfaches CGI-Script mit Python geschrieben werden. Dies ist ein Standard-Script, wie es in Kapitel 9 vorgestellt wurde, allerdings muss die Shebang-Zeile korrekt formuliert werden, damit der Python-Interpreter eingebunden wird.

Ein einfaches Beispielscript ist `hellopythoncgi.py` (Abbildung 12.24).

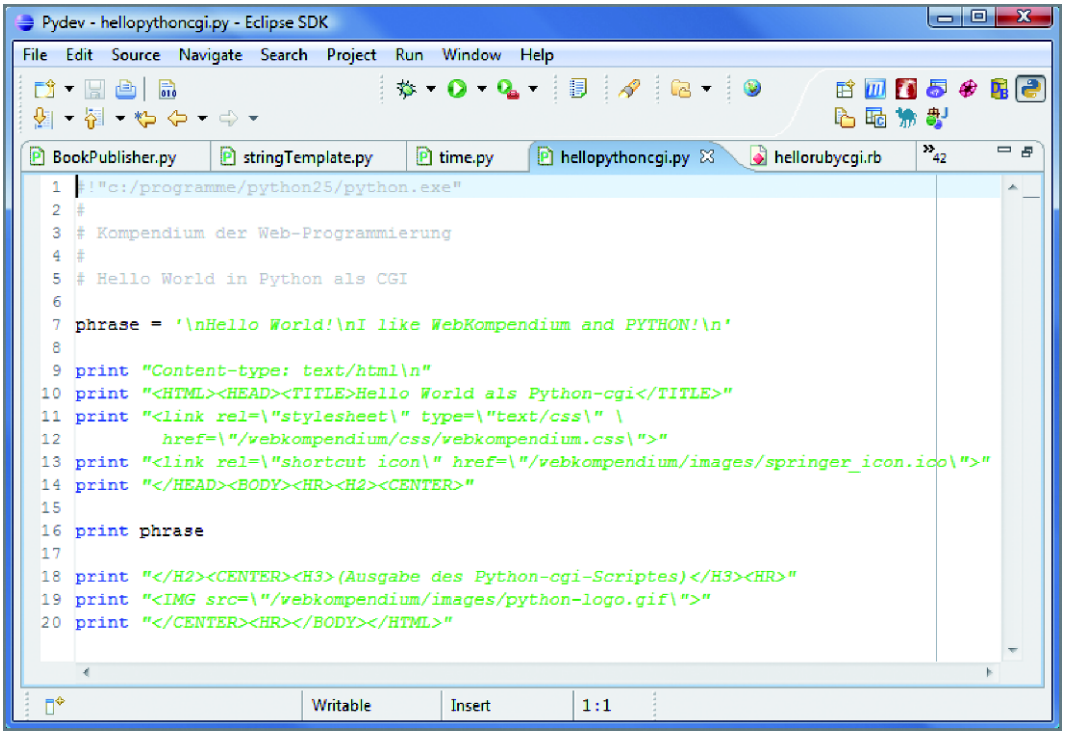



**Abbildung 12.24:**
Einfaches CGI in Python (Script `hellopythoncgi.py`)

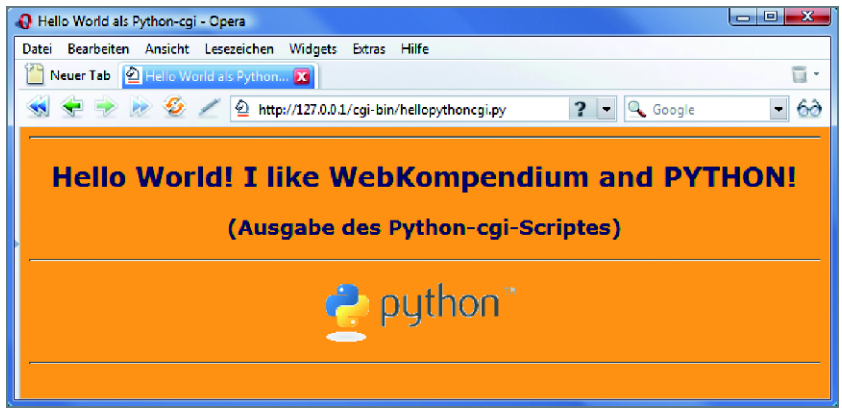



**Abbildung 12.25:**
Ausführen von `hellopythoncgi.py`

Mittels des Moduls `time` kann im nächsten Beispiel eine erste Dynamik erzeugt werden: Die Ausgabe der Serverzeit.

Dies leistet das CGI-Script `time.py` aus Abbildung 12.26, welches zusätzlich über JavaScript die Zeit auf dem Client ausgibt, also beide Methoden verbindet.

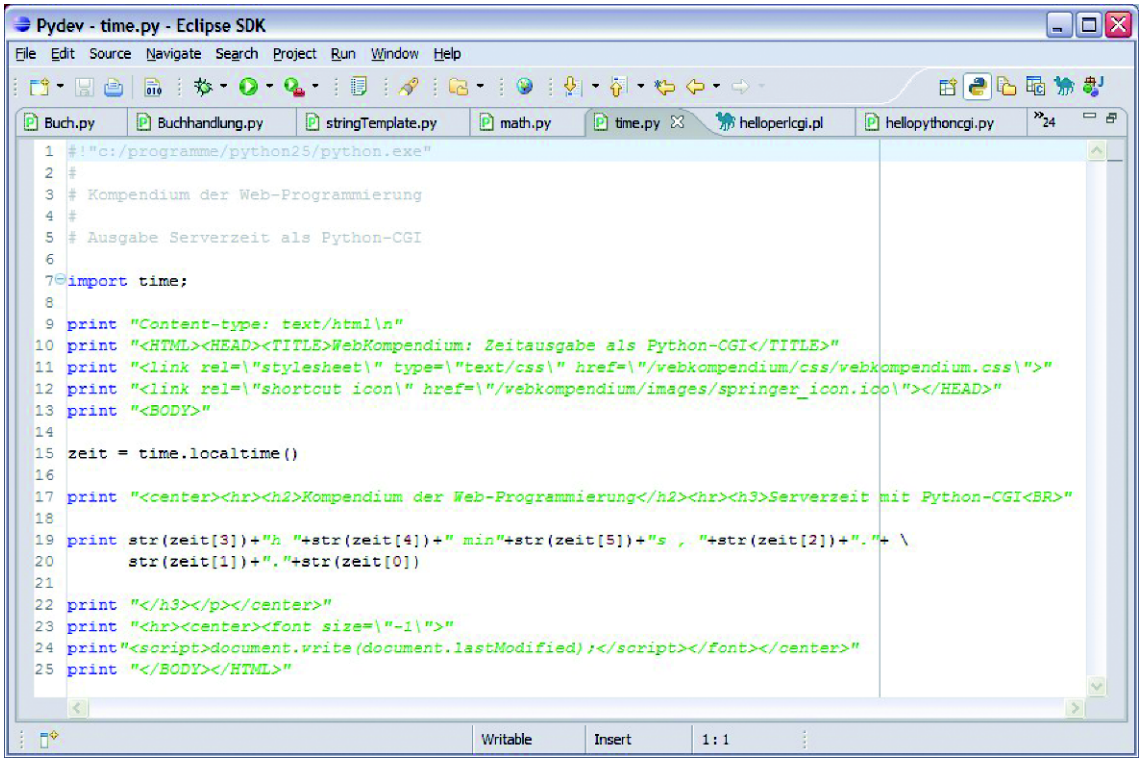



**Abbildung 12.26:**
Ausgabe der Serverzeit mit Python und der Clientzeit mit JavaScript (Script `time.py`)

**Abbildung 12.27:**
Ausführen von `time.py`

### 12.4.6.1 Here-Documents in Python

Wie Perl bietet auch Python Here-Documents (vgl. 10.4.3), die das Erstellen von CGI-Anwendungen deutlich vereinfachen.

In Python werden Here-Documents von dreifachen ’ oder “ eingeleitet und genauso beendet.

#### 12.4.6.2 Das Modul `cgi`

Um komplexere Anwendungen mit Python zu implementieren, stellt das Modul `cgi` zahlreiche nützliche Funktionen bereit – insbesondere die einfache Möglichkeit, Formulare zu verarbeiten.

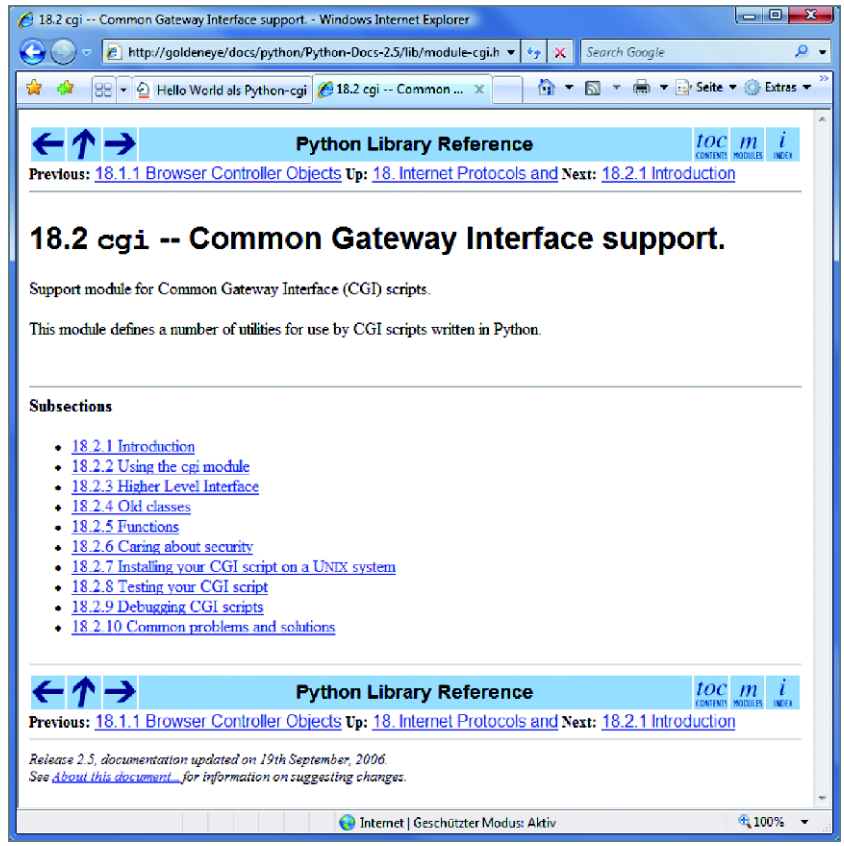



**Abbildung 12.28:** Online-Dokumentation zum CGI-Modul von Python

Für die Formularverarbeitung stellt das CGI-Modul die Klasse `FieldStorage` zur Verfügung, welche das übertragene Formular verwaltet. Eine Instanz dieser Klasse verhält sich wie eine assoziative Liste, welche die einzelnen Formularfelder enthält. Diese Formularfelder selbst sind als Instanzen der Klasse `MiniFieldStorage` hinterlegt.

Um auf ein Formularfeld zuzugreifen, muss deshalb zunächst eine Instanz der Klasse `FieldStorage` erzeugt werden; dies geschieht über den Konstruktoraufruf (vgl. 12.3.14.3):

```
import cgi
...
form = cgi.FieldStorage()
```

Dabei ist `form` der frei wählbare Bezeichner für das Formular.

Auf ein Feld kann dann mittels

```
form[key].value
```

zugegriffen werden; dies liefert den Wert zum Feld mit dem Namen `key`.

Das Script `cgi_1.py` zeigt die Anwendung dieses Mechanismus: Die Ausgabe aller übertragenen Formularfelder mit key und value.

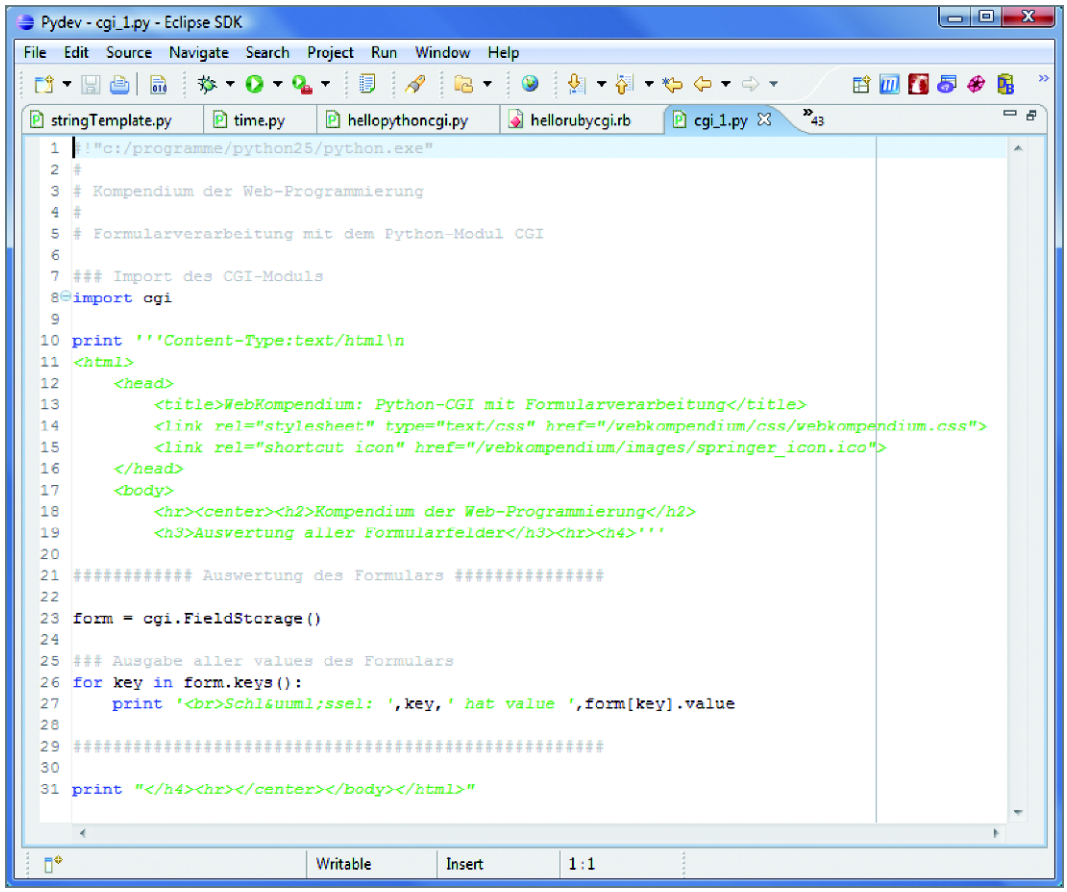



**Abbildung 12.29:**
Formularverarbeitung mit Python (Script `cgi_1.py`)

**Abbildung 12.30:**
Ausführen von `cgi_1.py` mit der URL `/cgi-bin/cgi_1.py?key1=value1&key2=value2`

Vorsicht ist hier bei älteren Python-Versionen (bis 2.3) geboten, da hier nur eine Klasse, die Klasse `SvFormContentDict`, zum Einsatz kam; dies entspricht `Vars` aus dem Perl-Modul CGI.

Das Modul `cgi` bietet weitere Funktionalität, etwa die Methode

```
cgi.escape(text)
```

welche `text` korrekt in HTML-Code übersetzt; so wird aus `&` die Codierung `&`.

#### 12.4.6.3 Fehlersuche und CGI: das Modul `cgitb`

*Fehlersuche* in CGI-Anwendungen ist ein schwieriges Geschäft; hier bietet Python eine wesentliche Erleichterung durch das Modul `cgitb`[3]: Es liefert eine Traceback-Funktionalität für cgi-Scripte.

Um das Modul zu verwenden, muss es wie üblich über

```
import cgitb
```

eingebunden werden. Mittels der Methode

```
cgitb.enable(...)
```

wird dann diese Traceback-Funktionalität aktiviert. Diese Methode `enable` verfügt über zahlreiche Parameter:

```
enable([display[, logdir[, context[, format]]]])
```

Dabei bedeutet:

- `display`: Der Standardwert 1 legt fest, dass die Traceback-Informationen an den Browser gesendet werden; bei 0 wird dieses unterdrückt.
- Wird `logdir` angegeben, erfolgt eine Ablage der Traceback-Informationen in einer Logdatei im lokalen Filesystem auf dem Server.
- `context` gibt an, wieviele Zeilen um die betreffende Stelle angegeben werden; der Default-Wert beträgt 5.
- `format` erlaubt die Formatsteuerung der Ausgabe; Default-Wert ist hier html, alle anderen Werte werden als „plain“ interpretiert und führen zu einer einfachen Text-Datei.

[3] verfügbar seit der Version Python 2.1.

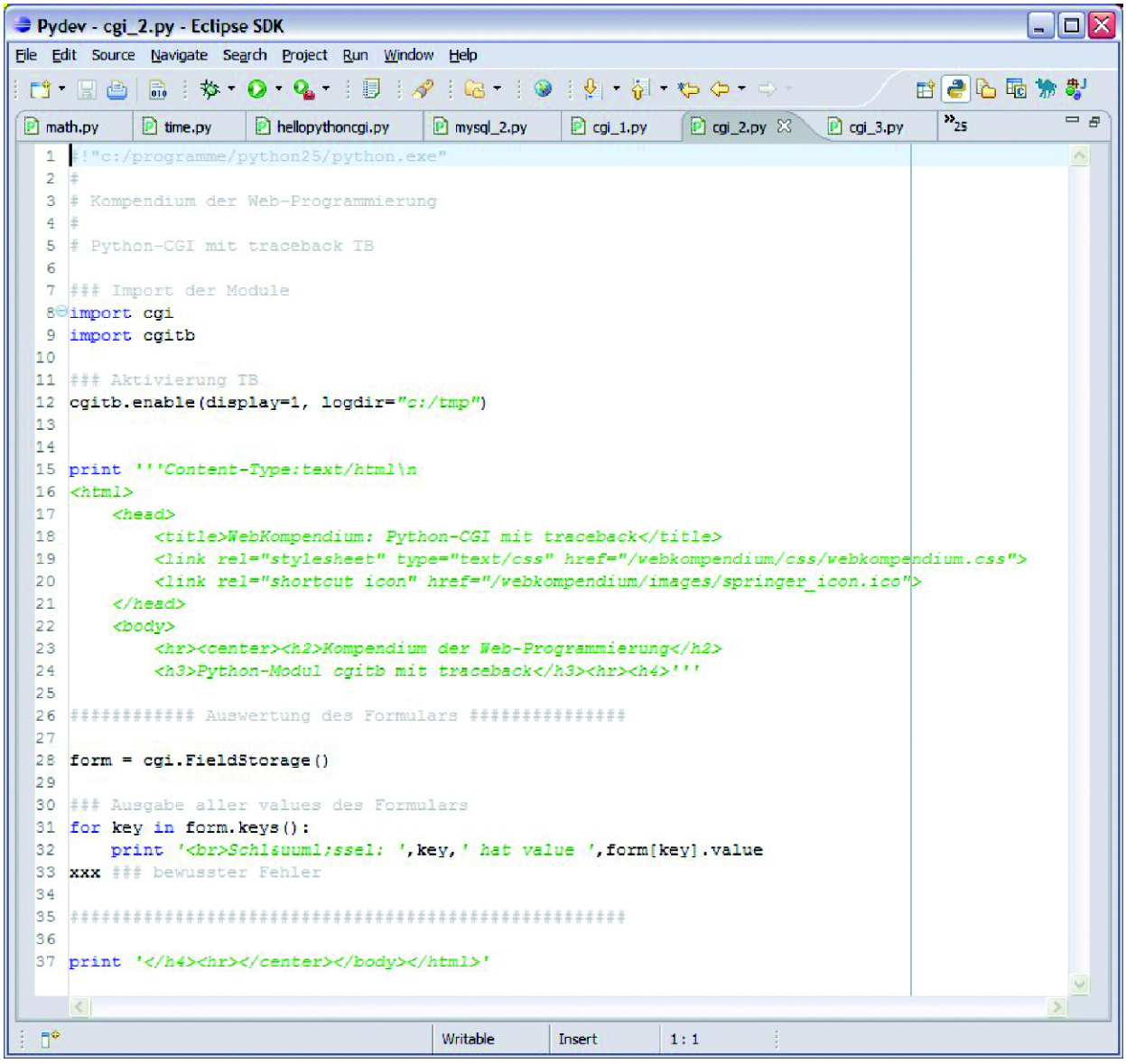



Abbildung 12.31 zeigt ein Script mit einem bewussten Fehler in Zeile 33; beim Ausführen ergibt sich bei diesen Parametern der `enable`-Methode die Browser-Anzeige nach Abbildung 12.32; außerdem wird eine Logdatei im HTML-Format geschrieben (Abbildung 12.33).

**Abbildung 12.31:**
Python-Script mit bewusstem Fehler (Zeile 33)

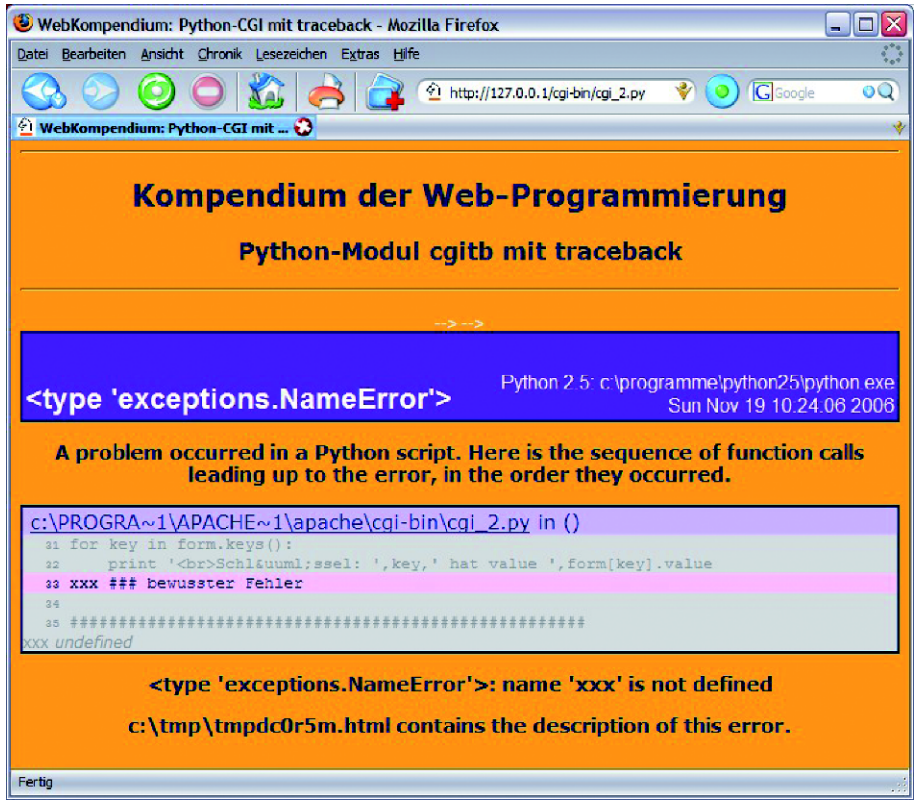



**Abbildung 12.32:**
cgitb-Meldung im Browser

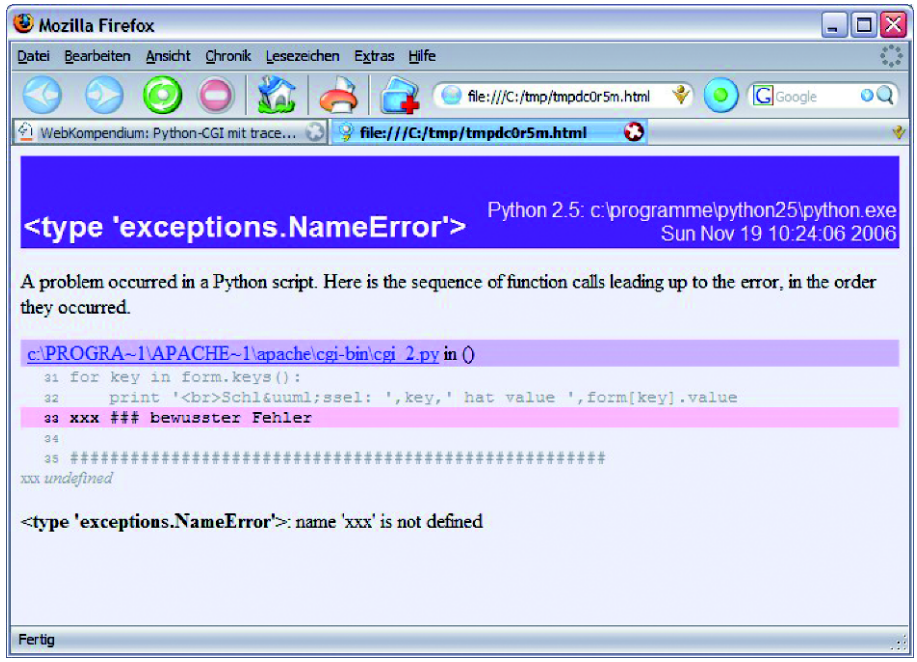



**Abbildung 12.33:**
Eine von cgitb erzeugte Logdatei im HTML-Format

### 12.4.7 Der Webclient mit Python: Das Modul `urllib`

In Perl stellt das Modul LWP die grundlegende Funktionalität bereit, um die Funktionalität eines *Webclients* zu implementieren, etwa für Performance-Analysen von Web-Applikationen. In Python leistet dies das Modul `urllib`.

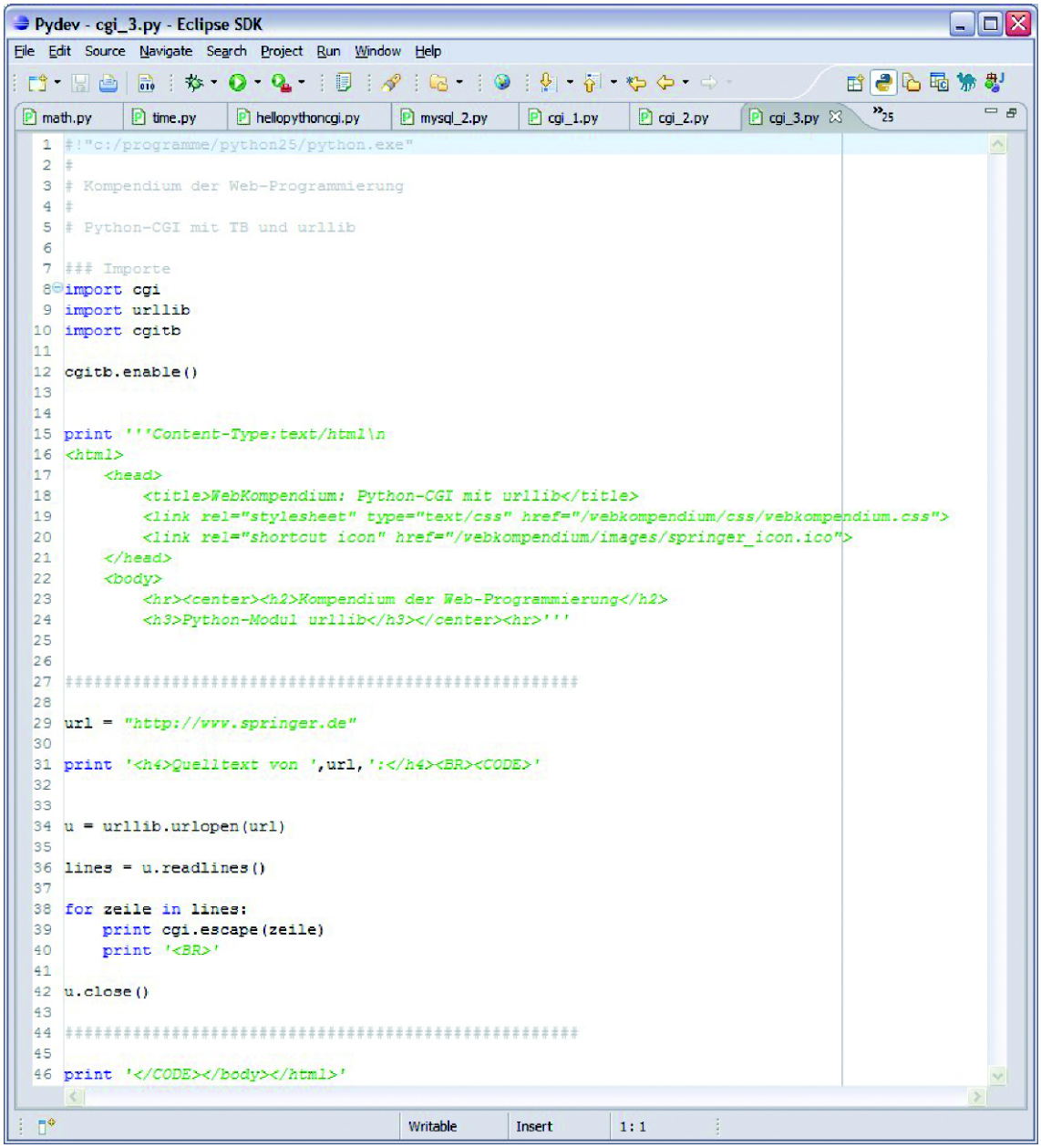



**Abbildung 12.34:**
Script `cgi_3.py` zum Auslesen des Sourcecodes von www.springer.de

Einige der wichtigsten Methoden des Moduls `urllib` sind:

- `urlopen(url)` öffnet eine URL wie eine Datei zum Lesen;
- `read` liest von dieser URL;
- `readlines()` liest mehrere Zeilen von der geöffneten URL
- `readline()` liest eine Zeile von der geöffneten URL
- `close` schließt die Verbindung.

Das Script `cgi_3.py` öffnet eine URL (www.springer.de) und gibt über die Methode `cgi.escape` (vgl. Seite 283) den Quelltext der Seite aus (Abbildungen 12.34 und 12.35).

**Abbildung 12.35:**
Ausführen von `cgi_3.py`

## 12.5 Python und Datenbanken

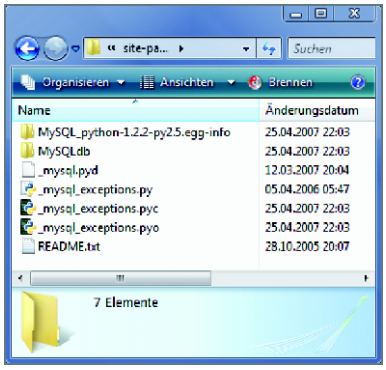



**Abbildung 12.36:**
Das installierte Modul `MySQLdb` unter Windows

Mit Python lassen sich selbstverständlich einfach *Datenbankapplikationen* entwerfen. Voraussetzung für die Verwendung eines Datenbankmanagementsystems mit Python ist das jeweilige DBMS-Modul; alle diese sind nicht Bestandteil der Python-Distribution, sondern müssen zusätzlich installiert werden.

Hintergrund dieser Datenbank-Module ist eine Python-eigene Datenbank-API, auf welcher diese Module aufbauen: mit dieser API entspricht Python Perl mit dem DBI-Modul (vgl. 10.7) – nur ist die Datenbankfunktion in Python direkt integriert und muss nicht zusätzlich installiert weren wie bei Perl-DBI. Aktuell ist diese Datenbank-API in der Version 2.0 Bestandteil von Python.

Für die MySQL-Datenbank ist ein Modul „MySQLdb“ auf Sourceforge [web] verfügbar; damit kann Python MySQL nutzen; die aktuelle Version des Moduls ist bis Python 2.5 nutzbar und unterstützt MySQL bis zur Version 5.1.

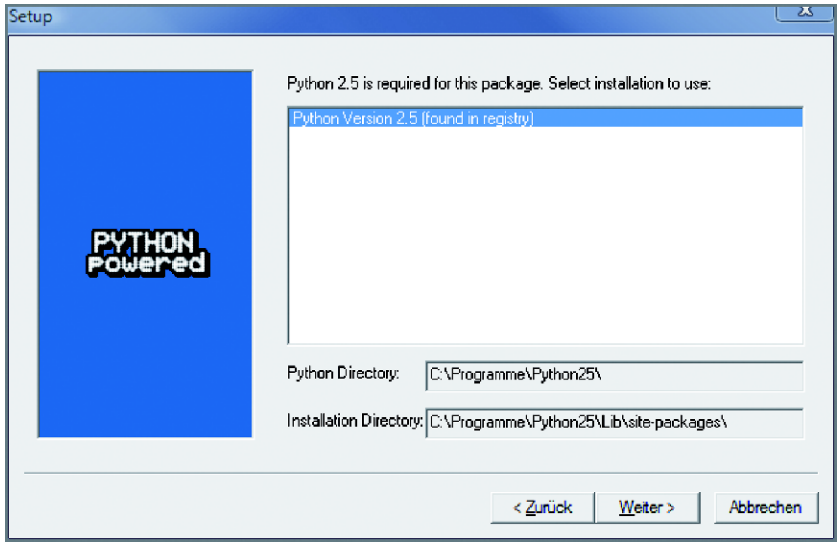



**Abbildung 12.37:**
Installieren des Moduls `MySQLdb` unter Windows

Mit dem Python DB-API Interface ist – wie bei Perl mit DBI – nach dem expliziten Laden des DB-Moduls nur der Verbindungsaufbau Datenbank-spezifisch; alles andere wird in einer vom jeweiligen DBMS unabhängigen Syntax formuliert.

Die DB-API stellt hierfür einen DB-Cursor für die Verarbeitung von Resultsets bereit, ähnlich zu JDBC bei Java.

Das Script `mysql_1.py` greift das Beispiel der Literaturdatenbank aus Kapitel 7 auf und gibt eine Übersicht über die Literatur bei Nennung des Erstautors aus (Abildungen 12.39 und 12.38).

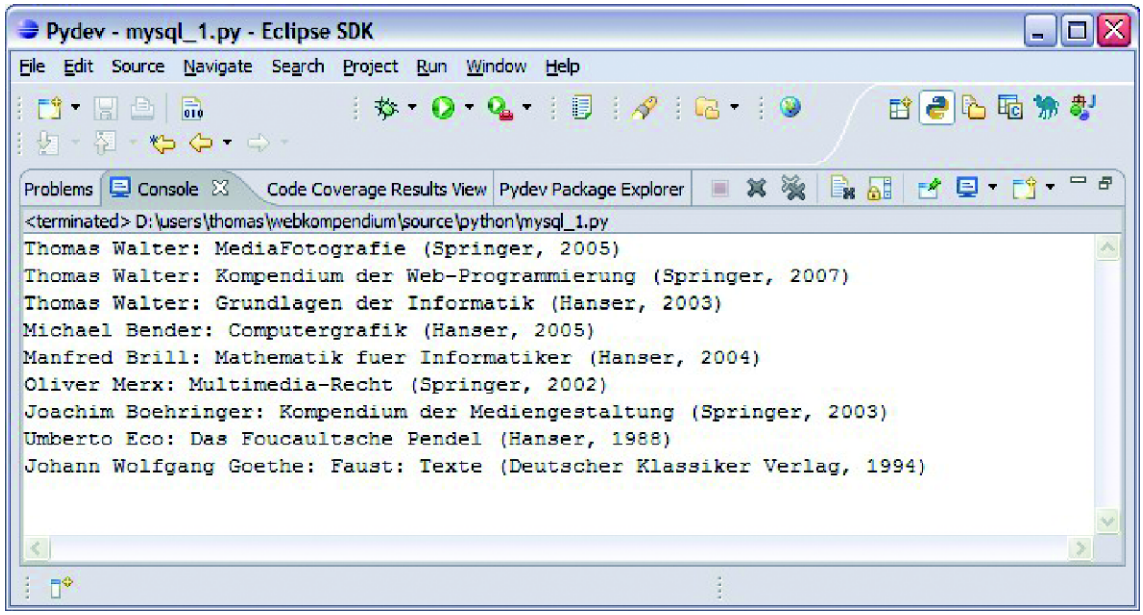



**Abbildung 12.38:**
Ausführen von `mysql_1.py` in Eclipse mit pydev

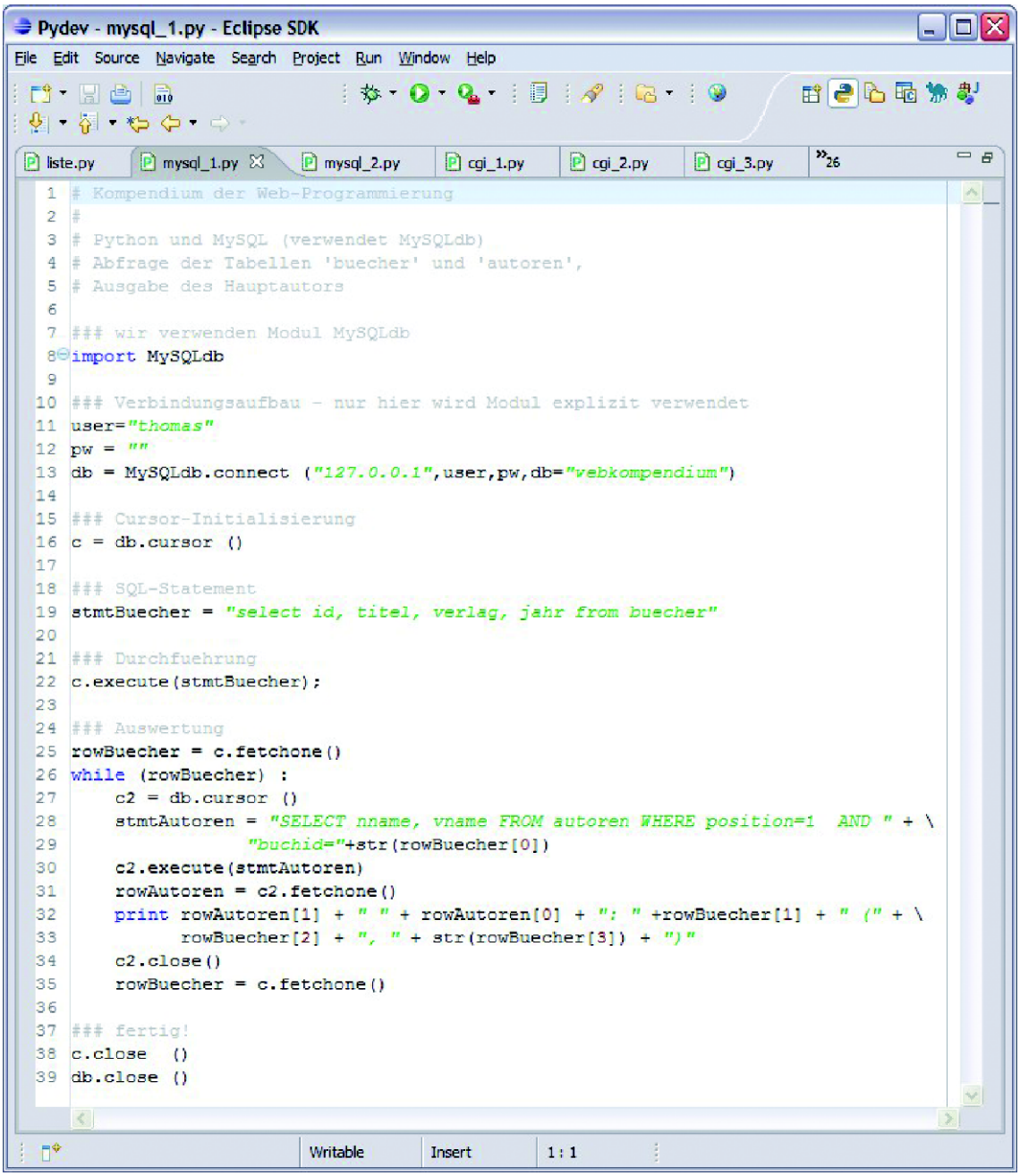



**Abbildung 12.39:**
Das Script `mysql_1.py` zur Abfrage der Literatur-Datenbank

Dieses Beispiel soll nun als CGI-Anwendung in das Web übertragen werden; das Script `mysql_2.py` (Abbildung 12.40) zeigt eine entsprechende Anwendung, welche ebenfalls das Modul cgitb verwendet; das Ergebnis ist in Abbildung 12.41 zu sehen.

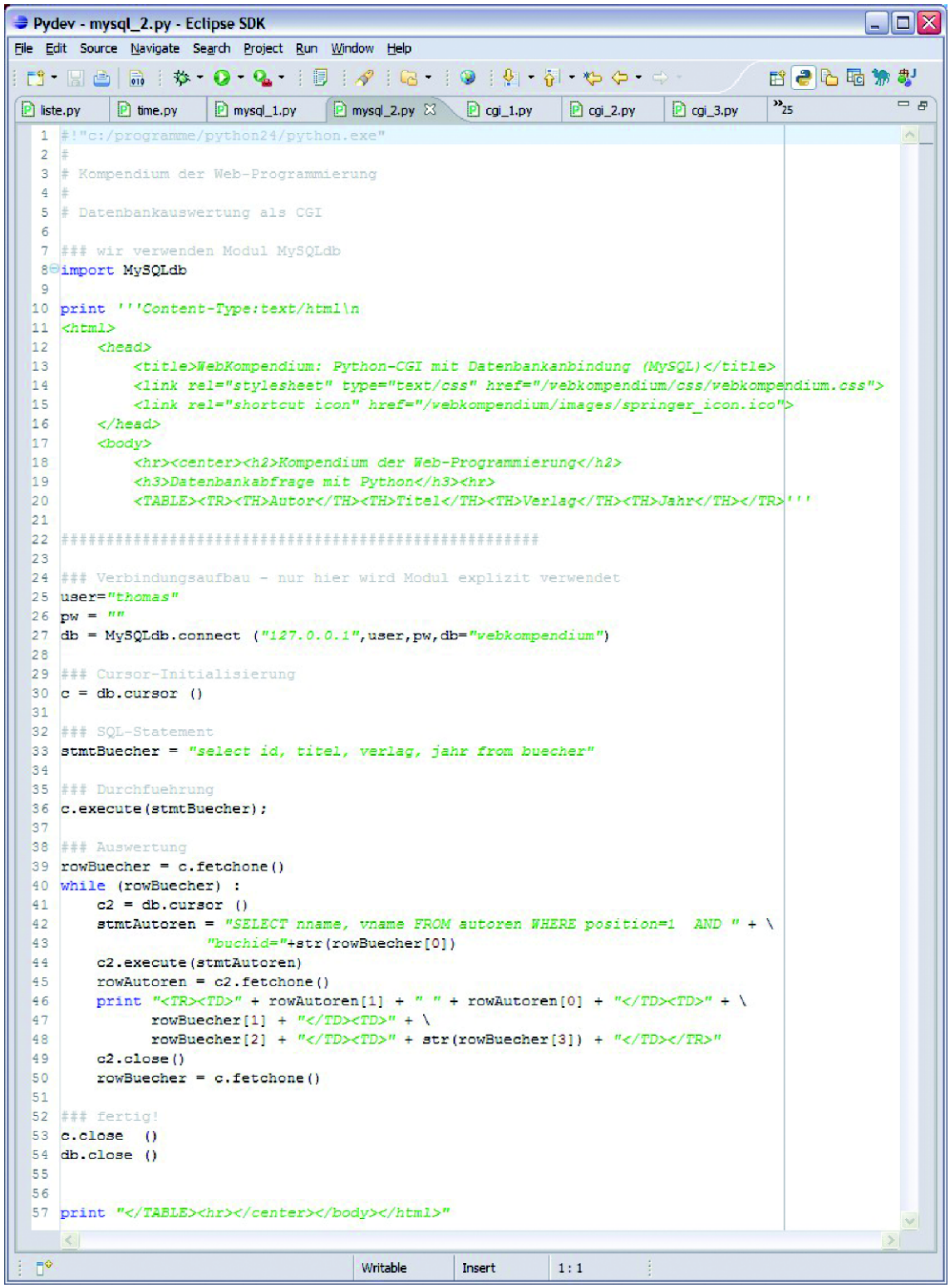



**Abbildung 12.40:**
Das CGI-Script `mysql_2.py` zur Abfrage der Literatur-Datenbank

**Abbildung 12.41:** Ausführen von `mysql_2.py`

Das Script `mysql_3.py` (Abbildung 12.43) arbeitet ähnlich wie `mysql_1.py`, allerdings wird hier die Klasse `BuchPublisher` verwendet – genauer eine Liste von Objekten dieser Klasse –, um die Datenbankeinträge intern zu speichern.

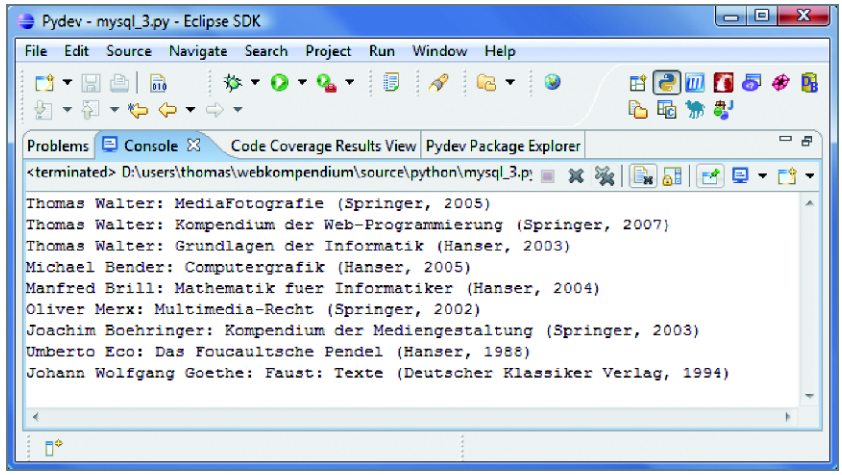



**Abbildung 12.42:** Ausführen von `mysql_3.py` in Eclipse mit pydev

**Python-Datenbanktreiber nach der Python DB-API 2.0 sind für eine Vielzahl von Datenbankmanagementsystemen verfügbar; neben MySQLdb für MySQL etwa psycopg für PostgreSQL und pysqlite für SQLite.**

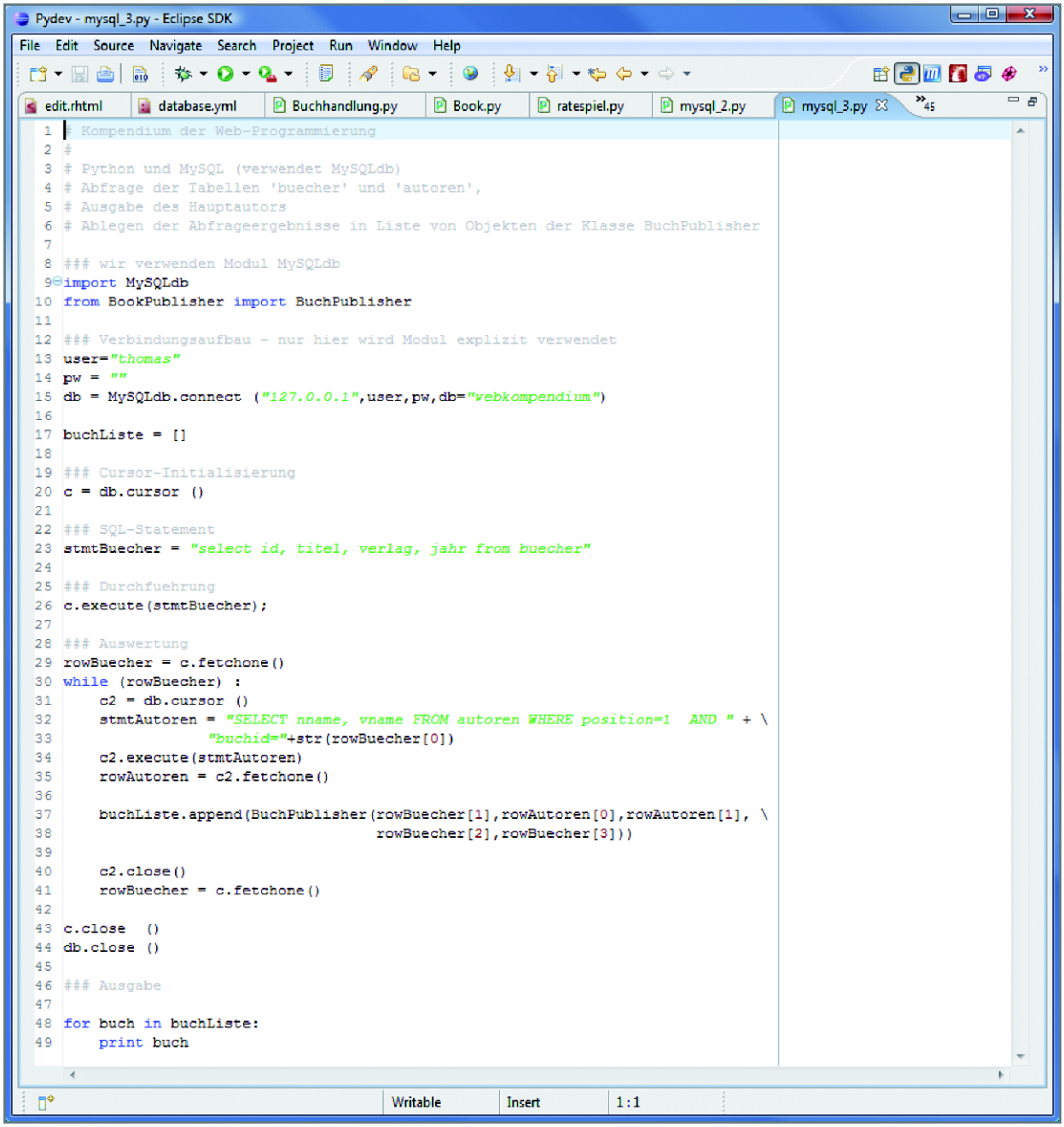



**Abbildung 12.43:**
Das Script `mysql_3.py` zur Abfrage der Literatur-Datenbank als Instanzen der Klasse `BuchPublisher`

## 12.6 Python und Java: Jython

Python selbst, also der Python-Interpreter, ist in C geschrieben; von daher gibt es zahlreiche Möglichkeiten, Teile von C in Python zu nutzen; davon profitiert etwa das Modul freeze. John Hugunin hat mit JPython einen Python-Interpreter in Java geschrieben [web]; der Name dieses Interpreters änderte sich inzwischen zu *Jython*.

Die Installation von Jython ist denkbar einfach; es existiert eine betriebssystemunabhängige class-Datei, welche mittels der Java Virtual Machine auszuführen ist (Abbildungen 12.44 und 12.45):

```
java -jar jython_installer-2.2b1.jar
```

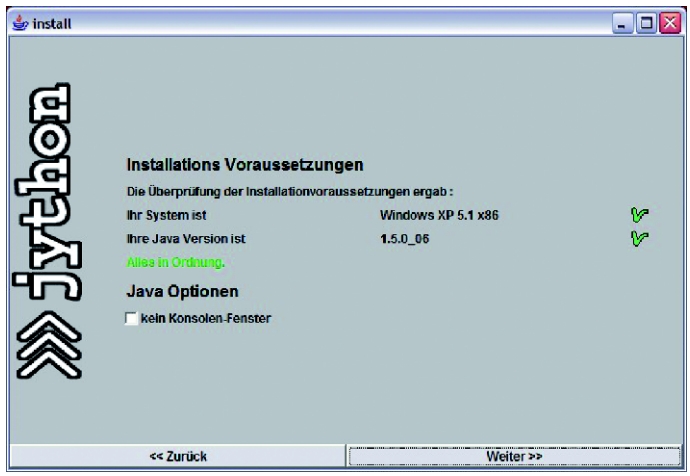



**Abbildung 12.44:** Installation von Jython (1)

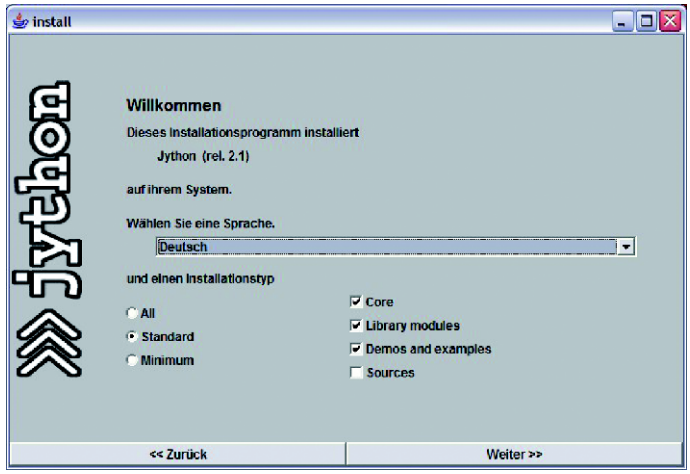



**Abbildung 12.45:** Installation von Jython (2)

Damit steht der interaktive Java-Python-Interpreter bereit: Es kann neben dem gesamten Python auch auf die Java-Bibliotheken zugegriffen werden. Abbildung 12.46 zeigt ein Beispiel.

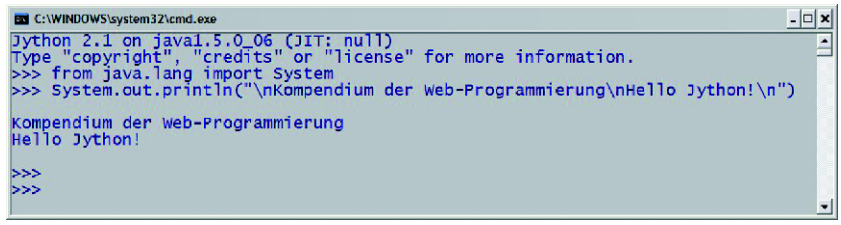



**Abbildung 12.46:** Jython-Interpreter: interaktives Java

### 12.6.1 Java-Applets mit Python

Jython bietet auch die Möglichkeit, eine Python-Anwendung in ein Applet zu überführen: Hierzu steht der Compiler `jpythonc` zur Verfügung, welcher aus einer Python-Anwendung ein echtes Java jar-Archiv erzeugt, welches entsprechend programmiert als Applet verwendet werden kann.

Der Sourcecode für ein minimales Applet lautet etwa:

```
from java.applet import Applet

class HelloWorld(Applet):
    def paint(self, g):
        g.drawString("Hello WebKompendium! Jython-Applet!!!" 20, 30)
```

IBMs WebSphere Application Developer (WSAD) kann inzwischen Jython-Code implementieren.

### 12.6.2 Jython und Datenbanken

Die effiziente Python DB-API 2.0 kann auch mit Jython verwendet werden; hierfür steht das Modul `zxJDBC` zur Verfügung, welches seit der Version 2.1 Bestandteil von Jython ist.

**Abbildung 12.47:**
Jython: die Verbindung von Python und Java

## 12.7 GUI-Programmierung mit Python

Python kann natürlich noch viel breiter eingesetzt werden. Wesentlich für die GUI-Programmierung mit Python ist das Modul `Tkinter`, welches eine Schnittstelle zu Tcl/Tk bereit stellt.

## 12.8 Ausblick

Python hat im Web nicht nur die Bedeutung als moderne(re) CGI-Sprache – Python bildet auch die Basis für viele Frameworks. Der Klassiker hiervon ist ZOPE (vgl. 22.4), auf welcher das Content Management System Plone basiert. Eine neuere Entwicklung hier ist etwa django [web] (vgl. 22).

Wesentlich für den performanten Einsatz von Python im Web ist das von Gregory Trubetskoy Apache-Modul mod_python [web], welches u.a. für das django-Framework benötigt wird. Durch dieses Modul kann der Apache Webserver – vergleichbar zu fastCGI, vgl. 19 – direkt Python-Code ausführen, ohne den Interpreter separat zu laden wie bei CGI.

Neben dem Einsatz im Web findet Python zunehmend weitere Verbreitung; so kann etwa die Grafik-Software Maya in der Version 8.5 neben ihrer eigenen Scriptsprache auch mittels Python programmiert werden.

**Abbildung 12.48:**
django-Logo

# 13

# Ruby

In diesem Kapitel wird die Scriptsprache *Ruby* vorgestellt. Bei Ruby handelt es sich um eine aktuelle Scriptsprache, welche zum einen viele moderne Sprachkonzepte, insbesondere die Objektorientierung, sehr konsequent umsetzt und zum anderen eine starke Verbreitung erfährt; nicht zuletzt ist Ruby die Grundlage für das populäre Framework Ruby on Rails (vgl. Kapitel 23).

Nachdem die Verbreitung von Ruby zunächst auf den asiatischen Raum beschränkt war, findet Ruby inzwischen weltweit immer mehr Anhänger, und es steht auch mehr englisch- und deutschsprachige Literatur für Ruby zur Verfügung, etwa [Höp07]. Natürlich gibt es auch zu Ruby viele nützliche Online-Ressourcen.

**Abbildung 13.1:** Ruby-Logo

## 13.1 Die Scriptsprache Ruby

Der Begründer der *Scriptsprache Ruby* ist der Japaner Yukihiro Matsumoto, der unter seinem Kürzel „Matz“ bekannt ist. Er begann angeblich genau am 24. Februar 1993 mit der Entwicklung der neuen Sprache.

Von Matz stammt sinngemäß folgendes Zitat, welches Ruby gut beschreibt:

*Ich glaube, dass das Ziel des Lebens (zumindest teilweise) ist, glücklich zu sein. Aufgrund dieser Überzeugung ist Ruby so konzipiert, dass Programmieren nicht nur einfach sein soll, sondern auch noch Spaß macht.*

Aus diesen grundlegen Zielen von Ruby leiten sich die Design-Prinzipien der Sprache ab.

Ruby ist zunächst eine interpretierte Scriptsprache mit sehr ausgeprägter Objektorientierung. Vorbilder bei der Entwicklung von Ruby waren zahlreiche andere moderne Sprachen oder programmierbare Shells, insbesondere Perl, Smalltalk, Python, LISP, Bash und CLU.

Der Name „Ruby“ ist kein Akronym, sondern eine Anspielung auf die Verwechslung von Perl mit der Perle „Pearl“.

Der Ruby-Interpreter samt einer umfassenden Online-Dokumentation und zahlreiche weitere Ressourcen sind im Web [web] frei verfügbar.

**Abbildung 13.2:** Die Ruby-Site

### 13.1.1 Prinzipien von Ruby

Ein wichtiges Prinzip der Scriptsprache Ruby ist das „principle of least surprise“, das Prinzip der geringsten Überraschung: Ruby soll frei von Überraschungen sein, also frei von unerwarteten Inkonsistenzen und Widersprüchen.

Ruby versucht, dieses Prinzip konsequent umzusetzen; entsprechend sind die wesentlichen Besonderheiten der Sprache:

- eine einfache Sprachsyntax;
- keine typisierten Variablen;
- die Sprache ist rein objektorientiert (vergleichbar zu Smalltalk); dies hat weitere Konsequenzen:
  - es gibt keine elementaren Datentypen, alles ist ein Objekt;
  - über Mixins (vgl. 13.4.5) steht ein Konzept für eine schwache Mehrfachvererbung bereit;
  - Ruby bietet Singleton-Methoden;
  - Klassen können zur Laufzeit umbenannt werden;
  - in Ruby ist es auch möglich, Operatoren zu überladen;
- Ruby bietet das Abfangen von Sondersituationen über Exceptions;
- Ruby bietet eine automatisierte Speicherbereinigung (Garbage Collection);

- reguläre Ausdrücke werden vergleichbar zu Perl bereit gestellt;
- Ruby ist weitgehend unabhängig vom verwendeten Betriebssystem;
- Ruby bietet eine einfache Möglichkeit der Anbindung von relationalen Datenbankmanagementsystemen;
- mit rdoc bietet Ruby die Möglichkeit, eine online-Dokumentation zu generieren – ein nützliches Tool, welches vergleichbar zu javadoc arbeitet;
- Ruby bietet auch Sprachbestandteile funktionaler Sprachen.

## 13.2 Installation und Entwicklungsumgebung

Ruby kann von der Haupt-Site (oder etwa auch von Active-State) bezogen werden; die *Installation* selbst läuft dann wie angegeben ab.

Ruby-Anwendungen können wieder mit einfachen Werkzeugen – der Kommandozeile zusammen mit einem Editor – entwickelt werden, oder man verwendet eine leistungsstärkere Entwicklungsumgebung. Empfehlenswert ist hier wieder ein Plugin für die IDE Eclipse (vgl. 8.1), speziell rubyeclipse (Abbildung 13.3); dieses Plugin kann direkt über den Update-Manager von Eclipse installiert werden.

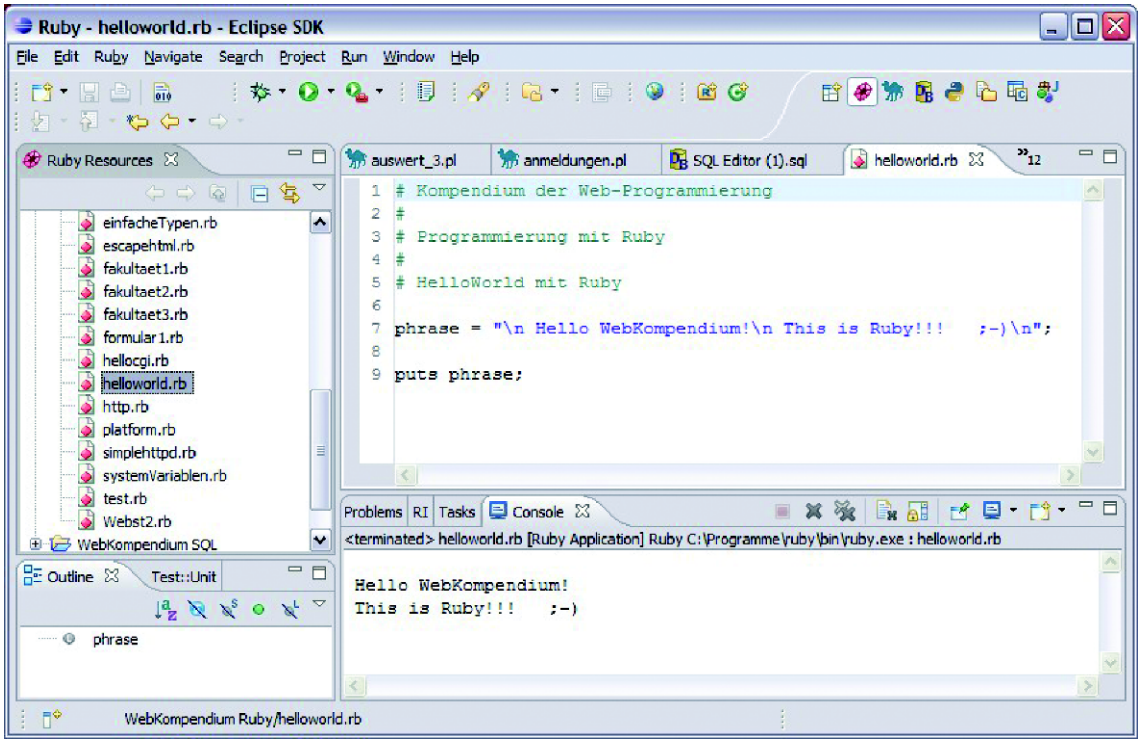



**Abbildung 13.3:**
Ruby-Entwicklung mit Eclipse und dem Plugin rubyeclipse

Mit diesem Plugin stehen zahlreiche nützliche Funktionalitäten für die praktische Arbeit mit Ruby zur Verfügung. Um Kommandozeilenanwendungen mit Eingabe in rubyeclipse sinnvoll zu nutzen, ist es notwendig, die von Eclipse standardmäßig verwendete Pufferung auszuschalten;[1] dies geschieht in Ruby durch die Anweisung

```
$stdout.synv = true
```

[1] vergleichbar zu `$| = -1;` in Perl, vgl. Seite 130.

Zum Ruby-Paket gehört aktuell auch eine weitere, freie IDE für Linux und Windows: FreeRIDE. Auch hiermit ist eine zügige Programmentwicklung mit Ruby möglich.

## 13.3 Grundlegende Syntax

### 13.3.1 Erste Schritte

Das erste Ruby-Script soll nun zunächst nichts anderes tun, als eine HelloWorld-Meldung auszugeben; `helloworld.rb` aus Abbildung 13.4 leistet genau dies.

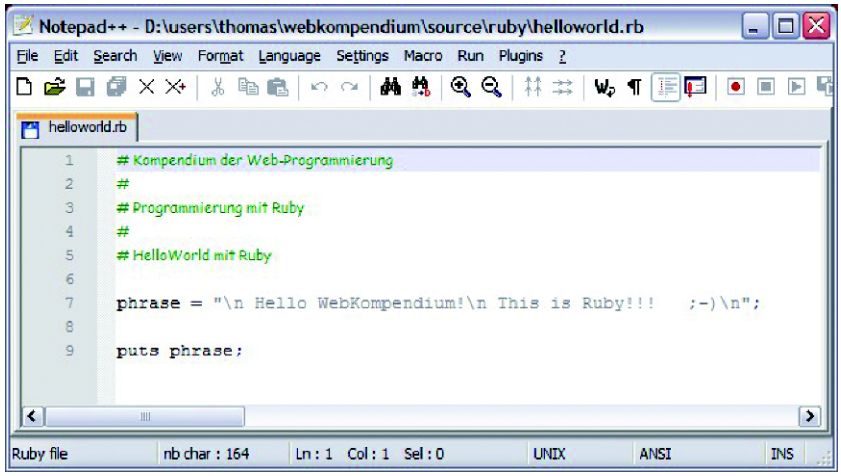



**Abbildung 13.4:** Erstes Ruby-Script `helloworld.rb`

Dieses Script wird direkt mittels des Ruby-Interpreters durch

```
ruby helloworld.rb
```

ausgeführt; das Ergebnis ist in Abbildung 13.5 zu sehen.

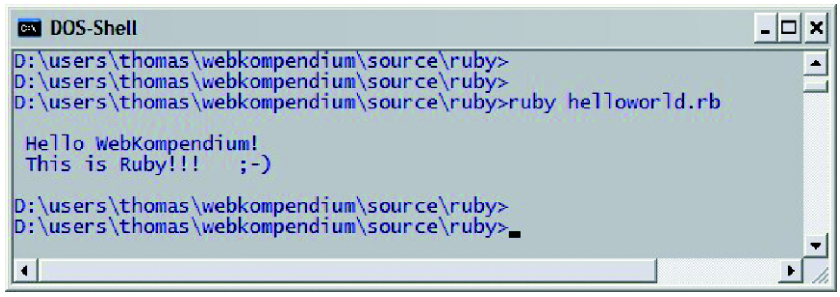



**Abbildung 13.5:** Ausführen von `helloworld.rb`

Der Ruby-Interpreter erlaubt neben der Ausführung eines Ruby-Scriptes ähnlich dem Python-Interpreter als weitere Betriebsart einen interaktiven Modus (Abbildung 13.6). Dieser wird durch `irb` gestartet. Dem Interpreter werden interaktiv Ruby-Anweisungen übergeben.

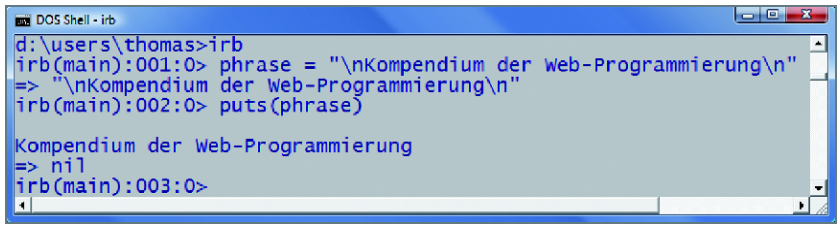



**Abbildung 13.6:**
Interaktives Ruby `irb`

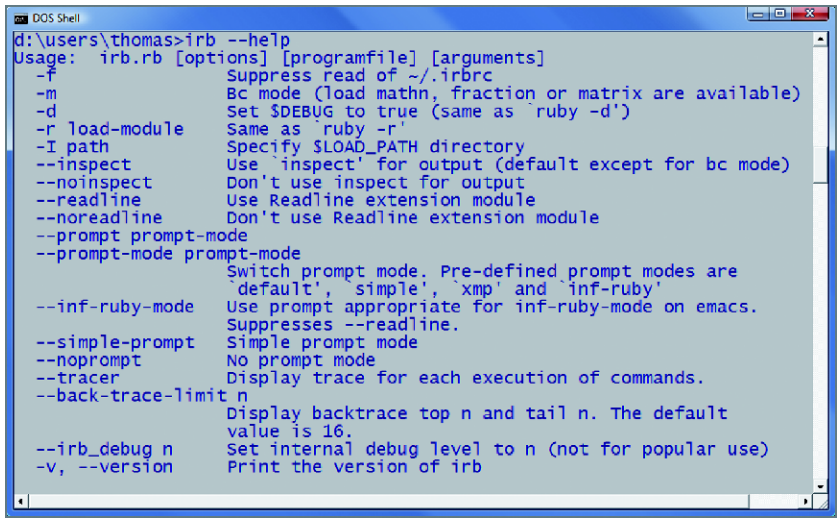



**Abbildung 13.7:**
Optionen für `irb`

Der Ruby-Interpreter kann über zahlreiche Parameter gesteuert werden (Abbildung 13.8); die wichtigeren hiervon sind

- `-w`: Ausgabe von Warnungen;
- `-W[level]`: Ausgabe von wenigen (Level 0) bis zu sehr vielen (Standard-Wert Level 2) Warnungen;
- `-c`: nur Syntax-Kontrolle des Scriptes;
- `-v`: Versions-Information des Ruby-Interpreters.

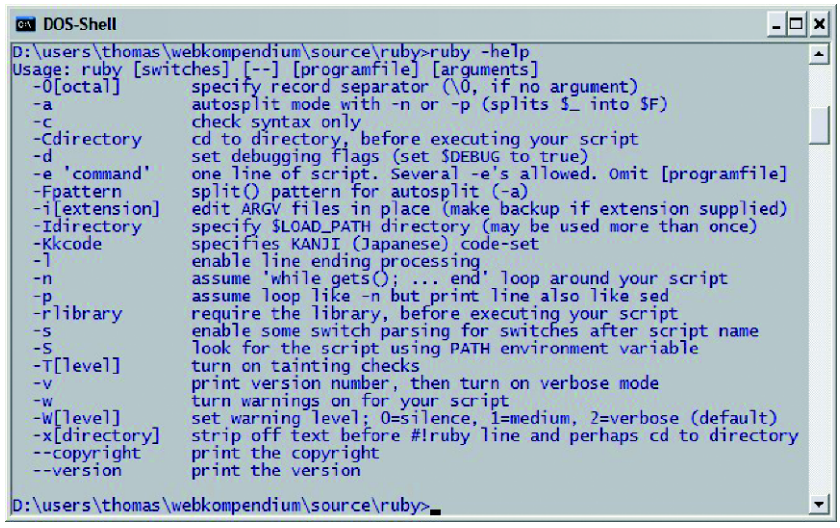



**Abbildung 13.8:**
Parameter des Ruby-Interpreters

Die Möglichkeit, den temporär vorhandenen compilierten Code zu speichern (wie Python mit der `pyc`-Datei), bietet Ruby momentan noch nicht, sie ist aber für die nächsten Versionen vorgesehen.

### 13.3.2 Grundlegende Syntax von Ruby

Eine Ruby-Scriptdatei (vgl. Abbildung 13.3) hat folgende prinzipielle Eigenschaften:

- es ist 7-bit ASCII-Code (über den Interpeter-Schalter `-Kkcode` kann auf den japanischen KANJI-Code umgeschaltet werden);
- Kommentare beginnen wie auf der Shell mit `#`; ein `;` am Zeilenende ist möglich, aber nicht zwingend;
- der Zeilenumbruch im Sourcecode wird durch \ ermöglicht.

Elementare Datentypen sind in Ruby im eigentlichen Sinne nicht vorhanden, da alle Variablen nur Referenzen auf Objekte sind. Folgende Arten von Datenstrukturen sind in Ruby standardmäßig durch Basisklassen angelegt:

- numerische Typen;
- Zeichenketten;
- Arrays bzw. Listen;
- Hashes;
- Ranges;
- Symbole;
- reguläre Ausdrücke.

#### 13.3.2.1 Numerische Typen

Auch *numerische Variablen* sind in Ruby keine elementaren Datentypen, sondern Instanzen entsprechender Basisklassen.

**In Java ist vergleichbares auch möglich, wenn beispielsweise anstelle des elementaren Datentyps `int` Instanzen der Klasse `java.lang.Integer` verwendet werden.**

Für ganze Zahlen stellt Ruby zwei Basisklassen bereit:

- `Fixnum` für die Zweierkomplementdarstellung ganzer Zahlen mit acht Byte;
- `Bignum` für beliebig große ganze Zahlen vergleichbar zur Klasse `java.math.BigInteger`, allerdings wegen der Operatorüberladung mit einfacherer Syntax in Ruby.

Interessant dabei ist, dass Ruby einen Überlauf im Typ `FixNum` automatisch erkennt und nach `BigNum` umwandelt, so dass der Anwender sich wenig Gedanken um diese Strukturen machen muss.

Für ganze Zahlen in Ruby gelten einige Besonderheiten:

- führendes `0x` kennzeichnet eine hexadezimale Darstellung: `0xcafe`;
- führendes `0` kennzeichnet eine oktale Darstellung: `0123`;
- führendes `0b` kennzeichnet eine binäre Darstellung: `0b110`;
- das Sonderzeichen `_` wird ignoriert: `4_711` ist eine korrekte Darstellung einer ganzen Zahl in Ruby.

Numerische Typen in Ruby sind Instanzen der entsprechenden drei Klassen; sie können dennoch einfach über

```
n = 313
```

erzeugt werden, es steht aber auch eine Erzeugung über

```
n = Fixnum.induced_from(313)
```

zur Verfügung.

Abbildung 13.9 zeigt ein Beispiel für den Umgang mit diesen Typen.

Neben diesen zwei Klassen für den Typ Ganzzahl stellt Ruby eine Klasse für Gleitkomma bereit: `Float`.

#### 13.3.2.2 Zeichenketten

Ruby stellt für *Zeichenketten* eine übliche Klasse `String` bereit. Instanzen dieser Klasse können sowohl über eine Zuweisung

```
wort = "WebKompendium"
```

als auch über den Konstruktor

```
wort = String::new("WebKompendium")
```

erzeugt werden.

Für Zeichenketten stehen die üblichen Escape-Sequenzen wie `\n` für einen Zeilenumbruch oder `\\` für `\` zur Verfügung.

Die schon aus Perl bekannten und praktischen Here-Documents bietet auch Ruby, in gleicher Syntax wie in Perl:

```
puts <<ende
Hallo WebKompendium
Here-Dokuments in Ruby
ende
```

Abbildung 13.9 zeigt ein Beispiel für die Erzeugung von Objekten für numerische Typen und für Zeichenketten.

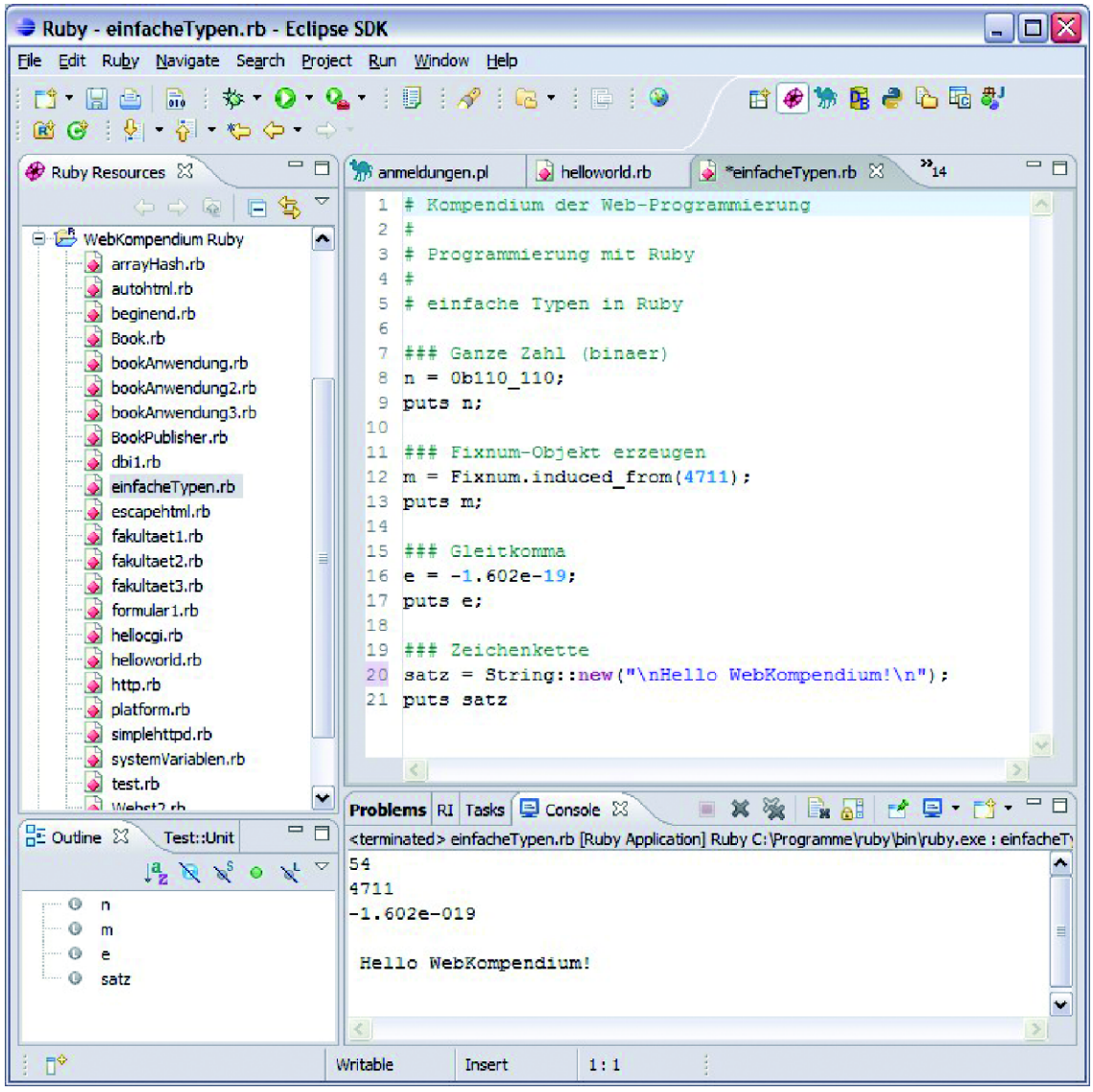



**Abbildung 13.9:** Einfache Typen in Ruby: Instanzen von Klassen (Script `einfacheTypen.rb`)

#### 13.3.2.3 Der Typ Range in Ruby

Ein *Range* in Ruby ist ein Bereich, vergleichbar zur Listen-Konstruktion in Perl (vgl. Seite 133):

- die Syntax `1..42` bezeichnet den Bereich von 1 bis einschließlich 42;
- im Gegensatz dazu bezeichnet `1...42` den Bereich von 1 bis 41.

#### 13.3.2.4 Arrays und Listen in Ruby

Wie in Scriptsprachen üblich, verfügt Ruby nicht über das Konzept des in der Größe festgelegten Arrays, sondern stellt direkt die Datenstruktur der *linearen Liste* bereit, die dynamisch erweiterbar ist.

Eine Liste in Ruby wird durch die Syntax

```
liste = [1,313,"WebKompendium"]
```

definiert; diese Liste kann – wie in diesem Beispiel – Elemente verschiedenen Typs enthalten; auf die einzelnen Elemente wird wieder über den Index, beginnend mit dem Index 0, zugegriffen:

```
puts liste[2]        # Ausgabe: WebKompendium
```

Durch typische Listenoperationen kann die Liste nun verändert werden, etwa legt

```
liste.push(4711)
```

ein weiteres Element auf der Liste ab (dann mit dem Index 3).

#### 13.3.2.5 Assoziative Listen in Ruby

Neben den über eine numerische Variable indizierten Listen stellt Ruby auch Key-Value-Strukturen wie das Hash in Perl bereit. Dies geschieht über die Syntax

```
hash = { key1 => value1, key2 => value2, ... }
```

Über den jeweiligen Schlüssel kann dann auf den Wert zugegriffen werden:

```
puts hash[key]
```

Abbildung 13.9 zeigt ein Beispiel für die Anwendung von Listen und Hashes in Ruby.

### 13.3.3 Bezeichner und Sonderzeichen in Ruby

Ähnlich zu Perl verfügt Ruby über einige Besonderheiten beim Aufbau von *Bezeichnern;* zwar gibt es das Symbol $ nicht mehr mit seiner Funktion für einfache Variablen, aber es bestehen einige andere Regeln:

- lokale Variablen in Klassen beginnen mit einem Kleinbuchstaben oder einem führenden _;
- Instanzvariablen von Ruby-Klassen beginnen mit führendem @;
- Klassenvariablen, also solche, welche in Java mit dem Attribut `static` gekennzeichnet werden, beginnen mit führenden @@.
- globale Variablen sind in Ruby ebenfalls möglich: diese beginnen mit $.

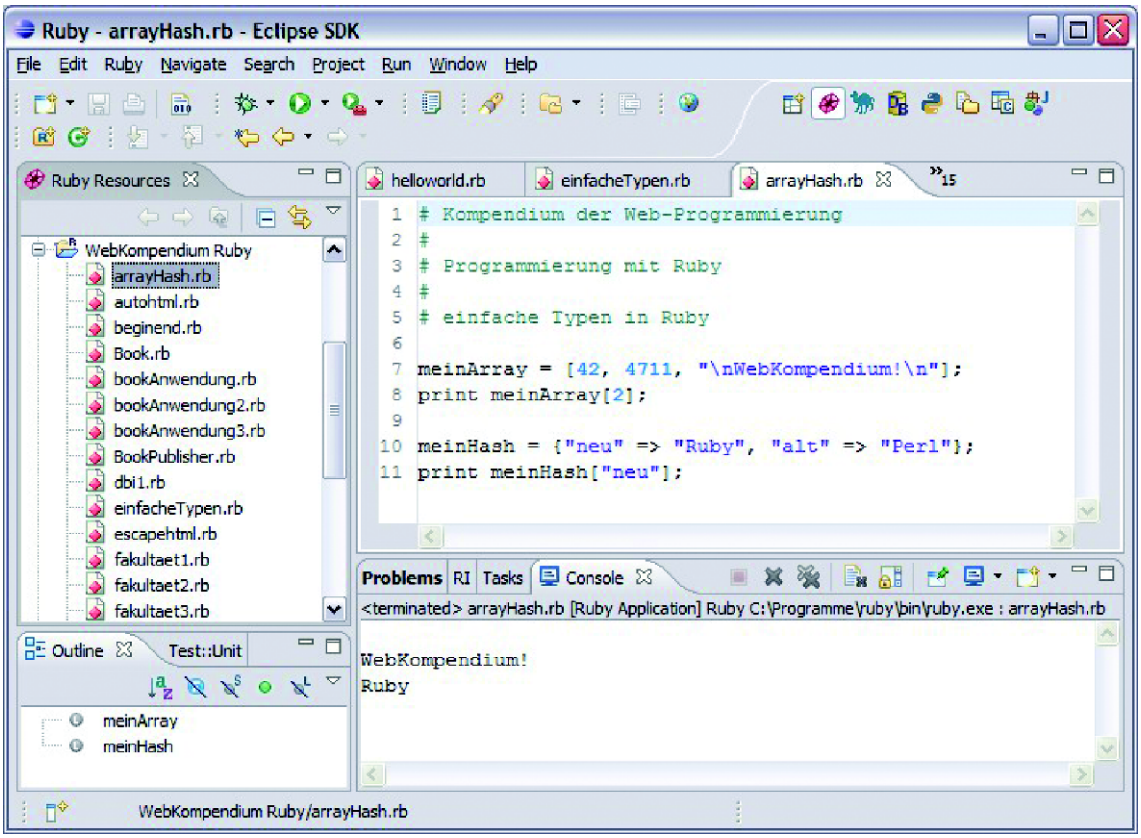



**Abbildung 13.10:** Listen und Hashes in Ruby (Script `arrayHash.rb`)

### 13.3.4 Systemvariablen in Ruby

Ruby stellt viele *Systemvariablen* bereit, welche für den Einsatz der Sprache sehr praktisch sind:

- die globale Variable `$>` bezeichnet das Ziel der Standardausgabe stdout;
- `$0` ist der Name des ausgeführten Ruby-Scriptes;
- `$*` ist eine Liste mit den beim Start übergebenen Parametern (entsprechend `String[] args` der main-Methode in Java);
- `$$` ist die Prozess-ID des Ruby-Programms;
- `ENV` bezeichnet einen Hash mit allen Umgebungsvariablen;
- `RUBY_VERSION` gibt die Ruby-Version des Interpreters an.

Abbildung 13.11 zeigt den Einsatz dieser Systemvariablen.

### 13.3.5 Operatoren in Ruby

Ruby verfügt über ein Set von *Operatoren*, das typisch für moderne Programmiersprachen ist, wie `+`, `-`, `*`, `/`, `%`, `+=`. Nicht vorhanden sind die beliebten Operatoren `++` und `--`, welche aber einfach durch `+=1` umschrieben werden können, allerdings ohne Steuerung der Priorität. Darüber hinaus gibt es in Ruby auch ein paar weitere, eher besondere Operatoren von `**` für das Potenzieren bis zu `not` für das logische nicht und `defined?` für das Überprüfen der Initialisierung.

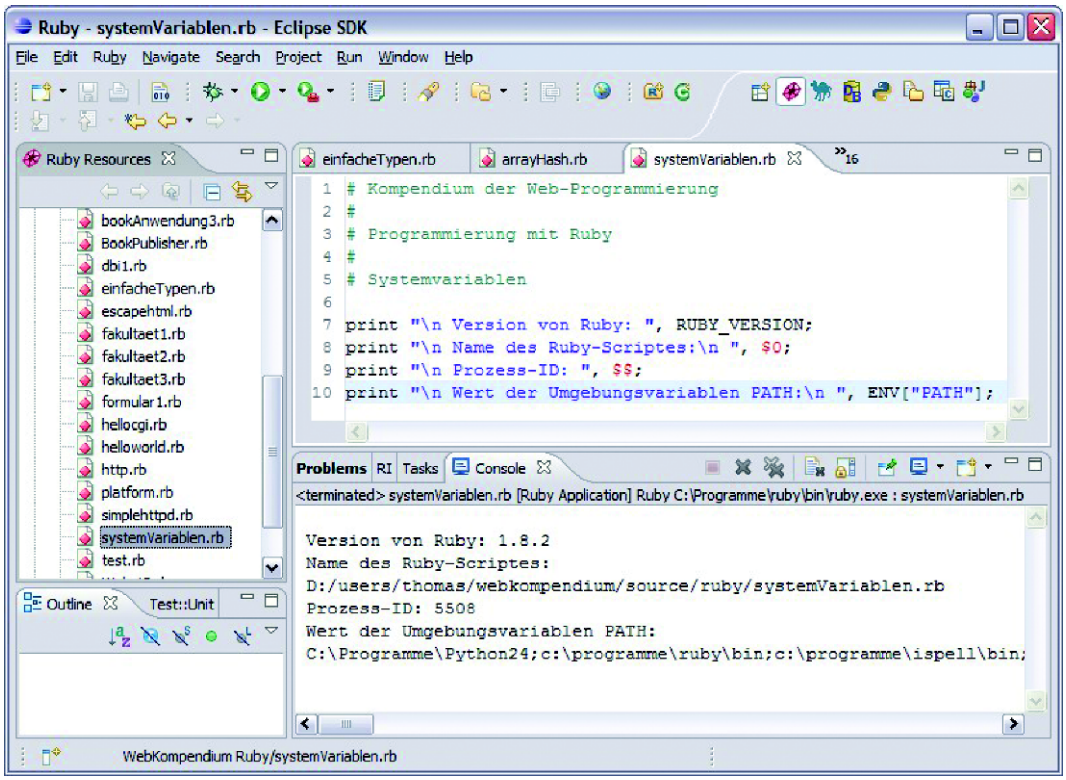



**Abbildung 13.11:**
Systemvariablen in Ruby (Script `systemVariablen.rb`)

### 13.3.6 Kontrollstrukturen in Ruby

Die *Kontrollstrukturen* entsprechen in Ruby weitgehend den üblichen Prinzipien, es gibt aber trotz des Ruby-Paradigmas der „kleinsten Überraschung“ ein paar bemerkenswerte Abweichungen.

#### 13.3.6.1 Logik in Ruby

Ruby implementiert keine Datenstruktur für logische Variablen der Art boolean, sondern geht wie üblich für Scriptsprachen vor:
Null-Werte werden als falsch interpretiert, alles andere ist wahr.

Darüber hinaus bietet Ruby die üblichen logischen Operatoren und Vergleichsausdrücke wie ==, !=, <, <=, && für „und“ sowie || für „oder“ .

Obwohl ein Datentyp boolean nicht vorhanden ist, verfügt Ruby über die Schlüsselwörter `false` und `true`.

#### 13.3.6.2 Sequenzielle Komposition in Ruby

Zu den eher unüblichen Syntax-Elementen einer modernen Sprache zählt in Ruby die *sequenzielle Komposition,* der Block; dieser wird nicht durch ein paar geschweifter Klammern `{ ... }` begrenzt – oder gar wie in Python durch Einrücken –, sondern schon wie in Pascal durch eine Struktur `BEGIN ... END`:

```
BEGIN
   anweisung
END
```

Dieses Prinzip gilt nicht nur für den einfachen Block, sondern für alle Kontrollstrukturen; so wird etwa eine while-Schleife ebenfalls durch ein `END` abgeschlossen.

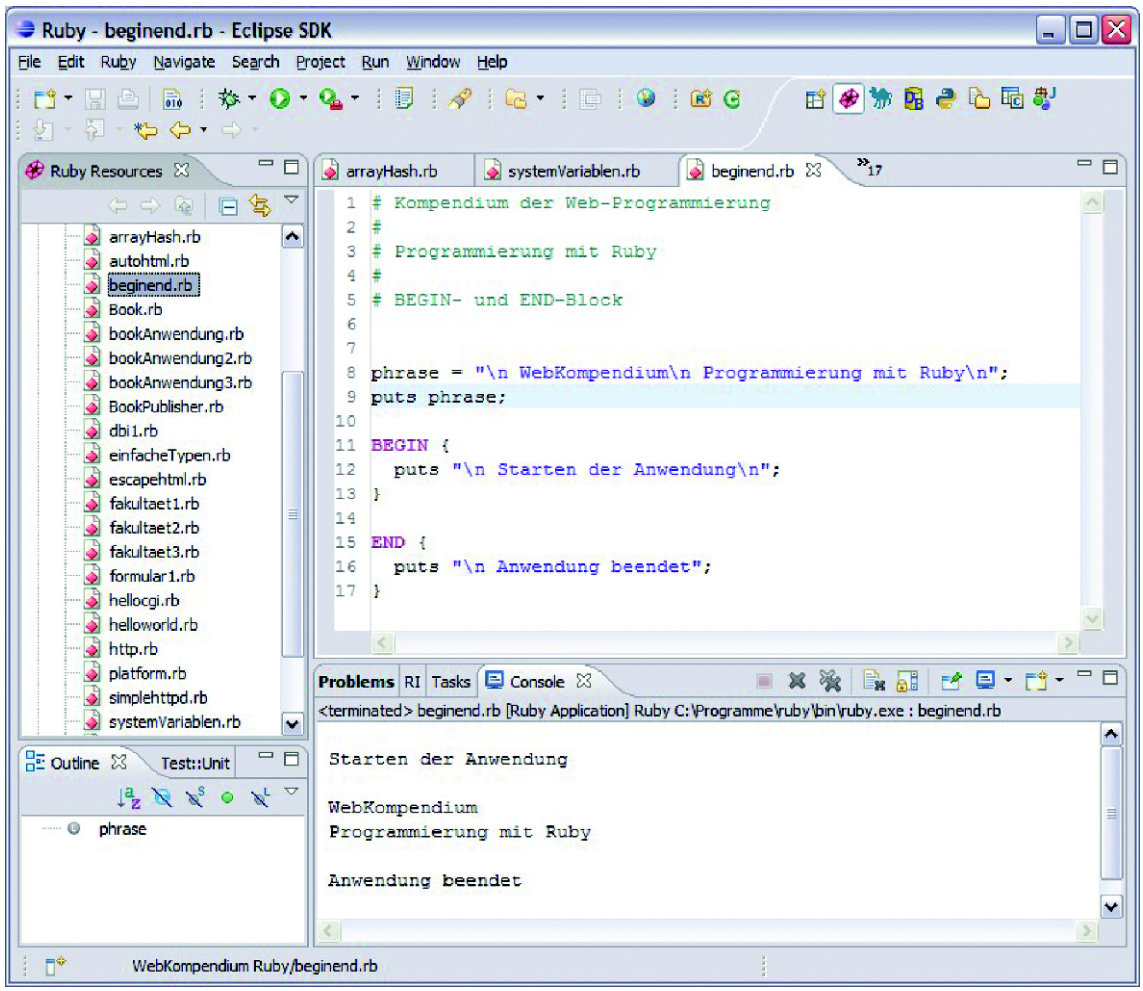



**Abbildung 13.12:** BEGIN- und END-Block in Ruby (Script `beginend.rb`)

### 13.3.6.3 BEGIN- und END-Block in Ruby

Eine weitere Besonderheit sind zwei spezielle Blöcke, welche beim Laden/-Starten der Ruby-Datei (BEGIN-Block) bzw. beim Beenden (END-Block) ausgeführt werden:

```
BEGIN {
   # Anweisungen beim Start
}

END {
   # Anweisungen beim Beenden
}
```

### 13.3.6.4 Entscheidung in Ruby

Neben den gerade vorgestellten Besonderheiten verfügt Ruby über die übliche Verzweigung wenn-dann-sonst mit der Syntax

```
if boole [then]
   anweisung
else
   anweisung
end
```

Das `then` ist optional und kann genauso wie der ganze else-Zweig auch entfallen; wie in Ruby üblich wird diese Kontrollstruktur mit `end` abgeschlossen.

Ruby bietet ebenfalls die komplexere Struktur mit einem else-if der Syntax

```
if boole1 [then]
   anweisung
elsif boole2 [then]
   anweisung
else
   anweisung
end
```

Hier können wieder beliebig viele `elsif`-Zweige verwendet werden.

#### 13.3.6.5 Schalter in Ruby

Neben der einfachen Verzweigung bietet Ruby auch eine *Schalterstruktur* (case). Diese hat in Ruby die Syntax

```
case schaltervariable
when wert1 [then] anweisung
when wert2 [then] anweisung
...
[else anweisung]
end
```

Der else-Zweig ist optional und wird ausgeführt, wenn kein anderer Fall eintritt. Ein verdeutlichendes Beispiel zeigt das Script `plattform.rb` (Abbildung 13.13).

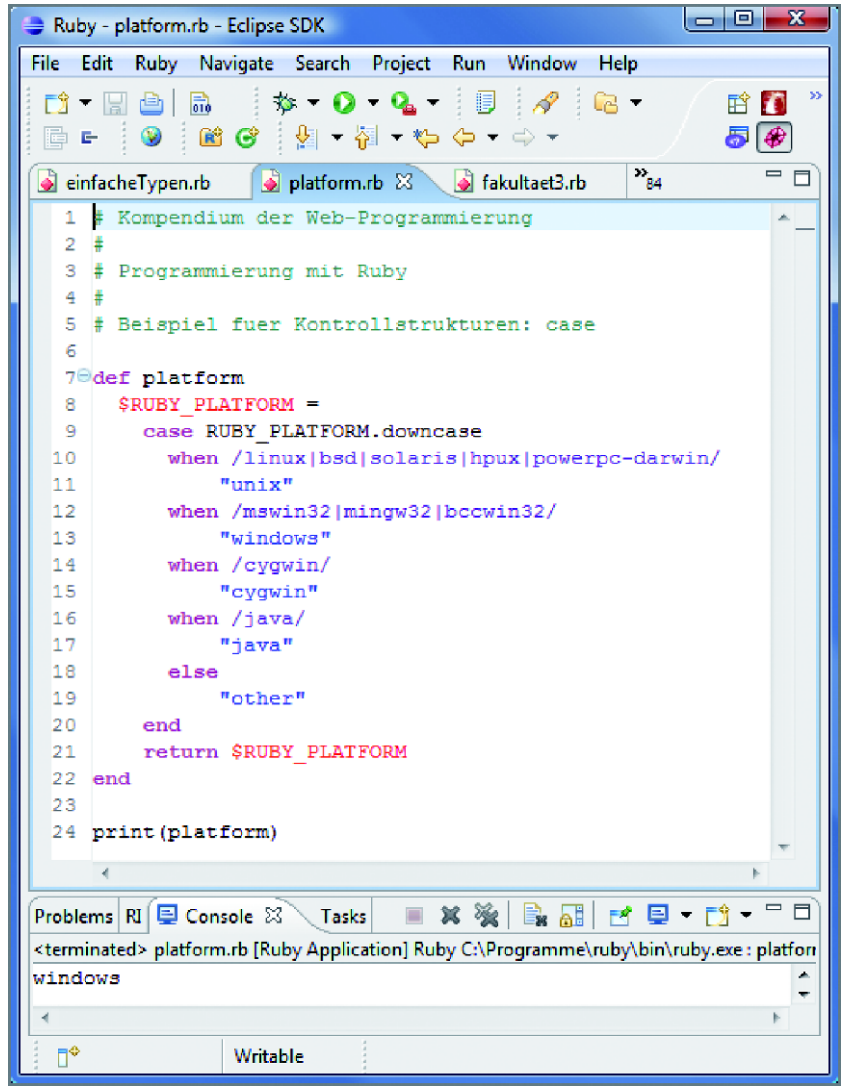



**Abbildung 13.13:** case-Struktur in Ruby (Script `plattform.rb`)

### 13.3.6.6 Alternative Syntax und Iteratoren in Ruby

Ruby bietet für die Kontrollstrukturen eine *alternative Syntax.* Am Beispiel der if-Verzweigung lautet diese:

```
anweisung if  boole
```

Dies bedeutet, dass `anweisung` nur ausgeführt wird, wenn `boole` als wahr im booleschen Sinne interpretiert wird.

Die entsprechende Syntax steht auch für Schleifen zur Verfügung; Ruby spricht dann von einem *Iterator;* die Syntax für Iteratoren hat Ruby von der Programmiersprache CLU übernommen.

#### 13.3.6.7 while-Schleife in Ruby

Die Standard-Schleife `while` ist in Ruby in der üblichen Syntax implementiert:

```
while boole [then]
   anweisung
end
```

Dies bedeutet, dass `anweisung` ausgeführt wird, so lange `boole` als wahr interpretiert wird.

Ruby bietet mittels `until` auch das logische Gegenteil, die Schleife, die ausgeführt wird, so lange die Schleifenbedingung falsch ist:

```
until boole [then]
anweisung
end
```

#### 13.3.6.8 for-Schleife in Ruby

Die `for`-Schleife ist in Ruby wie in Python implementiert und iteriert über die Elemente einer Liste:

```
for variable in liste [do]
   anweisung
end
```

Dabei nimmt die Variable `variable` alle Werte aus der Liste `liste` an. Das Script `fakultaet1.rb` zeigt ein Beispiel für diesen Schleifentyp, das zusätzich die Ruby-Ranges benutzt,

#### 13.3.6.9 Verlassen von Kontrollstrukturen

Ruby bietet auch Möglichkeiten zum *Verlassen von Kontrollstrukturen* (drei Möglichkeiten bei Schleifen, für die Schalter-Struktur `case` nur `break`):

- `break` beendet die Struktur;
- `redo` startet den aktuellen Durchlauf erneut;
- `next` beginnt den nächsten Durchlauf nach Überprüfung der Bedingung;
- `retry` überprüft die Schleifenbedingung erneut.

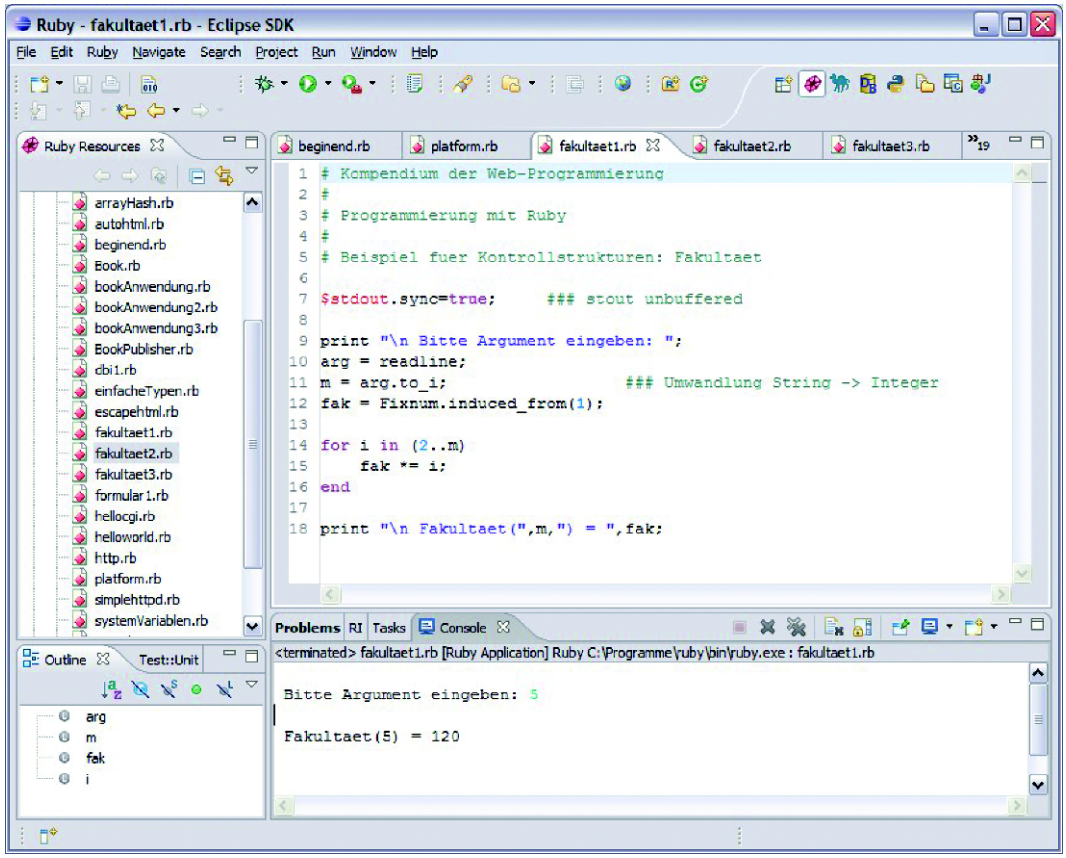



**Abbildung 13.14:** for-Schleife in Ruby (Script `fakultaet1.rb`)

## 13.4 Objektorientierung mit Ruby

Die Scriptsprache Ruby zeichnet sich besonders durch die *Objektorientierung* aus, da hier die OO-Prinzipien umfassend umgesetzt sind; dies ist sicherlich einer der Vorteile von Ruby.

### 13.4.1 Methodendefinition mit Ruby

*Methoden* in Ruby werden durch das Schlüsselwort `def` definiert; sie können eine Liste nicht-typisierter Argumente aufweisen, welche zusätzlich mit default-Werten versehen werden können, was eine Art Überladen ermöglicht:

```
def methodenBezeichner (arg [=val])
   anweisung
end
```

Das Script `fakultaet2.rb` (Abbildung 13.15) zeigt ein solches Beispiel.

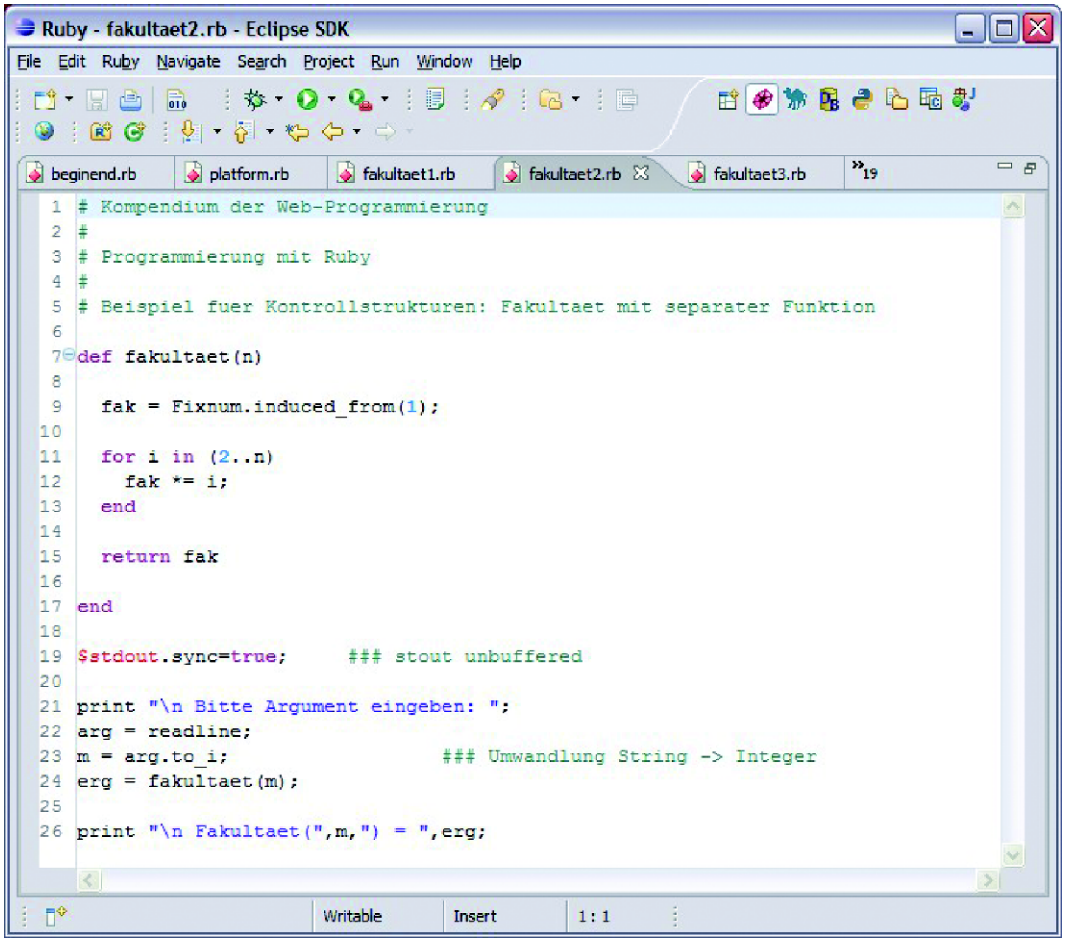



**Abbildung 13.15:**
Definition einer Methode (Script `fakultaet2.rb`)

### 13.4.2 Aliasing mit Ruby

In Ruby können für Bezeichner *Aliase* definiert werden durch die Syntax

```
alias alterBezeichner neuerBezeichner
```

### 13.4.3 Klassen mit Ruby

Der zentrale Begriff der *Klasse* wird durch das Schlüsselwort `class` in Ruby eingeführt; dabei kann auch direkt eine Elternklasse angegeben werden (vgl. 13.4.4):

```
class KlassenName<ElternKlasse
    anweisung
end
```

Die Angabe der Elternklasse ist dabei optional – das moderne Design von Ruby erkennt man aber daran, dass ohne Angabe einer Elternklasse die Klasse von der „Urklasse“ `Object` angegeben wird. Damit ist letztlich jede Klasse von `Object` abgeleitet.

Innerhalb einer Klasse wird in Ruby die Systematik der Bezeichner, wie in Abschnitt 13.3.3 vorgestellt, verwendet; so beginnen Instanzvariablen mit `@` und Klassenvariablen mit `@@`.

Als Beispiel ist die Klasse `Book` in der Ruby-Datei `Buch.rb` definiert (Abbildung 13.16).

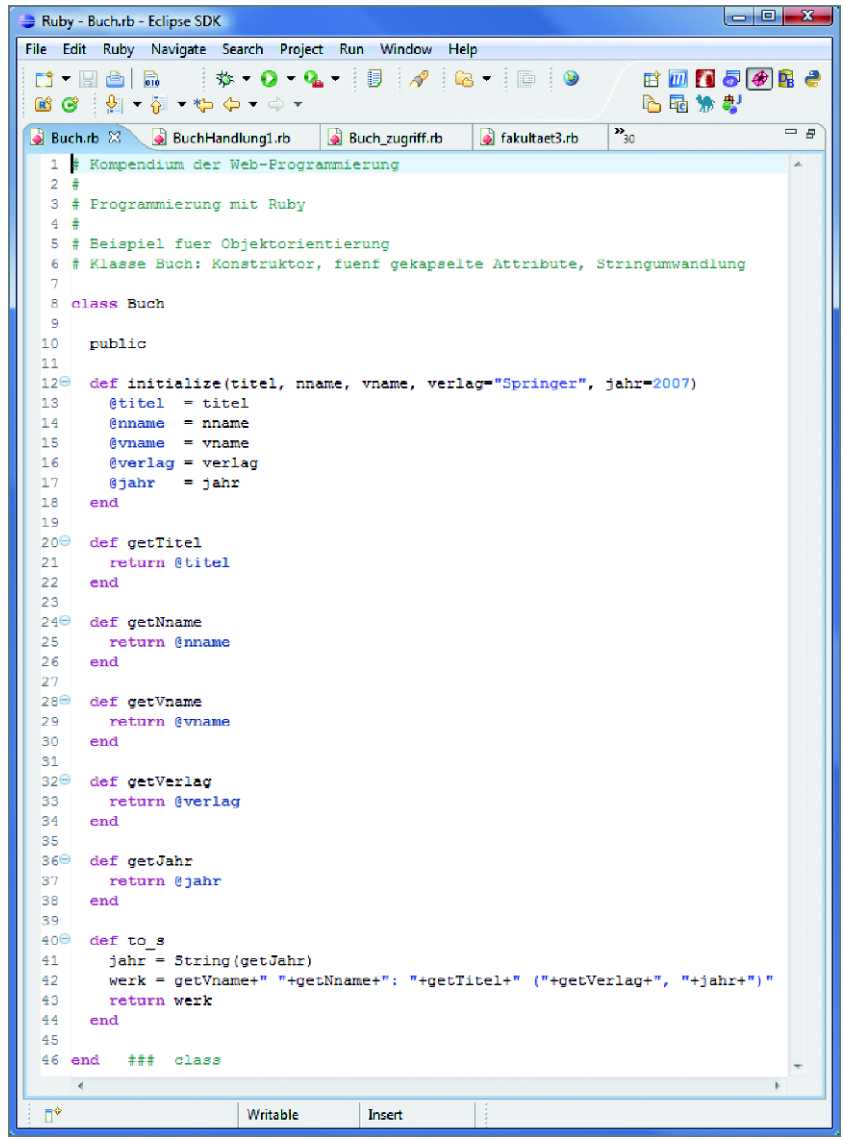



**Abbildung 13.16:** Definition der Ruby-Klasse `Book` (Datei `Buch.rb`)

Wie in 4.2.2 erläutert, sollten die Bezeichner von Klassen mit Großbuchstaben beginnen.

#### 13.4.3.1 Instanziieren von Klassen und der Konstruktor

*Instanzen einer Klasse* werden in Ruby mittels `new` erzeugt:

```
referenz = KlassenName.new([argumente])
```

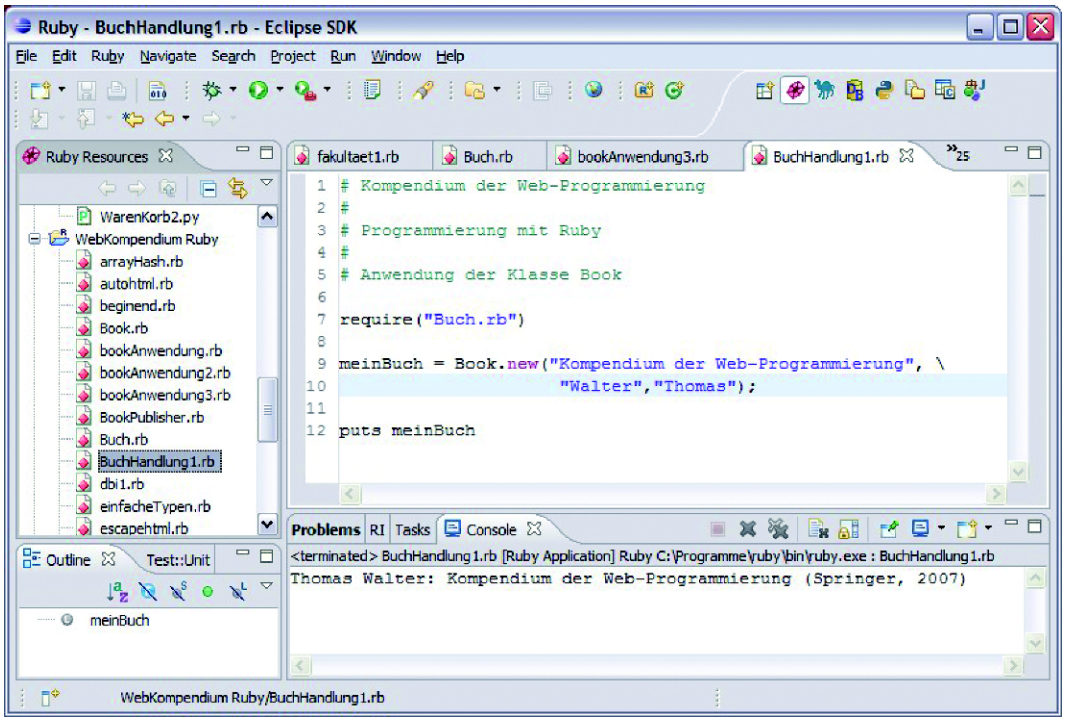



**Abbildung 13.17:**
Erzeugen einer Instanz der Klasse `Book` (Script `BuchHandlung1.rb`)

Durch die Methode `new` wird der *Konstruktor* einer Klasse aufgerufen. Der Konstruktor in Ruby ist eine Methode mit dem festen Bezeichner `initialize`, welchem über eine Argumentliste Werte übergeben werden, wobei auch Default-Werte zum Einsatz kommen können.

### 13.4.3.2 Bündelung von Software: Module in Ruby

Zur *Bündelung von Software* verwendet Ruby den Begriff des *Moduls*. Ein Ruby-Modul wird mit dem Schlüsselwort `module` eingeleitet (und wie üblich mit `end` abgeschlossen); sie bilden eine Softwarebibliothek mit einem lokalen Namensraum und stellen Mixins (vgl. 13.4.5), die Ruby-Variante der Mehrfachvererbung, bereit. Abbildung 13.18 zeigt eine Klasse (`Buchmodul`) innerhalb eines Moduls; Abbildung 13.19 zeigt die Anwendung dieser Klasse aus einem Modul, indem zunächst durch `require` die Ruby-Datei importiert und dann durch `include` das Modul eingebunden wird:

- `require` importiert ein Ruby-Script einmalig;
- `load` importiert bei jedem Aufruf die Ruby-Datei erneut;
- `include` ermöglicht den Zugriff auf ein Modul.

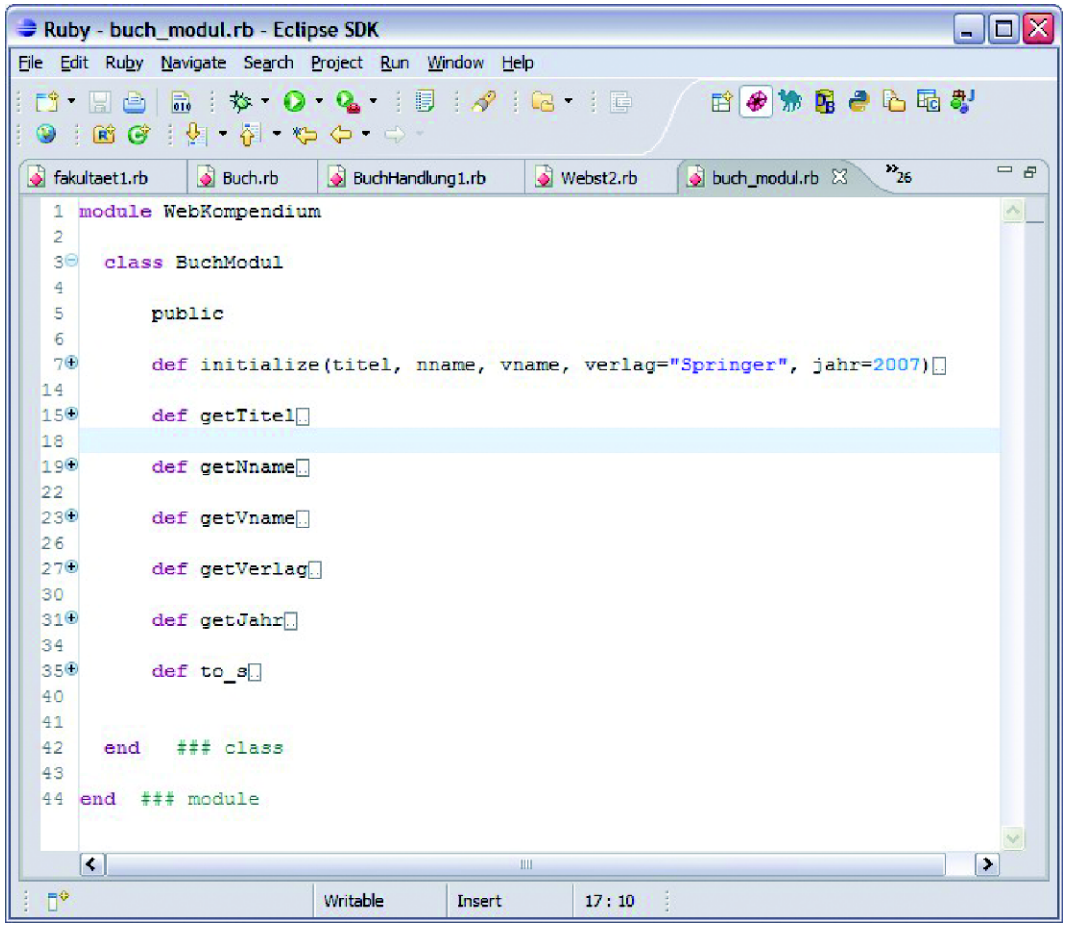



**Abbildung 13.18:** Klasse `BuchModul` im Modul `WebKompendium` (Script `buch_modul.rb`)

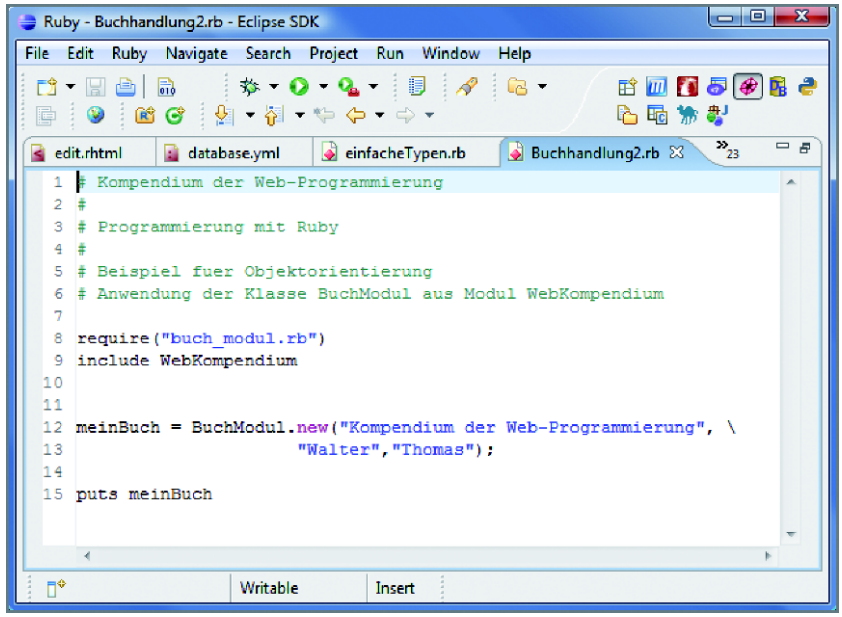



**Abbildung 13.19:** Erzeugen einer Instanz der Klasse `BuchModul` (Script `BuchHandlung2.rb`)

#### 13.4.3.3 Der Klassenpfad von Ruby

Ruby verwendet die Umgebungsvariable `LOAD_PATH` für die Angabe des *Klassenpfads,* also für die Angabe der Verzeichnisse, in welchen Ruby nach Modulen sucht. Diese Variable ist innerhalb des Ruby-Laufzeitsystems unter der Variablen `$:` direkt verfügbar (und kann dort auch gesetzt/verändert werden).

### 13.4.4 Vererbung in Ruby

Die Objektorientierung in Ruby ermöglicht selbstverständlich die Vererbung; über die Syntax

```
class KindKlasse < ElternKlasse
```

wird die Vererbung definiert. Wird keine Elternklasse angegeben, so wird – wie in Java – von der Basisklasse `Object` abgeleitet, von welcher letztlich jede Ruby-Klasse abgeleitet ist. Dies entspricht der Klasse `java.lang.Object` in Java.

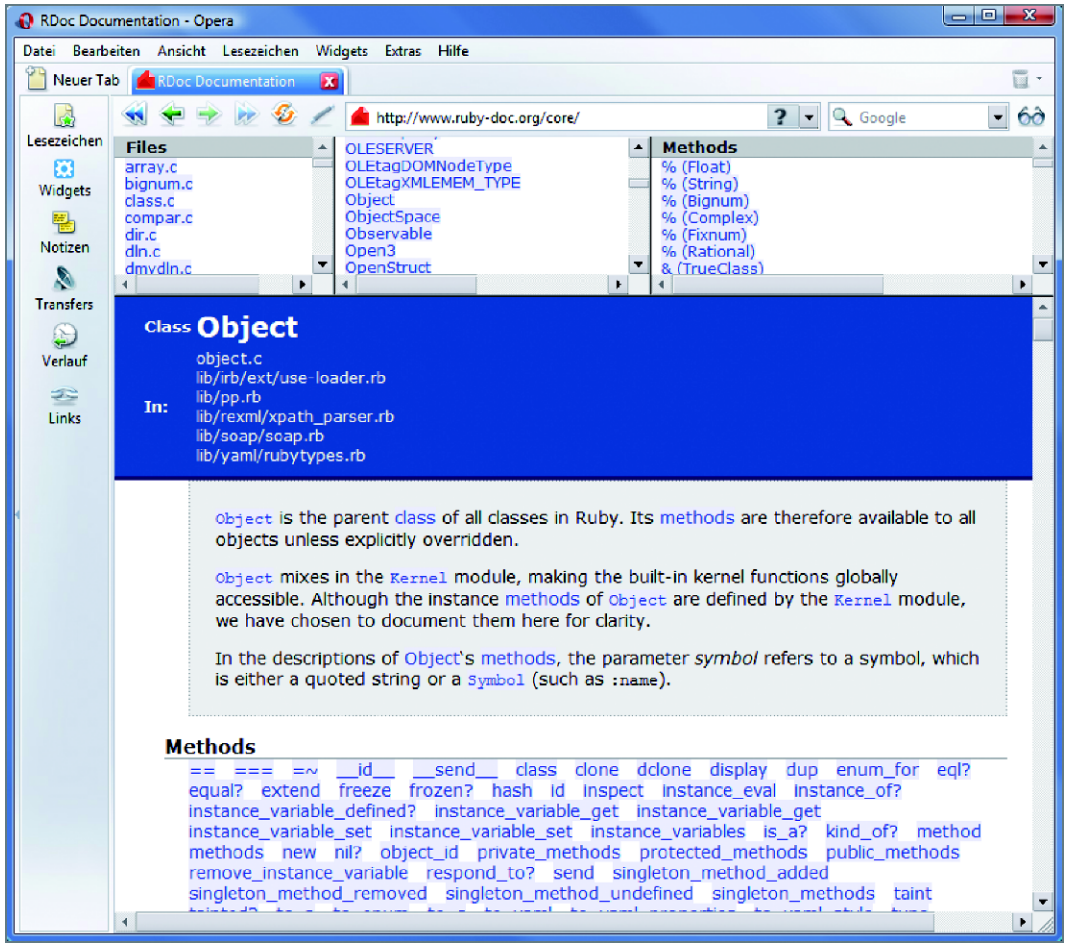



**Abbildung 13.20:**
rdoc für die Ruby-Klasse `Object`

Exemplarisch soll die Ausgangsklasse `Buch` zur Klasse `BuchIsbn` abgeleitet werden, welche ein weiteres Attribut `@isbn` hinzufügt. Der Konstruktor der neuen Klasse belegt die Attribute, wobei aber nur das neue Element explizit zu behandeln ist, die übrigen Attribute werden durch den gezielten Aufruf des Konstruktors der Elternklasse `super(...)` gesetzt (Abbildung 13.21).

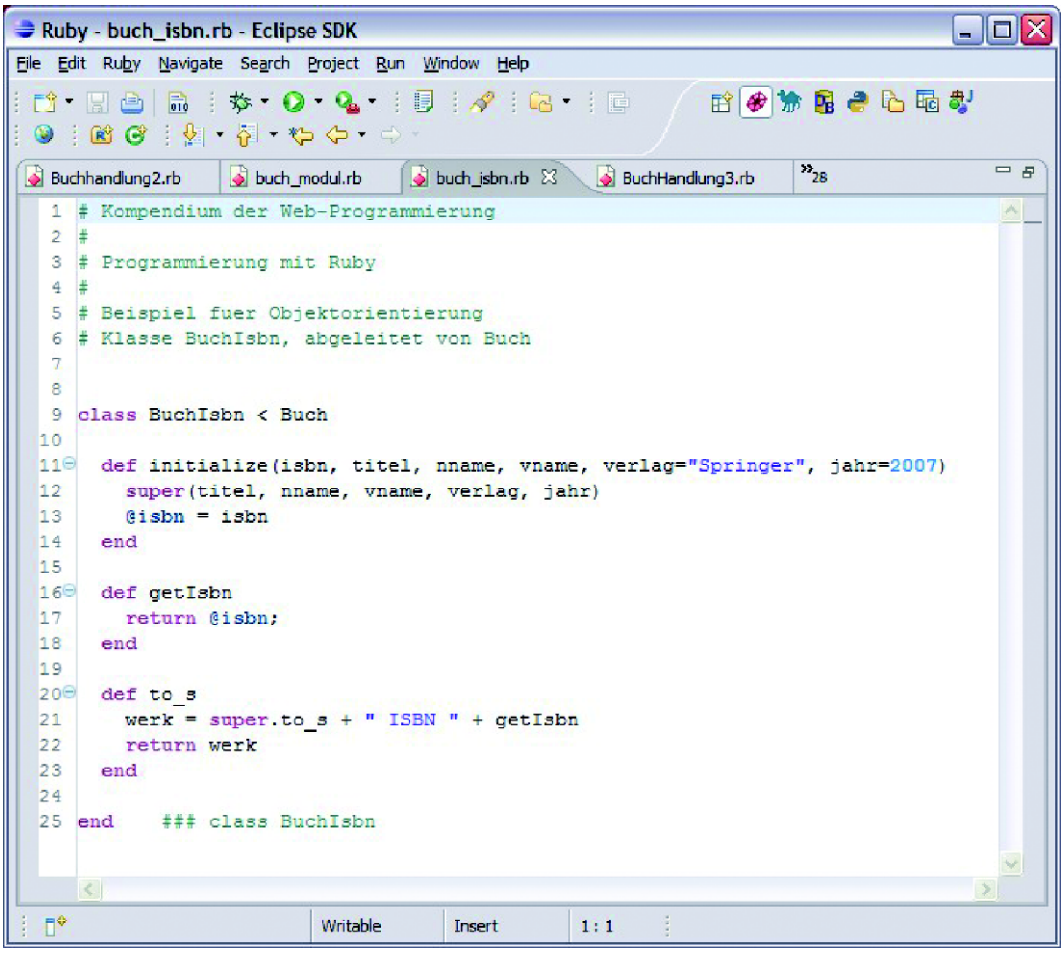



**Abbildung 13.21:** Die abgeleitete Klasse `BuchIsbn` (`buch_isbn.rb`)

In der Kindklasse wird außerdem die Methode `to_s` überschrieben, damit das neue Attribut ebenfalls ausgegeben wird; auch dies geschieht durch Verwendung der Elternklasse: Durch die Syntax `super.to_s` wird zunächst die Methode der Elternklasse aufgerufen und deren Rückgabewert dann erweitert.

Das Ruby-Script `Buchhandlung3.rb` erzeugt eine Instanz dieser Klasse (Abbildung 13.22).

### 13.4.5 Mehrfachvererbung in Ruby: Mixins

Ruby stellt wie Java keine *Mehrfachvererbung* bereit, hat hierfür aber eine „schwache Alternative“, die Ruby als *Mixins* bezeichnet: Methoden werden dabei direkt in Modulen und nicht in Klassen definiert. Diese Module sind nicht instanziierbar, es können aber Klassen diese Module „inkludieren“. Als Folge stehen dann die importierten Methoden in der Klasse zur Verfügung und können auch auf Instanzen angewendet werden.

Bei diesem Vorgehen werden die Methoden „beigemischt“: mixed in, was zum Namen dies Verfahrens führt.[2]

[2] auch für Instanzvariablen möglich.

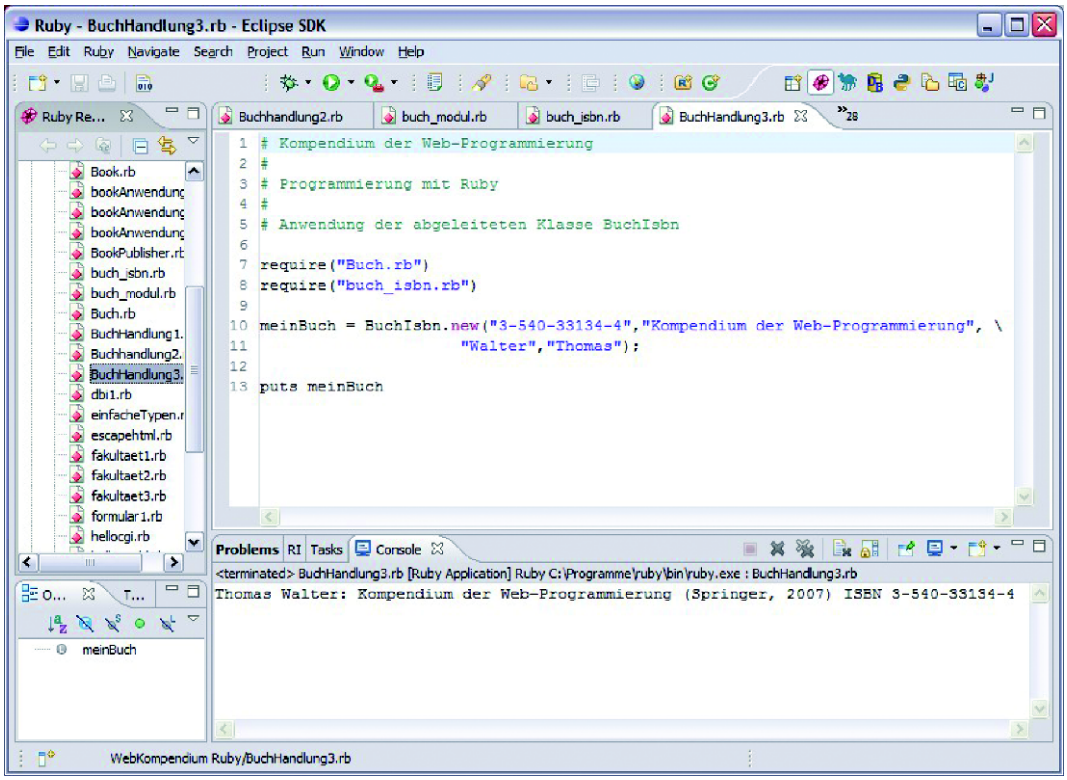



**Abbildung 13.22:**
Erzeugung einer Instanz von `BuchIsbn` (`buchhandlung3.rb`)

### 13.4.6 Zugriffsrechte und Datenkapselung

Die Objektorientierung in Ruby stellt *Zugriffsrechte* bereit, ähnlich dem, was Java ermöglicht:

- `public` für den öffentlichen Zugriff;
- `protected` erlaubt nur den Zugriff von Objekten der eigenen Klasse oder von Unterklassen;
- `private` erlaubt nur der eigenen Klasse mit dem Schlüsselwort `this` den Zugriff.

#### 13.4.6.1 Das Schlüsselwort `attr`

Ruby bietet über das Schlüsselwort `attr` eine einfache Möglichkeit, eine gekapselte Instanzvariable samt Zugriffsmethoden zu erzeugen:

```
attr :x, true
```

ist gleichwertig zur Definition von zwei Methoden:

```
def x
   return @x
end

def x=(val)
    @x = val
end
```

Der boolesche Parameter bei der Deklaration von `attr` gibt an, ob nur ein lesender (false) oder ein lesender- und schreibender Zugriff möglich sein soll. Diese Unterscheidung wird auch durch folgende Syntax möglich:

- ein lesender Zugriff über `attr_reader:x`;
- ein schreibender Zugriff über `attr_writer:x`;
- sowohl lesender als auch schreibender Zugriff: `attr_accessor:x`.

### 13.4.7 Exceptions in Ruby

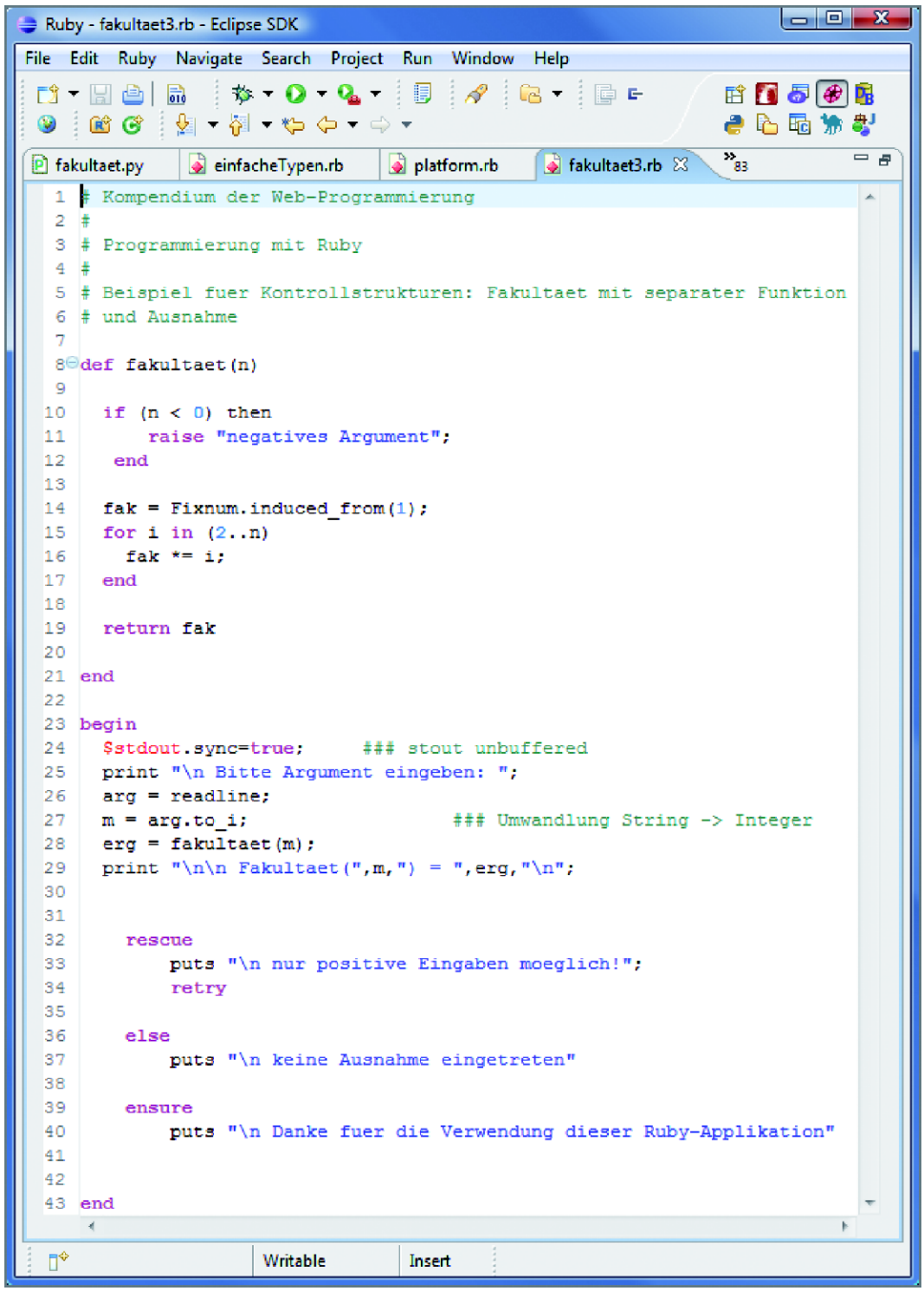
Ausnahmesituationen in Ruby werden über *Exceptions* behandelt: Innerhalb eines `begin–end`-Blockes kann über das Schlüsselwort `raise` eine Runtime-Exception zur Laufzeit ausgelöst werden; der Methode `raise` kann auch ein Argument übergeben werden:

```
raise "ungueltiges Argument"
```

löst eine solche Ausnahme mit der Meldung „ungueltiges Argument“ aus. Diese Ausnahme wird auf der Standardausgabe angezeigt und die Ruby-Ausführung endet – es sei denn, über `rescue [message]` wird die Ausnahme – optional nur spezifisch zur entsprechenden Informationsmeldung `message` – abgefangen. Es kann auch ein `else`-Block folgen, welcher nur ausgeführt wird, wenn kein `rescue` eingetreten ist, sowie ein `ensure`-Block, welcher in jedem Fall ausgeführt wird. Eine praktische Besonderheit in Ruby ist innerhalb des `rescue`-Blocks die Anweisung `retry`: Hierdurch wird der ursprüngliche `begin–end`-Block, in welchem das `raise` auftrat, erneut durchlaufen.

Abbildung 13.23 zeigt ein Beispiel: Die Methode `fakultaet` löst im Falle eines negativen Arguments eine Ausnahme aus; im aufrufenden Programmsegment wird diese durch einen `rescue`-Block abgefangen; dieser verwendet die Kontrollstruktur `retry`, um nach Auftreten der Ausnahme den Ursprungsblock mit der Eingabe erneut aufzurufen.

**Der `begin–end`-Block entspricht in Java dem `try`-Block, das `raise` entspricht in Java `throw`, `rescue` entspricht in Java dem `catch`, `ensure` entspricht in Java `finally`, `else` und `retry` haben keine direkten Entsprechungen in Java.**




```
1  # Kompendium der Web-Programmierung
2  #
3  # Programmierung mit Ruby
4  #
5  # Beispiel fuer Kontrollstrukturen: Fakultaet mit separater Funktion
6  # und Ausnahme
7
8  def fakultaet(n)
9
10   if (n < 0) then
11       raise "negatives Argument";
12    end
13
14   fak = Fixnum.induced_from(1);
15   for i in (2..n)
16     fak *= i;
17   end
18
19   return fak
20
21 end
22
23 begin
24   $stdout.sync=true;      ### stout unbuffered
25   print "\n Bitte Argument eingeben: ";
26   arg = readline;
27   m = arg.to_i;                  ### Umwandlung String -> Integer
28   erg = fakultaet(m);
29   print "\n\n Fakultaet(",m,") = ",erg,"\n";
30
31
32     rescue
33         puts "\n nur positive Eingaben moeglich!";
34         retry
35
36     else
37         puts "\n keine Ausnahme eingetreten"
38
39     ensure
40         puts "\n Danke fuer die Verwendung dieser Ruby-Applikation"
41
42
43 end
```

**Abbildung 13.23:**
Methode `fakultaet` löst für negative Argumente eine Exception aus, welche im aufrufenden Block abgefangen wird (Script `fakultaet3.rb`)

## 13.5 Ruby im Web

Die Scriptsprache Ruby bietet vielfältige Möglichkeiten des Einsatzes im Web, die sich meistens auf serverseitige Programmierungen beziehen. Kern für das

Verständnis ist zunächst der Einsatz als CGI-Sprache; hier ist Ruby – ebenso wie beim Datenbankzugriff – eng an Perl orientiert. Darauf aufbauend wird Ruby als Grundlagen für zahlreiche moderne Frameworks verwendet; das momentan bekannteste ist Ruby on Rails (vgl. Kapitel 23).

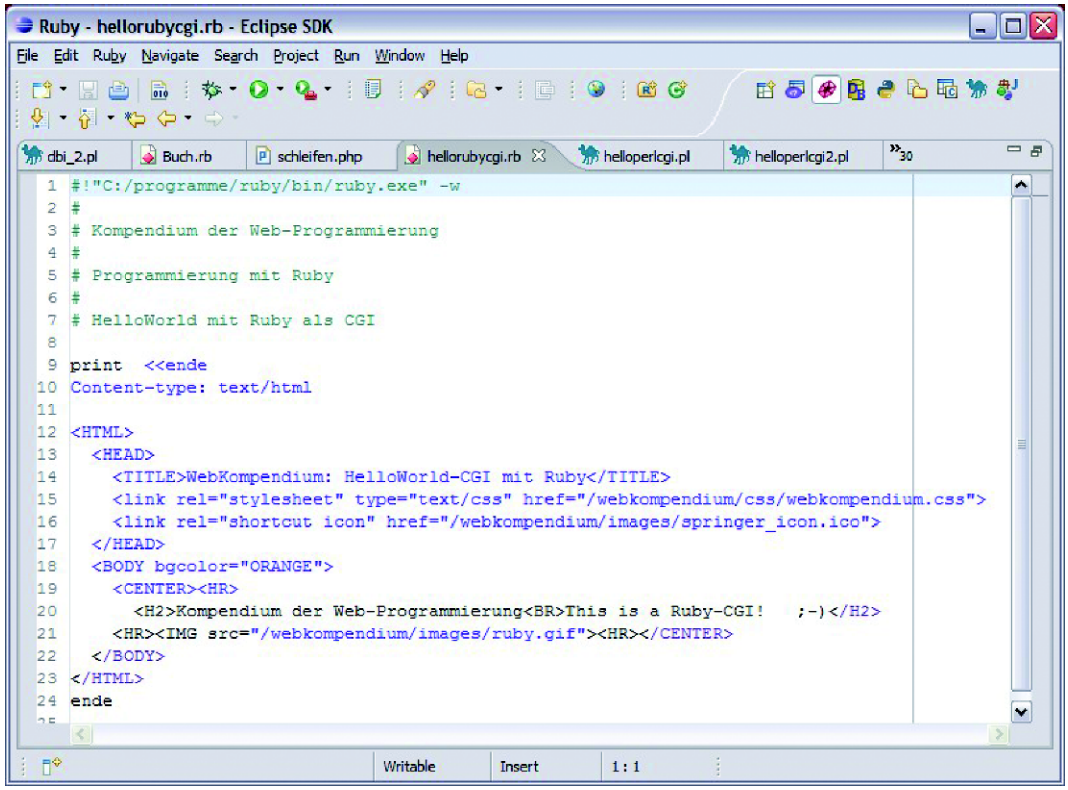



**Abbildung 13.24:** CGI-Script `hellorubycgi.rb` mit Here-Document

### 13.5.1 Ein einfaches CGI mit Ruby

Mit Ruby kann natürlich direkt ein einfaches CGI-Script entworfen werden; serverseitige Voraussetzungen sind neben dem für CGI konfigurierten Webserver und dem installierten Ruby-Interpreter nur noch die korrekte Shebang-Zeile. Nützlich sind dann natürlich wieder die praktischen Here-Documents, die bereits für Ruby vorgestellt wurden und in der Perl-Syntax

```
print <<terminierender_string

...Ausgabe...

terminierender_string
```

verwendet werden. Abbildung 13.24 zeigt ein solches einfaches CGI-Script (Ausführung in Abbildung 13.25).

**Abbildung 13.25:**
Ausführen von `hellorubycgi.rb`

### 13.5.2 Die Klasse CGI für Ruby

Zum Ruby-Standardpaket gehört die Klasse `CGI` im Modul `cgi`, welche zahlreiche nützliche CGI-Funktionalität bereitstellt. Die Syntax ist am CGI-Modul von Perl `cgi.pm` orientiert (vgl. 10.6).

Die Verwendung dieser `CGI`-Klasse in Ruby setzt das Einbinden durch

```
require 'cgi'
```

voraus.

Eine nützliche Klassen-Methode der CGI-Klasse ist

```
escapeHTML(...)
```

welche die korrekte Codierung des Arguments in HTML-Code erzeugt; das Beispiel `escapehtml.rb` (Abbildungen 13.27 und 13.26) zeigt die Anwendung dieser Methode.

**Abbildung 13.26:**
Ausführen von `escapehtml.rb`

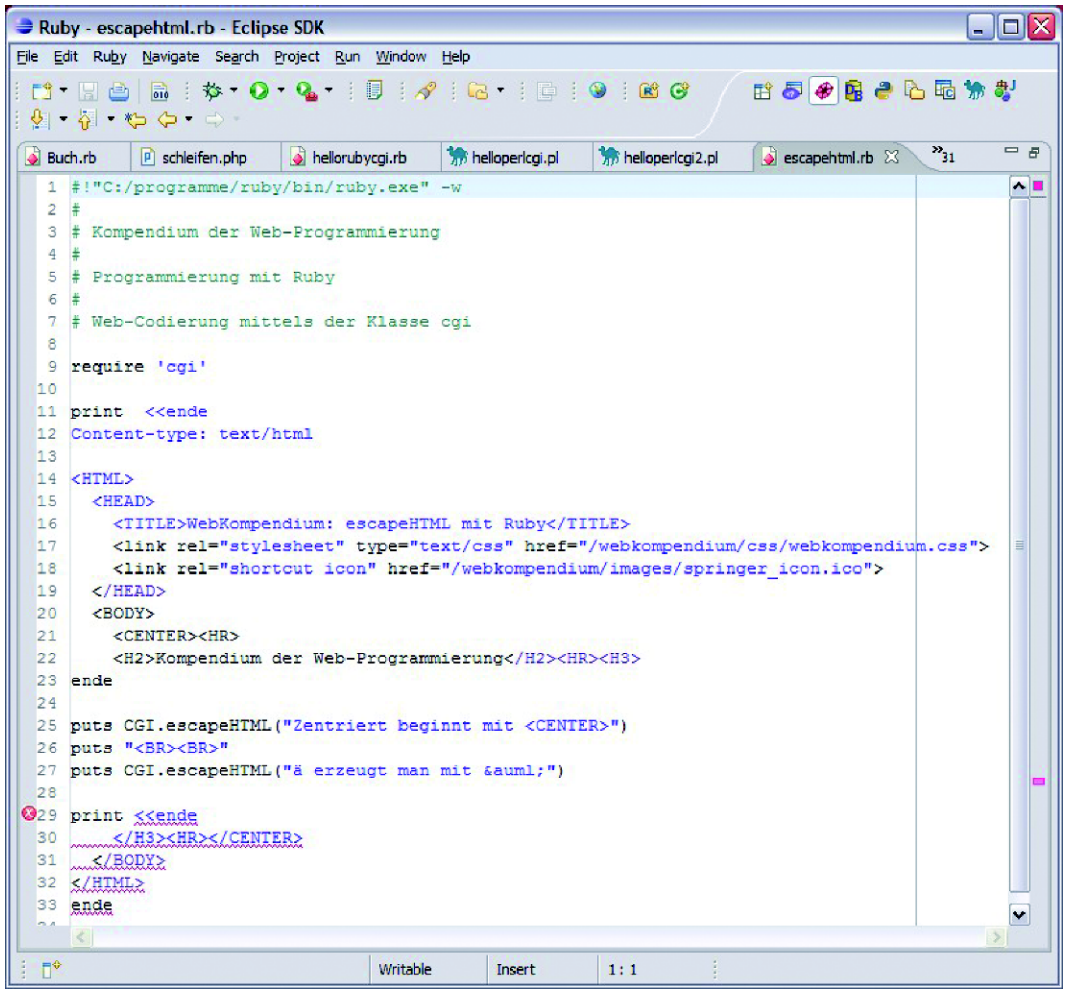



**Abbildung 13.27:** Anwendung der Klasse `CGI`: Methode `escapeHTML` (Script `escapehtml.rb`)

### Formularverarbeitung mit Ruby

Das Verarbeiten von HTML-Formularen in Ruby mittels der Klasse `CGI` ist stark angelehnt an das Vorgehen in Perl mit `CGI.pm`: Es wird ein Hash mit den Key-Value-Paaren des Formulars erzeugt, auf welches dann zugegriffen werden kann. Dies geschieht prinzipiell mit folgender Syntax:

```
require 'cgi'                ### Einbinden der Klasse cgi.CGI
formular = CGI.new           ### Erzeugen der Instanz der Klasse CGI
eingabe = formular.params    ### Erzeugung des Hashs zum uebermittelten Formular
```

Auf den Wert eines Eingabefeldes kann dann über

```
eingabe['key']               ### Value zum Key 'key'
```

zugegriffen werden.

Exemplarisch soll ein einfaches Formular mit mehreren Eingabefeldern (Abbildungen 13.28 und 13.29) an ein Ruby-Script `rubyformular.rb` (Abbildung 13.30) gesendet werden, welches eine HTML-codierte Antwort erzeugt (Abbildung 13.31).

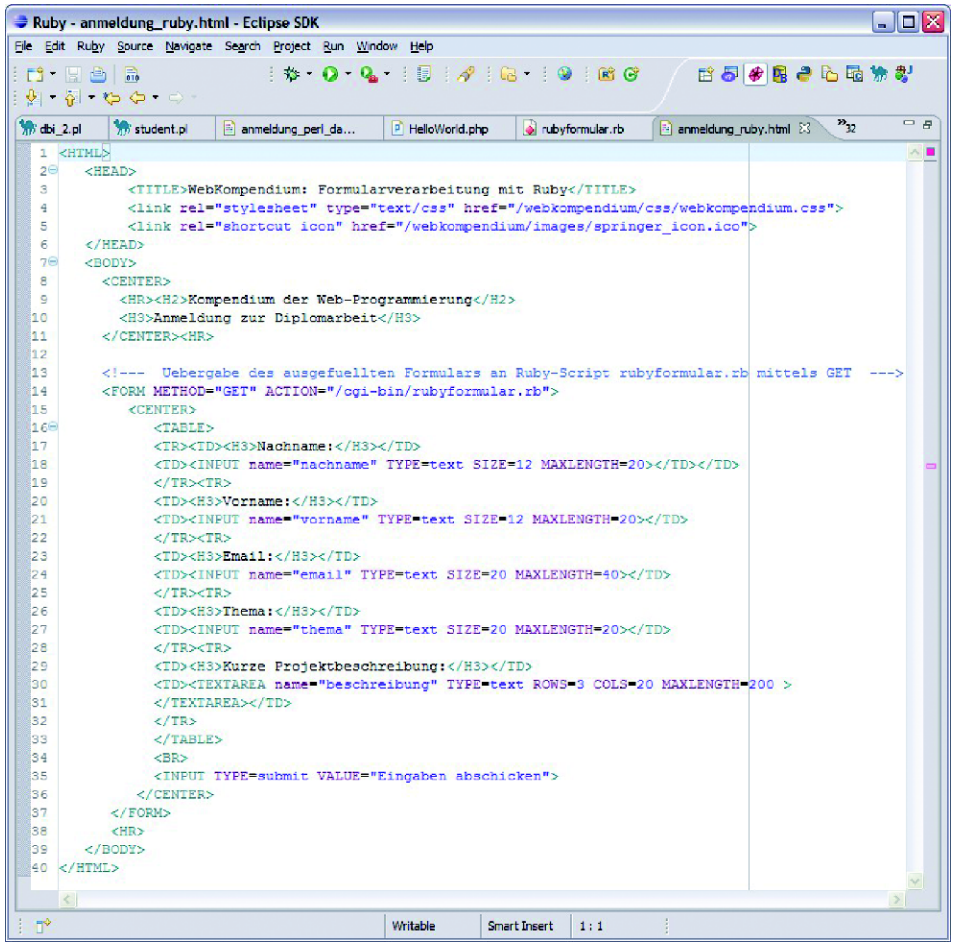



**Abbildung 13.28:**
HTML-Formular (Datei `anmeldung_ruby.html`)

**Abbildung 13.29:**
Das ausgefüllte Formular

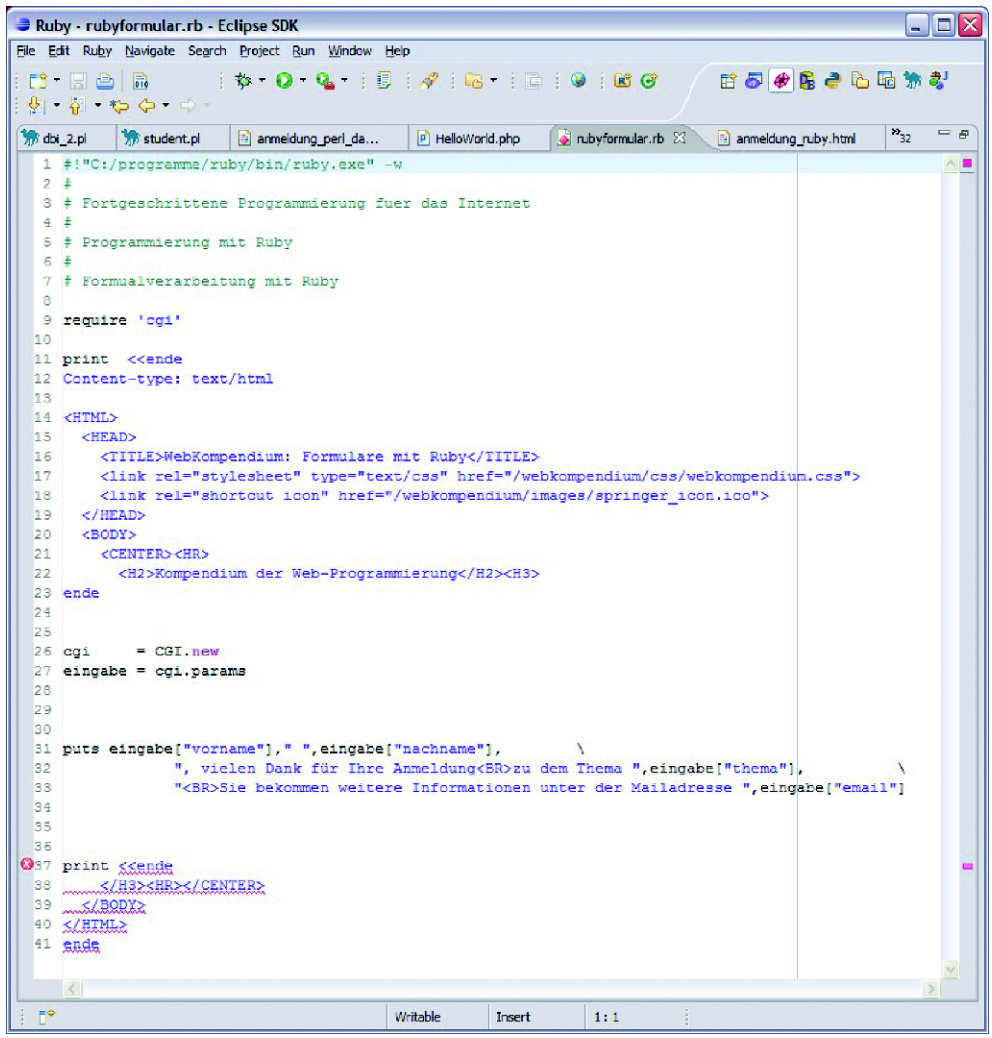



**Abbildung 13.30:** Erzeugen der Antwort durch Formularverarbeitung (Script `rubyformular.rb.r`)

**Abbildung 13.31:** Die konkrete Antwort

### HTML-Code aus Ruby generieren

Eine weitere Parallele zwischen der Ruby-Klasse `CGI` und dem Perl-Modul `CGI.pm` ist die Möglichkeit der Generierung von HTML-Code. Hierfür stellt

die Ruby-Klasse zahlreiche Methoden zur Verfügung. Der Ablauf einer derartigen Anfrage hat folgende Syntax:

```
require 'cgi'
cgi = CGI.new("html4")                    ### Erzeugung einer CGI-Insatz
                                          ### fuer Ausgabe in HTML 4
cgi.out{                                  ### Anfang der Ausgabe
   cgi.html{                              ### Anfang von HTML
         cgi.head{cgi.title{"WebKompendium"} } +
           cgi.body{cgi.hr +

      ...
```

Das Script `autohtml.rb` (Abbildung 13.32) wendet diese Funktionalität beispielhaft an, die Abbildungen 13.33 und 13.34 zeigen das Ergebnis.

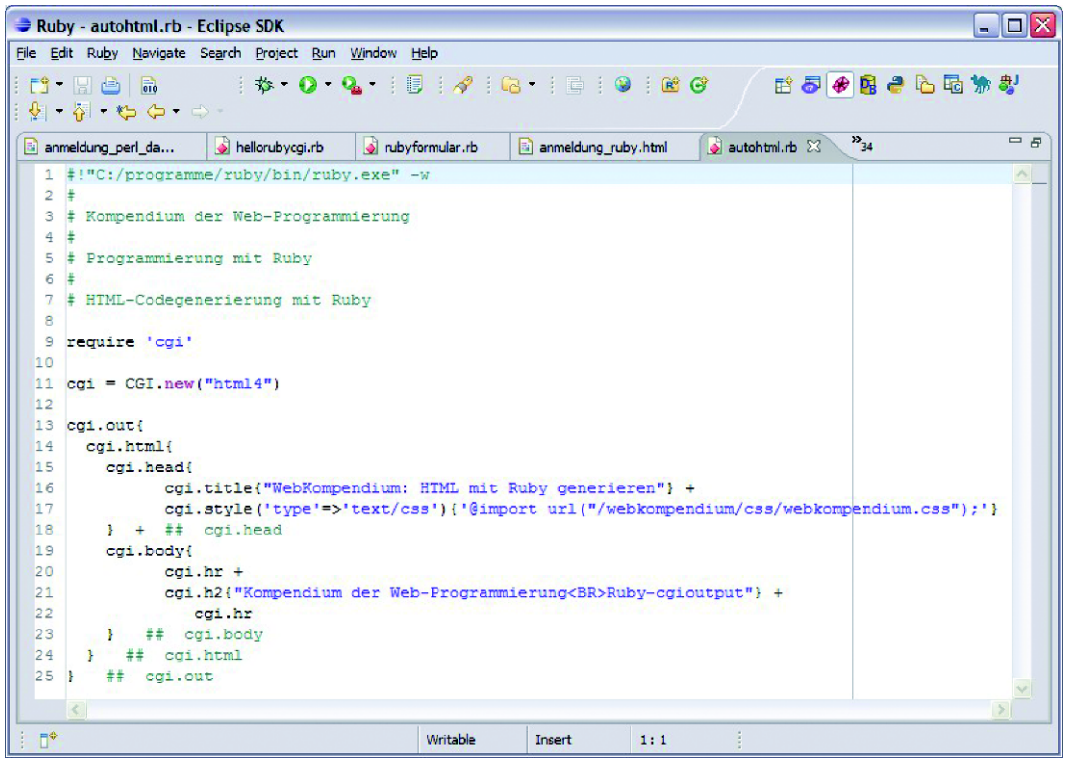


**Abbildung 13.32:** Generierung von HTML-Code durch Ruby (Script `autohtml.rb`)

**Abbildung 13.33:** Ausführen von `autohtml.rb`

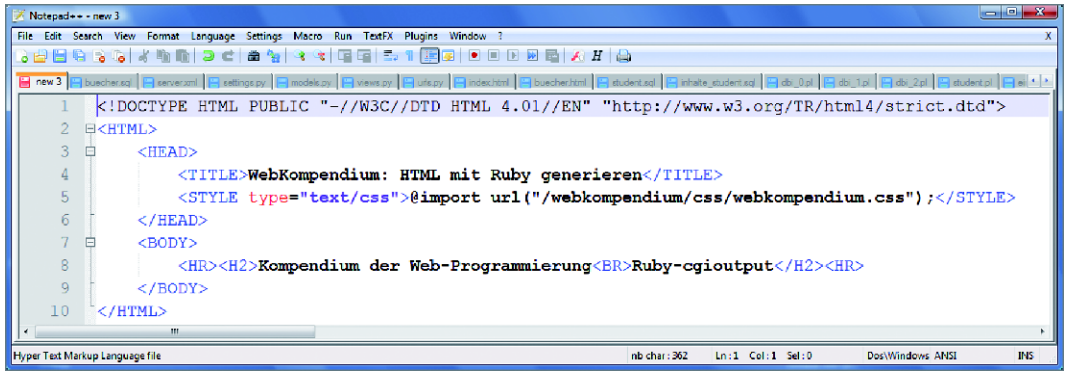



**Abbildung 13.34:**
HTML-Sourcecode der Antwort

### 13.5.3 Der umgekehrte Weg: Ruby in HTML einbinden

Wir haben nun den prinzipiellen Weg gesehen, wie durch Ruby HTML-Code erzeugt wird – der CGI-typische Weg, welcher entweder durch Here-Documents oder durch HTML-Code generierende Funktionen erleichtert werden kann. Dennoch zeigt die Verbreitung etwa von PHP oder von Java Server Pages, dass der umgekehrte Weg von größerem Erfolg ist: nicht in Programmcode wie Ruby oder Perl die HTML-Anteile zu integrieren, sondern umgekehrt in ein HTML-Dokument die aktiven Programmteile einzubinden.

Diesen Erfolg sehen natürlich auch die Entwickler von Ruby, und es gibt inzwischen Ansätze, auch mit Ruby den umgekehrten Weg zu gehen. Den Kern dieser Ansätze bildet jeweils eine Erweiterung des Apache-Webservers um Ruby-Funktionalitäten:

- **mod_ruby** ist ein typisches Apache-Modul für Ruby; es erhöht die Performance wesentlich;

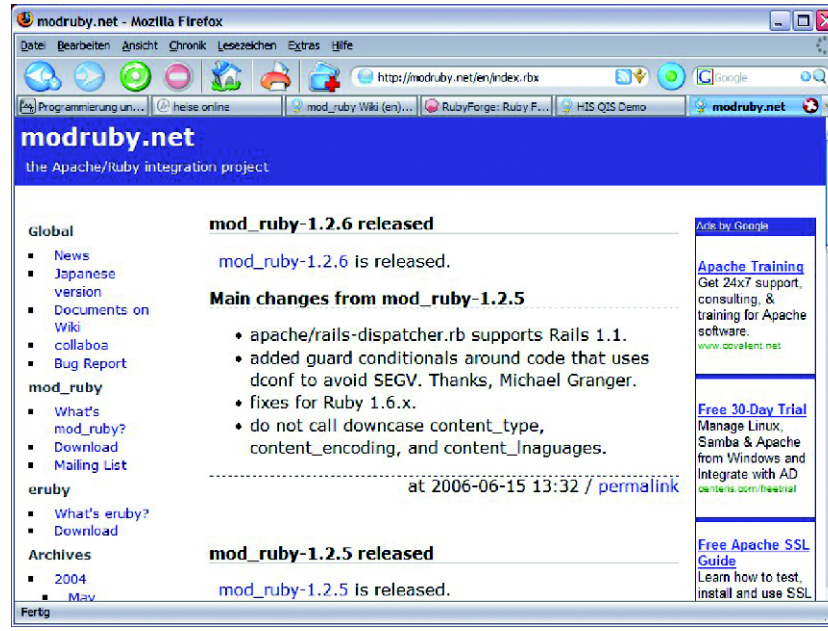



**Abbildung 13.35:**
mod_ruby im Netz

- **eRuby** ist eine auf CGI basierende Erweiterung, welche das Prinzip der Verwendung von Ruby im HTML-Code erlaubt:
  eRuby stellt drei Arten neuer Tags bereit, die sich auf Ruby-Code beziehen:
  – durch das Tag
    `<% Ruby-Code %>`
    wird der angegebene Ruby-Code ausgeführt;
  – durch
    `<%=Ruby-Ausdruck%>`
    wird der Ruby-Ausdruck ausgewertet und durch seinen Wert ersetzt;
  – das Tag
    `<%#Ruby-Code%>`
    wird ignoriert (etwa für Debugging).
  Der Einsatz von eRuby setzt voraus, dass der Apache-Server entsprechend konfiguriert wird; dies bedeutet etwa in der `httpd.conf` die Angabe folgender Direktiven:

```
AddType application/x-httpd-eruby .rhtml
Action application/x-httpd-eruby /cgi-bin/eruby

DirectoryIndex index.html index.shtml index.rhtml
```

**Sowohl mod_ruby als auch eRuby sind, wenn überhaupt, dann nur schwierig auf Windows einsetzbar; hier sollte unbedingt Unix/Linux verwendet werden.**

**Sollen unter Windows komplexere Ruby-Funktionalitäten in einen Apache-Webserver eingebunden werden, kann das Sourceforge-Projekt „rubyforapache“ [web] sehr hilfreich sein.**

Der Einsatz von fastCGI (vgl. Kapitel 19) ist prinzipiell auch für Ruby empfehlenswert.[3]

### 13.5.4 Ein HTTP-Server in Ruby

Zum Ruby-Paket gehört auch eine umfassende Netz-Bibliothek, welche alle gängigen Netz-Protokolle unterstützt. Wesentlich ist hier das Modul `socket`, welches Endpunkte von Netzwerkverbindungen verwaltet (Abbildung 13.36 zeigt die Struktur dieser Klassen).

Als Anwendung soll ein einfacher HTTP-Server mit Ruby realisiert werden, in Anlehnung an die Server-Implementierungen mit Java in Abschnitt 6.2. Hierzu wird in einer Ruby-Anwendung eine Instanz der Klasse `TCPServer` aus den Modul `socket` auf einem bestimmten Port (hier 8086) erzeugt; die Methode `accept` dieser Klasse liefert eine konkrete Verbindung, zu welcher mittels `print` eine Ausgabe erfolgt. Das Ruby-Script `simplehttpd.rb` (Abbildung 13.37) zeigt diese exemplarische Anwendung.

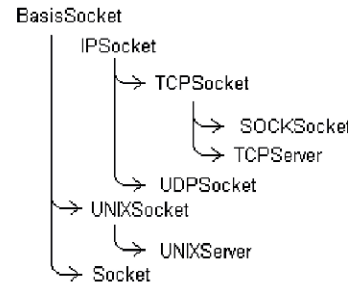



**Abbildung 13.36:**
Struktur der Socket-Klassen von Ruby

[3] insbesondere, weil der momentane Stand des Ruby-Interpreters keine Multi-Threads unterstützt.

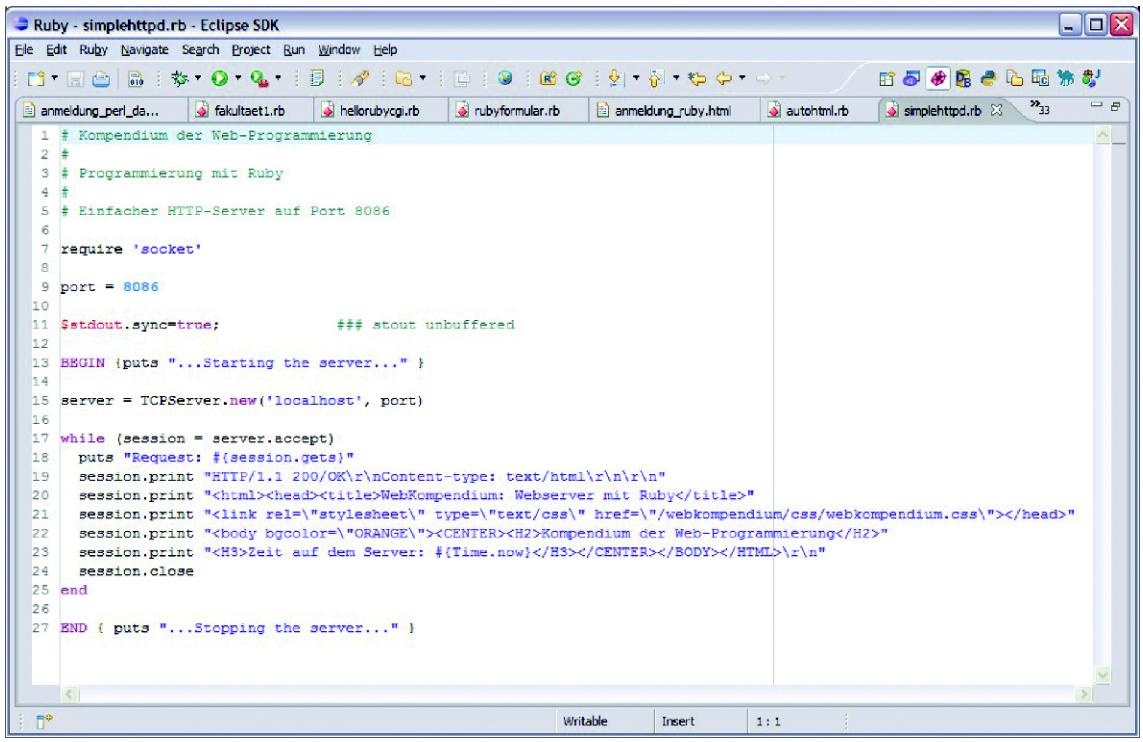



**Abbildung 13.37:**
Einfachster HTTP-Server mit Ruby (Script `simplehttpd.rb`)

### 13.5.5 Unterstützte Protokolle

Das Ruby-Modul `NET` unterstützt zahlreiche Netz-Protokolle (vgl. Abschnitt 1.6); hierzu zählen etwa:

- `NET::FTP` – das FTP-Protokoll;
- `NET::HTTP` – für HTTP;
- `NET::HTTPResponse` – zur Verwaltung der HTTP-Antwort;
- `NET::POP` – für das POP-Protokoll zur Abfrage eines Mail-Servers;
- `NET::SMTP` – für die Übertragung von E-Mails;
- `NET::Telnet` – f+ür das ungesicherte telnet-Protokoll.

### 13.5.6 Ein Webclient in Ruby

Entsprechend zum Perl-Modul `LWP` ist es natürlich auch in Ruby möglich, einen Webclient zu schreiben. Das Ruby-Script `http.rb` leistet dies (Abbildung 13.38), um etwa den HTTP-Header einer Web-Site auszuwerten.

## 13.6 Datenbankzugriff mit Ruby

Ähnlich wie bereits beim CGI-Verfahren orientiert sich Ruby beim *Datenbankzugriff* zunächst wieder stark an Perl und dessen Modul `DBI` (vgl. 10.7). Allerdings haben sich hier wesentliche aktuelle Erweiterungen im Zusammenhang mit dem Rails-Framework (vgl. Kapitel 23) ergeben. Hier werden der allgemeine Weg des Datenbankzugriffs sowie der Weg über das Ruby-Modul `DBI` vorgestellt.

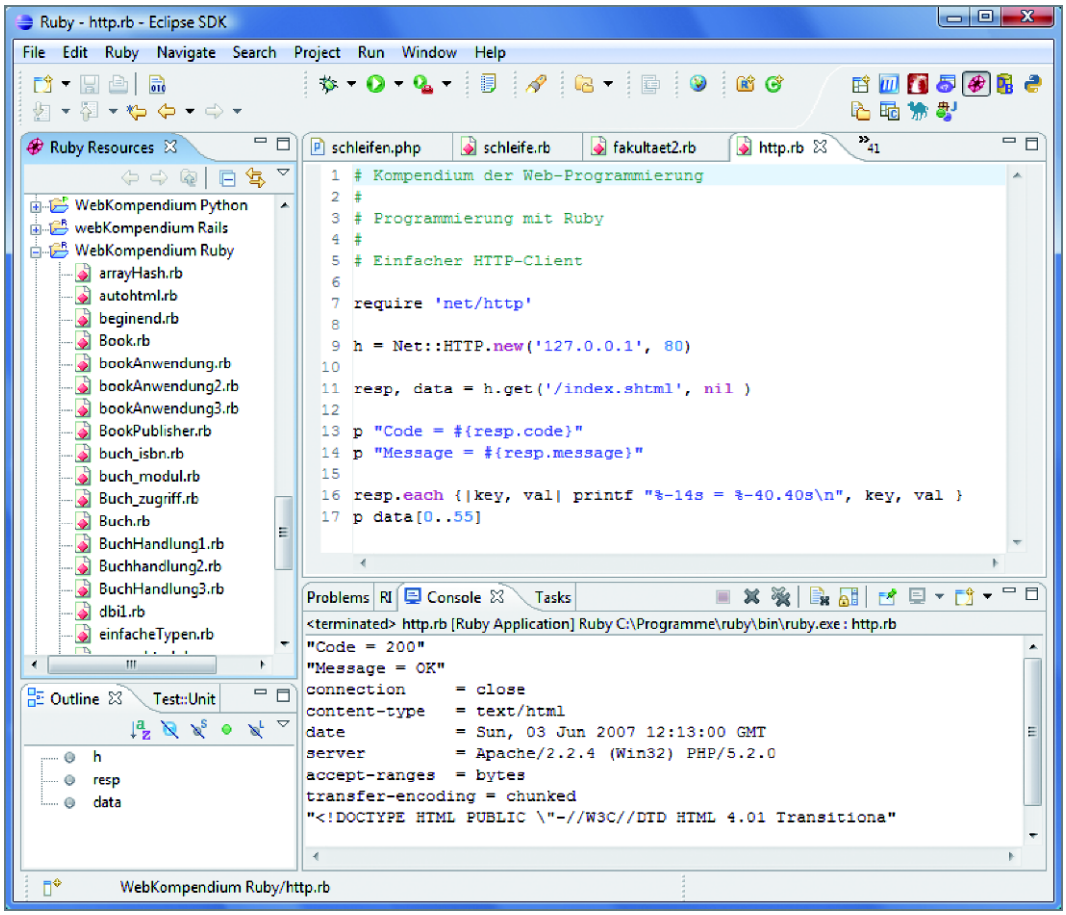



**Abbildung 13.38:**
Einfacher HTTP-Client mit Ruby (Script `http.rb`)

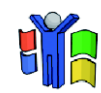
### 13.6.1 Direkter Datenbankzugriff mit Ruby

Zunächst stehen für Ruby für einzelne Datenbanken Schnittstellen zur Verfügung, die dann jeweils eine eigene Syntax verwenden.

Als Beispiel hierfür kann etwa das von Tomita Masahiro entwickelte mysql-Modul dienen [web]. Dieses ist in C geschrieben und wird über den üblichen Weg make und make install installiert. Einfacher geht es auf Unix-/Linux-Systemen über Paketinstaller.

Bei den einzelnen Datenbankpaketen zeigt sich momentan noch deutlich die Herkunft von Ruby, da viele Dokumentationen nur auf japanisch vorliegen.

**Die Ruby-Datenbankmodule sind unter Windows mühsam zu installieren. Üblicherweise wird eine C-Entwicklungsumgebung samt make benötigt.**



### 13.6.2 Das Ruby-Modul DBI

Es gibt weitere Möglichkeiten für den Datenbankzugriff mit Ruby. Da Ruby insgesamt noch eine vergleichsweise junge Sprache ist, deren Verbreitung im englischen Sprachraum zudem erst einsetzt, sind diese Wege aktuell noch nicht so weit entwickelt wie in anderen Sprachen.

Über das Modul `DBI` (Database Independend Interface) [web] stellt Ruby eine Abstraktionsschicht bereit, so dass die Entwicklung von Datenbankapplikationen

unabhängig vom jeweiligen Datenbankmanagementsystem ist. Wie in Perl ist `DBI` auch bei Ruby zumindest momentan kein Standard-Modul, sondern muss zusätzlich installiert werden.

Neben dem Abstraktions-Modul `DBI` werden spezifische Treiber für das jeweilige Datenbankmanagementsystem benötigt. Diese werden wie in Perl als `DBD` bezeichnet (Database Driver). Diese Treiber setzen aber meistens nur auf den proprietären Treibern für die einzelnen Datenbankmanagementsysteme auf, die also unabhängig davon auch zu installieren sind (vgl. Abschnitt 13.6.1).

#### 13.6.2.1 Installation von DBI

DBI wird als tar.gz-Archiv ausgeliefert; nach dem Entpacken wird durch

```
ruby setup.rb config
ruby setup.rb setup
ruby setup.rb install
```

das gesamte Paket installiert, mit allen integrierten Treibern (DBD). Der Parameter `-with` in Verbindung mit der `config`-Anweisung erlaubt hier ein gezielteres Vorgehen, etwa wird duch

```
--with=dbi,dbd_pg
```

nur das DBI-Modul sowie der PostgreSQL-Treiber installiert.

Aktuell sind folgende Treiber im DBI-Paket enthalten:[4]

- ADO für ActiveX Data Objects;
- DB2;
- Frontbase;
- InterBase;
- mSQL;
- MySQL;
- ODBC;
- Oracle und OCI8
- Pg für den PostgreSQL-Treiber;
- Proxy;
- SQLite;
- SQLRelay.

#### 13.6.2.2 Eine DBI-Anwendung

Das Script `dbi.rb` zeigt eine exemplarische *DBI-Anwendung* für eine SELECT-Abfrage (Abbildungen 13.39 und 13.40) über die Literaturtabellen nach Kapitel 7

Nach dem Importieren von DBI (DBD wird nicht explizit geladen) wird eine Verbindung zu einer Datenbank – hier wieder MySQL – aufgebaut, die über den Handle `dbh` verwaltet wird. Dies geschieht durch die statische `connect`-Methode von DBI, welche drei Parameter aufweist: die eigentliche Verbindung mit den Angaben des Treibers, des Datenbankhosts und -ports und dem Namen der Datenbank selbst, sowie der Benutzerkennung für die Anmeldung und das

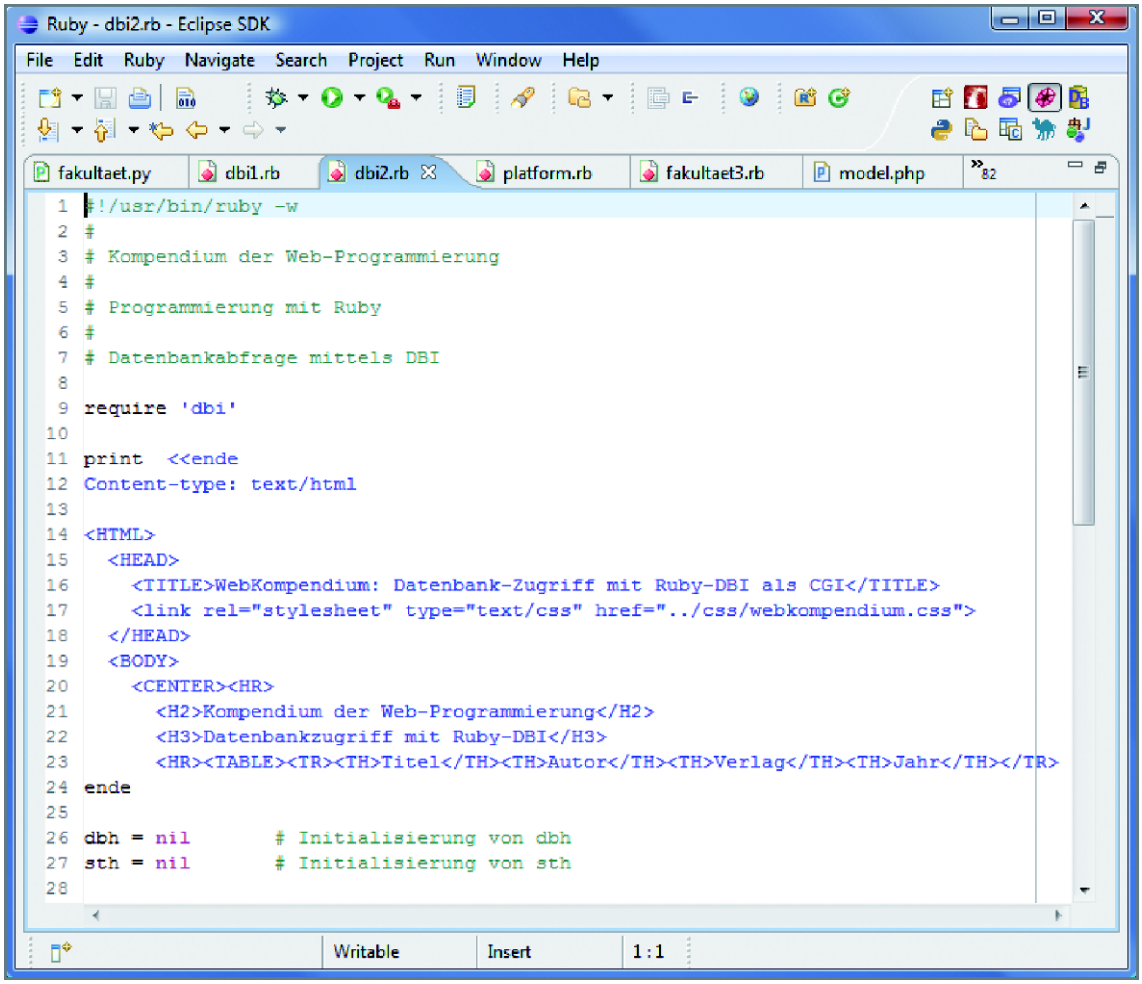
[4] wie schon erwähnt wird in den meisten Fällen zusätzlich der proprietäre Treiber für das einzelne Datenbankmanagemensystem benötigt.

für die Anmeldung verwendete Kennwort. Der hier verwendete MySQL-DBD-Treiber wird aktuell insbesondere von Paul DuBois weiterentwickelt.

Für diese Verbindung wird mittels der `prepare`-Methode ein Statement-Handle für eine spezielle SELECT-Abfrage erzeugt, welche dann mittels `execute` konkret ausgeführt wird.

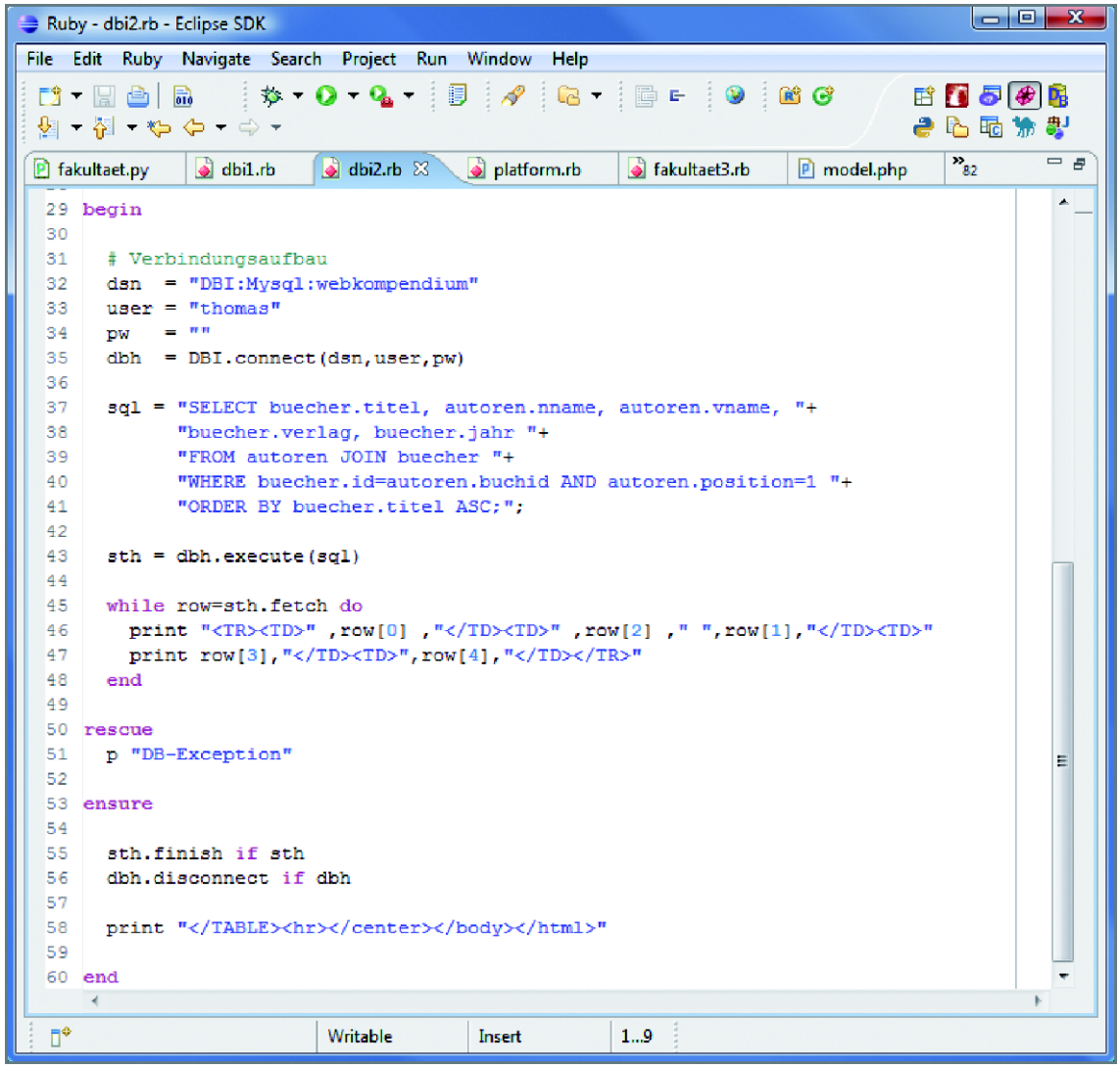
Das somit erhaltene Resultset wird wieder zeilenweise mit der Methode `fetch` verarbeitet.

Mittels `finish` werden das Statement-Handle sowie das Resultset gelöscht; `disconnect` baut die Datenbankverbindung ab.




```
1  #!/usr/bin/ruby -w
2  #
3  # Kompendium der Web-Programmierung
4  #
5  # Programmierung mit Ruby
6  #
7  # Datenbankabfrage mittels DBI
8
9  require 'dbi'
10
11 print  <<ende
12 Content-type: text/html
13
14 <HTML>
15   <HEAD>
16     <TITLE>WebKompendium: Datenbank-Zugriff mit Ruby-DBI als CGI</TITLE>
17     <link rel="stylesheet" type="text/css" href="../css/webkompendium.css">
18   </HEAD>
19   <BODY>
20     <CENTER><HR>
21       <H2>Kompendium der Web-Programmierung</H2>
22       <H3>Datenbankzugriff mit Ruby-DBI</H3>
23       <HR><TABLE><TR><TH>Titel</TH><TH>Autor</TH><TH>Verlag</TH><TH>Jahr</TH></TR>
24 ende
25
26 dbh = nil       # Initialisierung von dbh
27 sth = nil       # Initialisierung von sth
28
```

**Abbildung 13.39:**
Einfache SELECT-Abfrage mit Ruby/DBI (Script `dbi.rb`, Teil I)




```
begin

  # Verbindungsaufbau
  dsn  = "DBI:Mysql:webkompendium"
  user = "thomas"
  pw   = ""
  dbh  = DBI.connect(dsn,user,pw)

  sql = "SELECT buecher.titel, autoren.nname, autoren.vname, "+
        "buecher.verlag, buecher.jahr "+
        "FROM autoren JOIN buecher "+
        "WHERE buecher.id=autoren.buchid AND autoren.position=1 "+
        "ORDER BY buecher.titel ASC;";

  sth = dbh.execute(sql)

  while row=sth.fetch do
    print "<TR><TD>" ,row[0] ,"</TD><TD>" ,row[2] ," ",row[1],"</TD><TD>"
    print row[3],"</TD><TD>",row[4],"</TD></TR>"
  end

rescue
  p "DB-Exception"

ensure

  sth.finish if sth
  dbh.disconnect if dbh

  print "</TABLE><hr></center></body></html>"

end
```

**Abbildung 13.40:**
Einfache SELECT-Abfrage mit Ruby/DBI (Script `dbi.rb`, Teil II)

| Titel | Autor | Verlag | Jahr |
|---|---|---|---|
| Computergrafik | Michael Bender | Hanser | 2005 |
| Das Foucaultsche Pendel | Umberto Eco | Hanser | 1988 |
| Faust: Texte | Johann Wolfgang Goethe | Deutscher Klassiker Verlag | 1994 |
| Grundlagen der Informatik | Thomas Walter | Hanser | 2003 |
| Kompendium der Mediengestaltung | Joachim Boehringer | Springer | 2003 |
| Kompendium der Web-Programmierung | Thomas Walter | Springer | 2007 |
| Mathematik fuer Informatiker | Manfred Brill | Hanser | 2004 |
| MediaFotografie | Thomas Walter | Springer | 2005 |
| Multimedia-Recht | Oliver Merx | Springer | 2002 |

**Abbildung 13.41:**
Ausführen von `dbi.rb`, Teil I)

Die übrigen SQL-Anweisungen (alle außer SELECT) werden mit der DBI-Methode `do` durchgeführt, welche direkt auf den Database-Handle `dbh` angewendet wird; somit entfällt das Statement-Handle und die Methoden `prepare` und `execute`.

Soweit entspricht das DBI-Paket für Ruby in seiner Syntax sehr genau dem entsprechenden Paket von Perl (vgl. 10.7).

**Der Treiber für MySQL auf Unix/Linux ist in der aktuellen Version für manche SQL-Datentypen fehlerhaft; so wird etwa `DEC` nicht unterstützt. Für die Ausgaben von Abbildung 13.41 wurde in den Datenbanktabellen von Kapitel 7 deshalb statt des Typs `DEC` der Typ `INT` verwendet.**

## 13.7 Ein weiterer Ansatz: JRuby

*JRuby* ist ein neueres Projekt, welches das Ziel verfolgt – und faktisch auch erreicht –, einen Ruby-Interpreter vollständig in Java zu implementieren. Damit können in JRuby sowohl die Klassenbibliotheken von Ruby als auch die von Java verwendet werden.

JRuby ist damit ähnlich zur Java-Implementierung von Python, Jython, die in 12.6 vorgestellt wurde.

**Abbildung 13.42:**
JRuby-Logo

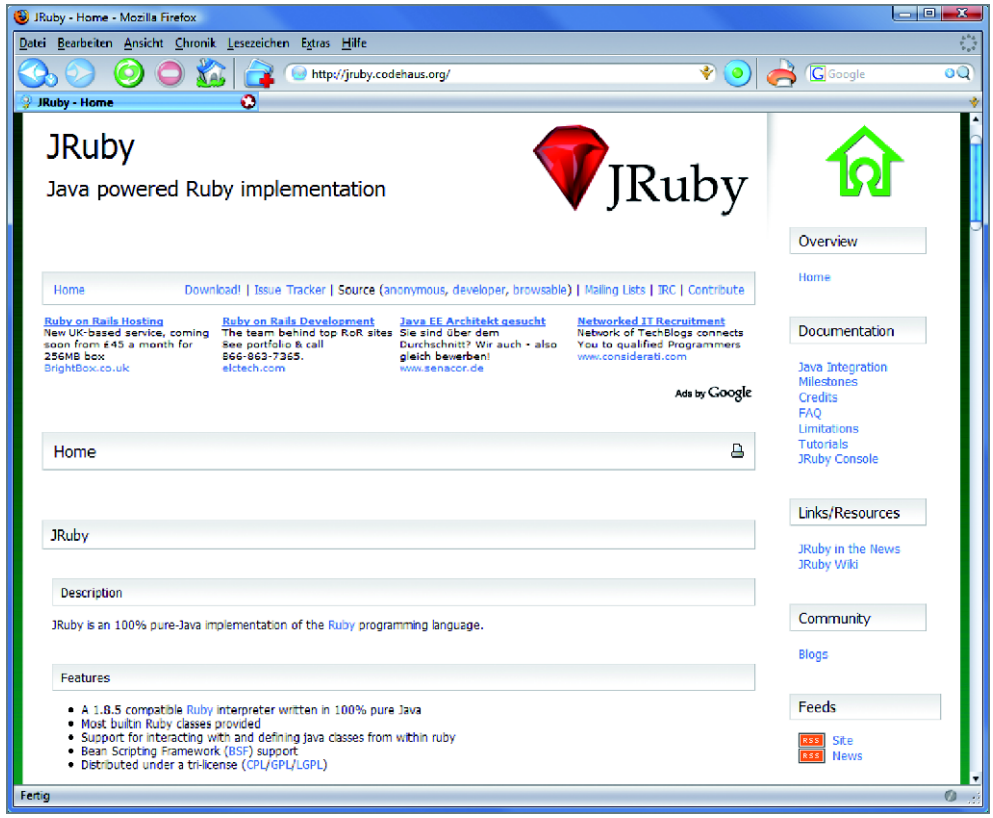



**Abbildung 13.43:**
JRuby im Web

# 14

# Server Side Includes

In diesem Kapitel wird eine einfache, aber auch sehr effiziente und für einige Aufgaben durchaus ausreichende, serverbasierte Technik vorgestellt: Server Side Includes.

## 14.1 Die einfache Alternative: SSI

*Server Side Includes* (SSI) sind eine der einfachsten, aber durchaus bemerkenswerten Techniken, um dynamischen Web-Content zu erzeugen. Die Vorteile von SSI sind:

- Es handelt sich um eine vollständig serverbasierte Technik;
- trotz der beschränkten Möglichkeiten reicht die Technik für vieles aus;
- SSI wird durch ein Apache-Standardmodul (vgl. 6.3.7) ermöglicht;
- Der Apache-Server seit der Version 2 verwendet selbst intensiv SSI für die Darstellung eigener Informationen.

## 14.2 Beispiele: Was kann SSI

Mit SSI können beispielsweise einfach HTML-Dokumente geladen – also zu einem Dokument hinzugefügt – werden; dies ist für den Aufbau von modularen Seiten sehr nützlich, in welchen etwa das Menü und eine abschließende Footer-Datei hinzugefügt wird. Die Pflege eines größeren Webauftritts wird damit wesentlich erleichtert, da nur diese inkludierten Dateien gepflegt werden müssen.

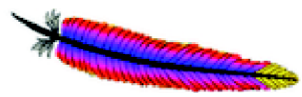


**Abbildung 14.1:** Logo der Apache Software Foundation

## 14.3 Voraussetzungen für SSI und Konfiguration des Apache

Voraussetzung für den Einsatz von SSI ist das Apache-Standardmodul mod_include. Dieses kann statisch oder dynamisch eingebunden werden. Da es ein Apache-Standardmodul ist, beinhaltet die Apache-Dokumentation auch die notwendigen Informationen zu dieser Funktionalität (Abbildung 14.2).

Neben dem Einbinden dieses Moduls muss die Apache-Konfiguration in den entsprechenden Verzeichnissen das Ausführen von SSI erlauben; dies regelt die Direktive `includes`. Dies kann beispielsweise folgendermaßen geschehen:

```
<Directory "d:/www/myssi">
   Options +Includes
</Directory>
```

Hierdurch wird im beschriebenen Verzeichnis SSI erlaubt. Die pauschalere Konfiguration `Options all` beinhaltet auch SSI.

**Abbildung 14.2:** Online-Dokumentation zu mod_include

Ferner müssen die entsprechenden Filter in der Apache-Konfiguration httpd.conf definiert werden:

```
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
```

### Kennzeichnung von SSI-Dateien

SSI-Dateien werden im normalen Verzeichnisbaum für Web-Dokumente (DocumentRoot) abgelegt. Es ist üblich, SSI-Dateien mit dem Suffix `shtml` zu kennzeichnen. Unter Unix/Linux gibt es eine interessante Alternative: Da für Web-Dokuemnte das Dateiattribut „ausführbar“ – das x-bit – nicht benötigt wird, kann dieses zur Kennzeichnung von SSI-Dateien verwendet werden; dies wird durch die Direktive

```
XBitHack on
```

ermöglicht, und es kann die übliche Dateiendung html verwendet werden; unter Windows steht dieser Weg nicht zur Verfügung.

## 14.4 Syntax und Beispiele

Die Grundsyntax von SSI ist denkbar einfach: Eine SSI-Anweisung ist für HTML ein Kommentar; intern besteht die Anweisung aus einem Element, dem mehrere Attribut-Wertepaare folgen können:

```
<!--#element  attribute=value ...  -->
```

Als Beispiel soll hier die Ausgabe der Serverzeit dienen:

```
<!--#echo var="DATE_LOCAL" -->
```

### 14.4.1 Mögliche SSI-Elemente

SSI verfügt über acht *Elemente:*

- `config`: Konfiguration der Ausgabe (Zeitformat, Mengeneinheiten wie MB, etc.;
- `echo`: Ausgabe einer Variablen;
- `exec`: Ausführen eines externen Programms, CGI oder Kommandozeilen-Anwendung;
- `fsize`: Ausgabe der Größe einer Datei;
- `flastmod`: Ausgabe des Datums der letzten Änderung einer Datei;
- `include`: Importieren einer Datei;
- `printenv`: Ausgabe von Umgebungsvariablen;
- `set`: Belegen von Variablen mit einem Wert.

Diese Elemente werden im Folgenden genauer vorgestellt.

| Element | Bedeutung |
|---|---|
| `config` | Konfiguration |
| `echo` | Ausgabe |
| `exec` | Ausführen |
| `fsize` | Größenausgabe |
| `flastmod` | Ausgabe Änderungsdatum |
| `include` | Import |
| `printenv` | Ausgabe Umgebungsvariablen |
| `set` | Wertzuweisung |

**Tabelle 14.1:** SSI-Elemente

#### 14.4.1.1 Das config-Element

Über `config` können insbesondere die Formate für Zeitausgabe (das Attribut `timefmt`) und das Attribut für die Größeneinheit von Dateien (das Attribut `sizefmt`) definiert werden; eine Anwendung lautet:

```
<!--#config timefmt="%d. %b %Y, %H.%mh" -->
Serverzeit <!--#echo var="DATE_LOCAL" -->
```

Dabei bedeutet d: Tag, Y: Jahr b: Monat (Kurzform) H: Stunde, m: Minute.

Für `sizefmt` können `byte` oder `abbrev` verwendet werden; erstere gibt die Größe immer in Byte an, zweitere auch in kB und MB, wobei wie in der technischen Informatik korrekt 1 kB = 1.024 B gilt.

Weitere Möglichkeiten für `config` sind die Definition von `errmsg` und `echomsg`.

#### 14.4.1.2 Das echo-Element

Mittels `echo` kann der Wert einer Variablen ausgegeben werden:

```
<!--# echo var="variable" -->
```

Für spezielle Zeichencodierung steht als zweites Attribut `encoding` zur Verfügung.

#### 14.4.1.3 Das exec-Element

`exec` ermöglicht – ähnlich wie CGI – das Ausführen von externen Programmen aus der SSI-Datei. Als Attribute stehen `cmd` und `CGI` bereit; ersteres führt in einer sh-Shell ein Programm aus, zweiteres führt eine CGI-Anwendung mit den hierfür üblichen Sicherheitsoptionen aus.

#### 14.4.1.4 Die fsize- und flastmod-Elemente

Die Größe einer Datei kann mittels `fsize` und das Änderungsdatum mittels `flastmod` ausgegeben werden, wobei die mittels `config` festgelegten Konfigurationen gelten.
Wichtiges Attribut ist `file`; so gibt

```
<!--#flastmod file="index.html" -->
```

das Änderungsdatum der Datei `index.html` aus.

#### 14.4.1.5 Das include-Element

Das `include`-Element ist von ganz besonderer Bedeutung. Es hat die Attribute `file` und `virtual`:

- `file` importiert eine HTML-Datei an dieser Stelle:

  ```
  <!--#include file="footer.html" -->
  ```

  Dabei sind auch geschachtelte SSI möglich: die includierte Datei ist auch eine SSI-Datei, die dann wieder „ausgeführt“ wird, also beispielsweise selbst ein include enthalten kann.
- `virtual` erlaubt das Importieren einer Web-Ressource über eine URL; hiermit kann auch einfach – und sicher – ein CGI importiert werden:

  ```
  <!--#include virtual="/cgi-bin/test.pl" -->
  ```

#### 14.4.1.6 Das printenv-Element

`printenv` gibt alle Variablen aus; es hat somit eine ähnliche Funktion wie das Standard-Perl-Script `printenv.pl` (vgl. Abbildung 9.2 in Kapitel 9).

#### 14.4.1.7 Das set-Element

Mittels `set` kann einer Variablen ein Wert zugewiesen werden:

```
<!--#set variable="42" -->
```

### 14.4.2 Flusssteuerung mit SSI

SSI erlaubt auch eine einfache Flusssteuerung im Sinne von geschachtelten Wenn-Dann-Entscheidungen; der zwischen der `<!-#if ... ->` und der `<!-#endif ->`-Anweisung befindliche HTML-Code wird nur im „Wahr-Fall“ ausgegeben. Beispielsweise implementiert

```
<!--#if expr="$HTTP_USER_AGENT=/MSIE/" -->
    Vorsicht: Sie benutzen InternetExplorer
<!--#endif -->
```

eine Warnung, falls in der CGI-Variablen `$HTTP_USER_AGENT` die Zeichenkette „MSIE“ enthalten ist (Abfrage über den regulären Ausdruck).

Es sind auch komplexere Verzweigungen mit else-if und else möglich:

```
<!--#elif expr="test_condition" -->
    ...
<!--#else -->
    ...
<!--#endif -->
```

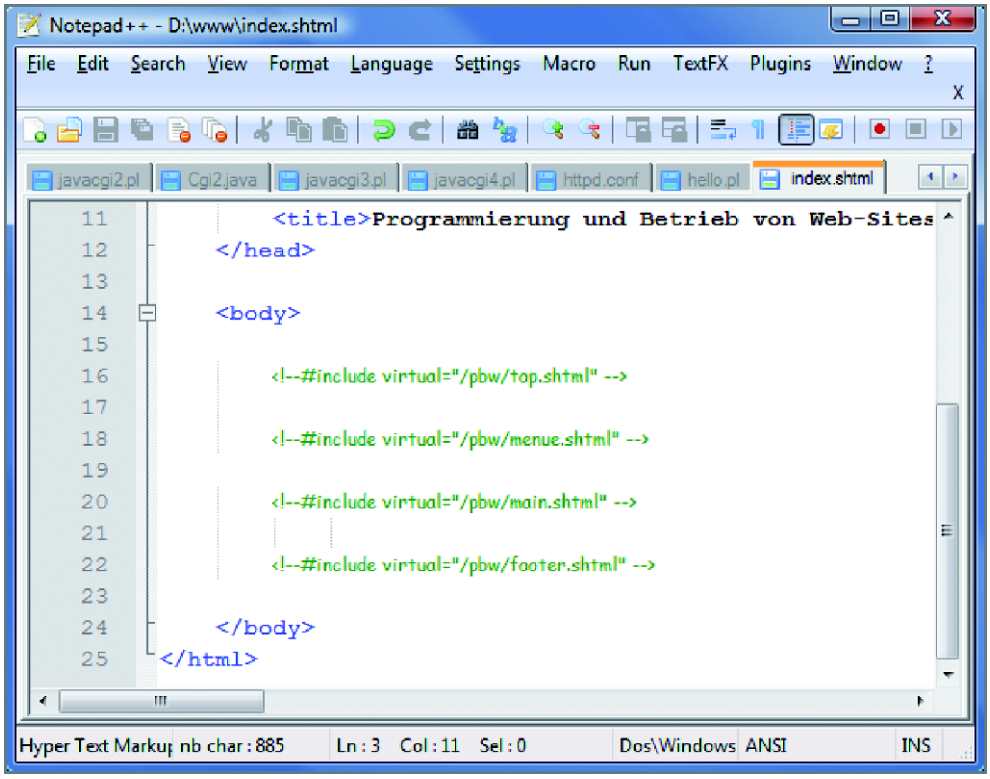



**Abbildung 14.3:**
index.shtml – die eigenliche Seite

## 14.5 Beispiele

Ein Beispiel soll den Einsatz von SSI an einer realen Site verdeutlichen; zum einen werden modular zentrale Dateien geladen, zum anderen erfolgt eine Ausgabe von Variablen, das Einbinden eines Perl-Scriptes für die Zufallsauswahl von Bildern und ein Abfangen im Falle eines speziellen Browsers.

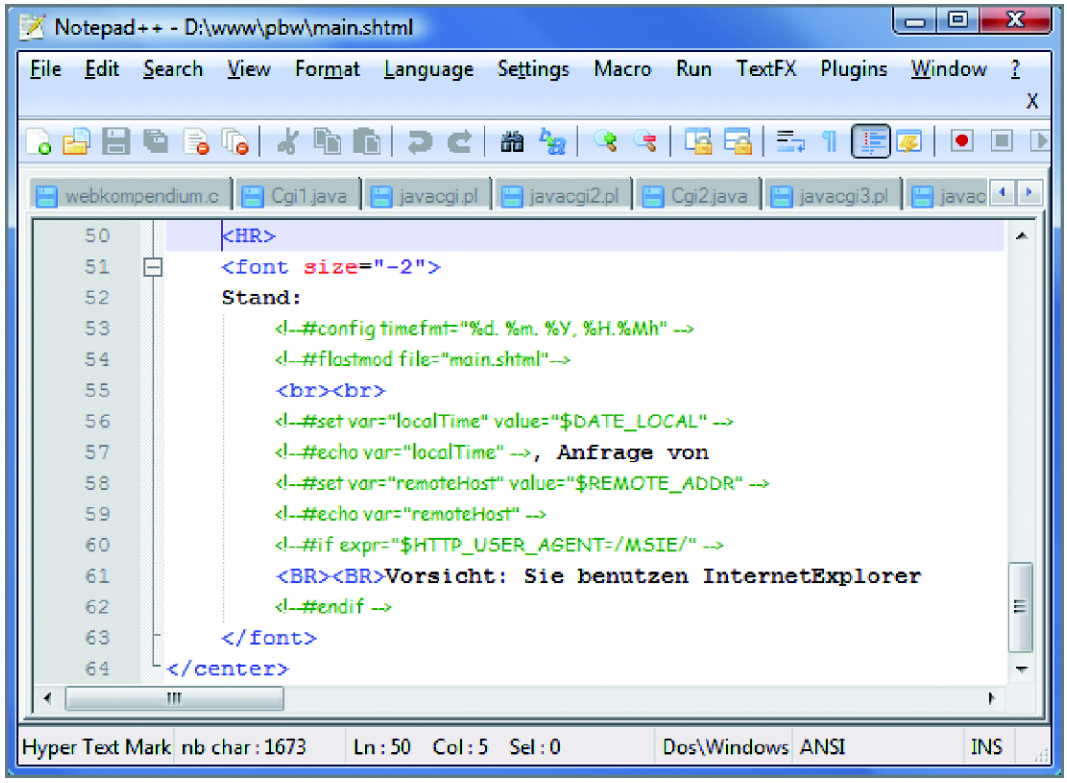



**Abbildung 14.4:** Das Hauptelement mit mehreren SSI-Anweisungen

### Modularisierung

Die zentrale Datei – index.shtml – lädt im wesentlichen nur weitere HTML- und SSI-Dateien hinzu (Abbildung 14.3).

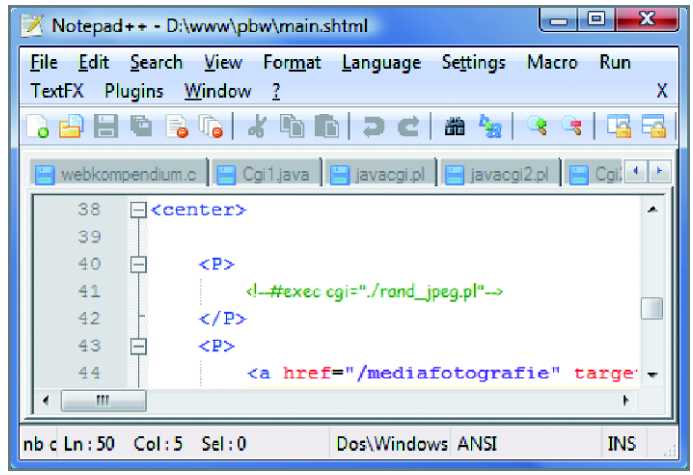



**Abbildung 14.5:** Starten des Perl-Scriptes für Zufallsbilder

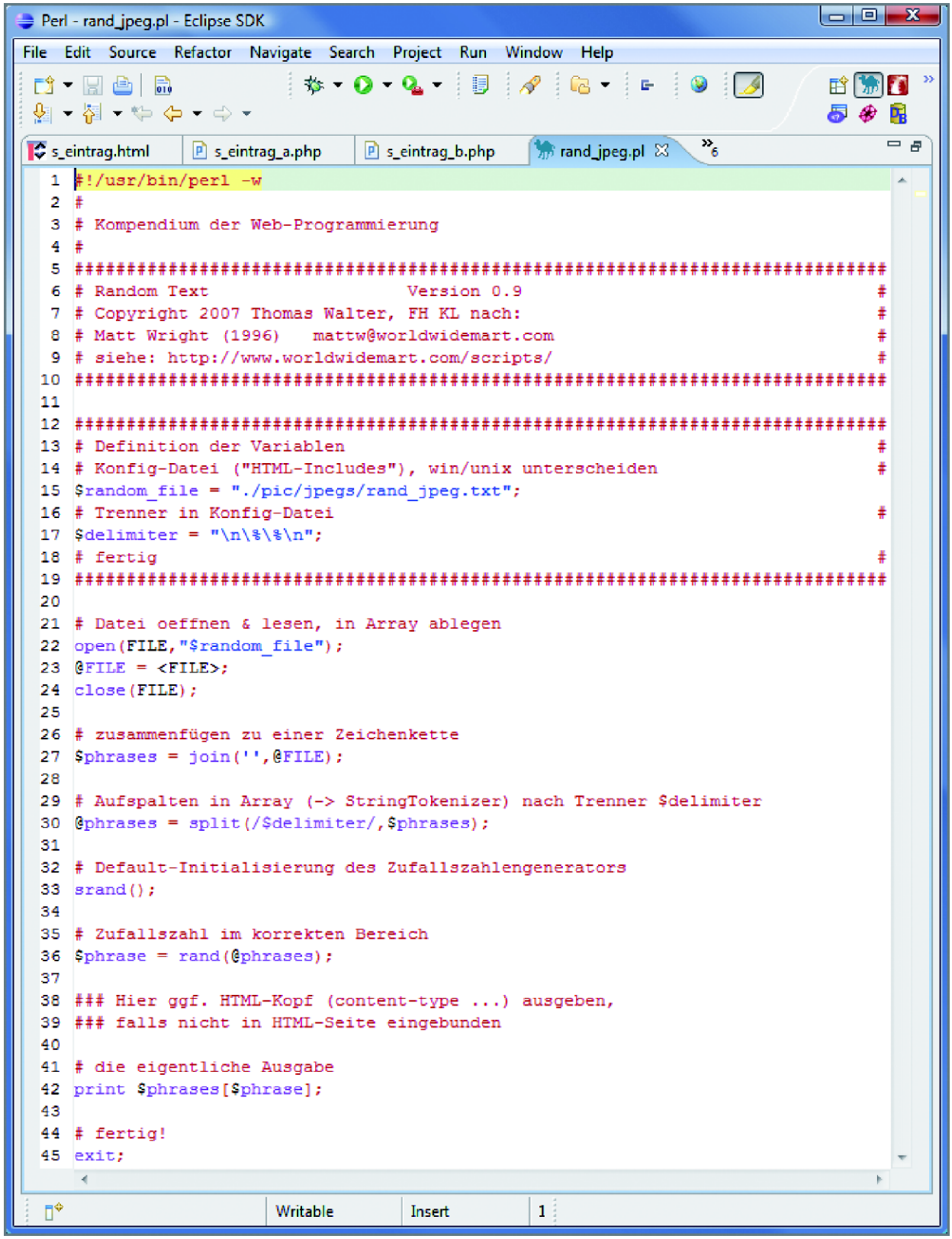



**Abbildung 14.6:**
Perl-Script `rand_jpeg.pl`

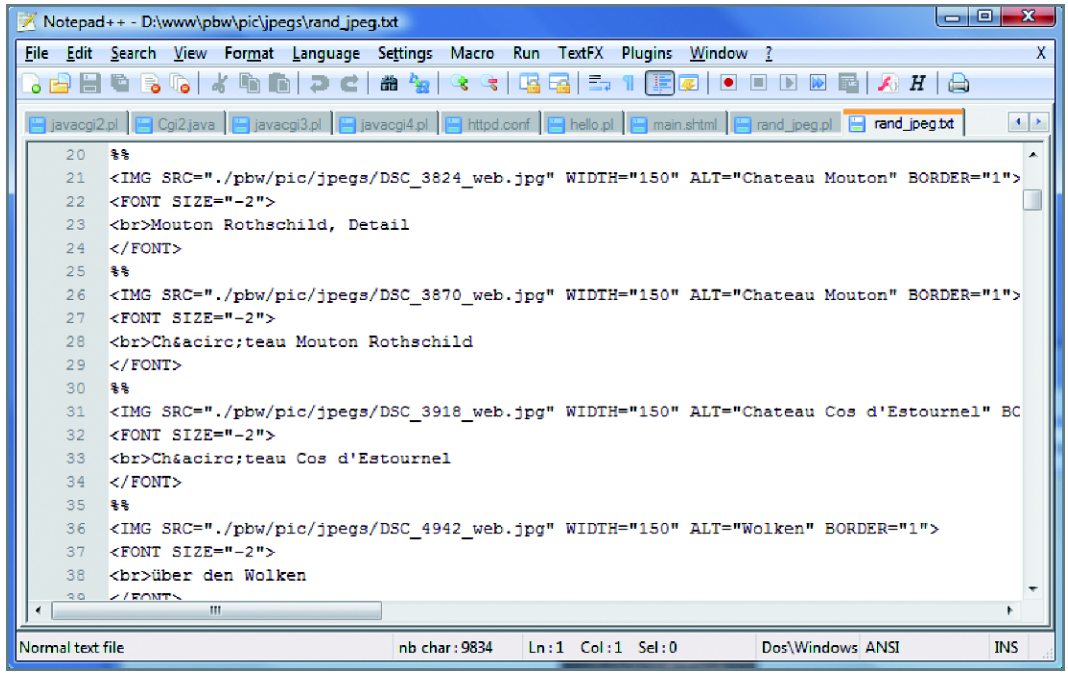



**Abbildung 14.7:**
Konfigurationsdatei der Zufallsbilder

**Abbildung 14.8:**
Das Endergebnis

# Teil III

# Clientseitige Programmierung

# 15

# JavaScript

Die Programmierung für das Internet spielt sich stark – das wird in diesem Buch sehr deutlich – auf der Serverseite ab, aber auch die clientseitige Programmierung ist für viele Anwendungsfälle unerlässlich. Dabei ist der „Klassiker“ bis heute JavaScript. In diesem Kapitel wird JavaScript deshalb genauer vorgestellt.

## 15.1 Dynamisches HTML: DHTML

Ein starkes Schlagwort aus den Anfangsjahren des Webs ist *dynamisches HTML: DHTML*. Von Kritikern auch als „Marktstrategie“ bezeichnet, versteht man unter DHTML allgemein alle Techniken, die es dem Autor einer Web-Site ermöglichen, während der Anzeige im Browser die Seite zu verändern. Diese Änderung kann automatisiert oder durch den Eingriff des Benutzers erfolgen.

Zu den vielen Ansätzen, die zu DHTML zählen, gehören etwa:

- im einfachsten Fall „animated gif“: die sich selbst verändernde Grafik;
- Flash-Animationen (vgl. 17), die heute breit verwendet wird;
- der „Klassiker“ des DHTML ist nach wie vor die Programmierung mittels JavaScript.

Natürlich sind server- und clientseitige Web-Programmierungen genau zu trennen. Beide haben ihre Vor- und Nachteile. Der wesentliche Vorteil der clientseitigen Techniken mit DHTML ist insbesondere die Möglichkeit, direkt auf das Benutzerverhalten zu reagieren: Der Browser weiß beispielsweise, wo sich der Mauszeiger befindet und kann sich entsprechend verhalten. Serverseitig ist dies nicht möglich.

Eine der ältesten, aber nach wie vor die am stärksten verbreitete clientseitige Technik ist JavaScript. Aktuell erlebt dieses durch Ajax (vgl. 16) wieder eine gesteigerte Bedeutung, da Ajax im Kern eine reine JavaScript-Technik ist.

Das klassische Lehrbuch zu JavaScript ist [Fla06], darüber hinaus gibt es ausgesprochen umfassende Literatur zu diesem populären Thema, etwa [Wen07] für eine praktische Kurzeinführung mit Bezug zu Ajax.

## 15.2 Dynamische Webseiten mit JavaScript

JavaScript ist eine der beliebtesten Techniken für „dynamisches“ Web – weniger im Geschäftsumfeld, deutlich aber bei der Gestaltung kleinerer Sites. Dabei

besteht durchaus eine nicht unerhebliche Verwirrung verschiedener Versionen und Dialekte dieser Sprache, da viele sich am Erfolg von JavaScript beteiligen möchten. Die European Computer Manufacturers Association (ECMA) hat unter der Bezeichnung ECMA-262 versucht, einen einheitlichen Standard für diese Scriptsprachen zu definieren [web]. JavaScript selbst erfüllt seit einigen Jahren diesen Standard.

JavaScript läuft vollständig[1] clientseitig, also im Browser ab: JavaScript sind aktive Code-Elemente innerhalb einer Web-Site, die der Browser ausführen kann, um damit „Dynamik“ zu erzeugen wie

- Reaktionen auf Maus-Aktionen wie Rollover-Effekte;
- Überprüfung von Formularen vor deren Versenden an einen Server (Plausibilisierung);
- Steuerung des Browser-Verhaltens bis hin zur Möglichkeit der Konfiguration von Browser-Fenstern.

Hier bietet jeweils JavaScript mehr als reines (X)HTML.

In die Kritik kam JavaScript in der Vergangenheit wegen zahlreicher Sicherheitslücken.

### 15.2.1 Entwicklung von JavaScript

*JavaScript* kam keineswegs mit diesem Namen in die Welt der Web-Programmierung: Die von Brendan Eich entwickelte Sprache wurde von Netscape zunächst als LiveScript vorgestellt. Wichtig ist hier zunächst der formale Zusammenhang zur Programmiersprache Java:

**JavaScript hat außer einige formale Ähnlichkeiten keinen Bezug zu Java. Netscape hat aus Marketing-Gründen 1996 LiveScript in JavaScript umbenannt. Allerdings bestehen zahlreiche syntaktische Parallelen zwischen Java und JavaScript.**

Wie es der Name schon sagt, wird JavaScript-Code nicht compiliert, sondern in einem Interpreter direkt ausgeführt.

1996 wurde JavaScript 1.0 mit dem Browser Netscape 2.0 veröffentlicht, es folgten die Versionen 1.1, 1.2 und 1.3. Die Version 1.4 war die erste ECMA-262-kompatible Version, allerdings ohne nennenswerte Implementierung in einem Browser. Bedeutsam war die Version 1.5 von 2002, welche sehr weit verbreitet ist; aktuell ist die Version 1.7, welche Firefox 2 unterstützt.

### 15.2.2 JavaScript und der Browser

Das Zusammenspiel von JavaScript und dem Browser ist von zentraler Bedeutung, hier kommt der zentrale Unterschied zur serverseitigen Web-Programmierung am stärksten zum Vorschein: Bei der serverseitigen Programmierung hat der Diensteanbieter die vollständige Kontrolle über den Ablauf, er weiß, welche Software in welchen Versionen verwendet wird und überträgt nur simples HTML zum Client. Bei der clientseitigen Web-Programmierung muss die richtige Software in der richtigen Version vorhanden sein – und es muss der Browser-Benutzer die Verwendung der Sprache erlaubt haben, was nicht selbstverständlich ist. Alle Browser haben mehr oder weniger versteckt

[1] es gab einen Versuch, serverseitiges JavaScript zu implementieren, der allerdings ohne Bedeutung geblieben ist.

die Option, die Ausführung aktiver Inhalte wie JavaScript zu erlauben oder zu verbieten.

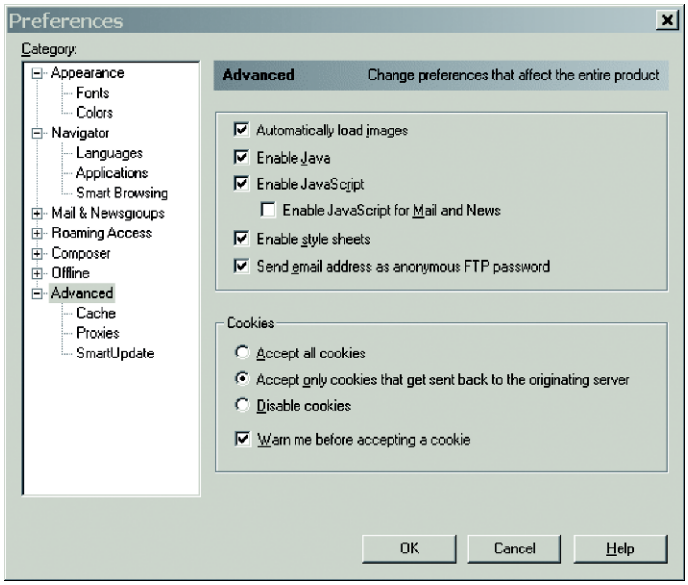



**Abbildung 15.1:**
JavaScript-Steuerung in Netscape 4.7

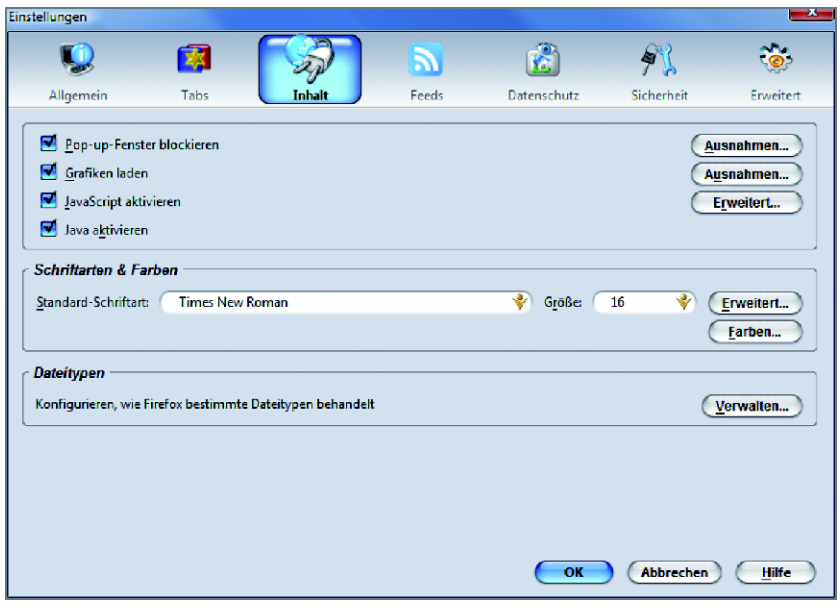



**Abbildung 15.2:**
JavaScript-Steuerung in Firefox 2

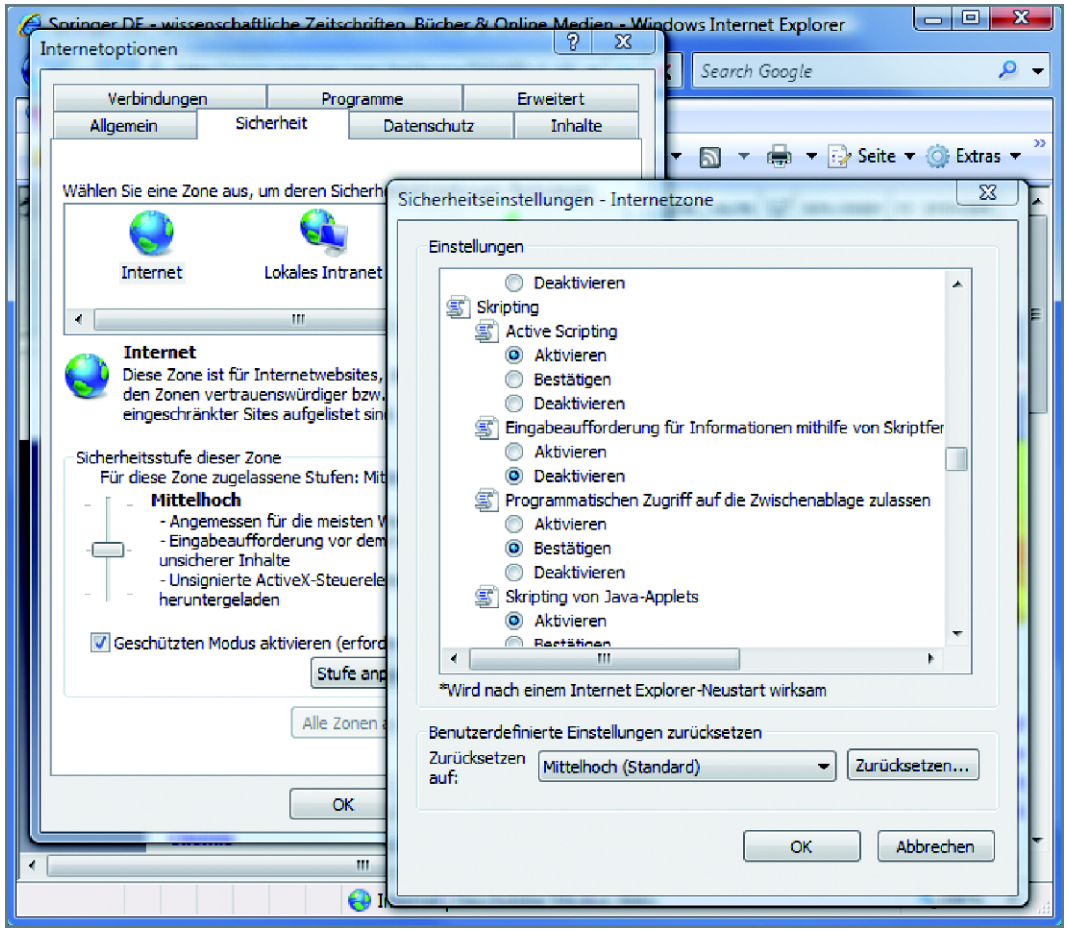



**Abbildung 15.3:** JavaScript-Steuerung im InternetExplorer 7

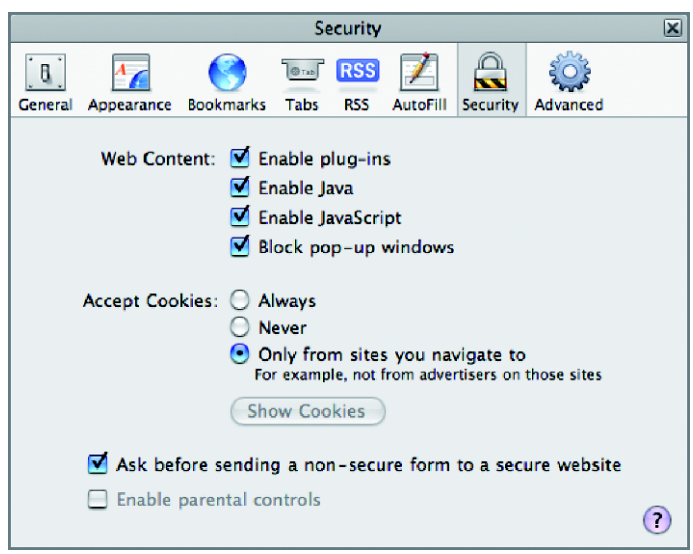



**Abbildung 15.4:** JavaScript-Steuerung in Safari für Windows

Neben dem Problem, dass der Diensteanbieter nicht sicher sein kann, ob JavaScript bei seinem Besucher zugelassen ist, steht noch das Problem, dass nicht sicher gestellt werden kann, ob die benötigte Version vorliegt.

### 15.2.3 Entwicklungsumgebung für JavaScript

Für Eclipse (vgl. 8.1) stehen mehrere gute JavaScript-Plugins zur Verfügung. Beliebt ist das von Adobe/Macromedia angebotene JSEclipse [web], welches auch hier zum Einsatz kommt.

### 15.2.4 Ein einfaches JavaScript

Um ein erstes JavaScript zu erzeugen, wird eine HTML-Datei mit üblicher Endung `.html` innerhalb der Web-Dokumente des Webservers benötigt (im Verzeichnisbaum „htdocs“). Innerhalb dieser Datei wird zunächst mit den Tags

```
<script>
     // JavaScript-Code
</script>
```

der JavaScript-Code begrenzt. Abbildung 15.5 zeigt ein solches einfaches Script, Abbildung 15.6 die Anzeige dieses Scriptes in einem Browser, der JavaScript verwendet.

Bereits das einfache Beispiel zeigt, dass auch in JavaScript die einzelnen Anweisungen mit „;“ abgeschlossen werden; bei einer einzelnen oder bei der letzten Anweisung kann dieses wiederum entfallen.

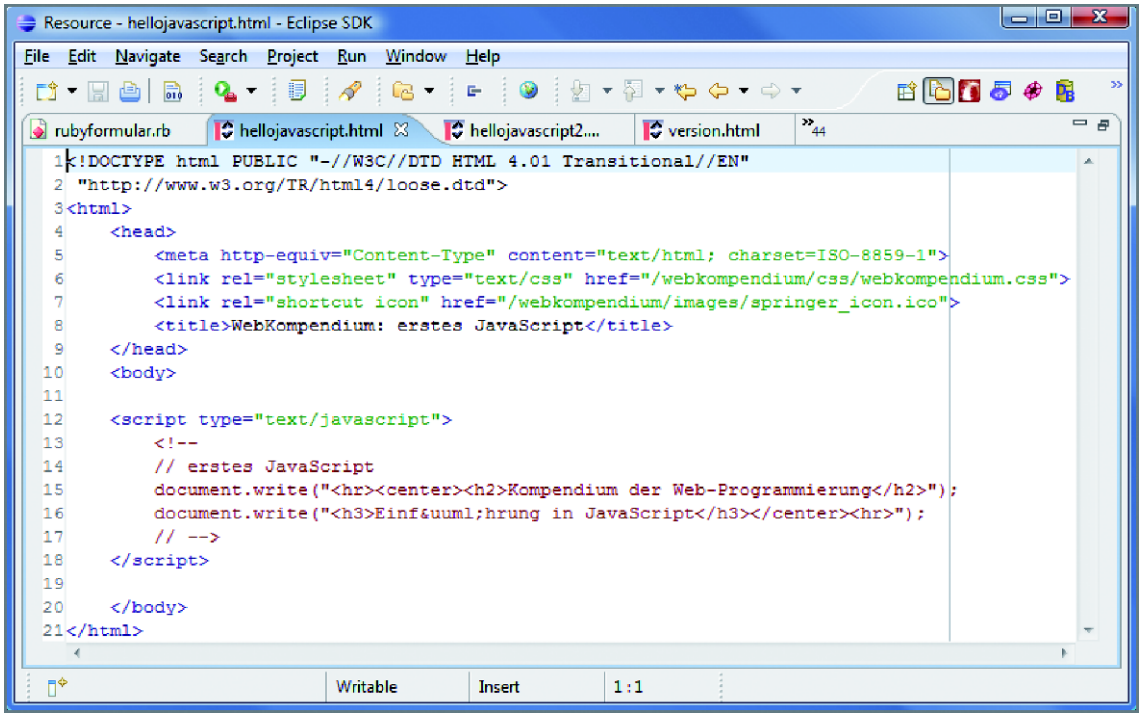



**Abbildung 15.5:** HelloWorld-Anwendung mit JavaScript: `hellojavascript.html`

**Abbildung 15.6:** Aufrufen von `hellojavascript.html`

Das Gegenteil bewirkt das Tag

```
<noscript>
    HTML-Code
</noscript>
```

Hier wird der HTML-Code nur dann angezeigt, wenn JavaScript nicht verfügbar ist (Abbildung 15.7). Dieser Mechanismus ist nützlich, um Anwender auf die Notwendigkeit von JavaScript hinzuweisen, falls dieses nicht genutzt werden kann.

### 15.2.5 Kommentare in JavaScript

Innerhalb eines JavaScript-Blockes können auf unterschiedliche Weise *Kommentare* geschrieben werden:

- der „Mehrzeilenkommentar“ mit der üblichen Syntax

```
/*
   Kommentartext
   ueber mehrere Zeilen
 */
```

- der übliche Zeilenkommentar der Form

```
// Kommentar bis zum Zeilenende
```

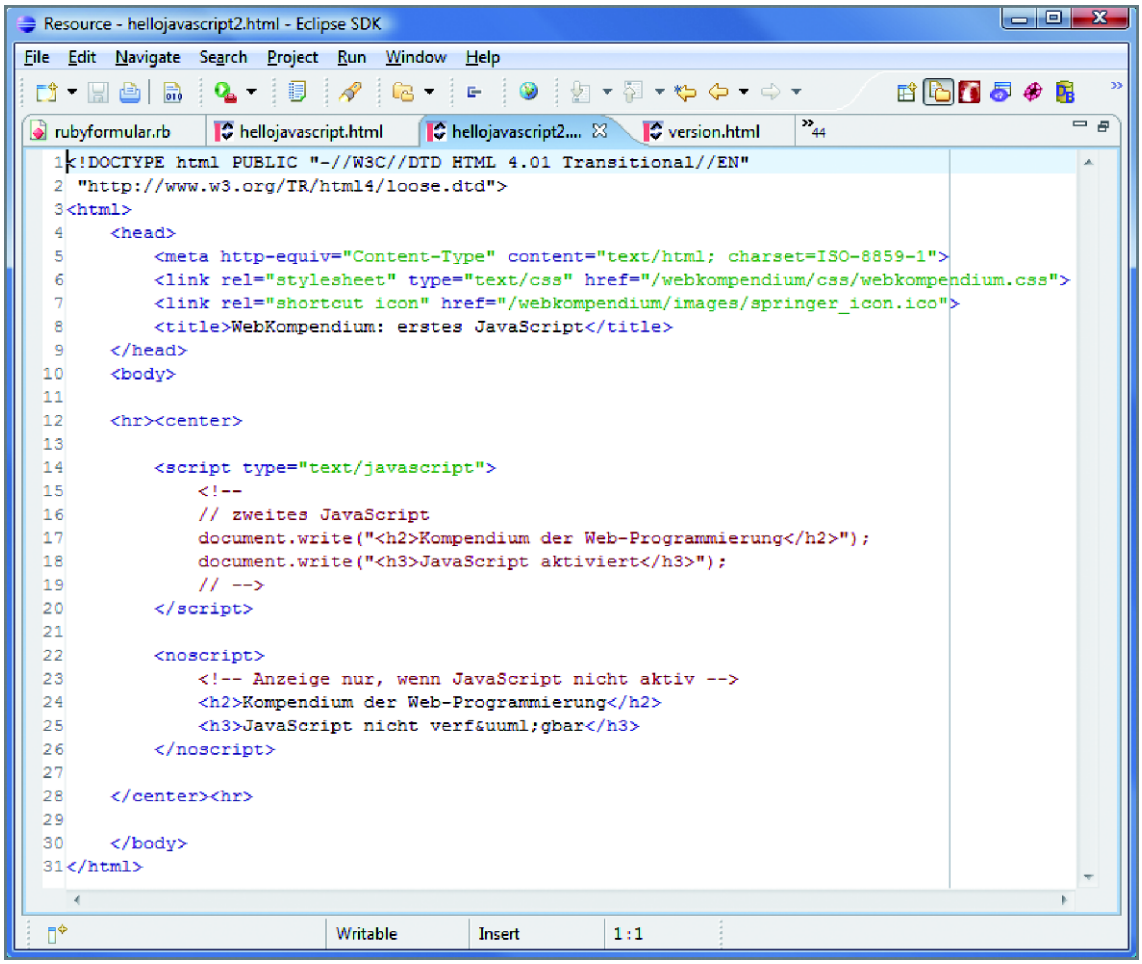



**Abbildung 15.7:**
Seite mit und ohne JavaScript
`hellojavascript2.html`

### 15.2.6 Tags für JavaScript

Das Beispiel aus Abbildung 15.1 verwendet die einfachste Form der *JavaScript-Tags:*

```
<SCRIPT>
   // JavaScript-Code
</SCRIPT>
```

Dies kommt aus der Zeit, als JavaScript „die“ Scriptsprache der Web-Programmierung war. Dies ist heute natürlich wesentlich anders: Passend zu den PHP-Tags (vgl. 11.3.2) bietet JavaScript auch die Version der Tags mit Nennung der Sprache:

```
<SCRIPT language="JavaScript">
   // JavaScript-Code
</SCRIPT>
```

Das öffnende Tag kann weiter durch die Angabe des MIME-Typs spezifiziert werden:

```
<script type="text/javascript">
```

Dies ist die seit der Version 4 von HTML vorgesehen Variante der JavaScript-Tags.

### 15.2.7 JavaScript-Version

Es gibt mehrere Möglichkeiten, die Version der verwendeten Javascript-Programmierung vorzugeben, so dass der Browser den Code nur dann ausführt, wenn er mindestens über diese Version verfügt. Leider verhalten sich hier die aktuellen Browser wieder uneinheitlich: Opera verarbeitet nur die alte Form des Tags

```
<script language="JavaScript1.5">
```

während der InternetExplorer und Safari die HTML4-konforme Variante

```
<script type="text/javascript1.5">
```

unterstützen; Firefox und der InternetExplorer arbeitet in beiden Fällen.

Das Script `version.html` zeigt eine solche Versions-Überprüfung (Abbildung 15.9).

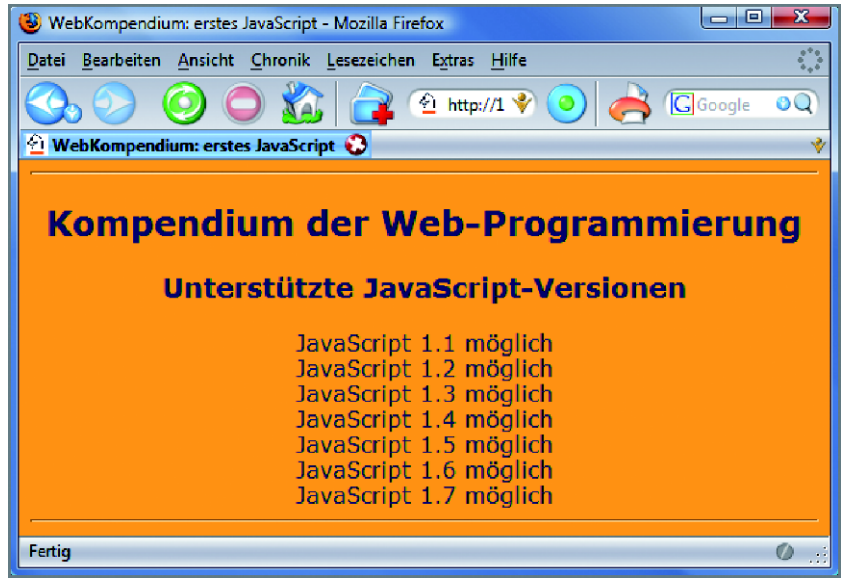
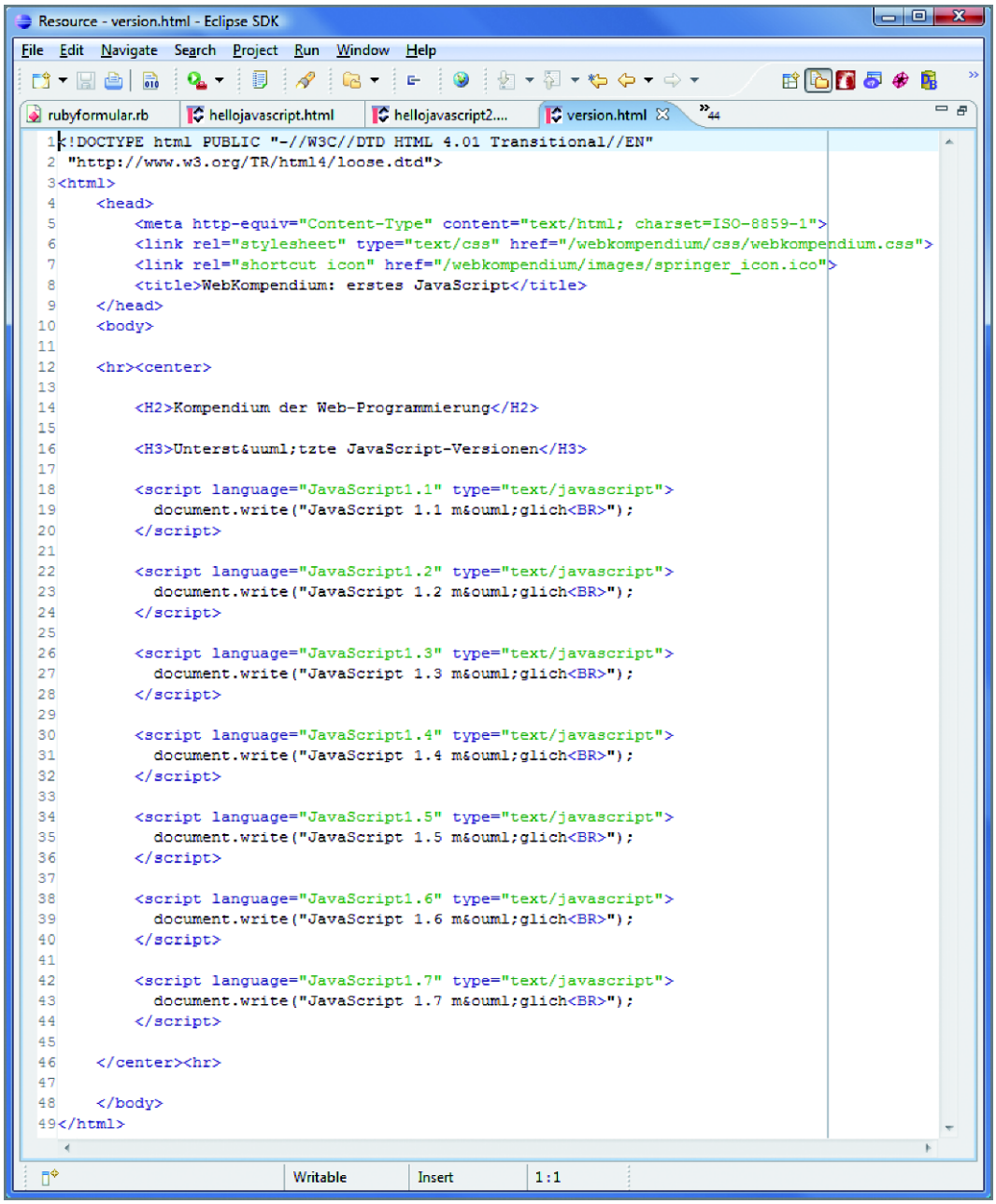



**Abbildung 15.8:**
Firefox 2 unterstützt JavaScript 1.7




```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
        <title>WebKompendium: erstes JavaScript</title>
    </head>
    <body>

    <hr><center>

        <H2>Kompendium der Web-Programmierung</H2>

        <H3>Unterst&uuml;tzte JavaScript-Versionen</H3>

        <script language="JavaScript1.1" type="text/javascript">
          document.write("JavaScript 1.1 m&ouml;glich<BR>");
        </script>

        <script language="JavaScript1.2" type="text/javascript">
          document.write("JavaScript 1.2 m&ouml;glich<BR>");
        </script>

        <script language="JavaScript1.3" type="text/javascript">
          document.write("JavaScript 1.3 m&ouml;glich<BR>");
        </script>

        <script language="JavaScript1.4" type="text/javascript">
          document.write("JavaScript 1.4 m&ouml;glich<BR>");
        </script>

        <script language="JavaScript1.5" type="text/javascript">
          document.write("JavaScript 1.5 m&ouml;glich<BR>");
        </script>

        <script language="JavaScript1.6" type="text/javascript">
          document.write("JavaScript 1.6 m&ouml;glich<BR>");
        </script>

        <script language="JavaScript1.7" type="text/javascript">
          document.write("JavaScript 1.7 m&ouml;glich<BR>");
        </script>

    </center><hr>

    </body>
</html>
```

**Abbildung 15.9:**
Versions-Abprüfung von JavaScript `version.html`

### 15.2.8 Fehlersuche bei JavaScript

Die *Fehlersuche* in JavaScript-Sites gestaltet sich abweichend vom üblichen Verfahren, da die heute verbreiteten Browser ausgesprochen tolerant sind. Als Beispiel wird im Script `version.html` in Abbildung 15.8 ein Syntaxfehler in Zeile 43 eingebaut. Als Folge zeigt aber der Browser die übrigen Scriptelemente korrekt an, nur das eine fehlt – der Benutzer bemerkt in diesem Fall den Fehler nicht, ebensowenig der Entwickler. Dafür bieten die verbreiteten Browser eine Fehlerkonsole an. Bei Firefox unter Extras|Fehler-Konsole zu finden oder mittels der Adresse `javascript:` direkt zu starten, erhält man ein nützliches Werkzeug wie es in Abbildung 15.10 zu sehen ist.

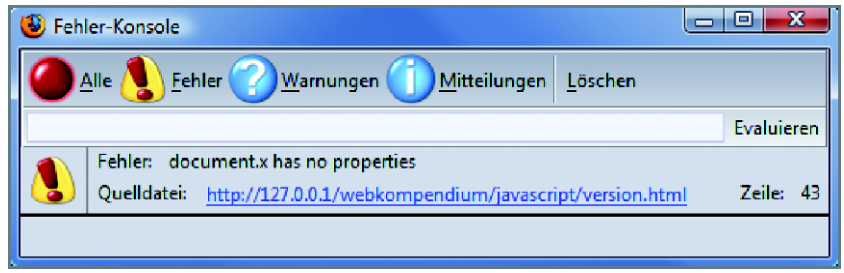



**Abbildung 15.10:**
Fehlerkonsole von Firefox

Eine leistungsfähigere Alternative für die Fehlersuche und -korrektur von JavaScript ist das Firebug-Plugin für den Firefox-Browser (vgl. 8.3).

## 15.3 Grundlegende Syntax von JavaScript

Die erste genauere Auseinandersetzung mit JavaScript muss sich mit der *Grundsyntax* der Sprache auseinandersetzen.

### 15.3.1 Variablen in JavaScript

Bezeichner von *Variablen* in JavaScript beginnen mit einem Buchstaben – üblicherweise in Kleinschreibung – oder mit dem Underscore `_`. Die Bezeichner sind dabei wie üblich case-sensitiv: Groß- und Kleinschreibung wird unterschieden.

Das Schlüsselwort `var` kann – muss aber nicht – für die erstmalige Verwendung einer Variablen genutzt werden, um mehr Übersichtlichkeit zu erreichen:

```
var sinn_des_lebens = 42;
```

JavaScript ist als Scriptsprache nicht-typisiert: Variablen werden nicht deklariert und haben keinen Typ, eine Typumwandlung erfolgt automatisch.

Für numerische Typen stehen die üblichen Operatoren `+`, `-`, `*`, `/` und `%` (mod) bereit.

Numerische Literale in JavaScript verwenden die übliche Notation des vorangestellten `0x` für die hexadezimale und der führenden `0` für die oktale Darstellung.

Die Gültigkeit von Variablen in JavaScript wird durch eine pragmatische Regelung festgelegt:

- Variablen aus dem normalen Haupt-Block haben globale Gültigkeit;
- Variablen, die innerhalb von Methoden definiert werden, sind lokal.

### 15.3.2 Zeichenketten in JavaScript

Konstante Zeichenketten – die String-Literale – werden in JavaScript entweder durch einfache oder durch doppelte Anführungszeichen begrenzt:

```
'Kompendium der Web-Programmierung'
"I don't like mondays"
```

Darüber hinaus gibt es die üblichen Escape-Sequenzen wie `\n` für den Zeilenumbruch.

Zeichenketten in JavaScript sind Instanzen der Klasse `String`; entsprechend können Sie durch direkte Zuweisung oder durch Aufruf eines Konstruktors erzeugt werden. Die beiden folgenden Anweisungen sind damit gleichwertig:

```
var s = "WebKompendium";
var s = new String("Webkompendium");
```

Zeichenketten werden in JavaScript wie in Java durch + zusammengefügt:

```
var wk = "Kompendium der " + "Web-Programmierung";
```

Die Methode `length` gibt die Länge einer Zeichenkette aus, in Fortführung des vorherigen Beispiels:

```
var n = wk.length;   // n bekommt Wert 33
```

Teilstrings werden durch die Methode `substring(m,n)` gebildet, wobei `m` der Index des ersten berücksichtigten und `n` der Index des ersten nicht mehr berücksichtigten Zeichens ist.

```
var s = wk.substring(15,18);   // s bekommt Wert "Web"
```

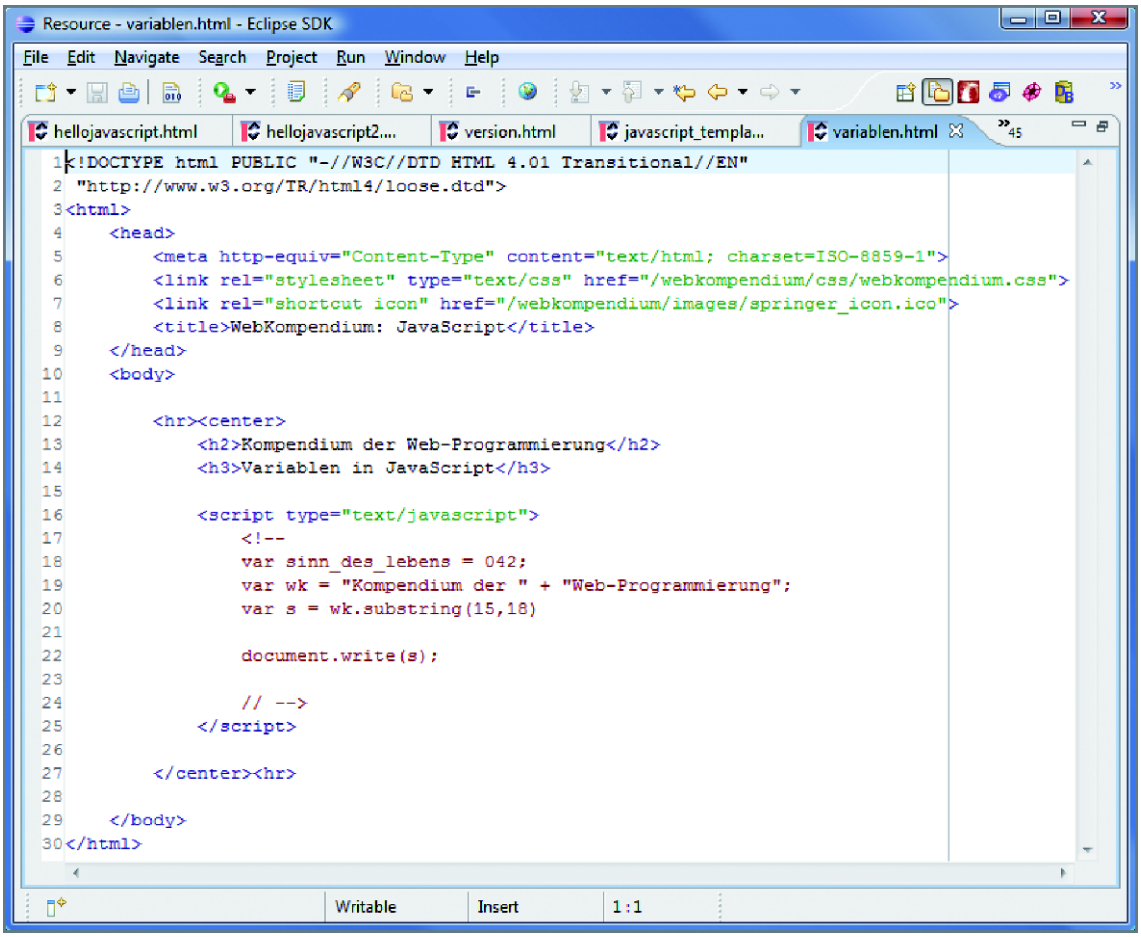



**Abbildung 15.11:** Variablen in JavaScript (Script `variablen.html`)

### 15.3.3 Arrays in JavaScript

Natürlich verfügt JavaScript auch über eine Datenstruktur für eine Liste wie das klassische Array – dies ist aber, ebenso wie in Java, letztlich nur die Instanz einer Klasse `Array`. Dabei kann JavaScript ebenso mit numerisch indizierten Arrays als auch mit assoziativen Arrays arbeiten.

Diese Listen werden über einen Konstruktor erzeugt (vgl. 15.4), welcher ohne Argument für Listen unbestimmter Größe oder mit Vorgabe der Größe für ein klassisches Array verwendet werden kann:

```
var array = new Array(42);    // fest definierte Groesse
var liste = new Array();      // variable Groesse
```

Indizierte Arrays, also solche mit numerischem Index, werden wie üblich verwendet:

```
var array = new Array();
array[0]  = "WebKompendium";
```

Assoziative Arrays arbeiten entsprechend mit beliebigem Index:

```
var array     = new Array();
array["buch"] = "WebKompendium";
```

Für diese Listen stehen zahlreiche nützliche Methoden in JavaScript zur Verfügung, dazu zählen:

- `concat(array1,array2)` fügt die Elemente des zweiten Arrays an das erste an;
- `pop` und `push` sind die üblichen Methoden für lineare Listen zum Entnehmen und Ablegen auf einer Liste, wobei hier `push` beliebig viele Argumente haben kann;
- `shift` bietet wie in Perl die Entnahme vom anderen Ende wie `pop`;
- `sort` sortiert die Elemente eines numerisch indizierten Arrays alphanumerisch.

### 15.3.4 Methoden in JavaScript

Sehr wichtig für die Arbeit mit JavaScript ist die Definition von *Methoden*. Dies geschieht mittels des Schlüsselwortes `function`:

```
// rekursive Fakultaet in JavaScript
function fakultaet(n) {
    if (n < 1) return 1;
    return n * fakultaet(n-1)
}
```

Wichtig ist die Positionierung selbstdefinierter Methoden: Üblicherweise werden diese innerhalb eines JavaScript-Blocks im HTML-HEAD des Dokuments plaziert.

Natürlich verfügt JavaScript über eine sehr große Zahl integrierter Methoden, wovon etliche in diesem Kapitel vorgestellt werden. Hierzu zählen:

- `alert(argument)` erzeugt ein Warnfenster mit der Meldung `argument`;
- `parseInt(argument)` wandelt das Argument in einen Ganzzahl-Wert um;
- `parseFloat(argument)` wandelt das Argument in einen Gleitkomma-Wert um;
- `isNaN(argument)` überprüft, ob das Argument ein numerischer Wert ist („is not a number“).

### 15.3.5 Logik und Operatoren in JavaScript

JavaScript verfügt über das übliche Set von *Operatoren* wie `++` (von links und von rechts anwendbar) sowie Zuweisungsoperatoren wie `+=`.

Da es keine Datentypen gibt, werden Variablen in logischen Ausdrücken wieder nach dem üblichen Schema ausgewertet: numerische Werte `0` und der leere String sind false, alles andere wird als true interpretiert.

Für logische Werte stehen Vergleichsoperatoren wie `==`, `<` und `<=` sowie das „Nicht“ `!` bereit. „und“ wird wieder durch `&&` und „oder“ durch `||` codiert.

### 15.3.6 Kontrollstrukturen in JavaScript

JavaScript verfügt über die üblichen Kontrollstrukturen der modernen Programmiersprachen, die hier in Kürze vorgestellt werden.

**Die Kontrollstrukturen von JavaScript ähneln denen von Java stark.**

#### 15.3.6.1 Sequenzielle Komposition: der Block

Die einfachste Struktur, die Bündelung einzelner Anweisungen zu einem *Block,* wird durch die geschweiften Klammern definiert:

```
{
    a1;
    ...
    an;
}
```

#### 15.3.6.2 Entscheidung

Die übliche „wenn-dann-sonst“-Entscheidung in JavaScript wird wie in Java durch die `if`-Klausel geschrieben:

```
if (boole)
    wahr_anweisung;
else
    falsch_anweisung;
```

Der `else`-Teil ist dabei wieder optional und kann entfallen.

#### 15.3.6.3 Schalter

Die mehrfache Entscheidung in JavaScript wird durch die *Schalterstruktur* `switch` – auch diese mit der aus Java bekannten Syntax – ermöglicht:

```
switch(n) {
     case 42:   alert("42");   break;
     case 313:  alert("313");  break;
     case 4711: alert("4711"); break;
     default:   alert("anderer Wert");
}
```

Die `break`-Anweisung ist notwendig, damit die Kontrollstruktur verlassen wird – sonst werden auch die übrigen Fälle ausgeführt. Der `default`-Fall ist wieder optional.

#### 15.3.6.4 while-Schleife und do-while-Schleife

JavaScript verfügt auch über den „Grundtyp“ der Schleifen, die `while`-Schleife:

```
while (boole)
    anweisung;
```

Neben dieser kopfgesteuerten Schleife gibt es auch die fußgesteuerte Form, welche mindestens einmal durchlaufen wird:

```
do {
    anweisung;
} while (boole);
```

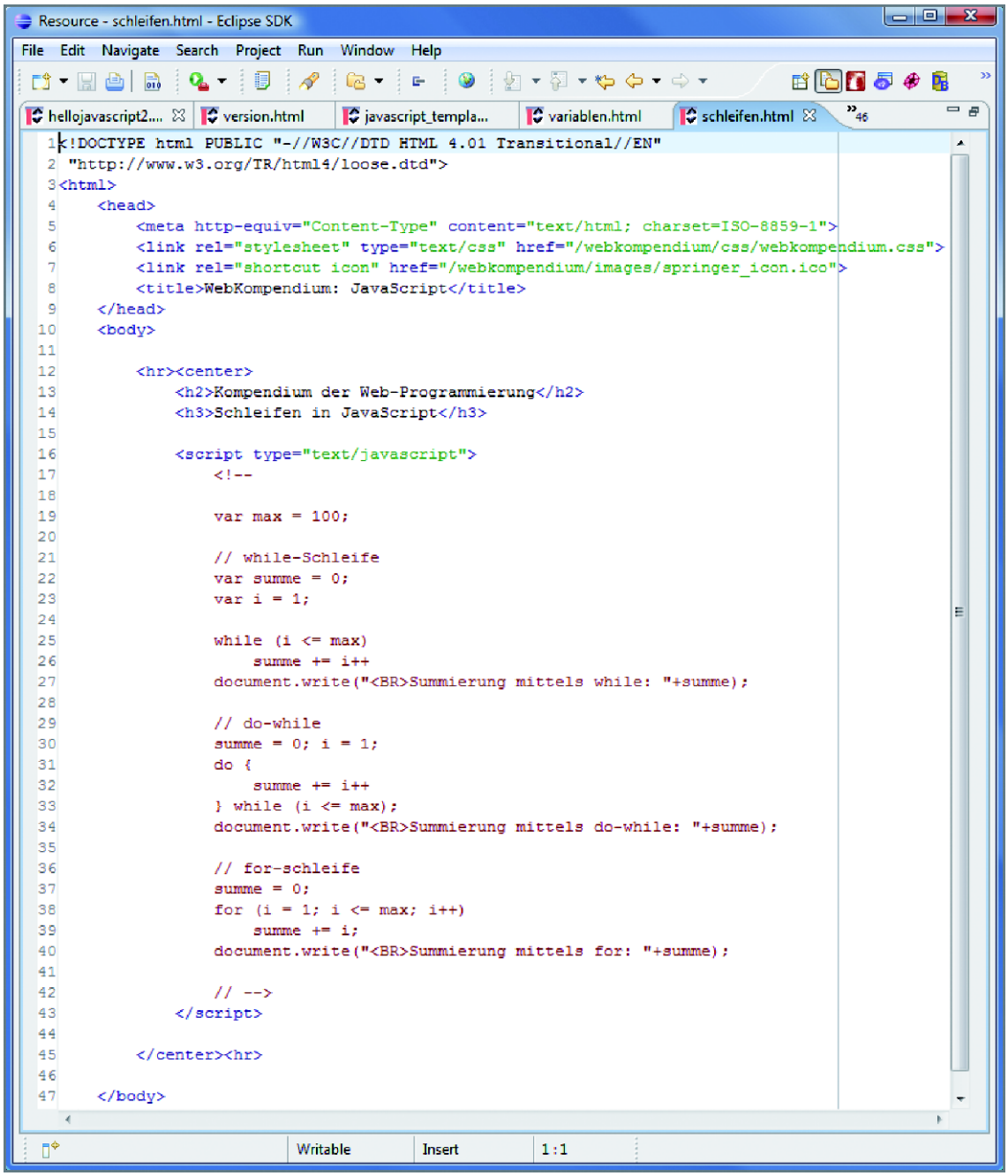



**Abbildung 15.12:**
Die Schleifen-Typen in JavaScript
`schleifen.html`

#### 15.3.6.5 for-Schleife

Die einfache Iteration mittels der `for`-Schleife in JavaScript wird über

```
for (init; boole; incr)
    anweisung;
```

definiert: Die `init`-Anweisung wird einmalig vor dem Schleifendurchlauf ausgeführt, die Schleife wird so lange ausgeführt, wie `boole` als true interpretiert wird, und nach jedem Durchlauf wird `incr` ausgeführt.

#### 15.3.6.6 Beispiel

Das Beispiel aus dem Script `schleifen.html` zeigt alle Schleifentypen zur Berechnung von

$$\sum_{i=1}^{100} i$$

#### 15.3.6.7 Verlassen von Kontrollstrukturen

Mittels der Anweisung `break` werden die Kontrollstrukturen (außer `if`) auch in JavaScript beendet, mittels `continue` wird die Ausführung von Schleifen mit dem nächsten Durchlauf fortgeführt.

#### 15.3.6.8 Iteratoren in JavaScript

JavaScript bietet auch die Möglichkeit eines *Iterators:* In einer Syntax der Form

```
for (laufvariable in struktur)
```

nimmt die Variable `laufvariable` alle Werte von `struktur` an; dabei kann `struktur` ein Objekt sein (vgl. 15.4) – dann werden die Werte aller Attribute durchlaufen – oder ein Array.

### 15.3.7 Importieren von JavaScript-Code

Seit der Version 1.1 von JavaScript ist es möglich, JavaScript-Code in eine externe Datei auszulagern; für diese Dateien ist die Endung `.js` üblich. Hierfür verfügt das `SCRIPT`-Tag über das Attribut `src`, vergleichbar zum Einbinden von Bildern mittels des `IMG`-Tags.

Der Import einer JavaScript-Datei `file.js` erfolgt dabei konkret durch die Syntax

```
<SCRIPT language="JavaScript" src="file.js"></SCRIPT>
```

Abbildung 15.13 zeigt ein Code-Beispiel für das Einbinden eines externen JavaScripts, welches im gleichen Ordner wie die HTML-Datei auf dem Webserver abgelegt ist.

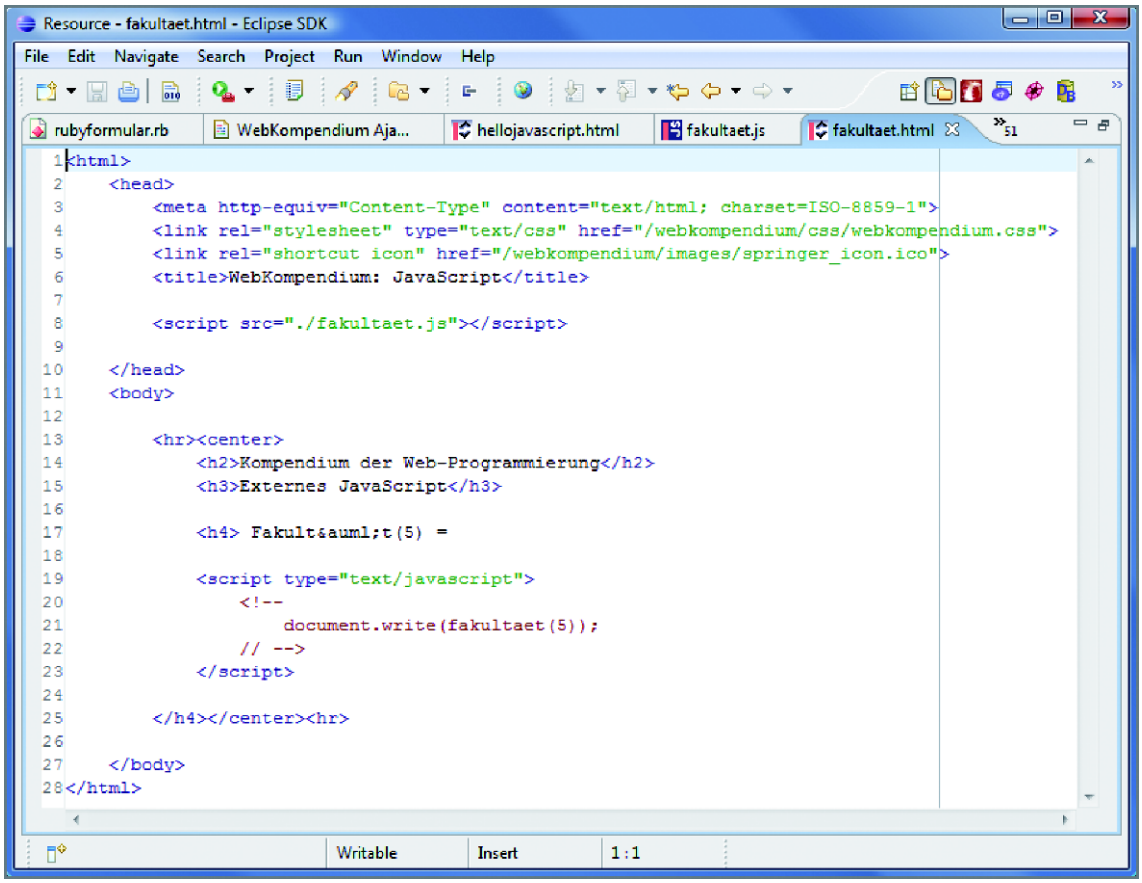



**Abbildung 15.13:** Einbinden eines externen JavaScripts (Zeile 8) `fakultaet.html`

Hier soll nochmal die vorliegende Systemarchitektur erläutert werden: Der Webclient greift über das HTTP-Protokoll auf den Webserver zu; dieser liefert das HTML-Dokument aus, welches das `SCRIPT src`-Tag enthält. Durch dieses Tag fordert der Webclient vom gleichen Server von der gleichen URL wie das Ausgangsdokument die js-Datei an, welche dann der Webserver ebenfalls ausliefert und vom Client ebenfalls eingebunden wird.

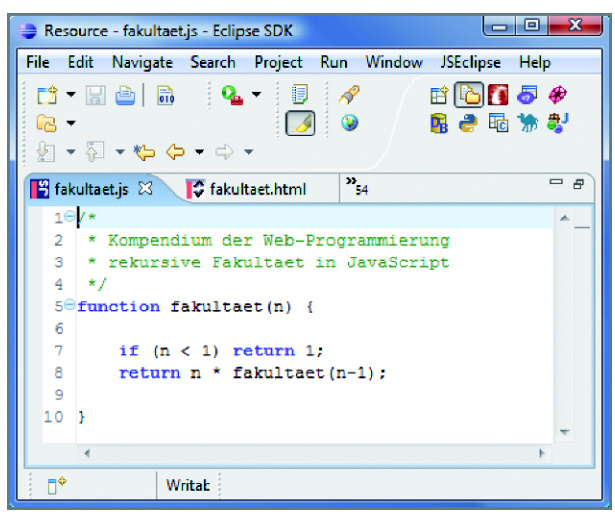



**Abbildung 15.14:** Das importierte JavaScript `fakultaet.js`

Über das `src`-Attribut kann auch ein JavaScript von einem anderen Webserver angefordert, wenn die vollständige HTTP-Adresse angegeben ist (Abbildung 15.15).

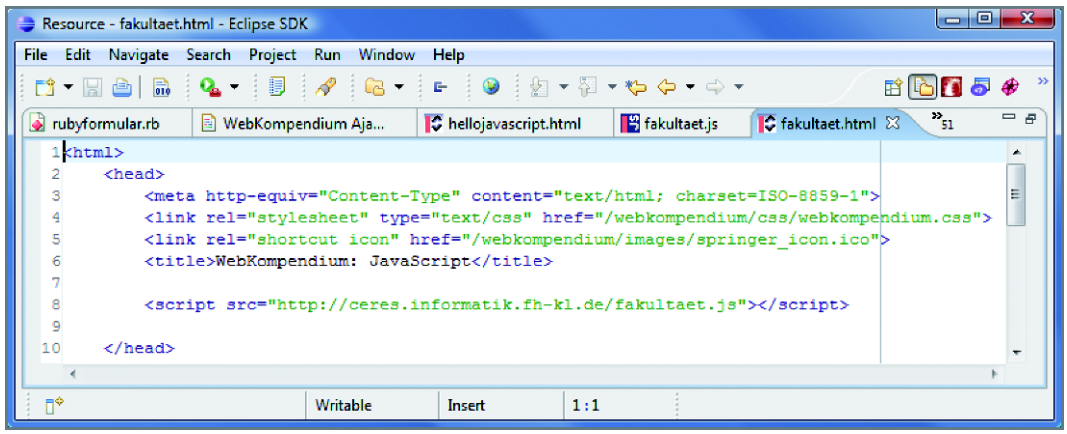



**Abbildung 15.15:** Einbinden eines externen JavaScripts von einem anderen Webserver

## 15.4 Objektorientierung in JavaScript

JavaScript verfügt als moderne Scriptsprache auch über eine Art der *Objektorientierung,* auch wenn diese von den in 4.2.2 vorgestellten Konzepten recht deutlich abweicht, weil sie nicht annähernd so weit entwickelt ist.

Ungewöhnlich ist, dass JavaScript kein Schlüsselwort „class“ verwendet: Klassen in JavaScript werden über den Funktionsbegriff des Konstruktors erklärt. Die Anwendung einer Methode auf ein Objekt geschieht – wie schon im ersten Script aus Abbildung 15.1 ersichtlich – über die normale Java-Syntax

```
objektname.methode(argumente)
```

und der Zugriff auf eine Eigenschaft eines Objektes wird über

```
objektname.eigenschaft
```

ermöglicht; für den Zugriff auf diese Objektattribute bietet JavaScript allerdings noch eine zweite Möglichkeit über die Syntax

```
objektname["eigenschaft"]
```

an. Für den Bezug auf das eigene Objekt steht wieder die Referenz auf das Objekt selbst über das Schlüsselwort `this` zur Verfügung.

Was JavaScript zunächst nicht bietet sind die wichtigen OO-Mechanismen für

- Vererbung;
- Datenkapselung durch Steuerung der Zugriffsrechte wie „private“.

Für das Attribut „static“ als Kennzeichner einer Klasseneigenschaft wird auf die Verwendung von `this` verzichtet.

Als Beispiel soll wieder in Anlehnung an Kapitel 7 eine Klasse zur Beschreibung eines Buches geschrieben werden. Diese wird durch folgenden Code definiert (Datei `Buch.js`):

```
/*
 * Kompendium der Web-Programmierung
 * Objektorientierung in JavaScript
 */

function Buch(titel, autor, verlag, jahr) {

this.titel  = titel;
this.autor  = autor;
this.verlag = verlag;
this.jahr   = jahr;

}  // Konstruktor
```

Dies ist letztlich nur die Definition des Konstruktors, welcher einfach über das Schlüsselwort `this` die Attribute belegt.

Eine Instanz dieser Klasse wird über das Schlüsselwort `new` erzeugt:

```
var objekt = new Konstruktormethode(argumente);
```

Am konkreten Beispiel wird etwa durch

```
var wk = new Buch("Kompendium der Web-Programmierung",
                  "Thomas Walter", "Springer", 2007);
```

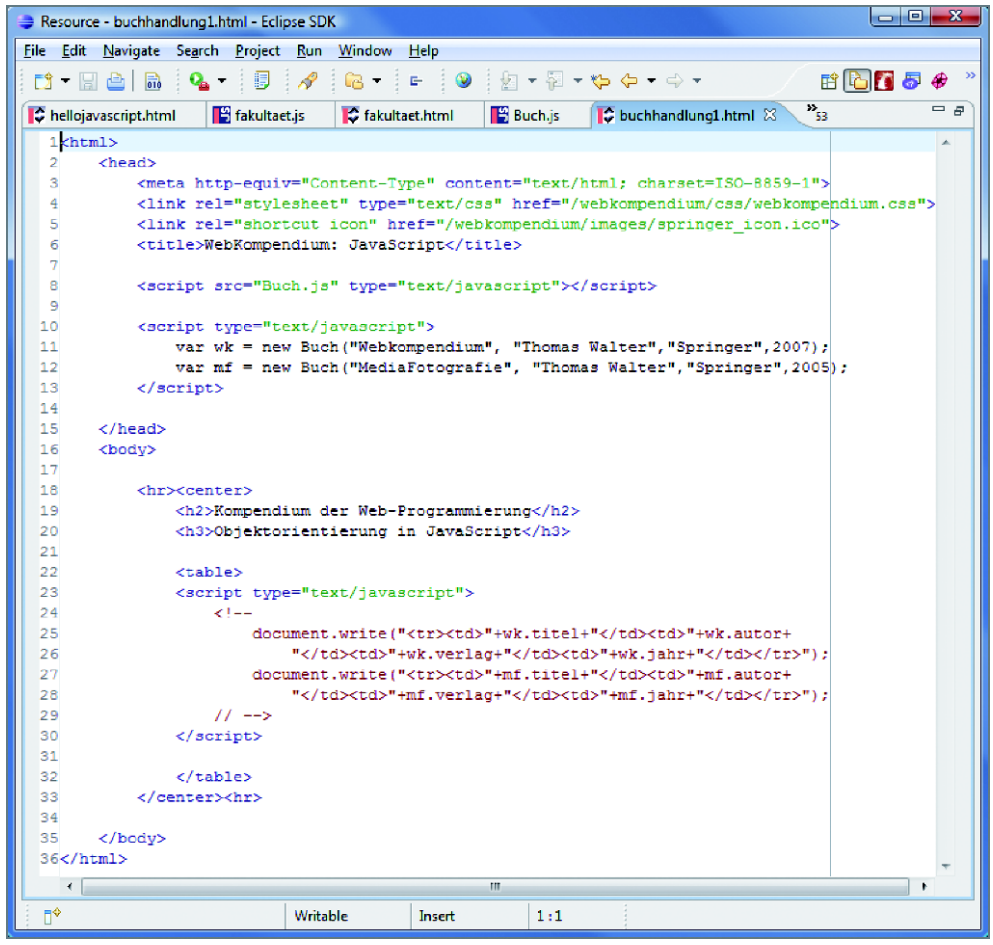
eine Instanz erzeugt.

Seit Version 1.2 verfügt JavaScript über einen Objektinitialisierer, mit welchem Instanzen ohne Klassenangabe über eine Syntax der Form

```
var objekt = {eigenschaft1:wert1, eigenschaft2:wert2, ...};
```

definiert werden können.

Ein einfaches Beispiel – die Datei `buchhandlung1.html` (Abbildung 15.16) – importiert zunächst `Buch.js` für die Klassendefinition und erzeugt anschließend zwei Instanzen dieser Klasse, alles im HTML-Header. Im Body werden dann die Objektattribute ausgegeben (Abbildung 15.17).




```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
        <title>WebKompendium: JavaScript</title>

        <script src="Buch.js" type="text/javascript"></script>

        <script type="text/javascript">
            var wk = new Buch("Webkompendium", "Thomas Walter","Springer",2007);
            var mf = new Buch("MediaFotografie", "Thomas Walter","Springer",2005);
        </script>

    </head>
    <body>

        <hr><center>
            <h2>Kompendium der Web-Programmierung</h2>
            <h3>Objektorientierung in JavaScript</h3>

            <table>
            <script type="text/javascript">
                <!--
                    document.write("<tr><td>"+wk.titel+"</td><td>"+wk.autor+
                        "</td><td>"+wk.verlag+"</td><td>"+wk.jahr+"</td></tr>");
                    document.write("<tr><td>"+mf.titel+"</td><td>"+mf.autor+
                        "</td><td>"+mf.verlag+"</td><td>"+mf.jahr+"</td></tr>");
                // -->
            </script>

            </table>
        </center><hr>

    </body>
</html>
```

**Abbildung 15.16:** Einbinden der Klassendefinition, Erzeugung zweier Objekte und Zugriff auf Objektattribute (Datei `buchhandlung1.html`)

**Abbildung 15.17:** Ausführen von `buchhandlung1.html` im Browser

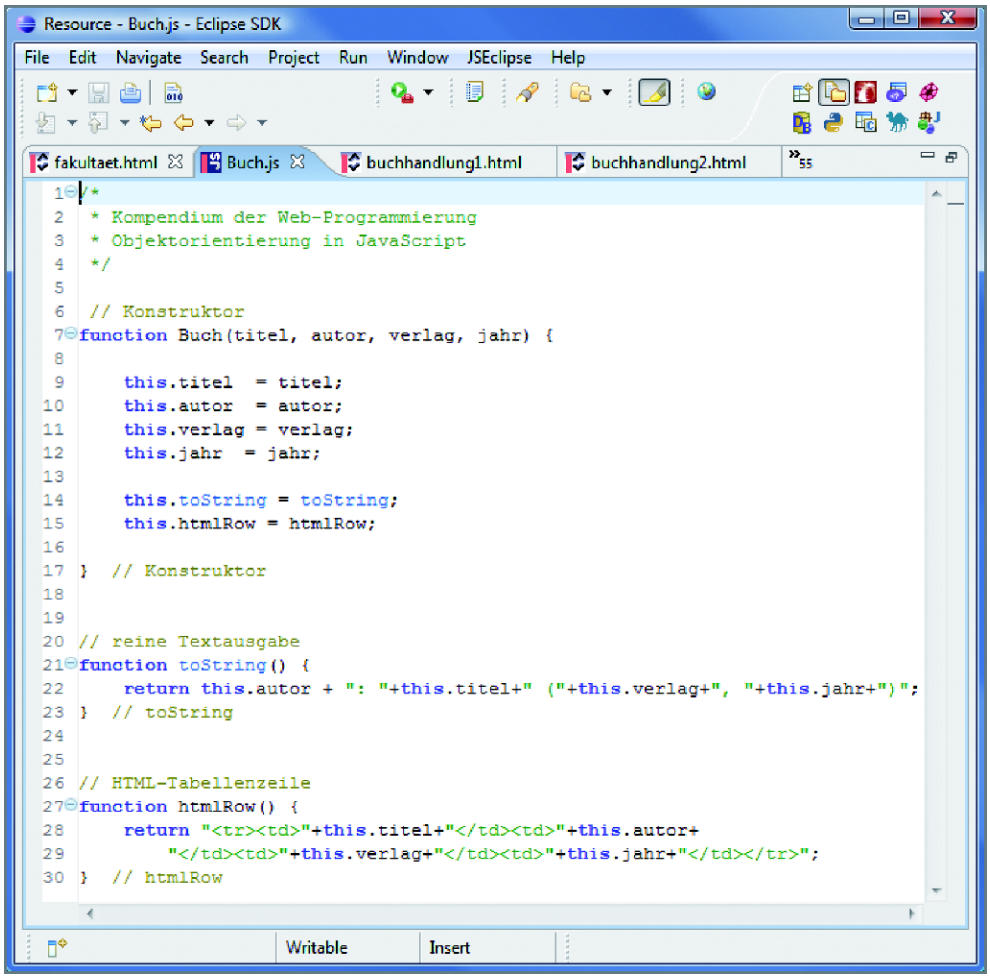



**Abbildung 15.18:**
Definition der Klasse `Buch` in der Datei `Buch.js`

Im nächsten Schritt sollen Methoden zu unserer Klasse `Buch` ergänzt werden, etwa eine Methode `htmlRow`, welche eine HTML-Tabellenzeile mit den Attributen eines Objektes generiert. Abbildung 15.18 zeigt die entsprechende JavaScript-Datei dafür. In dieser Datei geschehen zwei Dinge:

- Zwei Methoden werden ergänzt (Zeilen 21 und 27).
- wesentlich ist aber, dass diese Methoden mit der Klasse und dann mit jeweiligen Objekt verbunden werden; durch die Anweisungen in den Zeilen 14 und 15 im Konstruktor werden die neuen Methoden zur Klasse hinzugefügt; und da diese Definitionen noch das Schlüsselwort `this` verwenden, beziehen sich die Methoden auf Objekte der Klasse (sind also in der Java-Diktion nicht static).

Mit dieser Ergänzung kann dann die anwendende HTML-Datei wesentlich einfacher geschrieben werden; die Ausgabe der Objekte erfolgt nun über

```
<script type="text/javascript">
    <!--
    document.write(wk.htmlRow());
    document.write(mf.htmlRow());
    // -->
</script>
```

Die hier bereits verwendete Methode `toString` hat – wie aus Java bekannt – eine besondere Bedeutung:

**Wie in Java wird auch in JavaScript die Methode `toString` direkt bei jeder Ausgabe ausgeführt: `document.write(einObjekt);` führt also zur Ausgabe von `einObjekt.toString()`.**

### 15.4.1 Standardklassen von JavaScript

JavaScript verfügt über eine ganze Reihe von Standardklassen. Bereits vorgestellt wurde die Klasse `String` für die die Behandlung von Zeichenketten. Methoden der Klasse `String` sind etwa `charAt`, `toUpperCase` und `toLowerCase` sowie das ebenfalls schon vorgestellte `substring`. Die Methode `indexOf(string)` liefert den Index des ersten Auftretens des Arguments, wobei die erste Position den Index 0 hat; gibt es keinen Treffer, liefert die Methode $-1$.

Seit JavaScript 1.2 gibt es Klassen für numerische Datentypen; hier sind die Typumwandlung durch die Methoden `toString` in eine Zeichenkette und `valueOf` für die umgekehrte Umwandlung in einen numerischen Typ besonders nützlich.

Die Klasse `Math` bietet – vergleichbar zur Java-Klasse `java.lang.Math` – zahlreiche mathematische Funktionen wie `abs`, `sin` und `cos` und `exp` sowie `pow` zum Potenzieren mit zwei Argumenten. Nützlich kann auch die Methode `random` in dieser Klasse für die Berechnung von Zufallszahlen sein.

Für die Angabe eines Zeitpunkts ist die Klasse `Date` wichtig: Mittels

```
var jetzt = new Date();
```

wird das Objekt `jetzt` erzeugt, welches sich den Zeitpunkt seiner Erzeugung merkt. Für dieses sind dann zahlreiche Methoden anwendbar:

- `getDate()` liefert den Monatstag zwischen 1 und 31;
- `getDay()` bestimmt den Wochentag zwischen 1 und 7;
- `getHours()` sind die Stunden von 0 bis 23;
- `getMinutes()` sind entsprechend die Minuten zwischen 0 und 59;
- `getMonth()` bestimmt den Monat von 0 bis 11;
- `getYear()` ist die Jahresangabe, welche in verschiedenen Browsern leider nicht einheitlich implementiert ist;
- `getTime()` liefert das universelle Unix-Zeitformat: die Anzahl der Millisekunden seit Mitternacht am 1.1.1970.

## 15.5 JavaScript und HTML: DOM

Bis hierher ist JavaScript eine weitere, nette Scriptsprache. Das besondere, weshalb JavaScript einen wichtigen Platz in diesem Buch einnimmt, ist die Verbindung von JavaScript mit HTML: die Integration von JavaScript in eine Web-Site. JavaScript kann wie keine andere Sprache in eine Web-Site integriert werden und damit die Möglichkeiten von HTML/CSS wesentlich erweitern, was die Sonderstellung von JavaScript begründet.

JavaScript stellt zunächst eine Reihe von Objekten bereit, etwa für den Browser und das aktuell dargestellte Objekt; diese Objekte können in JavaScript verändert werden. Ferner können sehr einfach Ereignisse vom Browser wie Maus-Aktionen mit JavaScript-Methoden über Event-Handles verbunden werden.

Grundlage hierfür ist ein hierarchisches Modell der Web-Site, welches als *DOM* bezeichnet wird: *Document Object Model* [web].

### 15.5.1 DOM und JavaScript

JavaScript legt zahlreiche Objekte direkt an, die die Interaktion mit dem Client-System ermöglichen. Hierzu zählen auf der obersten Ebene drei unabhängige Objekte, die Wurzeln von DOM:

- `window` ist der Oberbegriff für alles, was im aktuellen Fenster geschieht; hierzu gehören insbesondere das angezeigte Dokument (`document`) mit allen seinen HTML-Elementen und mögliche Ereignisse wie ein Mausklick;
- `screen` stellt den Bildschirm dar und enthält u.a. Eigenschaften wie Farbtiefe und Bildschirmauflösung;
- `navigator` erinnert mit seinem Namen noch an die Herkunft von Netscape und beinhaltet Browser-Eigenschaften wie dessen Name und die verfügbaren Browser-Plugins.

Diese drei unabhängigen Objekte sind jeweils die Köpfe der komplexen Objekthierarchie von DOM; ein Teil dieser Hierarchie ist in den Abbildungen 15.19 und 15.20 zu sehen.

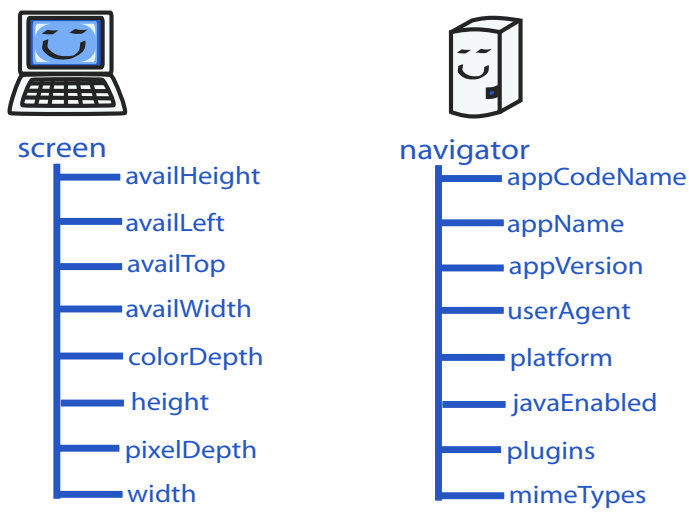



**Abbildung 15.19:** DOM-Hierarchie für screen und navigator

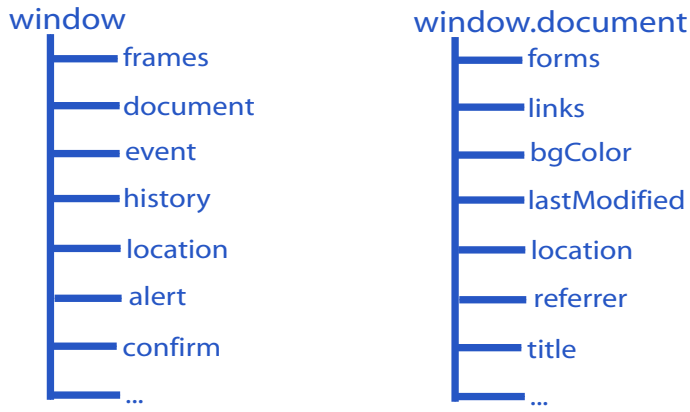



**Abbildung 15.20:** DOM-Hierarchie für window und window.document

### 15.5.2 Das navigator-Objekt

Das `navigator`-Objekt dient der Verwaltung des Browsers. Es verfügt über zahlreiche Attribute, die wichtigsten sind:

- `appName` ist der Name des Browsers;
- `appCodeName`: die Herstellerbezeichnung des Browsers;
- `appVersion` ist die Version des Browsers;
- `userAgent` entspricht der Namensnennung, welche der Browser an den Webserver sendet (dies entspricht damit der entsprechenden CGI-Umgebungsvariablen);
- `javaEnabled` gibt an, ob im Browser Java-Applets ausgeführt werden können (vgl. 18);
- `platform` liefert Informationen zur Systemarchitektur;
- `plugins` liefert ein Array mit allen verfügbaren Browser-Plugins; dieses Attribut ist nicht für alle Browser verfügbar;
- `mimeTypes` liefert – ebenfalls nicht in allen Browsern – ein Array mit allen unterstützen MIME-Types.

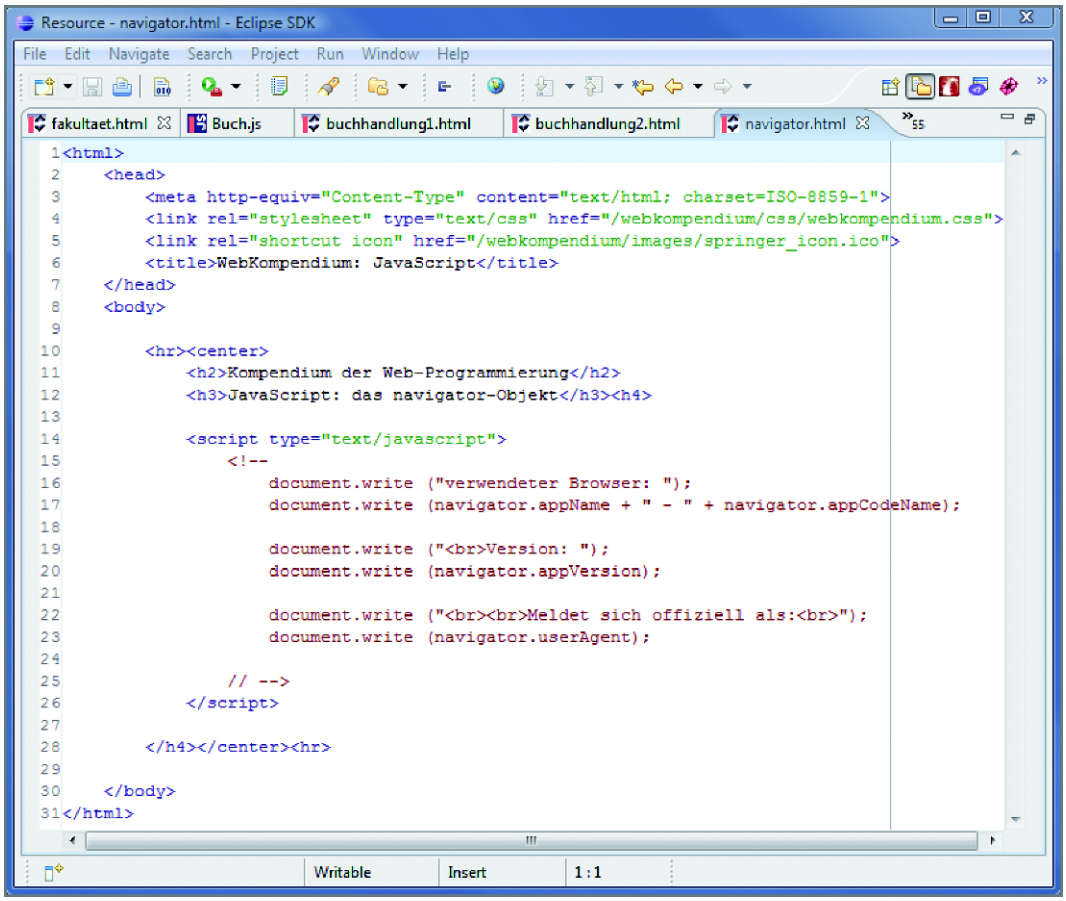



**Abbildung 15.21:**
Anwendung des navigator-Objektes (Datei `navigator.html`)

**Abbildung 15.22:**
Ausführen von `navigator.html`

### 15.5.2.1 Die Browserweiche

Eine gute Web-Entwicklung sollte stets unabhängig vom verwendeten Browser sein und mit sauberem CSS immer funktionieren. Das ist aber in der Praxis nicht immer vollständig erreichbar, so dass doch eine gewisse Notwendigkeit besteht, für verschiedene Browser die Implementierung variieren zu können; dies leistet die *Browserweiche*.

Schematisch kann man eine derartige Fallunterscheidung beispielsweise folgendermaßen erreichen:

```
if (navigator.appName.indexOf("'Internet Explorer"') != -1)
      document.write("InternetExplorer");
else
      document.write("anderer Browser");
```

### 15.5.3 Das screen-Objekt

Sehr ähnlich zu navigator arbeitet das *screen-Objekt*. Dieses erlaubt es, Informationen des Bildschirms zu erhalten.

Zu den verfügbaren Attributen des screen-Objektes zählen:

- `availableHeight` Anzahl der Bildschirmpixel in der Höhe;
- `availableWidth` Anzahl der Bildschirmpixel in der Breite;
- `colorDepth` Farbtiefe in bit.

Die Datei `screen.html` (Abbildung 15.24) enthält ein einfaches Beispiel dazu (Abbildung 15.23).

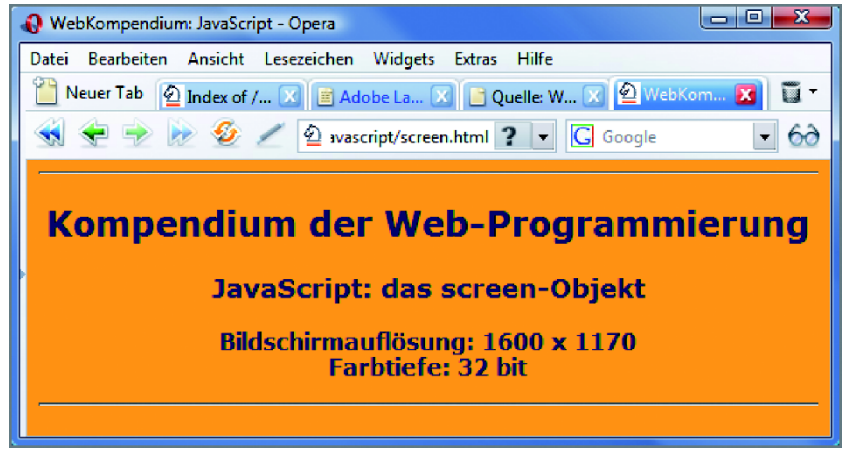
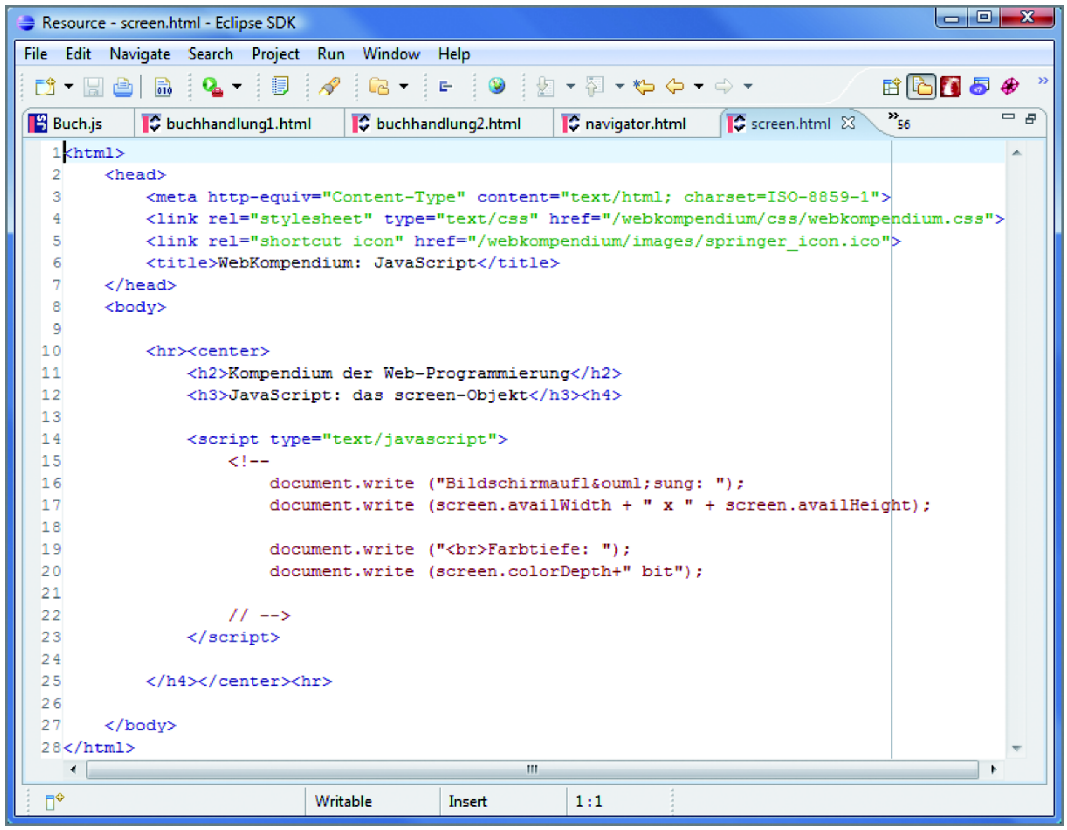



**Abbildung 15.23:** Ausführen von `screen.html`

Problematisch kann das screen-Objekt bei Systemen mit mehreren Monitoren werden; hier liefert aktuell nur Firefox das erwartete Ergebnis der Daten desjenigen Bildschirms, welcher den Browser anzeigt – die übrigen liefern die Daten des Hauptbildschirms.




```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
        <title>WebKompendium: JavaScript</title>
    </head>
    <body>

        <hr><center>
            <h2>Kompendium der Web-Programmierung</h2>
            <h3>JavaScript: das screen-Objekt</h3><h4>

            <script type="text/javascript">
                <!--
                    document.write ("Bildschirmaufl&ouml;sung: ");
                    document.write (screen.availWidth + " x " + screen.availHeight);

                    document.write ("<br>Farbtiefe: ");
                    document.write (screen.colorDepth+" bit");

                // -->
            </script>

        </h4></center><hr>

    </body>
</html>
```

**Abbildung 15.24:**
Anwendung des screen-Objektes `screen.html`

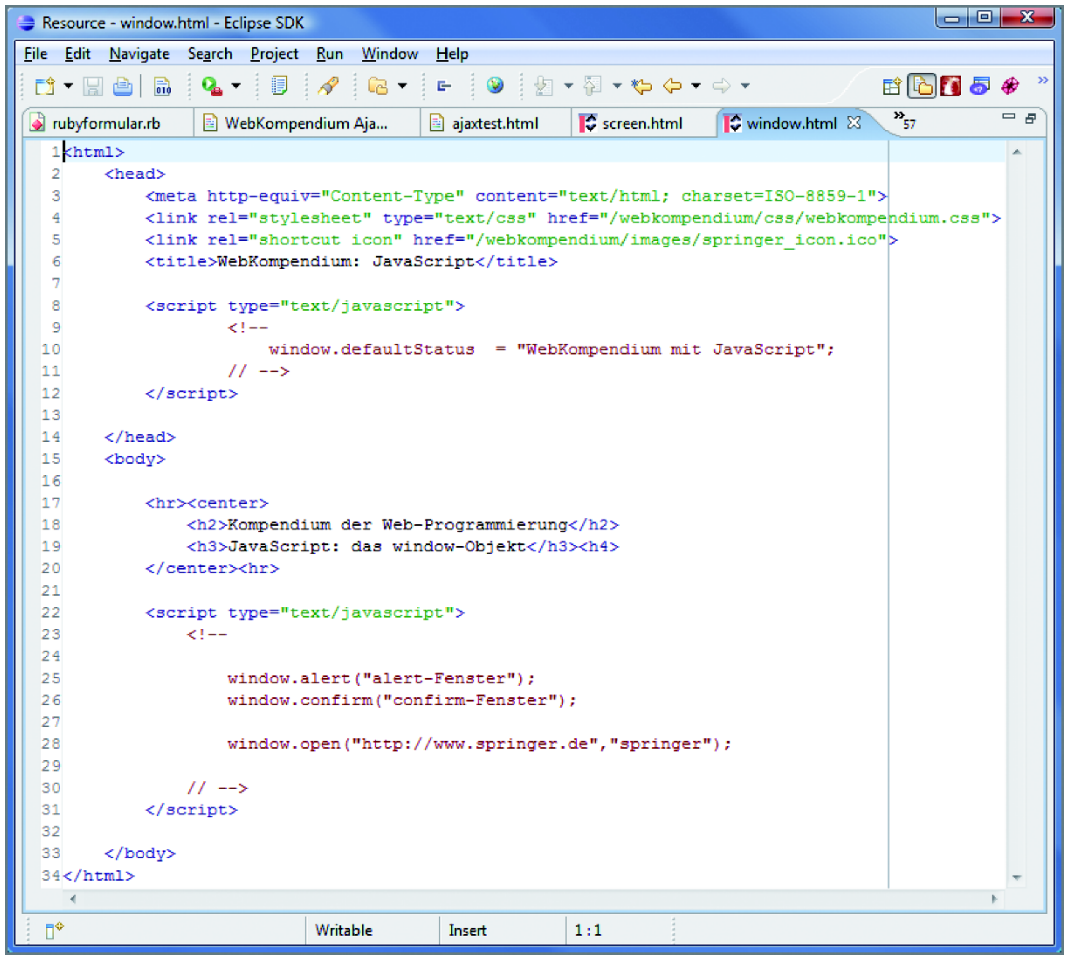
### 15.5.4 Das window-Objekt

Das window-Objekt stellt die Spitze der Dokumenthierarchie dar und ist wesentlich komplexer als navigator und screen. Zu window gehören zahlreiche wichtige Untergliederungen; die wichtigsten sind frames, document, event, history und location. Dazu kommen Eigenschaften und Methoden, welche direkt window zugeordnet sind.

Alles kann hier nicht vollständig vorgestellt werden; zu den wichtigsten Untergliederungen von window selbst zählen:

- `alert(message` öffnet ein Warnfenster mit der Meldung `message` (Abbildung 15.26);
- `blur()` ist eine Methode, um das aktive Browserfenster in den Hintergrund zu verschieben;
- `close()` schließt das Browserfenster;
- `confirm(message)` öffnet ein Bestätigungsfenster (Abbildung 15.26);
- `defaultStatus` definiert die Default-Textanzeige des Browsers in seiner status-Anzeige (Abbildung 15.26);
- `focus()` aktiviert das Fenster;
- `open(url,name,option)` öffnet die Adresse `url` in einem neuen Browserfenster, welches über den Bezeichner `name` gesteuert wird; unter `option` kann etwa die Größe des neuen Fensters wie „height=200,width= 300“ angegeben werden;
- `status` ist die aktuelle Textmeldung in seiner Statusanzeige.

Die Datei `window.html` (Abbildung 15.25) zeigt einige Möglichkeiten für das window-Objekt; das neue Fenster der Methode `open` wird dabei in einem neuen Tab geöffnet (Abbildung 15.26).




```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
        <title>WebKompendium: JavaScript</title>

        <script type="text/javascript">
                <!--
                    window.defaultStatus  = "WebKompendium mit JavaScript";
                // -->
        </script>

    </head>
    <body>

        <hr><center>
            <h2>Kompendium der Web-Programmierung</h2>
            <h3>JavaScript: das window-Objekt</h3><h4>
        </center><hr>

        <script type="text/javascript">
            <!--

                window.alert("alert-Fenster");
                window.confirm("confirm-Fenster");

                window.open("http://www.springer.de","springer");

            // -->
        </script>

    </body>
</html>
```

**Abbildung 15.25:** Anwendung des window-Objektes (`window.html`)

Häufig verwendet wird die Methode `open` für die „Popup-Fenster“[2]; diese sind inzwischen ausgesprochen unbeliebt und werden von den meisten Browsern standardmäßig direkt abgeblockt. Insgesamt ist die `open`-Methode sehr leistungsfähig, die Optionen bieten folgende Möglichkeiten:

- height / width : Höhe/Breite in Pixel
- directories = yes | no : Bookmarks;
- location = yes | no : URL-Anzeige
- menubar = yes | no : Anzeige der Menüleiste;
- resizeable = yes | no : steuert, ob der Benutzer die Fenstergröße ändern kann;
- scrollbars = yes | no : Anzeige von Scrollbars;
- status = yes | no : Anzeige der Status-Zeile;
- toolbar = yes | no : Anzeige der Toolbar;
- top/left : Position, an welcher das neue Fenster erscheint.

[2] nicht zu verwechseln mit dem HTML-Link in ein neues Fenster über das Attribut `target="_blank"`.

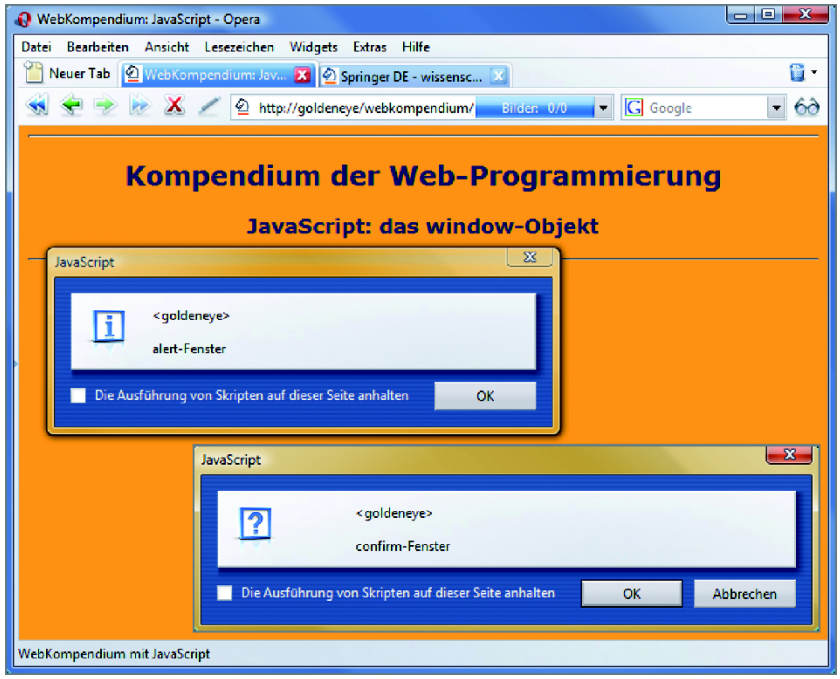



**Abbildung 15.26:**
Ausführen von `window.html`

Ebenso wie `open` sparsam verwendet werden sollte, ist heute auch die Möglichkeit der Veränderung der Status-Zeilenanzeige nicht mehr aktuell und sollte unbedingt vermieden werden.

**Die zahlreichen Möglichkeiten der Browsermanipulation durch JavaScript sollten heute nur sehr bewußt und sparsam eingesetzt werden.**

Und da das window-Objekt von so besonderer Bedeutung ist, kann in den meisten Fällen auf die Verwendung des Bezeichners „window“ verzichtet werden:

**Die meisten JavaScript-Objekte beziehen sich auf window, weshalb meistens auf diesen Bezeichner verzichtet werden kann, etwa nur `alert(...)` anstelle von `window.alert(...)` oder `document.write(...)` anstelle von `window.document.write(...)`**

### 15.5.4.1 JavaScript als Link

JavaScript-Methoden können auch als *Link* verwendet werden, etwa mittels

```
<a href="javascript:window.close()">Fenster schließen</a>
```

### 15.5.4.2 Das window.location-Objekt

Das *window.location-Objekt* verwaltet das vom Browser aktuell angezeigte Objekt.

Die Eigenschaften und Methoden von window.location sind:

- `protocol`: das verwendete Protokoll wie „http:“
- `hostname`: Host des Dokuments;
- `port`: verwendeter tcp-Port;
- `pathname`: Pfad des Dokuments;

- `href`: komplette URL des aktuellen Dokuments;
- `host`: Angabe von hostname:port;
- `reload()` lädt das aktuelle Dokument neu;
- `replace(url)` ersetzt das angezeigte Dokument durch das Dokument von der Adresse `url`.

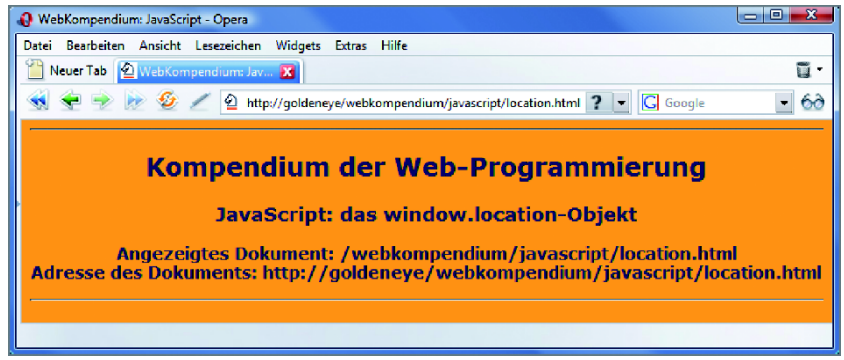
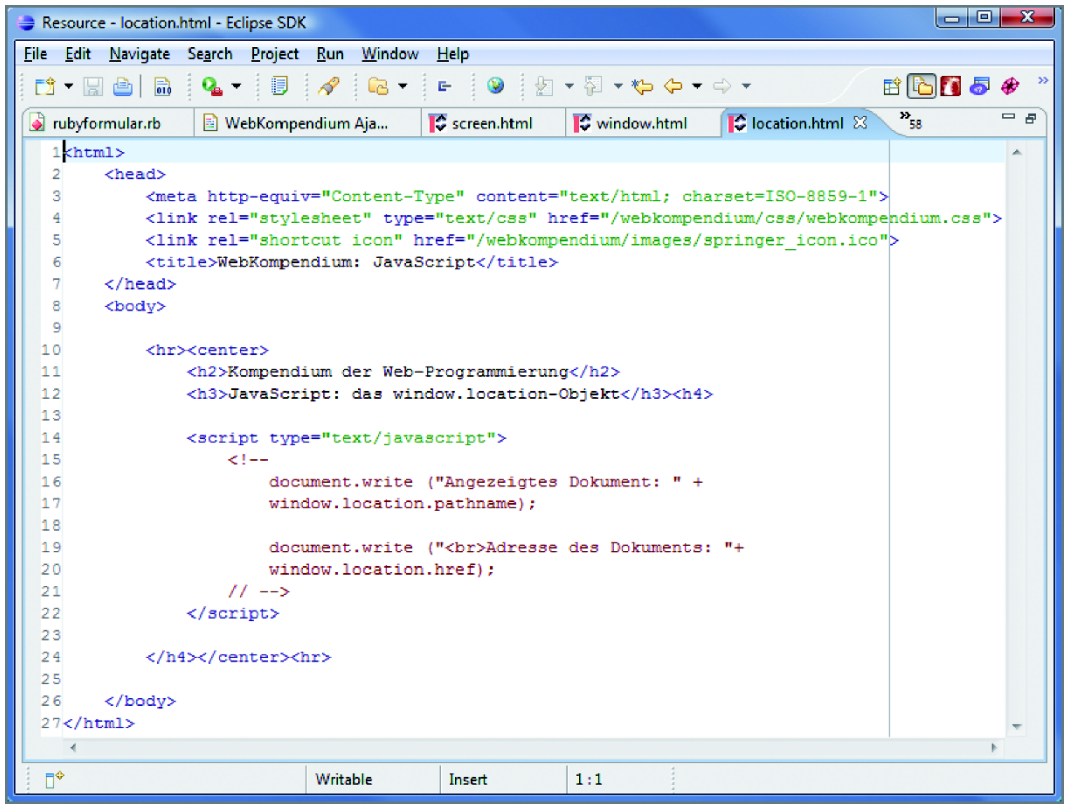



**Abbildung 15.27:** Ausführen von `location.html`




```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
        <title>WebKompendium: JavaScript</title>
    </head>
    <body>

        <hr><center>
            <h2>Kompendium der Web-Programmierung</h2>
            <h3>JavaScript: das window.location-Objekt</h3><h4>

            <script type="text/javascript">
                <!--
                    document.write ("Angezeigtes Dokument: " +
                    window.location.pathname);

                    document.write ("<br>Adresse des Dokuments: "+
                    window.location.href);
                // -->
            </script>

        </h4></center><hr>

    </body>
</html>
```

**Abbildung 15.28:** Anwendung des window.location-Objektes `location.html`

### 15.5.4.3 Das window.history-Objekt

Das *window.history-Objekt* steht für die Browser-History; es bietet die drei Methoden `back()` zum Laden des vorherigen Dokuments, `forward()` zum Laden des nachfolgenden Elements und `go(n)` für das Element, welches *n* Schritte zurück liegt.

#### 15.5.4.4 Das window.frames-Objekt

Das *window.frames-Objekt* verwaltet die verschiedenen Frames eines HTML-Dokuments; da die Frame-Technik aufgrund mehrerer Nachteile (keine Barrierefreiheit, kein Verweis in einen Frame) heute in der Regel nicht mehr verwendet wird, kommt diesem Objekt nur noch eine geringe praktische Bedeutung zu. Das frames-Objekt ist intern ein Array, die einzelnen Frames sind die Array-Elemente, welche über ihren Index oder ihren Namen ansprechbar sind; jedes Frame-Element hat sein eigenes document-Objekt.

Nützlich für Frames sind etwa `parent`, die Referenz auf das übergeordnete Frame und `self`, die Referenz auf sich selbst, sowie `top`, die Referenz auf das oberste Frame.

#### 15.5.4.5 Das window.document-Objekt

Dieses Objekt ist unterhalb der window-Objekte wiederum von der größten Bedeutung, denn es verwaltet das HTML-Dokument, das der Browser anzeigt; mithin hat es Zugriff auf die meisten HTML-Eigenschaften des Dokuments.

Zu den Eigenschaften und Methoden von window.document gehören:

- `write(message)`: Textausgabe;
- `title`: Dokumenten-Titel im HEAD;
- `referrer`: URI, von der das Dokument aufgerufen wurde;
- `lastModified`: Datum der letzten Änderung des Dokuments;
- `bgColor` und `fgColor`: Hintergrund- und Vordergrundfarbe.

Der übliche Weg der Angabe des Dokumentendatums ist somit

```
<script>
     document.write(document.lastModified);
</script>
```

Über Server Side Includes gibt es hierfür eine einfache, serverseitige Alternative (vgl. Kapitel 14).

#### 15.5.4.6 Das window.document.forms-Objekt

Das Objekt *window.document.forms* verwaltet die Formulare innerhalb eines HTML-Dokuments. Da ein HTML-Dokument mehrere Formulare enthalten kann, ist forms wie frames ein JavaScript-Array; jedes Array-Element ist ein eigenes Formular.

Wichtig und nützlich für forms sind:

- `elements`: ein Array mit allen Elementen des Formulars;
- `reset()`: Methode zum Löschen aller Einträge;
- `submit()`: Übermittlung des Formulars;
- `elements[n].value`: Wert des n-ten Elements des Formulars;
- `length` kann sowohl auf das forms-Array als auch auf das elements-Array angewendet werden und ergibt im ersten Fall die Anzahl der Formulare im HTML-Dokument und im zweiten Fall die Anzahl der Formularelemente;
- `action` ist die Empfängeradresse des Formulars.

Zur Verdeutlichung soll das erste Formular innerhalb einer HTML-Seite den Aufbau

```
<FORM ACTION="..." METHOD="..." NAME="formular1">
   <INPUT type="..." name="feld1">
   ...
</FORM>
```

haben; auf den Wert des ersten Feldes dieses Formulars kann JavaScript dann über

```
window.document.forms[0].elements[0].value
```

oder auch über

```
window.document.forms[0].feld1.value
```

und sogar über

```
window.document.formular1.feld1.value
```

zugreifen.

#### 15.5.4.7 Das window.document.forms.elements-Objekt

`document.forms[0].elements[0]` ist das erste Element des ersten Formulars des HTML-Dokuments; für ein solches Objekt stehen wieder Eigenschaften und Methoden bereit:

- `checked`: wahr, wenn Element angekreuzt;
- `name`: Bezeichner des Elements;
- `value`: eingetragener Wert;
- `click()`: Element anklicken;
- `focus()`: Eingabefokus auf Element legen.

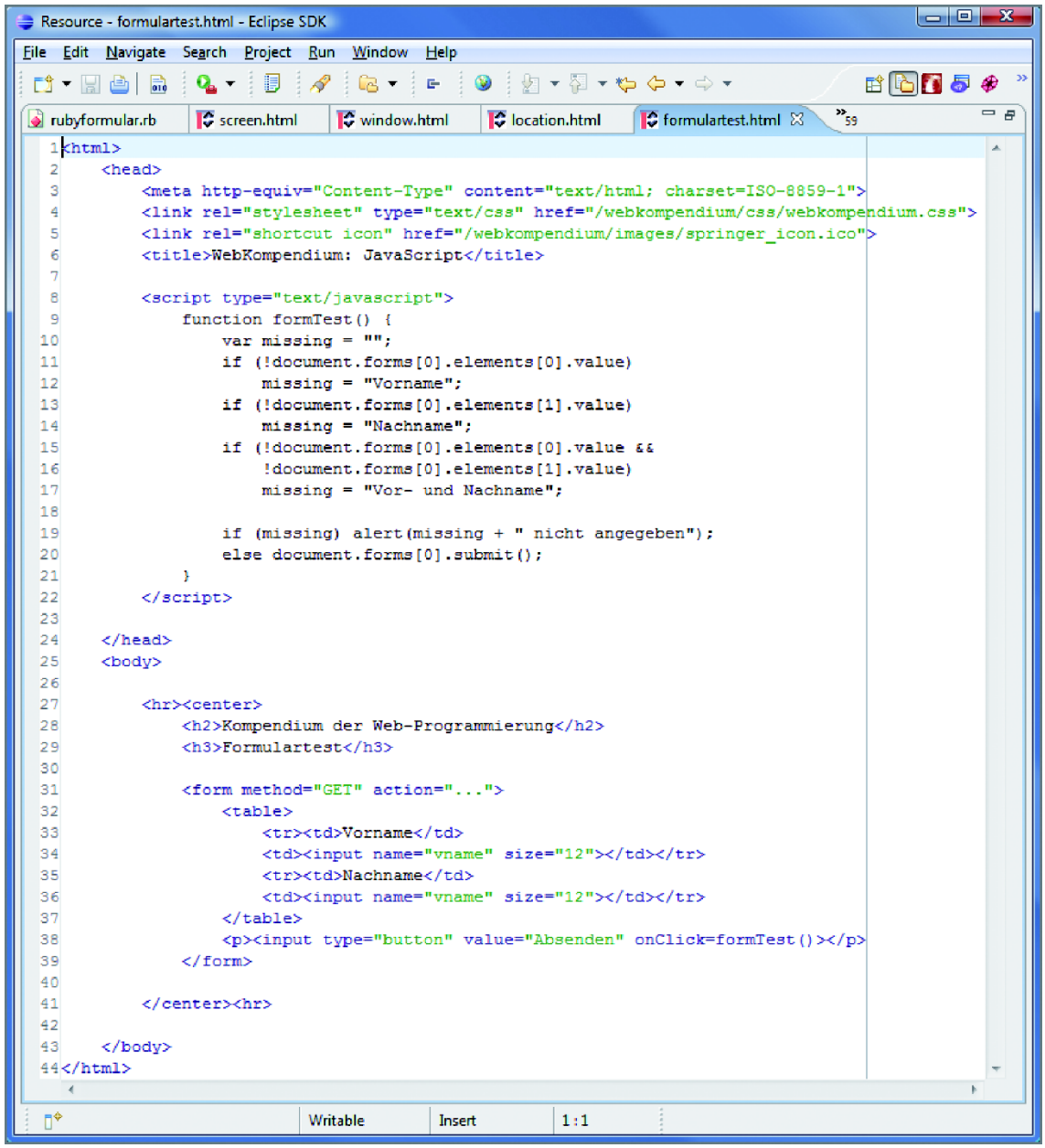
#### 15.5.4.8 Überprüfen von Formularen mit JavaScript

Mit dem bisher vorgestellten Möglichkeiten von JavaScript ist es nun möglich, ein HTML-Formular clientseitig vor dem Versenden zu überprüfen.

Dieses häufig angewendete Verfahren beruht auf der Möglichkeit, das HTML-Formular derart abzuändern, dass der Klick auf den Absendebutton nicht das Formular versendet, sondern eine JavaScript-Methode aufruft; diese überprüft die Formularfelder; im Erfolgsfall wird die Standard-Methode `submit()` auf das konkrete forms-Objekt angewendet, um das Formular zu versenden:

```
document.forms[0].submit();
```

Im anderen Fall muss der Fehler abgefangen werden, etwa durch ein alert-Fenster. Die Datei `formulartest.html` (Abbildung 15.29) enthält ein entsprechendes Beispiel. Die Aufruf der JavaScript-Methode durch den Maus-Klick geschieht durch das Event-Handle `onClick`, das im nächsten Abschnitt (15.6) genauer vorgestellt wird.




```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
        <title>WebKompendium: JavaScript</title>

        <script type="text/javascript">
            function formTest() {
                var missing = "";
                if (!document.forms[0].elements[0].value)
                    missing = "Vorname";
                if (!document.forms[0].elements[1].value)
                    missing = "Nachname";
                if (!document.forms[0].elements[0].value &&
                    !document.forms[0].elements[1].value)
                    missing = "Vor- und Nachname";

                if (missing) alert(missing + " nicht angegeben");
                else document.forms[0].submit();
            }
        </script>

    </head>
    <body>

        <hr><center>
            <h2>Kompendium der Web-Programmierung</h2>
            <h3>Formulartest</h3>

            <form method="GET" action="...">
                <table>
                    <tr><td>Vorname</td>
                    <td><input name="vname" size="12"></td></tr>
                    <tr><td>Nachname</td>
                    <td><input name="vname" size="12"></td></tr>
                </table>
                <p><input type="button" value="Absenden" onClick=formTest()></p>
            </form>

        </center><hr>

    </body>
</html>
```

**Abbildung 15.29:**
Formular mit clientseitiger Überprüfung `formulartest.html`

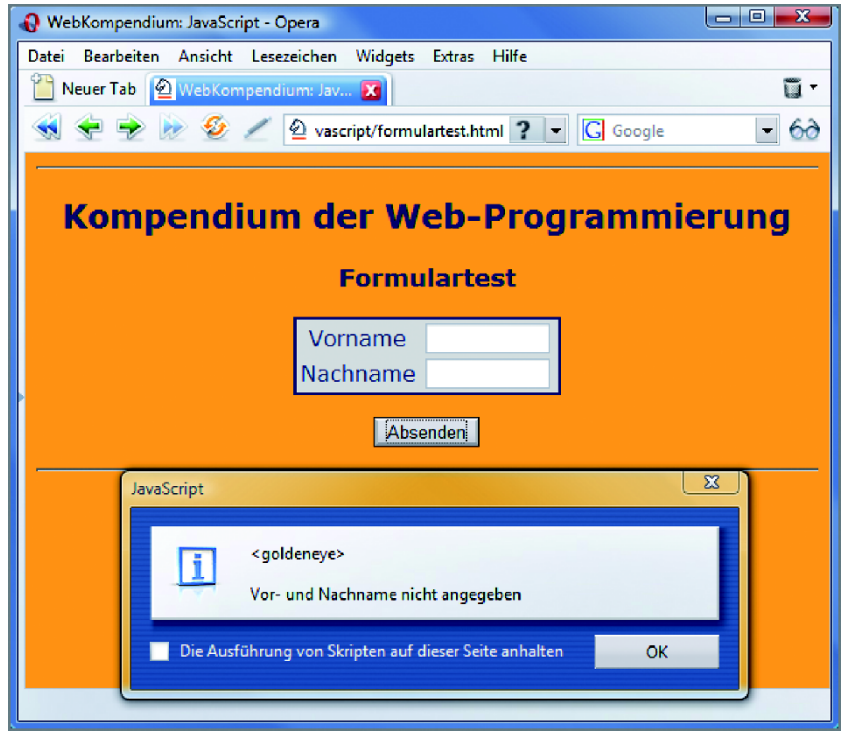



**Abbildung 15.30:**
Ausführen von `formulartest.html`

## 15.6 Event-Behandlung mit JavaScript

„Ereignisse“ *(Events)* sind ein klassischer Begriff der Programmierung grafischer Oberflächen. Unter diesen Events versteht man allgemein die Ereignisse, die eintreten können: ein Klick auf ein spezielles Element oder das Schließen des Fensters und weitere. Die serverseitige Web-Programmierung ist nicht in der Lage, diese Events zu behandeln, da der Server von den meisten Events nichts mitbekommt; hier liegt die Domäne der clientseitigen Programmierung und eine besondere Stärke von JavaScript.

Events werden durch Event-Handles bedient; diese verknüpfen das Eintreten eines Ereignisses mit einer damit ausgelösten Aktion, etwa dem Aufruf einer JavaScript-Methode. Event-Handles in JavaScript beginnen mit „`on`“, etwa das schon beim Formulartest (`formulartest.html` verwendete `onClick`. Sie werden als HTML-Attribute zu den HTML-Elementen definiert, auf welche sie sich beziehen.

Die inzwischen sehr enge Bindung von JavaScript und HTML zeigt sich auch daran, dass die JavaScript Event-Handles nunmehr zum festen Sprachbestandteil von HTML geworden sind.

Ein typisches Beispiel ist das Event-Handle

```
onFocus
```

welcher aufgerufen wird, wenn der Eingabefokus auf das entsprechende Element gelegt wird. So kann durch

```
<INPUT onFocus="jsfunction()" >
```

ein Eingabefeld definiert werden, welches dann, wenn es angewählt wird, die JavaScript-Methde `jsfunction` aufruft, welche entsprechend zu implementieren ist; alternativ kann in einfachen Fällen auch direkt der durch den Event ausgelöste JavaScript-Code angegeben werden. Ein solches Beispiel ist in `event1.html` implementiert; nach Aufrufen des Formulars sind zunächst die Formularfelder leer, sie füllen sich bei Anwahl des Feldes (bei jeder Anwahl erneut mit der Vorbelegung).

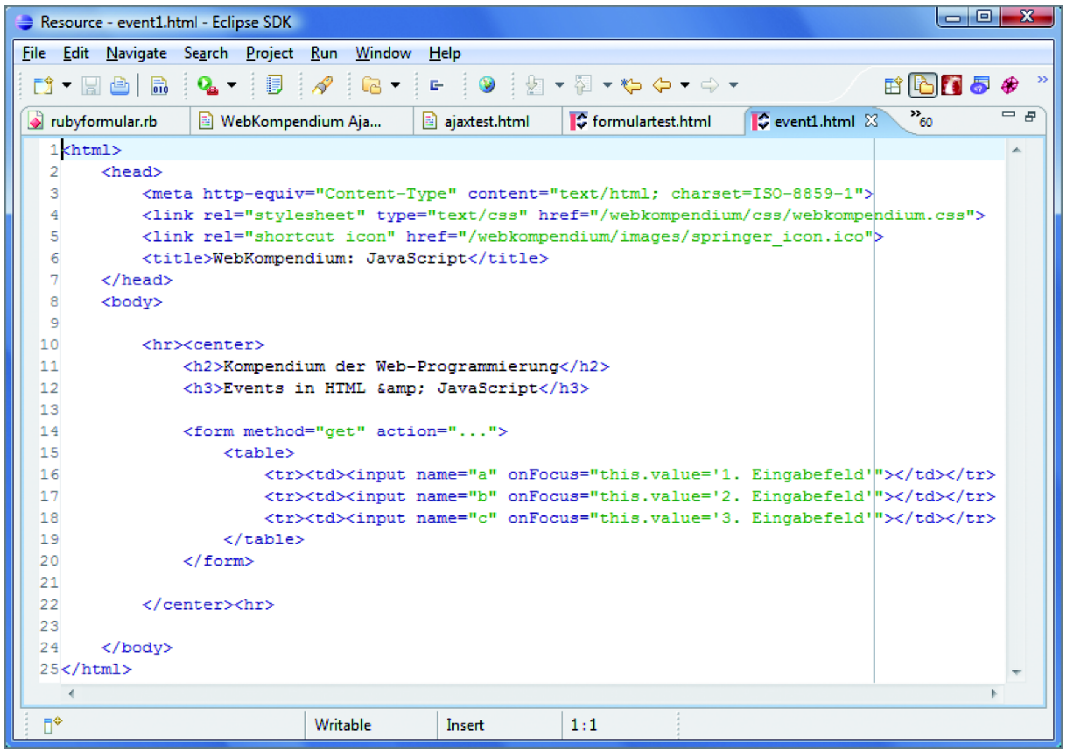



**Abbildung 15.31:**
Einfacher Event-Handle mit JavaScript `event1.html`

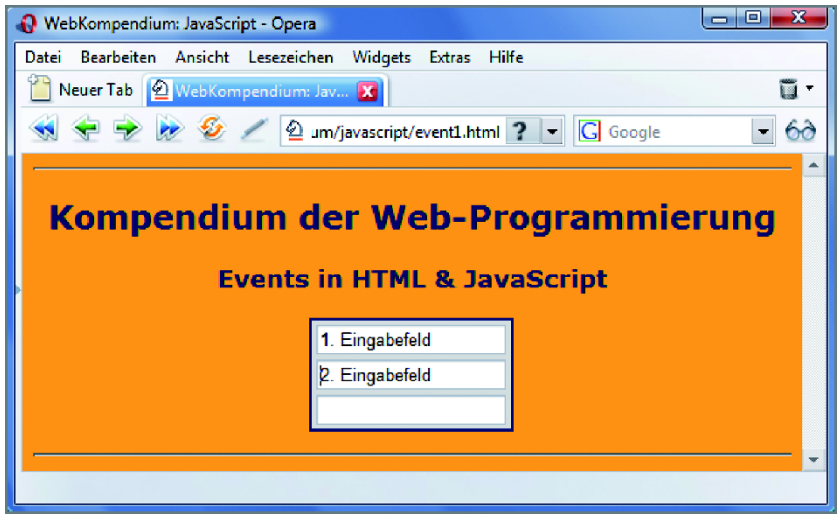
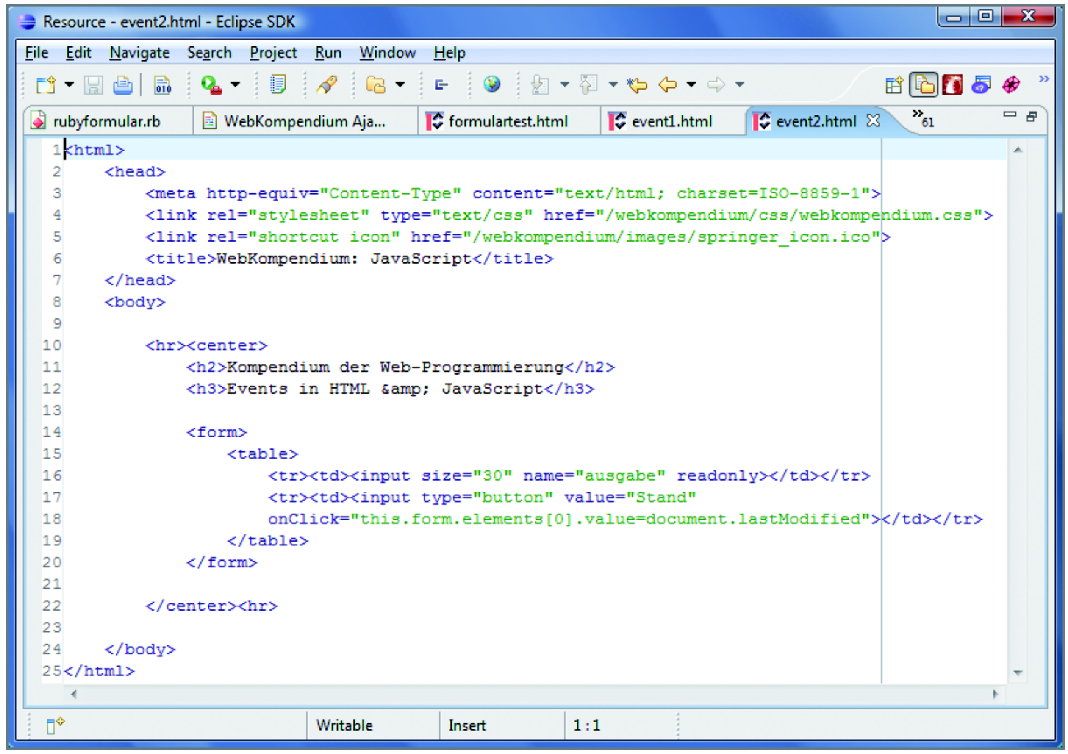



**Abbildung 15.32:**
Ausführen von `event1.html`

Neben `onFocus`, das Hanlde für den Eingabefokus, ist das Handle für den Maus-Klick sehr verbreitet: `onClick`. Bereits in 15.5.4.8 war ein entsprechendes Beispiel zu finden, hier folgt mit `event2.html` ein Weiteres.




```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
        <title>WebKompendium: JavaScript</title>
    </head>
    <body>

        <hr><center>
            <h2>Kompendium der Web-Programmierung</h2>
            <h3>Events in HTML &amp; JavaScript</h3>

            <form>
                <table>
                    <tr><td><input size="30" name="ausgabe" readonly></td></tr>
                    <tr><td><input type="button" value="Stand"
                    onClick="this.form.elements[0].value=document.lastModified"></td></tr>
                </table>
            </form>

        </center><hr>

    </body>
</html>
```

**Abbildung 15.33:** Event-Handle `onClick` `event2.html`

**Abbildung 15.34:** Ausführen von `event2.html`

## 15.7 Übersicht über die Event-Handles

JavaScript verfügt über zahlreiche Event-Handles – aber nicht alle werden von allen Browsern unterstützt. Prinzipiell ist es so, dass die älteren Browser deutlich weniger Handles unterstützen, es ist also bei der Entwicklung einer entsprechenden Site an das zu erwartende Publikum zu denken.

Komplexe HTML-Editoren wie Dreamweaver und GoLive ermöglichen die Auswahl der zum jeweiligen Browser passenden Event-Handles.

Die wichtigsten Event-Handles von JavaScript sind in Tabelle 15.1 aufgeführt.

**Tabelle 15.1:** Event-Handles in JavaScript

| Event-Handle | Bedeutung |
|---|---|
| `onAbort` | bei Abbruch |
| `onBlur` | beim Verlassen |
| `onChange` | bei erfolgter Änderung |
| `onClick` | beim Anklicken |
| `onDblClick` | bei doppeltem Anklicken |
| `onError` | im Fehlerfall |
| `onFocus` | beim Aktivieren |
| `onKeydown` | bei gedrückter Taste |
| `onKeypress` | bei erfolgtem Tastendruck |
| `onKeyup` | bei losgelassener Taste |
| `onLoad` | beim Laden einer Datei |
| `onMousedown` | bei gedrückter Maustaste |
| `onMousemove` | bei weiterbewegter Maus |
| `onMouseout` | beim Verlassen des Elements mit der Maus |
| `onMouseover` | beim Überfahren des Elements mit der Maus |
| `onMouseUp` | bei losgelassener Maustaste |
| `onReset` | beim Zurücksetzen des Formulars |
| `onSelect` | beim Selektieren von Text |
| `onSubmit` | beim Absenden des Formulars |
| `onUnload` | beim Verlassen der Datei |
| `javascript:` | bei Verweisen |

Dies ist eine möglichst umfassende Liste der Event-Handles in JavaScript; einige sollen hier noch etwas genauer erläutert werden:

- `onAbort` wird ausgelöst, wenn im Browser der Stop-Button gedrückt wir, bevor alle Grafiken geladen wurden;
- `onBlur` wird bei Verlassen eines Elementes aktiv;
- `onChange` wird ausgelöst, wenn das spezielle Element verändert wurde;
- `onMouseover` wird aktiv, wenn das spezielle Element mit der Maus überfahren wird;
- `onMouseout` wird aktiv, wenn das spezielle Element mit der Maus verlassen wird;
- `onSubmit` wird bei Absenden des Formulars ausgelöst;
- `javascript:` erlaubt die Verwendung von JavaScript in einem href-Tag.

## 15.8 Komplexere Strukturen: JSON

JavaScript bietet mit JSON – *JavaScript Object Notation* – eine Datenstruktur für komplexere Parameterstrukturen [web]. Dabei ist JSON eigentlich unabhängig von der Programmiersprache, sondern beschreibt ein Textformat für den Datenaustausch.

JSON bildet die sprachunabhängige Basis für zwei Arten von Datenstrukturen:

- Key-Value-Paare wie das Hash von Perl: Eine Sammlung von Schlüssel-Wert-Beziehungen ohne explizite Reihenfolge der Paare.
- Die lineare Liste: Eine Verkettung von Werten.

JSON ist eine Beschreibung dieser Strukturen ohne explizite Verwendung einer Programmiersprache. Der große Vorteil ist hiervon, dass eine Server-Applikation wie PHP mit einer Client-Applikation, etwa Ajax-gesteuertes JavaScript, im JSON-Format sehr performant komplexe Datenstrukturen austauschen kann.

Eine wesentlich komplexere Alternative zu JSON ist natürlich XML, allerdings zeigt sich immer wieder der Vorteil der JSON-Nomenklatur in der einfacheren Umsetzung und der großen Performance.

## 15.9 Zum Einsatz von JavaScript für die Web-Programmierung

Abschließend werden in diesem Abschnitt noch einige wichtige Bemerkungen zum Einsatz von JavaScript für die Web-Programmierung zusammengestellt.

### 15.9.1 Gemischte Client- und Serverscripte

Natürlich ist es problemlos möglich, Client- und Serverscripte zu mischen, beispielsweise aus einem PHP-Script heraus JavaScript zu verwenden:

```
<?php
    echo("<script>document.write(document.lastModified)".
         "</script>");
?>
```

Allerdings ist gerade dieses Beispiel interessant, da ein Server-generiertes Dokument kein „lastModified“ im klassischen Sinne hat, wird hier nur das Datum „1.1.1970“ – der Anfang der Zeitrechnung für Unix – angegeben.

### 15.9.2 Browserabhängigkeit

Leider ist die JavaScript-Implementierung deutlich vom jeweiligen Browser abhängig: Zum einen werden sehr unterschiedliche Versionen von JavaScript unterstützt, aktuell von 1.3 bis 1.7, zum anderen verwenden einzelne Hersteller proprietäre Sprachbestandteile. Besonders deutlich wird dies an der speziellen Implementierung des Ajax-Objektes im Microsoft InternetExplorer vor der Version 7 (vgl. 16).

Ein erfolgreicher Ausweg ist das Document Object Model (DOM), welches eine einheitliche Struktur vorgibt (vgl. 15.5).

### 15.9.3 JavaScript und Sicherheit

Der Einsatz von JavaScript hat in der Vergangenheit insbesondere deshalb gelitten, weil JavaScript als vergleichsweise unsichere Technik gilt, welche den Client gefährden kann.

Ein Problembereich der Sicherheitsverletzung durch JavaScript ist die Möglichkeit der Täuschung des Nutzers, da sich viele Anzeigen des Browsers – etwa die Statuszeile – verändern lassen. Aus diesem Grunde schränken moderne Browser auch diese Manipulationsmöglichkeiten häufig ein bzw. weisen darauf hin. Ebenso können durch JavaScript unbemerkt Benutzer-Aktionen ausgelöst werden, welche der Anwender bewusst so gar nicht ausführen möchte. Auch hat JavaScript Zugriff auf zahlreiche Konfigurationen des Browsers – etwa die installierten Plugins für einige Browsertypen –, deren Bekanntgabe durchaus sicherheitskritisch ist. Kritisch ist ebenfalls die Möglichkeit von JavaScript, Scripte schon beim Laden des Dokuments auszuführen; der Anwender hat hier keine Möglichkeit, die Ausführung zu unterbinden.

Der JavaScript-Prozess auf dem Clientrechner hat die gleichen Rechte wie der eigentliche Browser und damit hohe Zugriffsrechte auf den Client. Zum Vergleich verwendet das Java-Applet hier das Prinzip des „Sandkastens“ (sandbox) zur Kapselung des Prozesses (vgl. 18).

Ein anderes Problem ist die einfache Möglichkeit, durch rekursive Scripte einen Clientrechner praktisch lahm zu legen, vgl.15.9.5.

### 15.9.4 Tipps zur Sicherheit mit JavaScript

Natürlich könnte man nach dem vorherigen Abschnitt schnell zu dem Schluss kommen, JavaScript pauschal zu deaktivieren. Dies bedeutet aber, dass viele Sites nur noch sehr umständlich und unschön benutzbar wären, manche andere gar gänzlich ausfallen. Von daher wird man in vielen Fällen nicht dauerhaft auf JavaScript verzichten wollen.

Als Ausweg bieten sich zwei wichtige Mechanismen an:

- Der verwendete Browser sollte stets auf dem aktuellen Stand gehalten werden, damit bekannte Sicherheitslücken geschlossen werden. Viele bieten inzwischen einen automatisierten Update-Mechanismus an, der dies deutlich erleichtert.
- Die meisten Angriffe beziehen sich natürlich auf die am stärksten verbreiteten Browser, insbesondere den InternetExplorer. Die Gefahr eines erfolgreichen Angriffs sinkt stark, wenn ein weniger verbreiteter Browser verwendet wird.

### 15.9.5 Destruktives JavaScript

Das nachfolgende JavaScript in Listing 15.2 zeigt exemplarisch, wie einfach ein *destruktives JavaScript* geschrieben werden kann. Dieses ruft sich selbst jeweils rekursiv auf, allerdings immer in einem neuen Fenster. Damit entsteht ein einfacher DOS-Angriff.

### 15.9.6 Ein sehr nützliches JavaScript

Im Gegensatz zu dem abschreckenden Beispiel aus Abschnitt 15.9.5 gibt es natürlich zahllose ausgesprochen nützliche JavaScripts. Ein ganz einfaches wird von Google eingesetzt (Listing 15.1): Beim Aufrufen der Google-Hauptseite ist der Eingabefokus automatisch in das Eingabefeld gelegt. Hier wurde bereits die gleiche Technik beim „interaktiven Ratespiel mit Perl“ eingesetzt (Abschnitt 10.6.2.3): Das eigentliche Formular baut sich nach dem Gerüst

```
<BODY onload="self.focus(); document.formular.feld.focus()">

   <FORM method="..." action="..." name="formular">
       <INPUT name="feld">

       ...
```

**Listing 15.1:** Grundgerüst für benutzerfreundliches Web-Formular

auf; durch diesen JavaScript-Einsatz wird benutzerfreundlich der Eingabefokus automatisch beim Laden auf das Eingabefeld gelegt.

```
<HTML>
  <HEAD>
     <TITLE>WebKompendium: destruktives JavaScript</TITLE>
     <SCRIPT type="text/javascript">
        function hahaha()
        {
        var newWindow = window.open("./destruktiv.html","Fenster",
                                    "width=200,height=200");
        }
     </SCRIPT>
     <!-- das Script ruft rekursiv die Seite selbst wieder auf -->
  </HEAD>

  <BODY>
    <DIV ALIGN=CENTER>
      <H3>JavaScript ist etwas ganz besonderes!</H3>
    </DIV>

    <SCRIPT type="text/javascript">
       hahaha();
    </SCRIPT>

  </BODY>
</HTML>
```

**Listing 15.2:** Destruktives JavaScript

### 15.9.7 JavaScript beim Clienteinsatz und für den Diensteanbieter

Zusammenfassend sollen für den erfolgreichen Einsatz von JavaScript noch ein paar Hinweise gegeben werden.

Der Benutzer wird nicht dauerhaft auf JavaScript verzichten wollen. Hier sollte man sich der damit verbundenen Gefahren stets bewusst sein und seinen Browser auf dem neuesten Stand halten.

Für den Diensteanbieter ist die Situation wesentlich komplexer:

- Zunächst muss man beachten, dass nicht vorausgesetzt werden kann, dass bei allen Clients ein JavaScript in einer aktuellen Version ausgeführt werden kann. Dies wird allerdings noch je nach Zielgruppe deutlich variieren.
- Eine Site, die JavaScript verwendet, sollte prinzipiell auch ohne JavaScript benutzbar sein: JavaScript sollte stets bewusst eingesetzt werden.
- Jedes veröffentlichte JavaScript sollte gut – auch mit verschiedenen Browsertypen – getestet sein; ein nicht funktionsfähiges JavaScript ist eine sehr schlechte Werbung für die Site.

# 16

# Ajax

Eine der populärsten der neuen Techniken im Internet ist *Ajax*. Diese Abkürzung steht für „Asynchronous JavaScript And XML“ und bezeichnet eine Technologie, welche auf Jesse J. Garrett und die Firma AdaptivePath zurückgeht (Abbildung 16.1). Dabei ist Ajax keine an sich neue Technik, sondern nur ein geschickt zusammengefasstes Bündel bestehender Techniken, wobei die Kernfunktionalität auf JavaScript basiert. XML wird als Datenaustauschformat zwischen der Clientapplikation, welche durch das zentrale „Ajax-JavaScript-Objekt“ gesteuert wird, und einer Serverapplikation, verwendet – hier können aber auch andere Formate zum Einsatz kommen, etwa das einfachere, aber performante JSON (vgl. 15.8). Die Literatur zu Ajax ist inzwischen sehr umfassend, für einen weitergehenden Einstieg kann etwa [Wen07] verwendet werden.

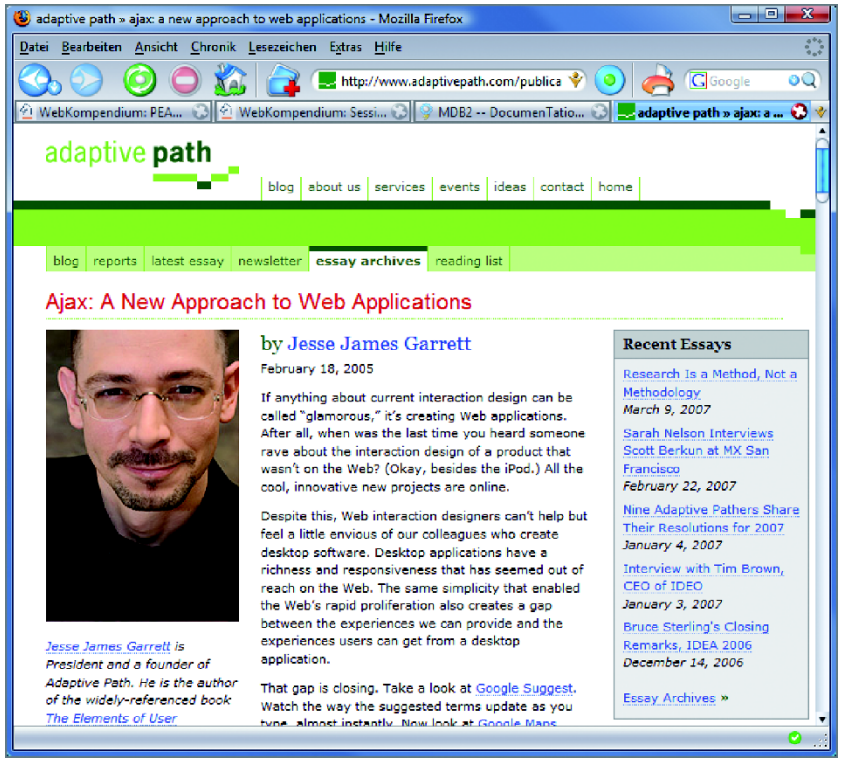



**Abbildung 16.1:** Initialartikel zu Ajax

## 16.1 Beispiele für Ajax

Ein erstes, prominentes Beispiel für Ajax findet sich unter `http://www.google.com/webhp?complete=1&hl=en` (Abbildung 16.2): Das Google-Suchfenster zeigt interaktiv bei der Eingabe des Suchbegriffes eine geschätzte Trefferzahl für die möglichen Vervollständigungen an. Zentral – und anders als die bisherigen Web-Applikationen – ist die interaktive Komponente dieser Technik, welche fortlaufend mit dem Webserver kommuniziert und damit das Verhalten einer klassischen Clientapplikation im Web bieten kann.

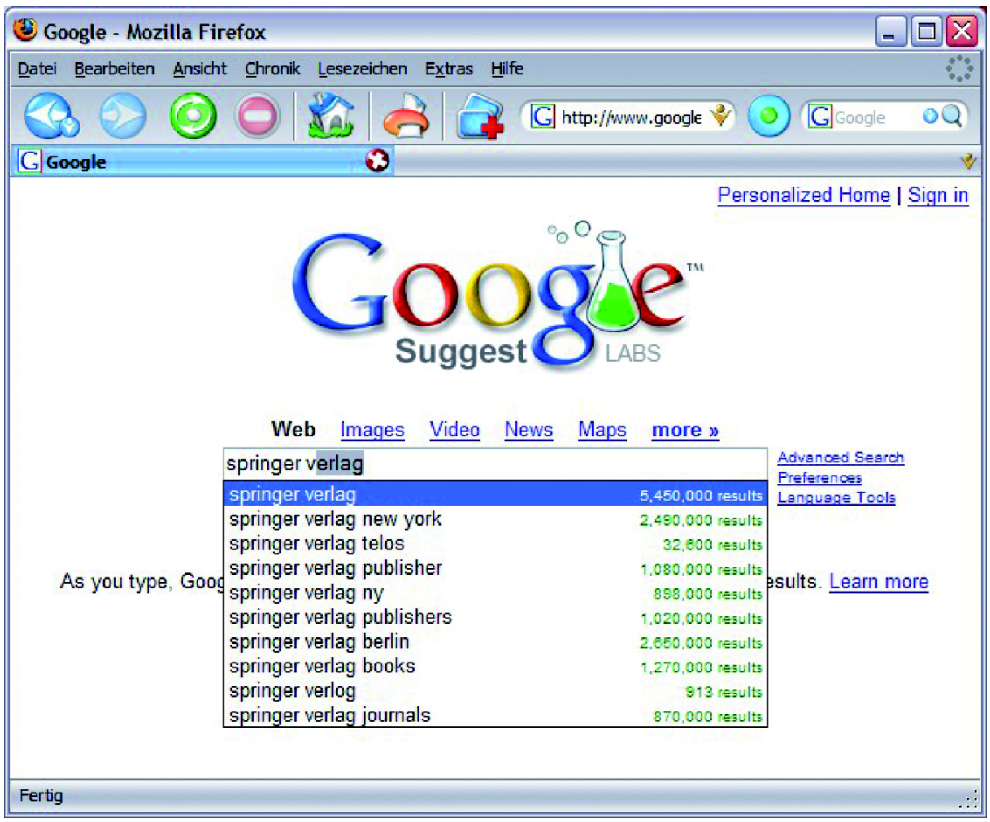



**Abbildung 16.2:** Ajax-Beispiel von Google

Unter `http://demo.neximage.com` findet sich ein weiteres Beispiel (Abbildung 16.3), welches Ajax anwendet, um eine interaktive Bildbearbeitung anzubieten; dabei wird interaktiv auf dem Client die Bilddatei verändert, und die eigentliche Bearbeitung erfolgt auf dem Server.

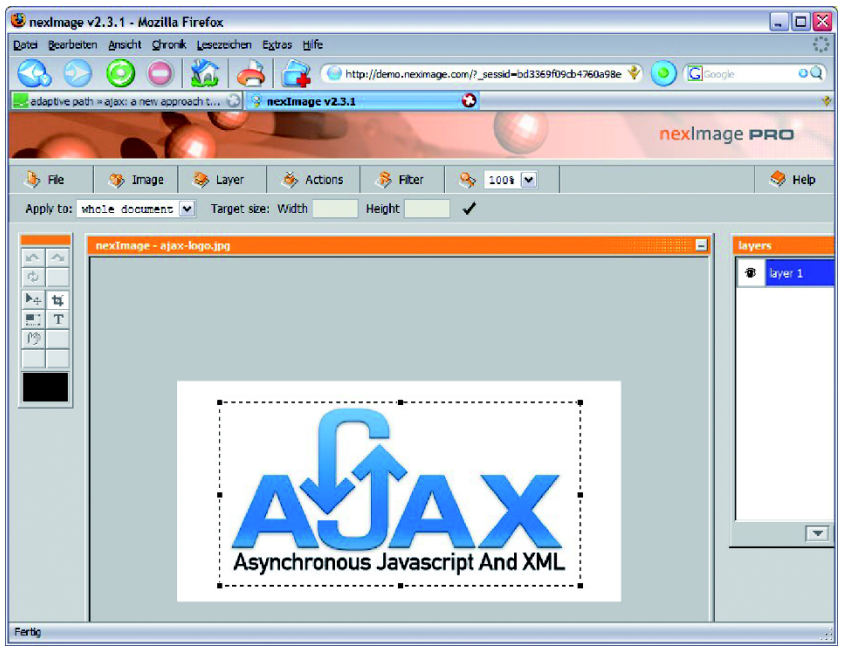



**Abbildung 16.3:**
Bildbearbeitung mit Ajax

## 16.2 Technische Grundlage für Ajax

Die Grundidee von Ajax, welche eine Interaktion für Web-Anwendungen ermöglicht, ist die Fähigkeit eines Browsers, im Hintergrund („asynchron“) einen HTTP-Request an einen Server zu schicken. Dies wurde mit der Version 5 des InternetExplorer eingeführt, allerdings innerhalb der Microsoft-Welt zunächst als proprietäre Lösung mittels ActiveX.

Die meisten Browser, auch der InternetExplorer ab der Version 7, verwenden das JavaScript-Object für Ajax:

```
XMLHttpRequest
```

Zu den Browsern, die dieses JavaScript-Objekt einsetzen können, gehören:

- Firefox ab Version 1.0;
- Netscape ab der Version 7.1;
- Safari ab der Version 1.2;
- Opera ab der Version 8;
- Konquerror ab der Version 3.

## 16.3 Entwicklungsumgebungen für Ajax

Um auf der Client-Seite Ajax-Applikationen zu entwickeln, wird eine Entwicklungsumgebung benötigt, welche JavaScript unterstützt. Hierfür ist Eclipse mit dem Adobe-Macromedia-Plugin für JavaScript, wie in 15.2.3 vorgestellt, ausreichend.

Speziell für Ajax liegt ein leistungsfähigeres Plugin für die Entwicklungsumgebung Eclipse (vgl. 8.1) vor: ATF, the Ajax Toolkit Framework [web].

Bei ATF handelt es sich um ein sehr umfassendes Plugin. Zunächst wird das eigentliche Plugin installiert. Anschließend sind einige Ergänzungen von Zusatzkomponenten von Hand vorzunehmen: die rico-Templates prototype.js und rico.js, das JavaScript-Framework dojo sowie jslint, eine Ergänzung von full-jslint. Anschließend ist ein Neustart von Eclipse mit Bereinigung des Workspaces durch `eclipse -clean` notwendig. Ajax-Projekte werden dann als ein static-Webprojekt angelegt.

Das ATS-Plugin stellt insbesondere die folgenden Eigenschaften bereit:

- einen umfassenden JavaScript-Support über das dojo-Toolkit;
- die Integration des Webservers: Projekte werden lokal entwickelt und auf dem eigenen Webserver veröffentlicht; anschließend können sie im Eclipse-integrierten Mozilla betrachtet werden.

**Abbildung 16.4:** dojo-Logo

Teil des ATS-Plugins ist, wie schon beschrieben, dojo [web], ein umfassendes JavaScript-Framework.

## 16.4 Ablauf einer Ajax-Anfrage

Eine *Ajax-Anfrage* unterscheidet sich von der klassischen Web-Anfrage, da nun der Browser JavaScript-gesteuert kontinuierlich an den Server weitere Anfragen senden kann; die Auswertung der Antworten erfolgt ebenfalls kontinuierlich und führt zum Neuladen eines Teiles der Browseranzeige (Abbildung 16.5).

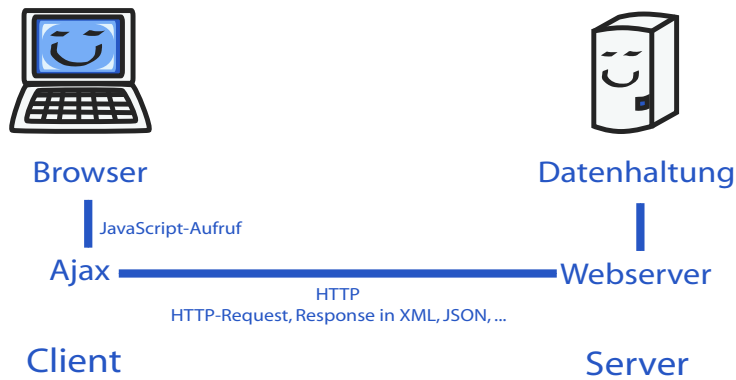



**Abbildung 16.5:** Prinzip einer Ajax-Anfrage

Im Einzelnen kann eine Ajax-Anfrage in drei Schritte untergliedert werden:

1. Erzeugung der Instanz von `XMLHttpRequest` im Client-Browser durch JavaScript:

   ```
   var http = new XMLHttpRequest();
   ```

   Nur für den InternetExplorer in der Version bis 6 ist dies entsprechend:

   ```
   var http = new ActiveXObject("Microsoft.XMLHTTP");
   ```

   Beide Wege können auch zusammengeführt werden, so dass ein vom Browser unabhängiger JavaScript-Code entsteht:

```
<script type="text/javascript">
var http;
if (window.XMLHttpRequest)
   http = new XMLHttpRequest();
else {
       if (window.ActiveXObject)
          http = new ActiveXObject("Microsoft.XMLHTTP");
      }
...

</script>
```

2. Im zweiten Schritt stellt das Ajax-Objekt eine HTTP-Verbindung zum Server her:
   - Anwendung der Methode `open(...)` der Klasse `XMLHttpRequest`
   - Parameter von `open`:
     - die HTTP-Methode: GET/ oder POST;
     - die Server-URL;
     - Art der Verbindung: synchron (false), asynchron (true);
       synchron: Script-Ausführung auf dem Client wird angehalten, bis die Serverantwort vorliegt;
       asynchron: Client-Script wartet nicht auf die Server-Antwort (Default-Fall)
       (in diesem Fall: Eigenschaft http.onreadystatechange verwaltet Ereignis „Änderung des Zustands der XMLHttpRequest-Instanz“).
3. Der letzte Schritt ist das Anzeigen des Ergebnisses:
   - Die JavaScript-Eigenschaft `readyState` und das Ereignishandle `onreadystatechange` verwalten das Eintreten einer Antwort, genauer die Änderung des Antwortzustandes: Beim Ändern des Antwortzustandes tritt das Ereignis `onreadystatechange` ein; dann kann über den Wert der Variablen `readyState` die Art des Antwortzustandes abgefragt werden. Dabei sind die folgenden fünf Fälle möglich:

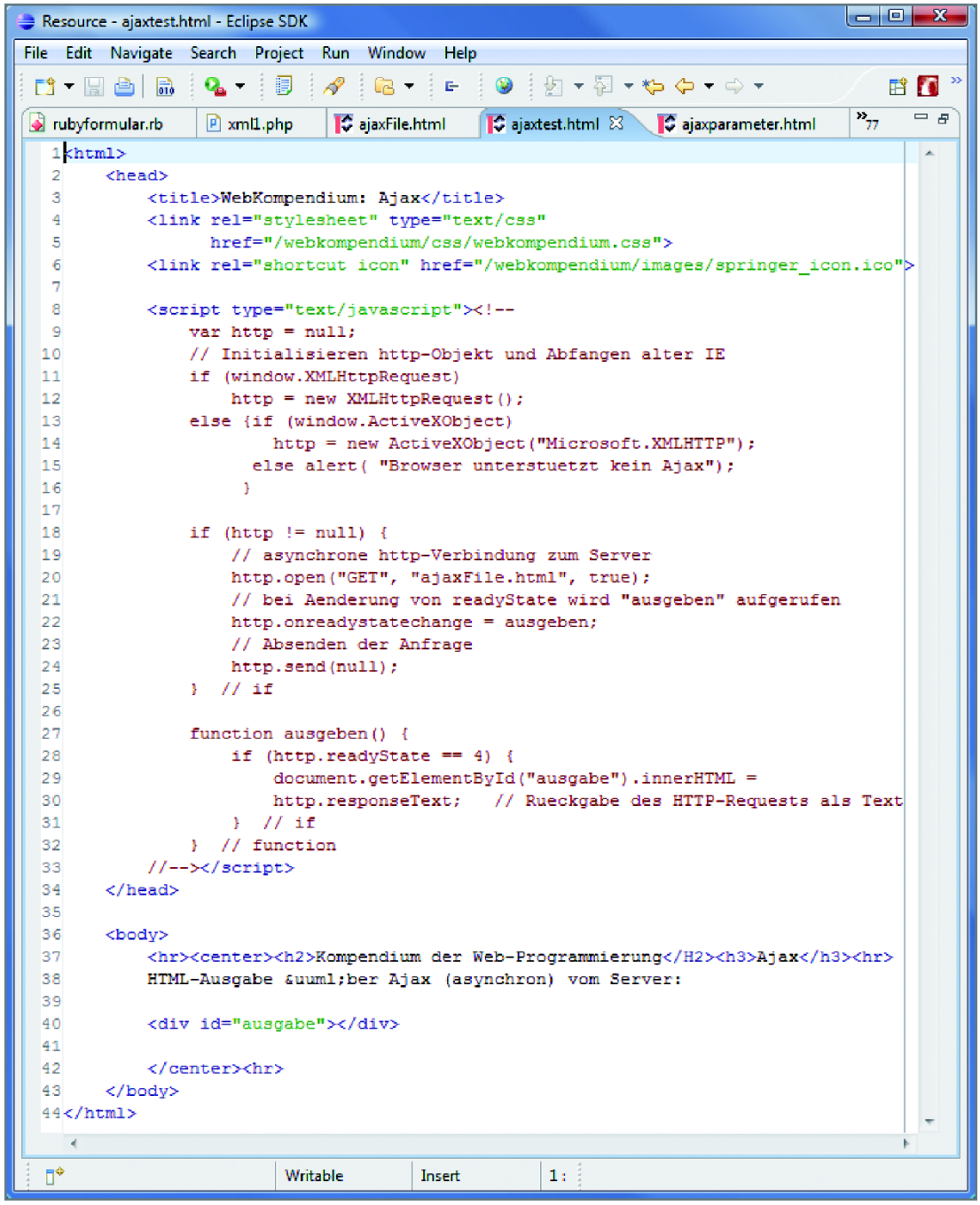
     0: nicht initialisiert;
     1: Laden;
     2: fertig geladen;
     3: Warten;
     4: beendet.
   - Im Erfolgsfall `readyState == 4` enthält die Eigenschaft `responseText` die Rückgabe des abgefragten Servers.

## 16.5 Beispiele für Ajax

Drei Beispiele sollen die beschriebene Arbeitsweise von Ajax verdeutlichen.

### 16.5.1 Beispiel: Dateinachladen mit Ajax

Eine Web-Site verwendet das XMLHttpRequest-Objekt, um durch eine asynchrone Anfrage an den Webserver eine Textinformation – die Datei `ajaxFile.html`, Abbildung 16.7 – anzuzeigen. Die Ajax-Datei `ajaxtest.html` erzeugt – je nach Browser – das JavaScript-Objekt oder das ActiveX-Control;




```
<html>
    <head>
        <title>WebKompendium: Ajax</title>
        <link rel="stylesheet" type="text/css"
              href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">

        <script type="text/javascript"><!--
            var http = null;
            // Initialisieren http-Objekt und Abfangen alter IE
            if (window.XMLHttpRequest)
                http = new XMLHttpRequest();
            else {if (window.ActiveXObject)
                    http = new ActiveXObject("Microsoft.XMLHTTP");
                  else alert( "Browser unterstuetzt kein Ajax");
                 }

            if (http != null) {
                // asynchrone http-Verbindung zum Server
                http.open("GET", "ajaxFile.html", true);
                // bei Aenderung von readyState wird "ausgeben" aufgerufen
                http.onreadystatechange = ausgeben;
                // Absenden der Anfrage
                http.send(null);
            }  // if

            function ausgeben() {
                if (http.readyState == 4) {
                    document.getElementById("ausgabe").innerHTML =
                    http.responseText;   // Rueckgabe des HTTP-Requests als Text
                }  // if
            }  // function
        //--></script>
    </head>

    <body>
        <hr><center><h2>Kompendium der Web-Programmierung</H2><h3>Ajax</h3><hr>
        HTML-Ausgabe &uuml;ber Ajax (asynchron) vom Server:

        <div id="ausgabe"></div>

        </center><hr>
    </body>
</html>
```

**Abbildung 16.6:**
Die HTML-Datei `ajaxtest.html` in JSEclipse

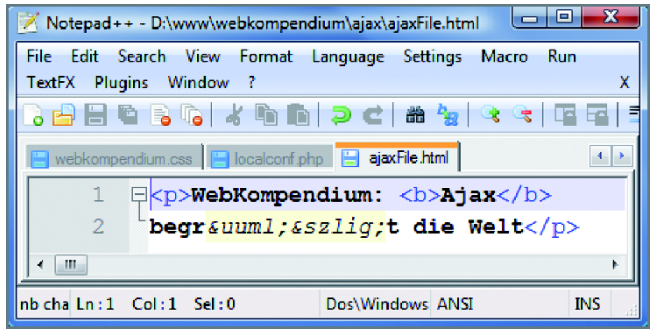



**Abbildung 16.7:**
Die mittels Ajax nachgeladene Datei HTML-Datei `ajaxFile.html`

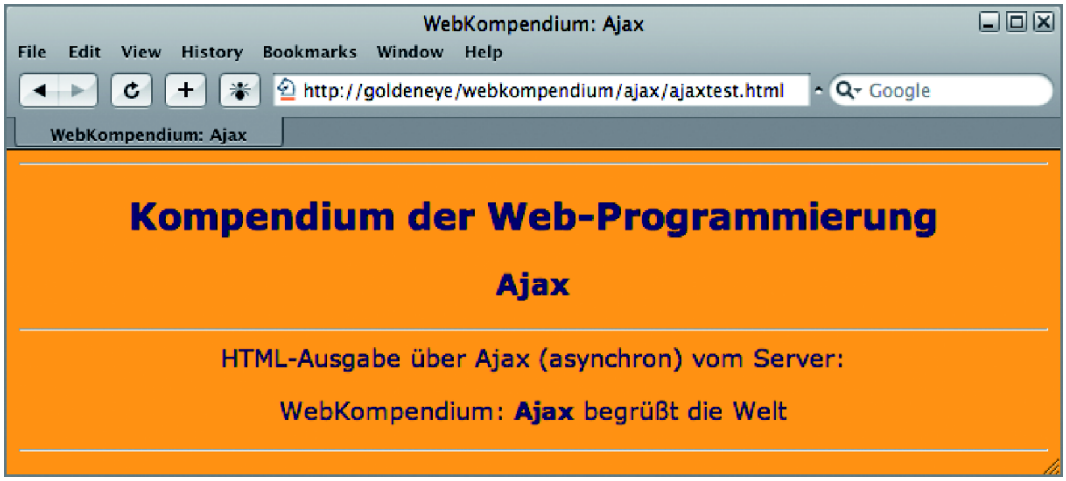
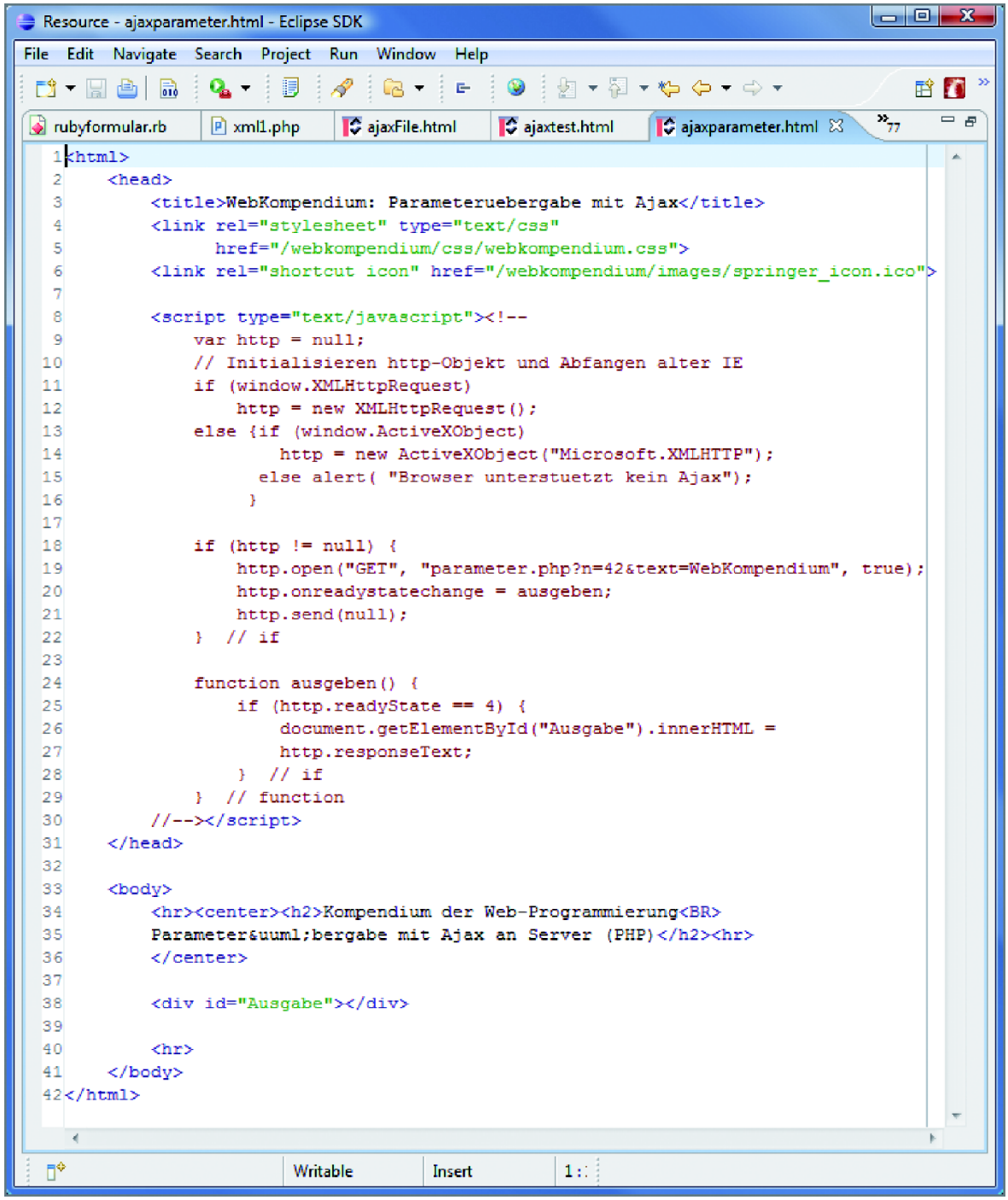



**Abbildung 16.8:**
Ausgabe von `ajaxtest.html` in Safari für Windows

### 16.5.2 Beispiel: Parameterübergabe mit Ajax

Ein zweites Beispiel soll die Übergabe von Parametern aus Ajax zeigen (Datei `ajaxparameter.html`, Abbildung 16.9). Hierbei sendet das Ajax-Objekt einen HTTP-GET-Request an das PHP-Script `parameter.php`, welches alle übertragenen Parameter ausgibt (Abbildung 16.10). Als Ergebnis entsteht die Ausgabe von Abbildung 16.11.




```
<html>
    <head>
        <title>WebKompendium: Parameteruebergabe mit Ajax</title>
        <link rel="stylesheet" type="text/css"
              href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">

        <script type="text/javascript"><!--
            var http = null;
            // Initialisieren http-Objekt und Abfangen alter IE
            if (window.XMLHttpRequest)
                http = new XMLHttpRequest();
            else {if (window.ActiveXObject)
                    http = new ActiveXObject("Microsoft.XMLHTTP");
                  else alert( "Browser unterstuetzt kein Ajax");
                 }

            if (http != null) {
                http.open("GET", "parameter.php?n=42&text=WebKompendium", true);
                http.onreadystatechange = ausgeben;
                http.send(null);
            }  // if

            function ausgeben() {
                if (http.readyState == 4) {
                    document.getElementById("Ausgabe").innerHTML =
                    http.responseText;
                }  // if
            }  // function
        //--></script>
    </head>

    <body>
        <hr><center><h2>Kompendium der Web-Programmierung<BR>
        Parameter&uuml;bergabe mit Ajax an Server (PHP)</h2><hr>
        </center>

        <div id="Ausgabe"></div>

        <hr>
    </body>
</html>
```

**Abbildung 16.9:**
Die HTML-Ajax-Datei `ajaxparameter.html`

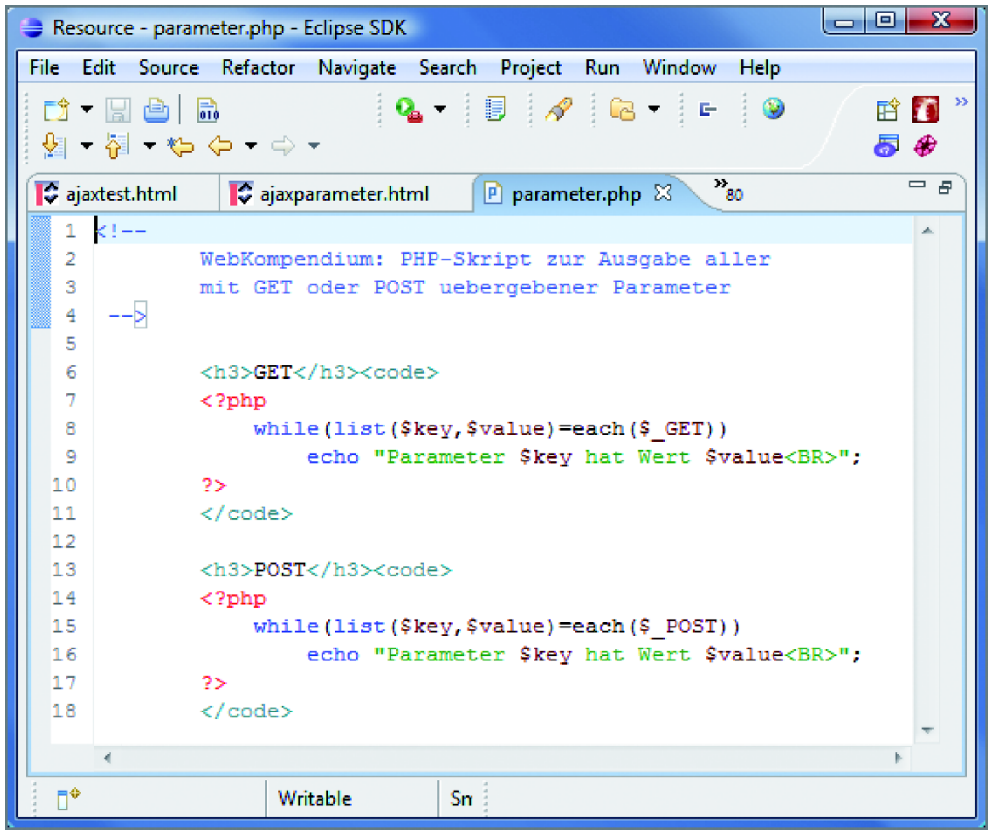



**Abbildung 16.10:**
Die mittels Ajax nachgeladene Datei HTML-Datei `ajaxFile.html`

**Abbildung 16.11:**
Ausgabe von `ajaxparameter.html`

Das Beispiel `ajaxparameter.html` überträgt die Parameter mit der HTTP-Methode GET; um POST anwenden zu können, ist ein komplexeres Vorgehen notwendig: Hier muss die Methode `setRequestHeader()` verwendet werden. Die Ajax-Datei `ajaxparameterpost.html`, Abbildung 16.12, wendet diese Methode für die Übertragung mit POST an das gleiche PHP-Script an.

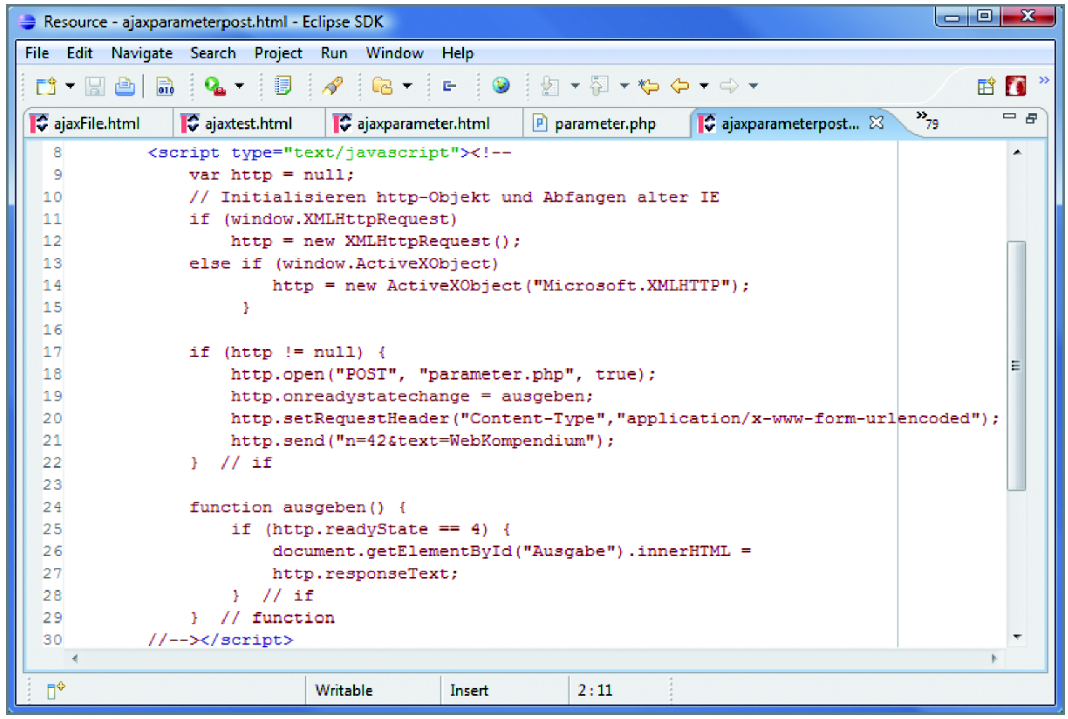



**Abbildung 16.12:** Übertragung mittels POST (`ajaxparameterpost.html`)

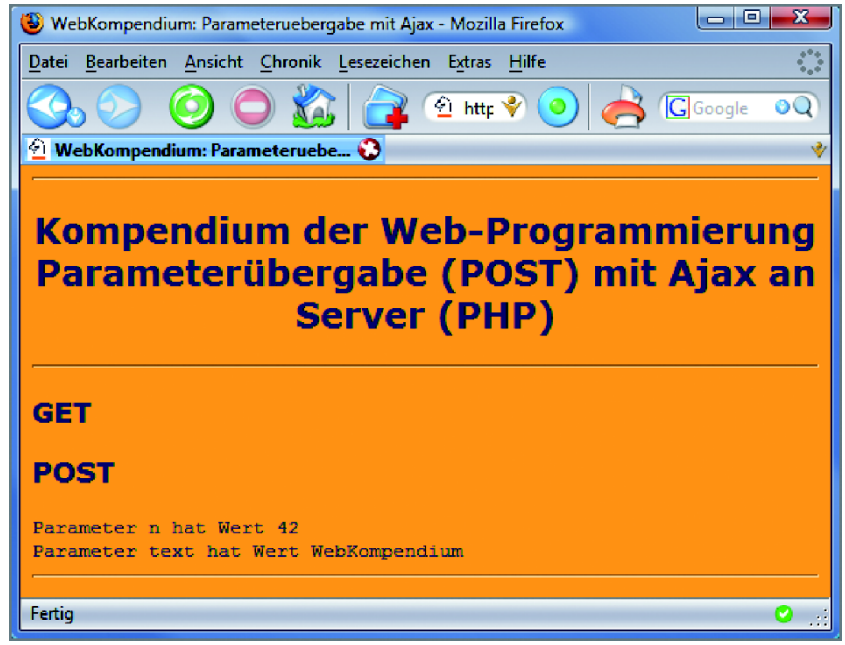



**Abbildung 16.13:** Ausgabe von `ajaxparameterpost.html`

### 16.5.3 Beispiel: Beenden von Ajax

Ein anderes Problem stellt das *Beenden* von Ajax dar; hierfür verfügt das `XMLHttpRequest`-Objekt über die Methode `abort`.

Zur Verdeutlichung wird das Server-Script `langsam.php` verwendet; dieses liefert – allerdings nach einer zufällig gesteuerten Verzögerung bis zu zehn

Sekunden – eine einfache Textrückgabe (Abbildung 16.14). Das Ajax-Script `ajaxabbruch.html` (Abbildung 16.15) lädt dieses nach, wartet dazu aber maximal fünf Sekunden. Erfolgt bis dahin keine Rückmeldung vom Serverscript, wird das `XMLHttpRequest`-Objekt beendet und eine Standard-Ausgabe erzeugt (Methode `abbrechen`).[1]

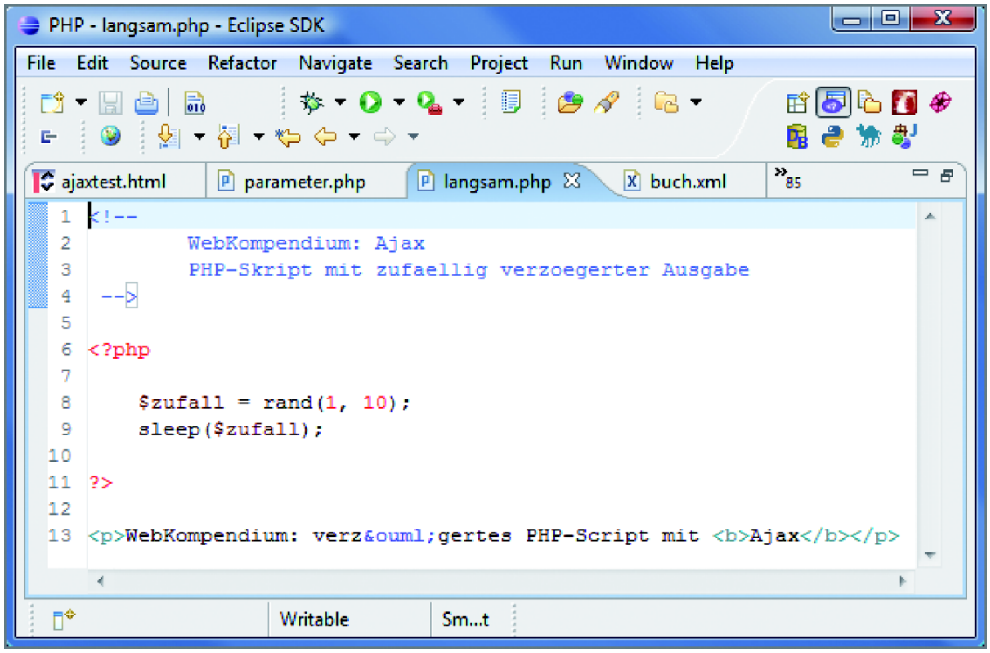
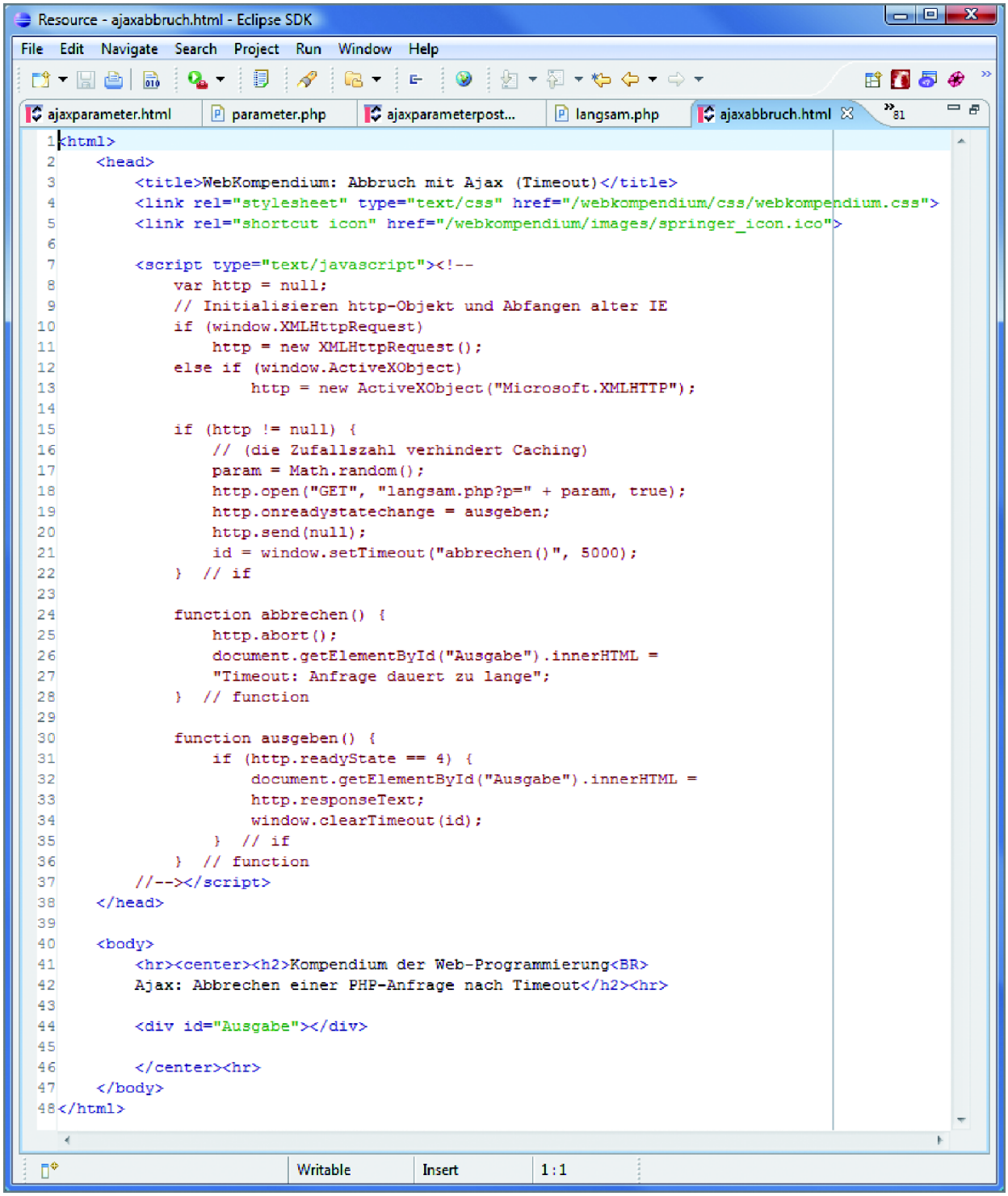



**Abbildung 16.14:**
Verzögertes PHP-Serverscript (`langsam.php`)

[1] die URL der `open`-Methode verwendet dabei einen Zufallszahlenparameter, um ein Caching zu verhindern.




```
<html>
    <head>
        <title>WebKompendium: Abbruch mit Ajax (Timeout)</title>
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">

        <script type="text/javascript"><!--
            var http = null;
            // Initialisieren http-Objekt und Abfangen alter IE
            if (window.XMLHttpRequest)
                http = new XMLHttpRequest();
            else if (window.ActiveXObject)
                    http = new ActiveXObject("Microsoft.XMLHTTP");

            if (http != null) {
                // (die Zufallszahl verhindert Caching)
                param = Math.random();
                http.open("GET", "langsam.php?p=" + param, true);
                http.onreadystatechange = ausgeben;
                http.send(null);
                id = window.setTimeout("abbrechen()", 5000);
            }  // if

            function abbrechen() {
                http.abort();
                document.getElementById("Ausgabe").innerHTML =
                "Timeout: Anfrage dauert zu lange";
            }  // function

            function ausgeben() {
                if (http.readyState == 4) {
                    document.getElementById("Ausgabe").innerHTML =
                    http.responseText;
                    window.clearTimeout(id);
                }  // if
            }  // function
        //--></script>
    </head>

    <body>
        <hr><center><h2>Kompendium der Web-Programmierung<BR>
        Ajax: Abbrechen einer PHP-Anfrage nach Timeout</h2><hr>

        <div id="Ausgabe"></div>

        </center><hr>
    </body>
</html>
```

**Abbildung 16.15:**
Die Ajax-Datei `ajaxabbruch.html`

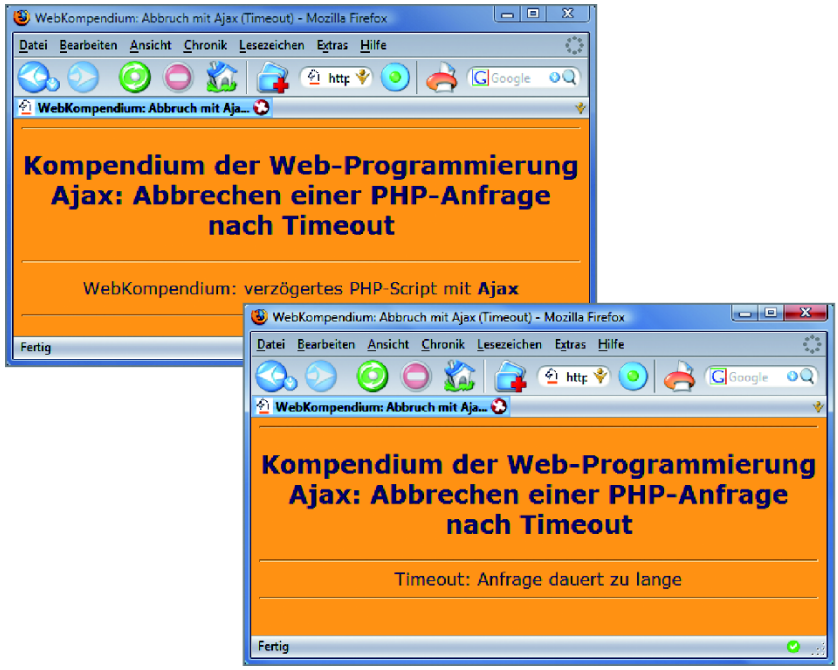



**Abbildung 16.16:**
Ausführen von `ajaxabbruch.html` ohne und mit Abbruch

## 16.6 Das XMLHttpRequest-Objekt

Der „Motor“ des Ajax-Mechanismus ist die zentrale JavaScript-Klasse `XMLHttpRequest`, weshalb diese noch genauer betrachtet werden soll.

Die Erzeugung einer Instanz der entsprechenden JavaScript-Klasse geschieht durch

```
var http = new XMLHttpRequest();
```

Auf diese Instanz der Klasse `XMLHttpRequest` können zahlreiche Methoden angewendet werden, darunter:

- `abort` Abbruch der asynchronen Ajax-Operation (s.o.);
- `getAllResponseHeaders` HTTP-Headerinformationen;
- `getResponseHeader` HTTP-Headerinformationen;
- `open` öffnet eine URL (s.o.);
- `send` Ajax-Übertragung (s.o.);

Ebenso verfügt die Instanz der Klasse `XMLHttpRequest` über Attribute wie:

- `onreadystatechange` Ereignis bei Änderung des Antwortzustands (s.o.);
- `readyState` Zustand der Antwort (s.o.);
- `responseText` Antwort in Textdarstellung (s.o.);
- `responseXML` Antwort als XML-Struktur;
- `status` der HTTP-Statuscode der Antwort als numerischer Wert;
- `statusText` der HTTP-Statuscode in Textdarstellung.

## 16.7 Zusammenfassung: Vorteile und Probleme von Ajax

Ajax ist sicherlich eine der spannendsten neuen Techniken der Web-Programmierung. Die Technik selbst ist nicht revolutionär neu, ihr kommt aber eine besondere Bedeutung zu, da sie wichtige neue Möglichkeiten für Web-Anwendungen eröffnet: mit Ajax werden interaktive, desktopähnliche Applikationen im Web möglich, ohne die Nachteile einer Applet-Entwicklung oder von Flash in Kauf zu nehmen (vgl. Kapitel 17 und 18). Dabei ist Ajax clientseitig an JavaScript gebunden; serverseitig können beliebige Techniken verwendet werden.

Zu den wichtigen Vorteilen gehört auch eine Netzwerkentlastung und eine Performance-Steigerung, da nicht bei jedem Request eine komplette Seite neu geladen werden muss; der Datenverkehr zwischen Server und Client wird deutlich geringer. Clientseitig ist reines JavaScript für Ajax ausreichend, es werden keine Browser-Plugins benötigt, ein weiterer Vorteil.

Im Kern erweitert Ajax den Begriff des Webs wesentlich:

**Eine Ajax-Site ist keine zustandslose Web-Site mehr!**

Aus dieser Tatsache resultieren aber auch nach dem momentanen Verständnis des Webs Nachteile; dazu zählen ein mitunter unübliches Verhalten des „Backbuttons“ des Browsers, Ajax-Anwendungen können nicht per Lesezeichen in einem speziellen Zustand abgespeichert werden und Suchmaschinen haben Schwierigkeiten mit Ajax.

Die Möglichkeiten von Ajax stehen momentan noch am Anfang. Viele Anwendungen beginnen momentan, Ajax verstärkt einzusetzen, etwa das beliebte Content Management System TYPO3 für zentrale Backend-Funktionen (vgl. Kapitel 28.2). Insgesamt führt dies alles zu einem „interaktiven Web“: Ajax ist eine Kerntechnologie des Web 2.0.

# Adobe Flash

*Flash* ist eine clientbasierte Softwaretechnik für den Einsatz im Web, welche einen besonderen Bereich abdeckt: die Animation. Schöne grafische Effekte bis hin zum bekannten „Flash-Intro“, einer kleinen Filmsequenz, werden damit möglich. Im gemischten Einsatz mit einem Server kommt Flash beim Streaming zum Einsatz. Eine weitere wichtige Domäne von Flash auf dem Client ist die Einbindung von Audio. Exemplarisch für alle diese Anwendungen sei unter `http://www.dradio.de` das „Flash-Radio“ von Deutschlandfunk genannt (Abbildung 17.1).

Dabei ist von vornherein Flash als Technik für das Web konzipiert, weshalb die gerade bei Filmtechniken wichtige Frage der Dateigrößen bzw. der Datenkomprimierung im Mittelpunkt der Technik stehen.

Es liegt zahllose Literatur zu Flash vor, für die Zielsetzung „Film mit Flash“ ist insbesondere [PR07] empfehlenswert.

**Abbildung 17.1:**
Flash-basiertes Streaming „à la carte“ des Deutschlandfunks

## 17.1 Das Prinzip von Flash

Flash ist ein proprietäres Produkt, ursprünglich von Macromedia, heute von Adobe vertrieben. Es war zunächst als „Shockwave Flash Player“ von Macro-

medias Director-Software bekannt, heute ist der Flash Player ein eigenständiges, sehr verbreitetes Produkt.

Ursprünglich ist Flash im gewissen Sinne als Nachfolger von animated gif für die Darstellung eines Intros als reines Animationswerkzeug konzipiert. Seit der Version 4 bietet Flash aber auch eine zugehörige Programmierung mit ActionScript, so dass auch einfache clientbasierte Anwendungen in Flash möglich sind.

Flash basiert auf dem swf-Format, „Small Web Format“, einem vektorbasierten Grafikformat.[1]

Zum „Abspielen“ von Flash-Animationen im Browser braucht dieser den Flash-Player, welcher – inzwischen auch für Linux – frei verfügbar ist. Einige Browser werden direkt mit diesem Plugin ausgeliefert, ansonsten muss es einmalig installiert werden. Adobe gibt an, dass circa 95 % der Browser über den Player verfügen, es ist also von einer sehr großen Verbreitung auszugehen.

Die Flash-Entwicklungsumgebung verwendet eine Vielzahl von Dateiformaten; die wichtigsten sind das fertige („compilierte“) Flash-Format `swf` und die Flash-Sourcedatei `fla`.

### 17.1.1 Die Flash-Entwicklungsumgebung

Zentraler Bestandteil von Flash ist die Entwicklungsumgebung nach Abbildung 17.2. Diese ist vom Aufbau vielen Filmbearbeitungsprogrammen recht ähnlich: Das zentrale „Filmfenster“, die Flash-Bühne, zeigt in WYSIWYG-Art die entstehende Animation, die Zeitleiste darüber steuert den temporären Ablauf und die einzelnen Effekte der Animation; eine leichte Ähnlichkeit zum Dreamweaver-Webeditor nach Abbildung 8.5) ist erkennbar. Daneben sind die Werkzeugleiste und weitere Arbeitsbereiche vorhanden.

[1] eine freie Alternative zu swf ist das bisher noch nicht so verbreitete SVG in Verbindung mit der Auszeichnungssprache SMIL.

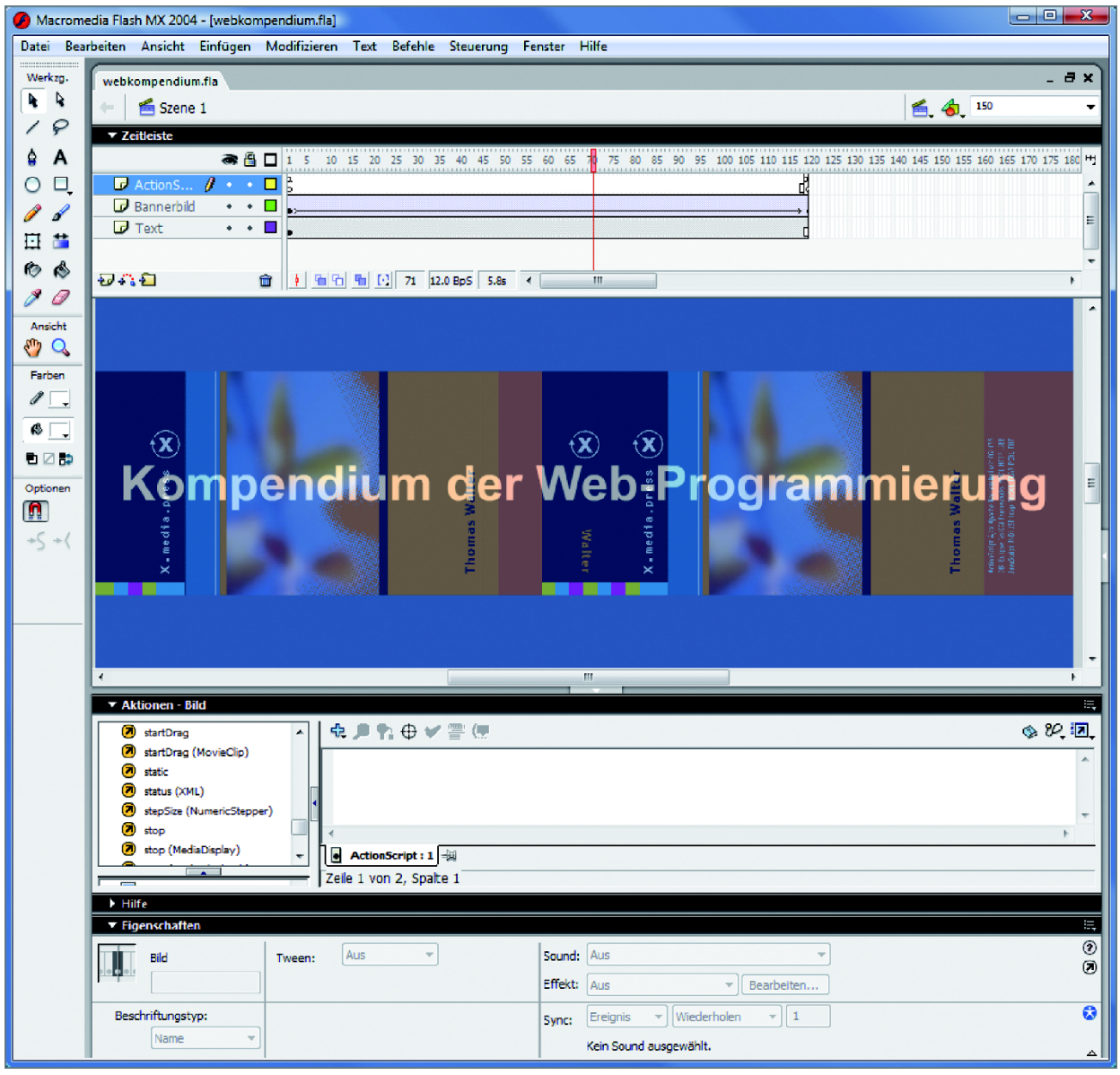



**Abbildung 17.2:**
Die Flash-Entwicklungsumgebung: Beispiel `webkompendium.fla`

Der zentrale Bereich der Flash-Entwicklungsumgebung ist die Bühne; dies ist derjenige Teil, den letztlich der Betrachter im Browser sehen wird. Hier werden die für die Flash-Animation benötigten Objekte abgelegt, wobei Flash auch das Konzept der übereinanderliegenden Ebenen und Masken, ähnlich zu Bildbearbeitungen wie Photoshop, verwendet.

Für ein neu erstelltes Flash-Dokument sind im ersten Schritt grundlegende Eigenschaften festzulegen, darunter die Größe des Bühnenbereichs (im Entwurf kann der Platz darum auch genutzt werden, im fertigen Produkt aber nicht mehr), die Hintergrundfarbe und vorallem die Bildrate für den fertigen Film; hier ist der Default-Wert von 12 Bildern pro Sekunde meistens ausreichend.

Die Zeitleiste oberhalb der Bühne zeigt den in der Bühne angezeigten Zeitpunkt an, dieser kann per Maus einfach verschoben werden. Sie ist in die einzelnen Bilder (Frames) unterteilt, die Spielzeit ergibt sich durch Division durch die gewählte Bildrate. In Abbildung 17.2 ist beispielsweise das 71. Bild gewählt, was bei 12 Bildern pro Sekunde die Anzeige nach 5,8 Sekunden be-

stimmt, wie es unter der Zeitleiste angezeigt wird. Über die Zeitleiste werden auch die Ebenen verwaltet.

Zentral für Flash sind ferner die Begriffe „Bild“ und „Schlüsselbild“ (Key-Frame): Ein Bild ist eine unveränderliche Anzeige (ein Rechteck im Icon der Zeitleiste), während Schlüsselbilder den Anfang und das Ende einer Animation bilden (ein Kreis im Icon der Zeitleiste) und durch eine Linie für die Animation verbunden sind (in der Zeitleiste in Abbildung 17.2 in der zweiten Ebene „Bannerbild“ zu sehen).

Der Begriff der Szene in Flash hilft bei größeren Animationen zur Untergliederung der Entwicklung. Für die einzelne Animationen ist dabei noch der Begriff der „Tween-Animation“ nützlich: Hier werden das erste und letzte Schlüsselbild erstellt und Flash interpoliert den Übergang.

Das fertige Projekt ist abschließend zu veröffentlichen. Hierfür steht Datei|Exportieren|Film exportieren zur Verfügung, was direkt zur swf-Datei führt. Abbildung 17.3 zeigt die dabei wählbaren Optionen.

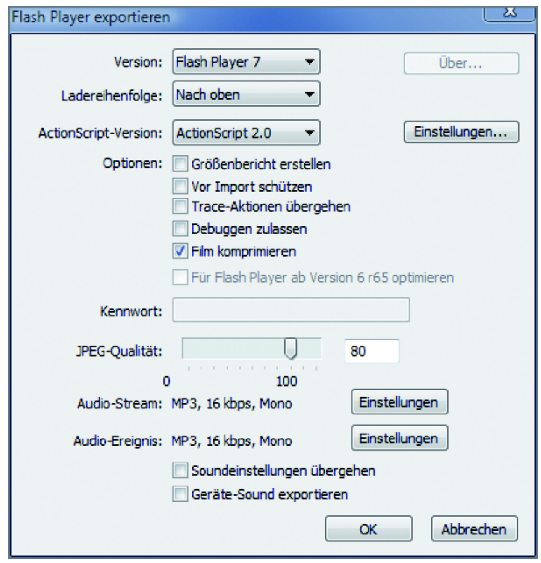



**Abbildung 17.3:**
Optionen für den swf-Export

#### 17.1.1.1 Ein einfaches Beispiel

Die Flash-Datei `webkompendium.fla` enthät eine einfache Animation in 120 Bildern bei 12 Bildern pro Sekunde, also 10 Sekunden Spielzeit (Abbildung 17.2).

Als Ausgangsbild dient, vor einem Hintergrund der Größe 600 × 300 Pixel im Farbton dieses Buches, die Covergrafik; innerhalb der 10 Sekunden wird diese Ausgeblendet: Start- und Endpunkt sind die Schlüsselbilder, am Ende ohne Transparenz (Alpha-Kanal mit 0%); zwischen beiden wird mittels Tween-Animation interpoliert. Dies alles läuft in der mittleren Ebene ab. Die darunterliegende enthält einen Schriftzug, der mit verblassen der oberen Schicht sichtbar wird. Die oberste Ebene enthält lediglich die ActionScript-Anweisung `stop()` in Frame 120, damit das Script nicht in einer Endlosschleife läuft.

In `flash.html` wird der erzeugte swf-Film in eine HTML-Seite eingebunden (Abbildung 17.4), das Ergebnis im Browser ist die Animation; eine Detailaufnahme zeigt Abbildung 17.5.

### 17.1.2 Einbinden von Flash in HTML

Nachdem eine swf-Datei erzeugt wurde, muss diese noch so in HTML eingebunden werden, dass der Flash-Player des Browsers die Datei ausführt; hierfür stehen mehrere Möglichkeiten bereit:

- über das `<embed>`-Tag:

```
<embed src="name_der_flashdatei.swf" width="600" height="300"
       play="true" loop="true" quality="high"
       type="application/x-shockwaveflash"></embed>
```

- über das `<object>`-Tag:

```
<object type="application/x-shockwave-flash"
     data="name_der_flashdatei.swf" id="name_des flashbereichs"
     width="600" height="300">
    <param name="movie" value="name_der_flashdatei.swf">
    <param name="bgcolor" value ="#000000">
    <param name="quality" value ="high">
    <param name="loop" value="true">
    <param name="menu" value="false">
<!--
     HTML-Code für Hinweis, wenn Flash-Plugin fehlt
-->
</object>
```

Die Parameter `<param>` legen das Verhalten des Flash-Players fest, etwa ob die Animation in einer Endlosschleife läuft.

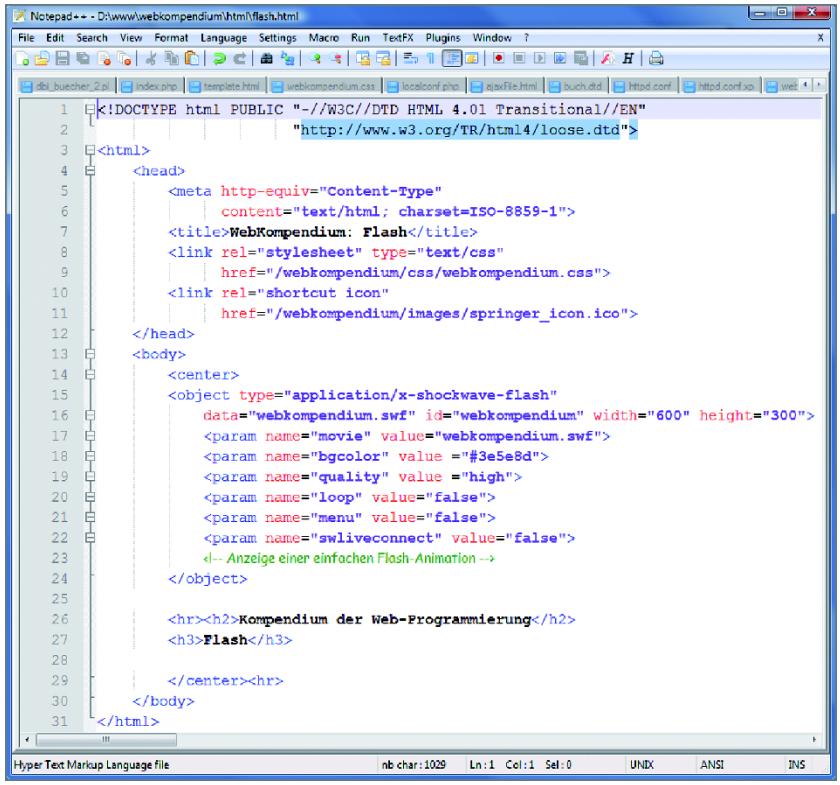



**Abbildung 17.4:**
Einbinden von `webkompendium.fla` in eine Webseite (`flash.html`)

Danach steht die Flash-Animation prinzipiell für die Anwendung bereit.

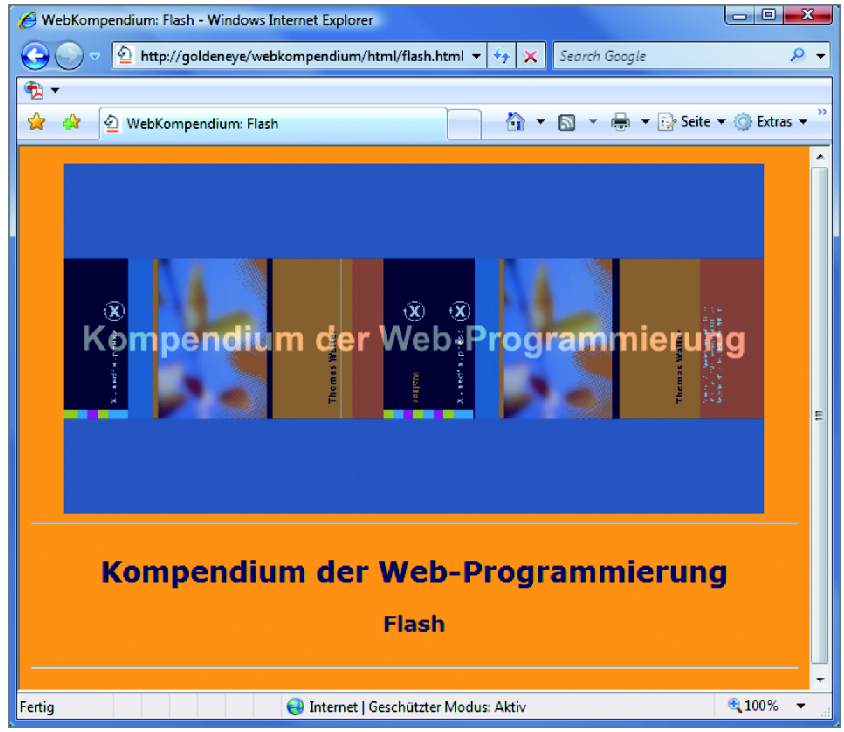



**Abbildung 17.5:** `flash.html` im Browser

## 17.2 ActionScript

*ActionScript* erweitert die Möglichkeiten von Flash wesentlich, da hiermit die Animation mit einer Programmierung verbunden werden kann. So können beispielsweise Eingaben erfolgen und in deren Abhängigkeit die weitere Flash-Animation verzweigen.

ActionScript ist eine objektorientierte Sprache; die einzelnen Flash-Elemente sind Instanzen entsprechender Klassen.

Mit ActionScript lassen sich auch komplexere Softwarearchitekturen realisieren: Beispielsweise ist es damit möglich, vom Flash-Player aus eine Serverapplikation über HTTP abzufragen, welche eine Datenbankanbindung implementiert; auf diesem Weg können Flash-Anwendungen mit Datenbanken verbunden werden.

## 17.3 Probleme von Flash

Flash ist eine mächtige Technik für Animationen im Web. Allerdings ist Flash auch mit zahlreichen Problemen verbunden, die teilweise sehr ähnlich zu denen der Java Applets sind (vgl. 18.4):

- zunächst handelt es sich bei Flash um einen nicht-offenen, proprietären Standard;
- der Flash-Player muss auf dem Client vorhanden sein;
- es muss auf dem Client auch die passende Version des Flash-Players vorhanden sein, wobei neuere Player gut alte swf-Versionen wiedergeben können;

- die Ausführung von Flash verzögert durch das Laden der Animation und das Starten des Players die Web-Site;
- Flash ist für Suchmaschinen schlecht, da diese die Information aus dem Grafikformat nicht erkennen können, allerdings arbeitet Google intensiv an Möglichkeiten zur Indizierung von swf-Informationen;
- für barrierefreie Webseiten (vgl. 2.6) ist Flash ungeeignet;
- der vielleicht größte Nachteil: Flash-Seiten sind im Allgemeinen langsam und schwer zu navigieren (der Backbutton des Browsers hat keine klare Funktion mehr, ähnlich zur Situation bei Ajax).

Dennoch sind insgesamt die Nachteile nicht so gravierend wie beim Applet, Flash wird insbesondere im kommerziellen Bereich häufig eingesetzt, auch wenn seine Verbreitung aktuell etwas zurück geht: die vor einiger Zeit zu beobachtende Tendenz, eine komplette Site mit Flash zu gestalten, ist heute nicht mehr erkennbar, Flash wird mehr bewusst als Gestaltungselement innerhalb einer Site genutzt, welche prinzipiell auch ohne Flash verwendbar wird. Eher wird dann Flash für einen separaten Werbe-Teaser verwendet, welcher als getrenntes Element reines Flash bietet, aber keine Navigation enthält.

## 17.4 Alternativen zu Flash

Es gibt natürlich auch einige Alternativen zu Flash, allerdings hat sich bisher keine nennenswert durchgesetzt:

- es gibt einige kommerzielle Protokolle für Filmformate/Filmstreaming im Web, etwa QuickTime von Apple;
- prinzipiell lassen sich die Möglichkeiten von Flash auch mit Java Applets erreichen (siehe Kapitel 18); Vorteil hiervon ist ein offener Standard, allerdings treten auch zahlreiche Nachteile auf;
- für PHP bietet das Open Source-Modul Ming die Möglichkeit, eine Ausgabe im swf-Format zu erzeugen;
- ebenso Open Source ist das Projekt Gnash (GNU Flash), welches künftig einen ähnlichen Funktionsumfang wie der Flash-Player bieten soll, aber noch nicht weit entwickelt ist; das proprietäre swf-Format ist allgemein ein Hindernis für Open Source-Projekte.

# 18

# Gescheiterte Technik: das Applet

Das *Applet,* ein „Programmchen“, war Ende der 90er Jahre ein großes Thema, und hat ganz wesentlich zur raschen Verbreitung von Java beigetragen. Allerdings hat das Applet das „Tal der Ernüchterung“ in Abbildung 31.1 nach der ersten Euphorie nicht mehr verlassen.

Hier werden die Grundidee und die prinzipielle Arbeitsweise der Applets vorgestellt, und die Technik in ihrer aktuellen Bedeutung geschildert.

## 18.1 Idee des Applets

Nachdem die ersten Techniken des World Wide Web zum Standard geworden waren und mit Java eine plattformunabhängige Sprache vorlag (vgl. 1.5.2), kamen beide Techniken zusammen: Mit Java kann ein universell ausführbares Programm geschrieben werden, welches in eine Web-Site integriert ist; der Client lädt mit der Site das Java-Programm und führt dieses aus. Somit lassen sich praktisch beliebige Clientprogramme über das Web verteilen, ein zunächst großer Vorteil.

Aus den in Abschnitt 18.4 genannten Gründen sind Applets heute nicht mehr verbreitet. Zu den heutigen Einsatzbereichen gehören Car-Konfiguratoren (Abbildung 18.1) – wenn hier auch Ajax künftig viele Applets ablösen wird – und die Visualisierung im wissenschaftlichen Bereich: So ist etwa aus [Roj98] eine Simulation von Konrad Zuses Z3 mittels Applet entstanden.

**Abbildung 18.1:**
Beispiel für ein Applet

### 18.1.1 Softwarearchitektur von Applets

Mit Applets entsteht eine andere Softwarearchitektur, als wir sie bisher kennengelernt haben (vgl. 4.2.4): Es ist im Kern nur noch eine 2-Schicht-Architektur (Abbildung 18.2): Nach dem Ausliefern des Applets durch den Webserver scheidet dieser aus der Kommunikation faktisch aus, das Applet im Browser kommuniziert direkt mit dem Datenbankserver.

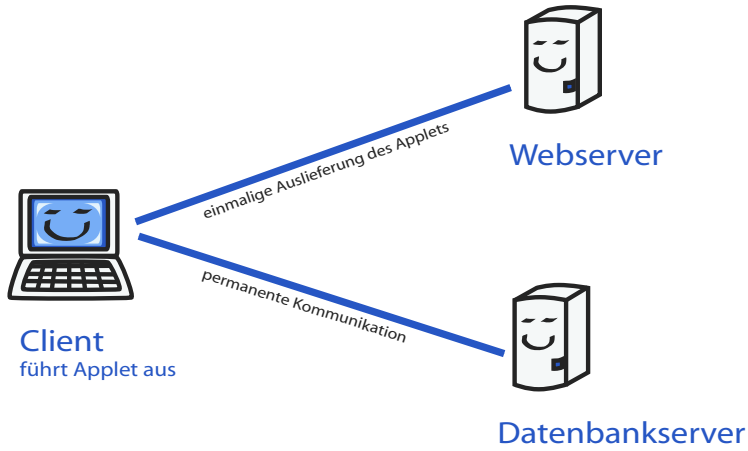



**Abbildung 18.2:**
Softwarearchitektur im Web mit einem Applet

In diesem zweischichtigen Modell liegt bereits ein wesentliches Problem, da nun der Datenbankserver weltweit erreichbar – und damit angreifbar – sein muss.

### 18.1.2 Applets und Sicherheit

In Bezug auf die Sicherheit ist das primäre Problem des Applets, dass auf dem Client ein „beliebiges“ Programm ausgeführt wird. Um hier jede Schadensmöglichkeit zu unterbinden, wird das Applet in der „Sandbox“ – also gekapselt

in einem speziellen Laufzeitsystem – ausgeführt und hat nur minimalen Zugriff auf Systemressourcen wie die lokale Festplatte.

Dieses Prinzip kann durch signierte Applets umgangen werden: Durch digitale Signatur und Zustimmung des Clients können einem Applet höhere Zugriffsrechte eingeräumt werden.

Das Prinzip der Sandbox hat sich in der Vergangenheit gut bewährt, die Zahl der Sicherheistlücken war hier – etwa im Gegegensatz zu JavaScript, welches ohne einen solchen Schutz auskommen muss – klein.

## 18.2 Einbinden eines Applets

Damit der Webserver das Applet an den Client ausliefern kann, muss dieses in die Webseite eingebunden werden. Hierfür stehen verschiedene HTML-Tags zur Verfügung:

- das `<applet>`-Tag: dies ist der klassische Weg zum Einbinden eines Applets; folgende Attribute sind wichtig für das Tag:
  - `width` und `height` sind zwingende Attribute, damit der Browser den Platzbedarf des Applets kennt;
  - `code` gibt den Namen der compilierten Java-class-Datei mit dem Applet-Code an;
  - `codebase` kann ein abweichendes Verzeichnis mit dem Applet-Code festlegen;
  - `align` gibt die Ausrichtung des Textes um das Applet vor;
  - `hspace` und `vspace` definiert den Abstand des Applets zum umgebenden Text.

  Zwischen dem öffnenden und schließenden `<applet>`-Tag können Parameterwerte zur Übergabe an das Applet durch die Syntax

  ```
  <param name="n" value="42">
  ```

  übergeben werden.
- das `<object>`-Tag ist neuerer und flexiblerer HTML-Standard; hier ist das Attribut

  ```
  codetype="application/java-vm"
  ```

  für die Kennzeichnung von Java Applets zusätzlich notwendig; den Namen der Java-Klasse gibt das Attribut

  ```
  classid="java:appletname.class"
  ```

  vor.

Die HTML-Datei `applet.html` (Abbildung 18.3) bindet das Applet `webkompendium.applet.WebKompendiumApplet` mittels des `<applet>`-Tags ein. In diesem Fall gehört das Applet zum Paket `webkompendium.applet`, was bedeutet, dass es bezogen auf diese HTML-Datei in einem Unterordner `./webkompendium/applet` abgelegt werden muss.

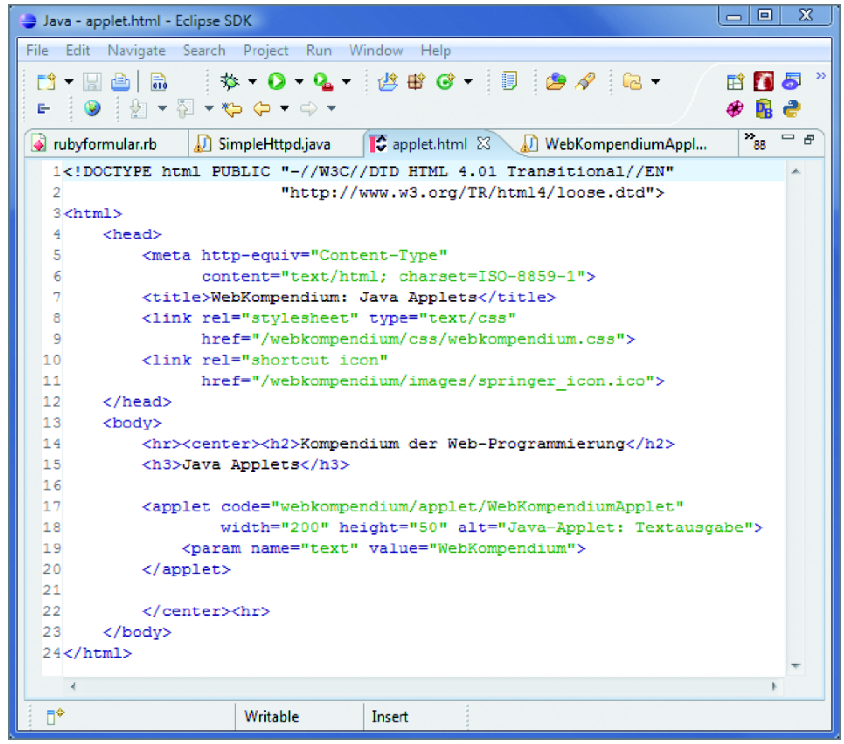



**Abbildung 18.3:**
Einbinden eines Applets

Komplexere Fälle, in welchen mehrere Java-Klassen benötigt werden, binden ein Java-Archiv ein: Alle benötigten class-Dateien werden dabei zu einem jar-Archiv gebündelt, was die Verteilung des Applets wesentlich vereinfacht.

## 18.3 Applet-Klassen in Java

Das eigentliche Applet, also die Java-Programmierung, ist eine Klasse, welche von der Klasse `java.applet.Applet` abgeleitet ist. Im Gegensatz zu Java-Kommandozeilenapplikationen hat das Applet keine main-Methode und auch keinen Konstruktur, sondern verwendet folgende Methoden:

- `init()`: diese Methode wird einmalig beim Laden des Applets in den Browser ausgeführt; sie hat somit die Rolle eines Konstruktors;
- `start()` wird jedesmal ausgeführt, wenn das Applet sichtbar wird;
- `paint(...)` zeichnet das Applet;
- `stop()` wird ausgeführt, wenn das Applet verdeckt wird; es ist somit die gegenteilige Methode zu `start`;
- `destroy` hat die Rolle des Destruktors: diese Methode wird automatisch aufgerufen, wenn das Applet aus dem Speicher entfernt wird, die gegenteilige Methode zu `init`.

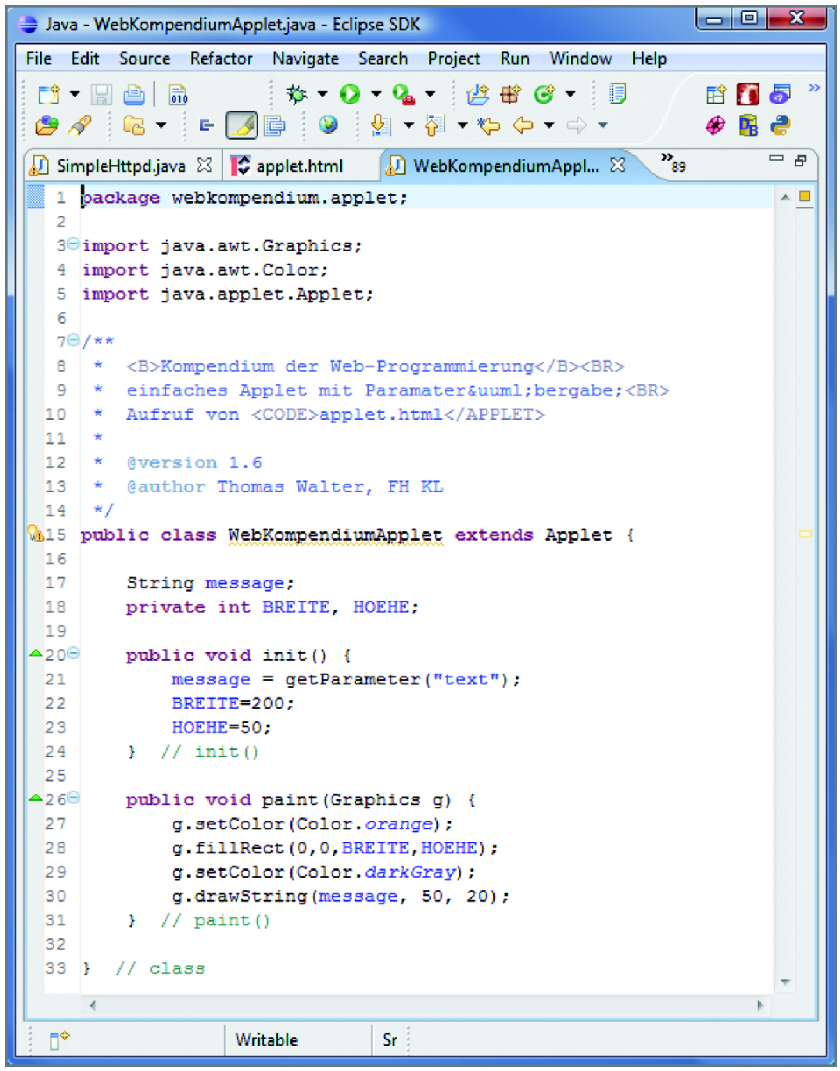
Die mittels des `<param>`-Tags übergebenen Parameterwerte können innerhalb des Applets mittels

```
getParameter("name_des_parameters")
```

abgerufen werden.

Das übliche Vorgehen für den Entwurf eines Applets führt über das Abstract Window Toolkit (AWT), der Java-Weg der Grafikprogrammierung vor der Einführung von Swing.

Die Klasse `webkompendium.applet.WebKompeniumApplet` nach Abbildung 18.4 zeigt ein derart geschriebenes einfaches Applet: Beim Laden des Applets wird `init` ausgeführt und damit der übergebene Parameter ausgelesen; die Methode `paint` zeichnet über eine Instanz der Klasse `java.awt.Graphics` das eigentliche Applet im Browser.




```
1  package webkompendium.applet;
2
3  import java.awt.Graphics;
4  import java.awt.Color;
5  import java.applet.Applet;
6
7  /**
8   *  <B>Kompendium der Web-Programmierung</B><BR>
9   *  einfaches Applet mit Paramater&uuml;bergabe;<BR>
10  *  Aufruf von <CODE>applet.html</APPLET>
11  *
12  *  @version 1.6
13  *  @author Thomas Walter, FH KL
14  */
15 public class WebKompendiumApplet extends Applet {
16
17     String message;
18     private int BREITE, HOEHE;
19
20     public void init() {
21         message = getParameter("text");
22         BREITE=200;
23         HOEHE=50;
24     }  // init()
25
26     public void paint(Graphics g) {
27         g.setColor(Color.orange);
28         g.fillRect(0,0,BREITE,HOEHE);
29         g.setColor(Color.darkGray);
30         g.drawString(message, 50, 20);
31     }  // paint()
32
33 }  // class
```

**Abbildung 18.4:**
Das Applet `WebKompendiumApplet` im Paket `webkompendium.applet`

Die durch Compilieren aus der Java-Datei nach Abbildung 18.4 entstandene class-Datei ist dann wegen der Paketstruktur wie beschrieben im Unterordner `./webkompendium/applet` der HTML-Datei aus Abbildung 18.3 abzulegen; Ausführen im Browser liefert dann die Ausgabe nach Abbildung 18.5.

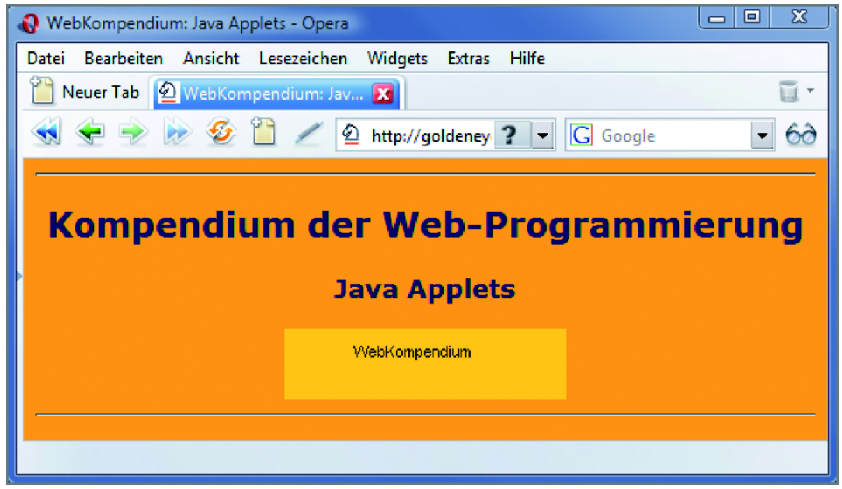



**Abbildung 18.5:**
Ausführen von `applet.html` mit dem Applet `WebKompendiumApplet`

## 18.4 Probleme der Applets

Vom Ansatz her bietet das Applet viele Möglichkeiten, die die ursprüngliche Webtechnik von Tim Berners-Lee nicht anbieten kann. Es zeigten sich aber rasch auch gravierende Nachteile;

- Zum Ausführen eines Java-Applets benötigt der Browser eine Java Virtual Machine (JVM). Zwar haben viele, aber nicht alle Browser eine JVM; ist diese nicht vorhanden, muss sie nachgeladen werden, was ein flüssiges Ausführen verhindert; insgesamt sind Seiten mit Applets durch das aufwändige Starten der JVM nicht flüssig ausführbar.
- Selbst wenn eine JVM vorhanden ist, kann nicht davon ausgegangen werden, dass die richtige Version vorliegt. Insbesondere benötigen einige Applets spezielle ältere Java-Versionen, die im allgemeinen nicht verfügbar sind.
- Suchmaschinen können nicht mit Applets umgehen.
- Applets provozieren geradezu Probleme mit der Barrierefreiheit nach Abschnitt 2.6.
- Wie auch bei JavaScript kann der Diensteanbieter nicht voraussetzen, dass der Client Java-Applets erlaubt.[1]
- Sicherheitsprobleme:
  - Der Client muss darauf vertrauen, dass die Sandbox fehlerfrei implementiert ist.
  - Datenbankbasierte Applet-Anwendungen laufen auf eine Zweischichtarchitektur nach Abbildung 18.2 hinaus, was eine hohe Gefährdung des Datenbankservers bedingt.

Aus diesen Gründen sind Applets heute nicht mehr verbreitet, sie haben aber eine wichtige Fortsetzung auf der Seite des Servers: das Servlet (vgl. Abschnitt 24.2).

[1] dies kann aber über JavaScript abgeprüft werden.

# Teil IV

# Fortgeschrittene Web-Programmierung

# 19

# Von CGI zu fastCGI

In Kapitel 9 wurde der zentrale CGI-Prozess vorgestellt: Das Starten eines Programmes auf dem Webserver durch einen entsprechenden Request an eine URL, welche üblicherweise auf die Adresse `cgi-bin` verweist. Dieses, bereits um 1992 vorgestellte Vorgehen ist bemerkenswert einfach – es hat aber insbesondere in der Performance deutliche Nachteile.

Fortschrittlicher ist der Ansatz von fastCGI; dieser hat sich in seiner ursprünglichen Form nicht wesentlich durchgesetzt, ist aber die Basis für sehr viele moderne Entwicklungen, weshalb er hier vorgestellt werden muss; zudem kommt fastCGI durch neuere Frameworks wie Ruby on Rails heute wieder verstärkt zum Einsatz.

## 19.1 Nachteile von CGI

Während der CGI-Prozess leicht aufzusetzen ist – es ist nur das Standard-Modul mod_cgi sowie ein ausführbares Programm notwendig –, hat er doch einige weitreichende Nachteile:

- Da der Prozess über die CGI-Umgebungsvariablen des Webservers seine Parameter erhält, kann er nicht auf einen separaten Server ausgelagert werden, es müssen also immer Webserver und CGI auf der gleichen Hardware betrieben werden; lediglich ein Datenbankserver kann getrennt werden.
- Bei jedem Client-Request wird ein neuer Prozess gestartet; bei interpretierten Scripten wird dabei jedesmal der Scriptsprachen-Interpreter neu geladen und das Programm übersetzt.

Als Konsequenz hieraus ergibt sich eine sehr eingeschränkte Performance, welche CGI für größere Web-Anwendungen ausschließt.

**Abbildung 19.1:** fastCGI-Logo

## 19.2 Die Ideen von fastCGI

Der um 1995 vorgestellte fastCGI-Ansatz geht hier entscheidende Schritte weiter: Die Grundidee ist das „Daemonize-It“ – die Überführung des einzelnen CGI-Prozesses in einen permanenten Daemon auf dem Webserver. Hierfür sind einige weitere Voraussetzungen auf dem Webserver notwendig: das entsprechende fastCGI-Modul sowie für die jeweiligen Programmiersprachen benötigte Zusatzpakete.

Neben dieser Überführung in einen Server-Prozess führt fastCGI noch eine wesentliche zweite Komponente ein: Der fastCGI-Prozess kann auch über normale Netzwerkverbindungen mit dem Webserver kommunizieren – dies erlaubt im Gegensatz zu CGI die Trennung des Webservers vom Anwendungsserver auf verschiedene Serversysteme. Entsprechend überzeugt gibt sich die fastCGI-Dokumentation: „fastCGI is a fast, open, and secure Web server interface that solves the performance problems inherent in CGI, without introducing the overhead and complexity of proprietary APIs“.

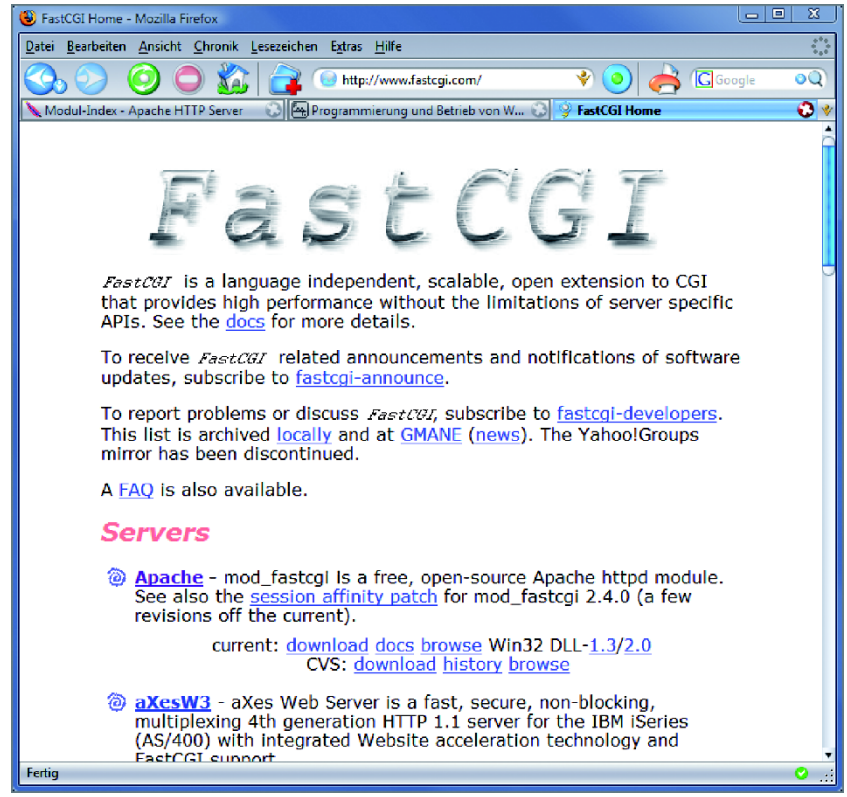



**Abbildung 19.2:** Die fastCGI-Site

### 19.2.1 Die Vorteile von fastCGI

Die Vorteile von fastCGI liegen damit auf der Hand:

- ein deutlicher Performance-Gewinn;
- Einfachheit, mit unkomplizierter Migration von bestehenden CGI-Anwendungen (fastCGI ist nur unwesentlich komplexer als natives CGI);
- Sprachunabhängigkeit: fastCGI ist mit fast allen relevanten Sprachen einsetzbar;
- fastCGI bietet eine Prozess-Isolation;
- fastCGI ist ein offener, nichtproprietärer Standard;
- fastCGI ist architekturunabhängig;
- fastCGI unterstützt verteilte Systeme.

Der eigentliche Motor des Erfolges von fastCGI ist:

- fastCGI-Prozesse sind Speicher-persistent: Nach Beendigung einer Anfrage werden sie nicht beendet, sondern warten auf die nächste Anfrage.

- fastCGI verwendet keine Umgebungsvariablen und keine pipes zur Kommunikation mit dem Webserver, sondern tcp-Verbindungen; als Folge können Webserver und fastCGI-Server auch getrennt werden.

### 19.2.2 Typischer Ablauf einer fastCGI-Anwendung

Eine fastCGI-Anwendung läuft nach folgendem Muster ab:

1. Der Webserver startet einen fastCGI-Anwendungsprozess; dies kann beim Start des Webservers selbst oder bei erster Anfrage an die URL erfolgen, je nach Konfiguration des fastCGI-Moduls.
2. Der fastCGI-Prozess initialisiert sich selbst und wartet auf Anfragen.
3. Bei Client-Anfragen baut der Webserver eine tcp-Verbindung zu dem fastCGI-Prozess auf und sendet darüber die Werte der CGI-Umgebungsvariablen und stdin.
4. Der fastCGI-Prozess sendet stdout und stderr entsprechend zurück.
5. fastCGI wartet auf weitere Anfragen.

### 19.2.3 fastCGI und Security

Ein weiterer Vorteil von fastCGI liegt in der Sicherheit: Durch die Möglichkeit, den Anwendungsserver vom eigentlichen Webserver zu trennen, können etwa mit lokalen Firewalls oder mit entsprechender Konfiguration der aktiven Netzwerkkomponenten weitere Sicherheitsbarrieren eingebaut werden (vgl. Kapitel 30). So sollte etwa der fastCGI-Server nur Requests von der ip des Webservers akzeptieren. Denkbar ist auch eine Verschlüsselung des Verkehrs zwischen Webserver und fastCGI-Server.

## 19.3 Das fastCGI-Protokoll

fastCGI verwendet ein spezielles Protokoll für die Kommunikation zwischen fastCGI-Prozess und Webserver. Beispiele hierfür sind etwa

- `FCGI_PARAMS`: name/value-Paare für CGI-Umgebungsvariablen;
- `FCGI_STDIN` für die Standardeingabe;
- `FCGI_STDOUT` für die Standardausgabe;
- `FCGI_STDERR` für Fehlermeldungen;
- `FCGI_END_REQUEST`: Ende der Anfrage.

Weitere Details sind auf der fastCGI-Web-Site [web] zu finden.

Dieses Protokoll wird durch das fastCGI-Modul auf dem Webserver implementiert; die einzelnen Programmiersprachen müssen durch entsprechende, zur fastCGi-Entwicklungsumgebung (dem Developer's Kit) gehörende Pakete erweitert werden.

## 19.4 fastCGI Developer's Kit

Auf der fastCGI-Site [web] ist das *fastCGI Developer's Kit* zu finden. Dieses umfasst viele für die Erstellung von fastCGI-Anwendungen notwendigen Bibliotheken, etwa für die Sprachen C, Perl und Java, sowie eine genauere Beschreibung des fastCGI-Protokolls.

## 19.5 Das fastCGI-Servermodul

Notwendig für den Einsatz von fastCGI ist das zugehörige *Server-Modul;* es ist für den Apache-Webserver und für einige andere, weniger verbreitete Systeme verfügbar.

Natürlich handelt es sich hierbei nicht um ein Standard-Modul des Apache, sondern um ein echtes „Fremdmodul“. Es ist aktuell in zwei Versionen verfügbar, für Apache 1.3 und für 2.0/2.2; es liegen Sourcen sowie die für Windows compilierten dlls bereit.

Das Modul wird jeweils über die LoadModule-Direktive der Apache-Konfiguration httpd.conf dynamisch eingebunden; ist das Modul server-info vorhanden, kann die konkrete Konfiguration des fastCGI-Moduls überprüft werden. Das Einbinden geschieht konkret über eine Direktive der Art

```
LoadModule fastcgi_module modules/mod_fastcgi-2.4.2.so        ### Unix

LoadModule fastcgi_module modules/mod_fastcgi-2.4.2-AP20.dll  ### Windows
```

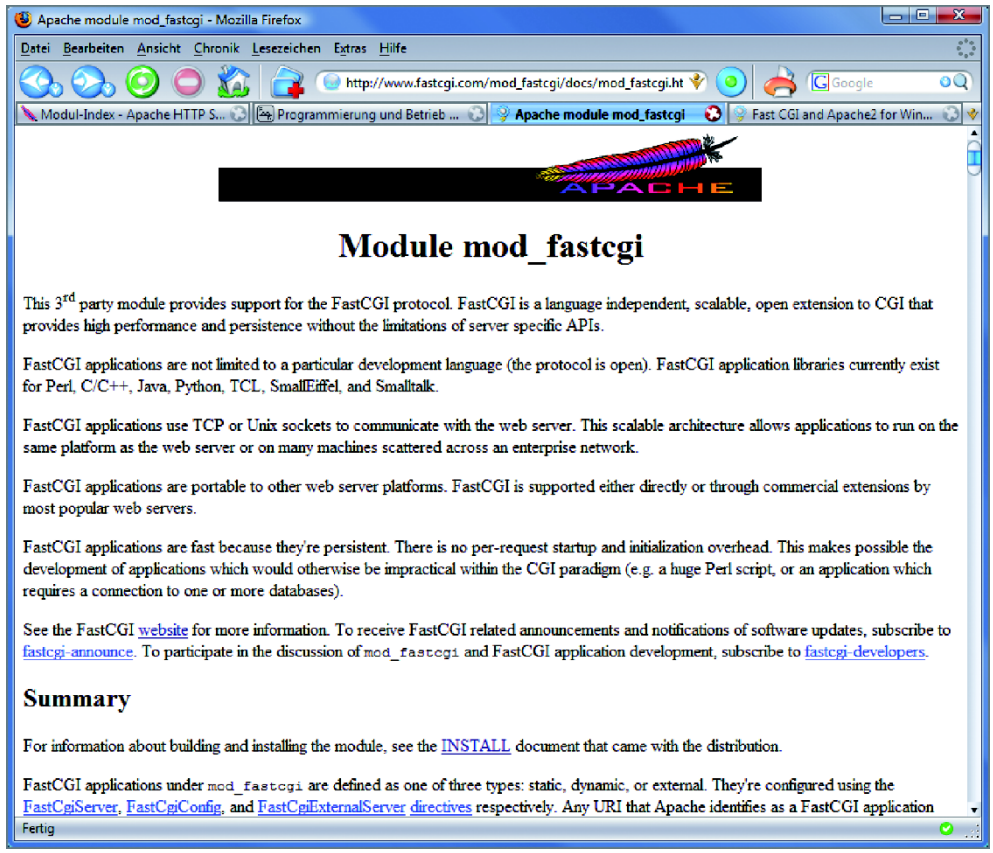



**Abbildung 19.3:** Online-Dokumentation zum fastCGI-Modul für Apache

Neben dem Laden des reinen Moduls muss ein Handle gesetzt werden:

```
<Directory "...">
     SetHandler fastcgi-script
</Directory>

AddHandler fastcgi-script fcgi fpl ...
```

Ersteres erlaubt fastCGI in einem bestimmten Verzeichnis (und dessen Unterverzeichnissen), zweiteres belegt den fastCGI-Handle mit Dateien mit der Endung `.fcgi` und `.fpl`.

### 19.5.1 Der fastCGI-Serverprozess

Neben dieser Konfiguration des Webservers wird ebenfalls in der httpd.conf festgelegt, welcher fastCGI-Serverprozess bereit steht; hier gibt es zwei grundsätzlich verschiedene Ansätze, der interne Server und der externe Server:

- Interner Server: Der Server läuft auf der gleichen Betriebssystem-Instanz wie der Webserver; durch eine Direktive der Art

  ```
  FastCgiServer /data/www/fcgi/webkompendium.fcgi
  ```

  wird ein interner Server definiert; dieser wird automatisch mit dem Start des Apache ebenfalls als Hintergundprozess gestartet – Apache übernimmt mit fastCGI das komplette Prozess-Management für diesen Prozess (ebenso wird der Prozess auch gemeinsam mit dem Apache beendet).
- Externer Server: Der externe Server läuft (normalerweise) auf einem anderen System; der Webserver und dieser Serverprozess kommunizieren über tcp-Netzwerkverbindungen. Der externe Server wird definiert durch die Apache-Direktive

  ```
  FastCgiExternalServer /data/www/fcgi/webkompendium
                                  -host 123.456.123.456:12345
  ```

  Dies bedeutet, dass ein Request nach `/fcgi/webkompendium` auf den tcp-Port 12345 zu dem Rechner mit der ip 123.456.123.456 weitergeleitet wird (wobei wieder /data/www als DocumentRoot angenommen wird).

  Da nun der Serverprozess auf einem entfernten Rechner läuft, kann Apache – natürlich – nicht das Prozessmanagement hierfür übernehmen.

Es sind einige wichtige weitere Konfigurationen möglich, eine Übersicht gibt die angegebene Online-Dokumentation zum fastCGI-Modul.

## 19.6 fastCGI-Anwendungen programmieren

fastCGI-Anwendungen können vergleichsweise einfach – aus bestehenden CGI-Anwendungen – entwickelt werden. Im Kern sind zwei Schritte notwendig:

- Einbinden des jeweiligen fastCGI-Paketes für die Kommunikation des Programmes mit dem Apache-Modul;

- Einbetten der Anwendung in eine Endlosschleife, damit sie konstant läuft.

Das Developer's Kit enthält die fastCGI-Libraries unter anderem für folgende Sprachen: C/C++, Perl, Java, Schema, Eiffel, Python, Ruby, TCL und Smalltalk.

### 19.6.1 Die fastCGI-Endlosschleife

Eingebettet wird die eigentliche Anwendung in eine Schleife der Art

```
// Initialisierung;

while (FCGI_Accept() >= 0 )  {
    ### Verarbeitung des Requests
}
```

Die – von den jeweiligen Paketen bereitgestellte – Methode `FCGI_Accept` wartet auf neue Anfragen – oder beendet über den fastCGI-Prozess-Manager die Anwendung.

### 19.6.2 fastCGI in Perl

Um fastCGI-Anwendungen in Perl zu schreiben, wird das entsprechende Perl-Modul FCGI.pm benötigt; dieses kann als Bestandteil des Developer's Kit compiliert und installiert werden, oder einfacher über den Perl Packet Manager ppm (vgl. 10.5.6) installiert werden (Abbildung 19.4).

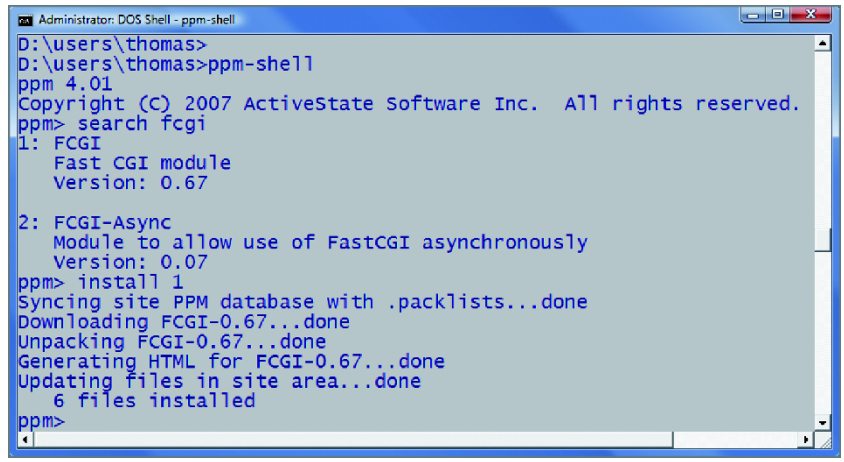
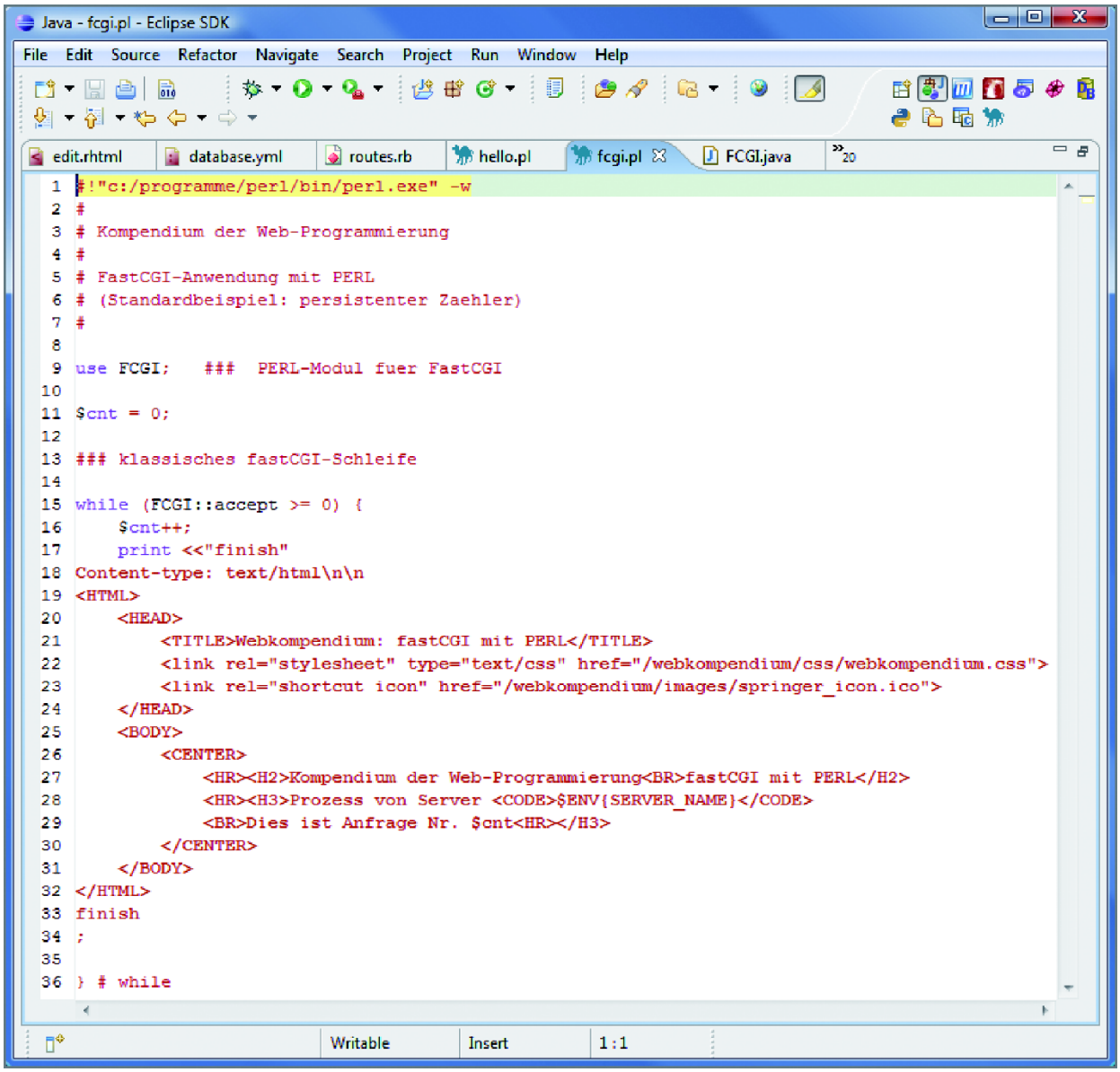



**Abbildung 19.4:**
ppm-Installation von FCGI.pm

Die Beispielanwendung `fcgi.pl` zählt die Anzahl der Zugriffe auf eine Seite – durch eine interne Variable `$cnt`. Bei einer normalen CGI-Anwendung würde bei jedem Aufruf diese Variable mit `0` initialisiert werden, es wird also immer der gleiche Zählerstand angezeigt; bei fastCGI läuft hingegen der Prozess permanent, es wird also korrekt weiter gezählt.




```
#!"c:/programme/perl/bin/perl.exe" -w
#
# Kompendium der Web-Programmierung
#
# FastCGI-Anwendung mit PERL
# (Standardbeispiel: persistenter Zaehler)
#

use FCGI;   ###  PERL-Modul fuer FastCGI

$cnt = 0;

### klassisches fastCGI-Schleife

while (FCGI::accept >= 0) {
    $cnt++;
    print <<"finish"
Content-type: text/html\n\n
<HTML>
    <HEAD>
        <TITLE>Webkompendium: fastCGI mit PERL</TITLE>
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
    </HEAD>
    <BODY>
        <CENTER>
            <HR><H2>Kompendium der Web-Programmierung<BR>fastCGI mit PERL</H2>
            <HR><H3>Prozess von Server <CODE>$ENV{SERVER_NAME}</CODE>
            <BR>Dies ist Anfrage Nr. $cnt<HR></H3>
        </CENTER>
    </BODY>
</HTML>
finish
;

} # while
```

**Abbildung 19.5:**
Das Perl-Script `fastcgi.pl`

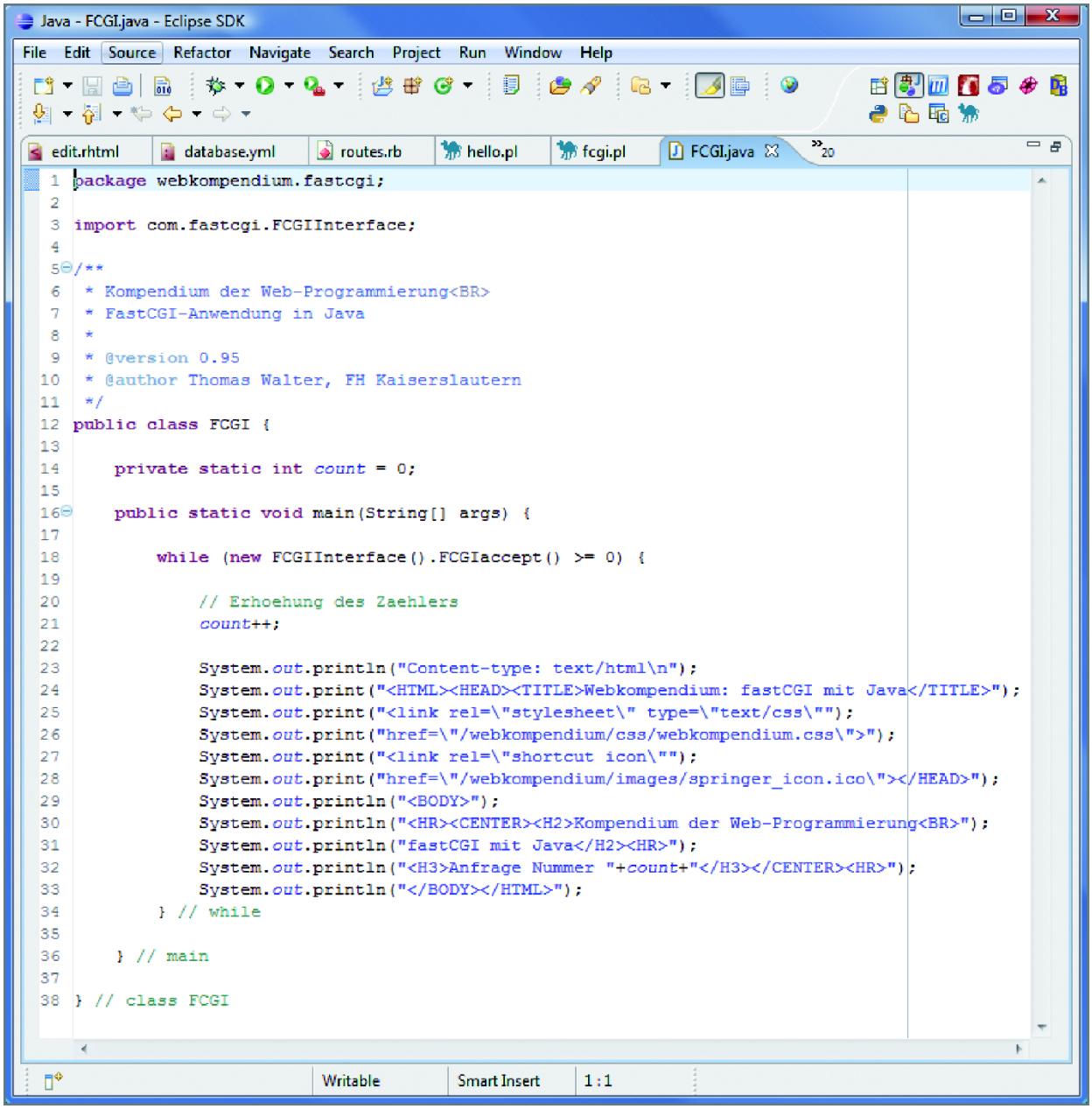
### 19.6.3 fastCGI in Java

Java verwendet für die fastCGI-Schnittstelle das Paket `com.fastcgi`, welches ebenfalls Teil des Developer's Kit ist. Benötigt wird aus diesem Paket zunächst das Interface `com.fastcgi.FCGIInterface`.

Das Beispiel der Klasse `webkompendium.fastcgi.FCGI` stellt hier einen externen fastCGI-Server da – im Falle von Java ist dies notwendig, da Apache nicht direkt die JVM starten kann. Diesem externen Server muss der verwendete tcp-Port als Parameter `FCGI_PORT` übergeben werden. Dies geschieht durch Starten der Anwendung mittels

```
java -DFCGI_PORT=12345 webkompendium.fastcgi.FCGI
```

wenn beispielsweise der tcp-Port 12345 verwendet werden soll. Abbildung 19.6 zeigt die entsprechende Java-Klasse.




```
1  package webkompendium.fastcgi;
2
3  import com.fastcgi.FCGIInterface;
4
5  /**
6   * Kompendium der Web-Programmierung<BR>
7   * FastCGI-Anwendung in Java
8   *
9   * @version 0.95
10  * @author Thomas Walter, FH Kaiserslautern
11  */
12 public class FCGI {
13
14     private static int count = 0;
15
16     public static void main(String[] args) {
17
18         while (new FCGIInterface().FCGIaccept() >= 0) {
19
20             // Erhoehung des Zaehlers
21             count++;
22
23             System.out.println("Content-type: text/html\n");
24             System.out.print("<HTML><HEAD><TITLE>Webkompendium: fastCGI mit Java</TITLE>");
25             System.out.print("<link rel=\"stylesheet\" type=\"text/css\"");
26             System.out.print("href=\"/webkompendium/css/webkompendium.css\">");
27             System.out.print("<link rel=\"shortcut icon\"");
28             System.out.print("href=\"/webkompendium/images/springer_icon.ico\"></HEAD>");
29             System.out.println("<BODY>");
30             System.out.println("<HR><CENTER><H2>Kompendium der Web-Programmierung<BR>");
31             System.out.println("fastCGI mit Java</H2><HR>");
32             System.out.println("<H3>Anfrage Nummer "+count+"</H3></CENTER><HR>");
33             System.out.println("</BODY></HTML>");
34         } // while
35
36     } // main
37
38 } // class FCGI
```

**Abbildung 19.6:**
Die Java-Klasse `webkompendium.fastcgi.FCGI`

## 19.7 Leistungen, Grenzen und Ausblick

fastCGI kann herkömmliche CGI-Anwendungen deutlich beschleunigen: bis zum Faktor 5 werden Anwendungen performanter. Genauer betrachtet ergibt sich – nach Herstellerangaben – folgender Zusammenhang:

- Auslieferung einer statischen HTML-Datei: 21 ms + 0.19 ms / kB;
- fastCGI-Anwendung: 22 ms + 0.28 ms / kB;
- CGI-Anwendung: 59 ms + 0.37 ms / kB;

Die Grenzen von fastCGI liegen insbesondere im problematischen Multitasking-Verhalten: Bei mehreren gleichzeitig eingehenden Requests verhält sich fastCGI nicht optimal.

Zum anderen hat eine strenge Lizenz-Politik fastCGI eher gebremst: fastCGI unterliegt der (strengeren) GNU-Policy und nicht dem neueren FSF-Lizenz-Modell. Es ist damit frei für jeden Einsatz und darf verändert werden, allerdings muss die Lizenz mitverteilt und insbesondere muss der damit entwickelte Code offengelegt werden, was sich sehr nachteilig auf die Verbreitung dieser Technik auswirkt.

fastCGI hat sich nicht direkt durchgesetzt – aber:

- die Ideen von fastCGI waren Wegbereiter für viele nachfolgende Techniken von PHP bis hin zu Java Servlets;
- fastCGI erlebt gerade eine gewisse Renaissance: Frameworks wie Ruby on Rails (vgl. Kapitel 23) basieren für eine performante Ausführung auf fastCGI.

# 20

# Das PHP-Framework PEAR

Ein sehr populäres und verbreitetes Framework ist das auf PHP basierende *PEAR*. Der Name ist dabei die Abkürzung für „PHP Extension and Application Repository“ – und eigentlich handelt es sich um viel mehr als „nur“ um ein Framework!

PEAR wurde 2003 von Stig Bakken vorgestellt und erfreut sich seither einer großen Beliebtheit. Die Idee für dieses Framework ist – wie so häufig – an Perl angelehnt: Mit CPAN (vgl. 10.5.4) steht für Perl ein umfassendes, einfach zugreifbares Archiv bereit; PEAR möchte ähnliches für PHP liefern. Das ist mit PEAR gelungen.

**Abbildung 20.1:**
PEAR-Logo

## 20.1 Struktur von PEAR

PEAR ist im Vergleich zu vielen anderen Frameworks eigentlich mehr: Entsprechend der eingangs beschriebenen Idee besteht PEAR aus einer ganzen Familie von Bibliotheken, die zugreifbar werden.

Dabei sind die eigentlichen PEAR-Bibliotheken plattformunabhängig in PHP geschrieben; es gehören aber auch die PECL-Bibliotheken zu PEAR: PHP Extension Code Library. Diese sind in C geschrieben.

## 20.2 Installation von PEAR

Die aktuellen PHP-Distributionen werden schon seit längerem mit einem speziellen PHP-Script ausgeliefert:

```
go-pear.php
```

welches üblicherweise im Unterordner `PEAR` der PHP-Installation zu finden ist oder von der PEAR-Site bezogen werden kann. Dieses Script ist für die Installation des PEAR-Frameworks wesentlich; zum Ausführen sollte der PHP-Interpreter im Pfad aufgenommen sein,[1] dann wird die PEAR-Installation mittels

```
php go-pear.php
```

gestartet; dabei ist darauf zu achten, dass die notwendigen Berechtigungen zum Schreiben in das Installationsverzeichnis vorhanden sind.

[1] dies bietet sich sowieso für das direkte Ausführen zum Testen von PHP-Scripts an.

Im Prinzip können die PEAR-Klassen in jedes beliebige Verzeichnis installiert werden, es ist aber üblich, den Unterordner `PEAR` der PHP-Installation hierfür zu nehmen. Wichtig ist, dass in der PHP-Konfigurationsdatei `php.ini` der Parameter

```
include_path
```

entsprechend angepasst wird; dies kann auch das Installationsscript `go-pear` übernehmen, dann erfolgt die Setzung etwas unübersichtlich ganz am Ende der Konfigurationsdatei.

Unter Windows sind nach der Installation noch Registry-Variablen zu setzen, wofür `go-pear` auch ein Registry-Script `PEAR_ENV.reg` erzeugt.

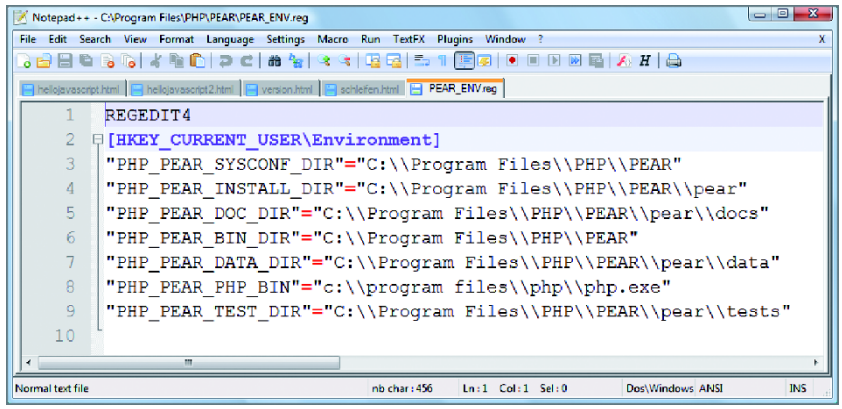



**Abbildung 20.2:** PEAR-Einträge in die Registry von Windows

## 20.3 Das Dienstprogramm PEAR

Nach erfolgreicher Installation steht das Kommandozeilenscript

```
pear
```

bereit; dieses ist der eigentliche Paketmanager von PEAR. Das Script verfügt über zahlreiche Optionen (Abbildung 20.3), die insbesondere die Suche und das einfache Installieren von PEAR-Paketen ermöglichen (Abbildung 20.4).

**Abbildung 20.3:**
Optionen des Paketmanagers `pear`

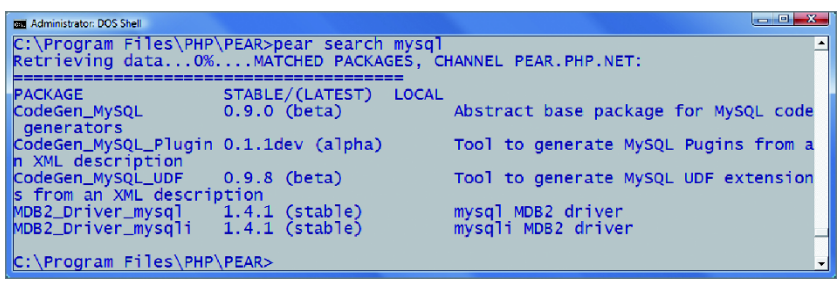



**Abbildung 20.4:**
Suche mittels `pear` nach „mysql“

In seiner Arbeitsweise ist der Paketmanager `pear` gut mit dem Paketmanager `ppm` für Perl vergleichbar (vgl. 10.5.6).

## 20.4 Die PEAR-Pakete

Das wesentliche an PEAR sind somit die damit einfach verfügbaren *PEAR-Pakete*. Diese können mittels des Paketmanagers `pear` gesucht und installiert werden, über die PEAR-Site [web] steht zusätzlich eine umfassende Dokumentation bereit.

Aktuell verfügt PEAR über rund 200 Pakete; im Web sind diese nach Gruppen sortiert und man kann sich einfach über die Pakete informieren (Abbildung 20.6).

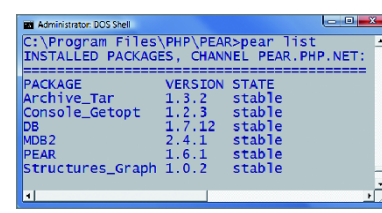



**Abbildung 20.5:**
Übersicht über die installierten PEAR-Pakete

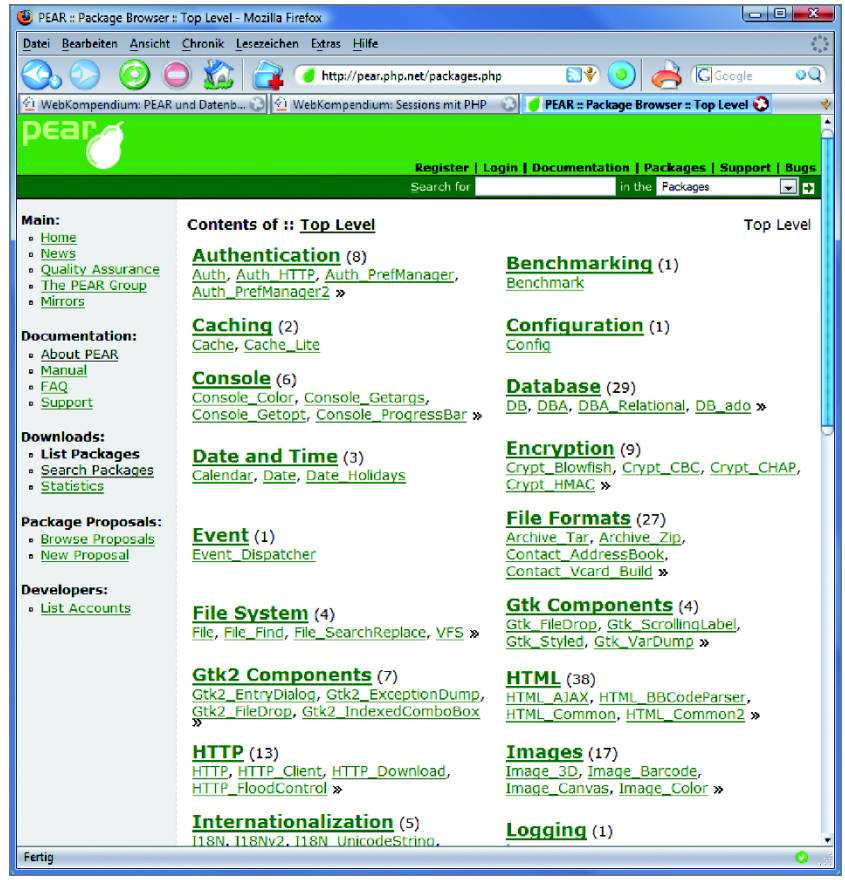



**Abbildung 20.6:** Online-Dokumentation zu den PEAR-Paketen nach Themengruppen

## 20.5 Das PEAR-Paket DB

In 10.7 haben wir gesehen, wie vorteilhaft eine Datenbankabstraktionsschicht ist. Die ursprünglichen PHP-Ansätze für das Arbeiten mit Datenbanken bieten diese Funktionalität nicht, es wird mit einem konkreten Treiber wie `mysql` oder `mysqli` mit einer jeweils völlig eigenen Syntax gearbeitet. Der bisherigie Ausweg würde in der sauberen Kapselung der Datenbankoperationen als Teil des Models (vgl. 4.3 und 11.6.2) bestehen, also Angelegenheit des Entwicklers sein. Einen eleganten Ausweg bietet PEAR: Das *PEAR-Paket DB* bietet eine Abstraktionsschicht für verschiedene Datenbankmanagementsysteme vergleichbar zu DBI oder jdbc.

### 20.5.1 Arbeitsweise von PEAR::DB

Die *Arbeitsweise* des DB-Paketes von PEAR ist vergleichbar zum Vorgehen bei DBI oder den proprietären Datenbanktreibern von PHP:

- im ersten Schritt wird eine Datenquelle definiert und die Verbindung dazu aufgebaut, es besteht dann ein Database Handle;
- Nun können SQL-Anweisungen über diesen Database Handle ausgeführt werden, wobei wieder `SELECT`-Abfragen zu trennen sind von den übrigen; im ersten Fall wird ein Result Set zurückgeliefert;

- vor dem Beenden des PHP-Scripts sind die offenen Verbindungen zu schließen.

Dabei ist insgesamt PEAR::DB sehr flexibel in seiner Syntax, was das Arbeiten vereinfacht. So kann die Definition der Datenquelle vielfältig geschehen; notwendig sind die Parameter

- Name des tieferliegenden Basistreibers (entsprechend DBD bei Perl-DBI);
- Benutzername;
- Kennwort;
- Protokoll;
- Datenbank-Host;
- Datenbankport;
- Name der Datenbank.

Für einige dieser Werte existieren Default-Werte, die teilweise auch in der Konfigurationsdatei `php.ini` gesetzt werden können. Dies kann in PEAR::DB übersichtlich über ein PHP-Array geschehen:

```
$dsn = array (
          'phptype'  => "mysqli",
          'username' => "thomas",
          'password' => "strengGeheim",
          'protocol' => "tcp",
          'hostspec' => "localhost",
          'port'     => "3306",
          'database' => "webkompendium"
       );
```

es existiert aber auch eine konventionelle Art wie

```
$dsn="treiber://user:kennwort@host:port/db";
```

Die eigentliche Verbindung wird dann durch

```
$dbh = DB::connect($dsn);
```

aufgebaut, der Rückgabewert ist ein Database Handle.

Die Durchführung einer SQL-Anweisung `$sql` für diese bestehende Verbindung zu einer Datenbank erfolgt anschließend durch

```
$sth = $dbh->query($sql);
```

Im Falle einer `SELECT`-Abfrage ist der Rückgabewert der `query`-Methode nun ein Result Set. Dieses kann wieder zeilenweise mittels der Methode

```
$zeile = $sth->fetchrow($modus)
```

abgearbeitet werden; der Rückgabewert dieser Methode ist je nach Argument `$modus` unterschiedlich:

- `DB_FETCHMODE_ORDERED` ist der default-Fall und liefert ein numerisches Array für jede Trefferzeile;
- `DB_FETCHMODE_ASSOC` liefert ein assoziatives Array, wobei wieder die Keys des Hashs den abgefragten Attributen der `SELECT`-Abfrage entsprechen;
- `DB_FETCHMODE_OBJECT` liefert ein PHP-Objekt pro Trefferzeile, welches als Attribute die abgefragten Tabellenspalten enthält.

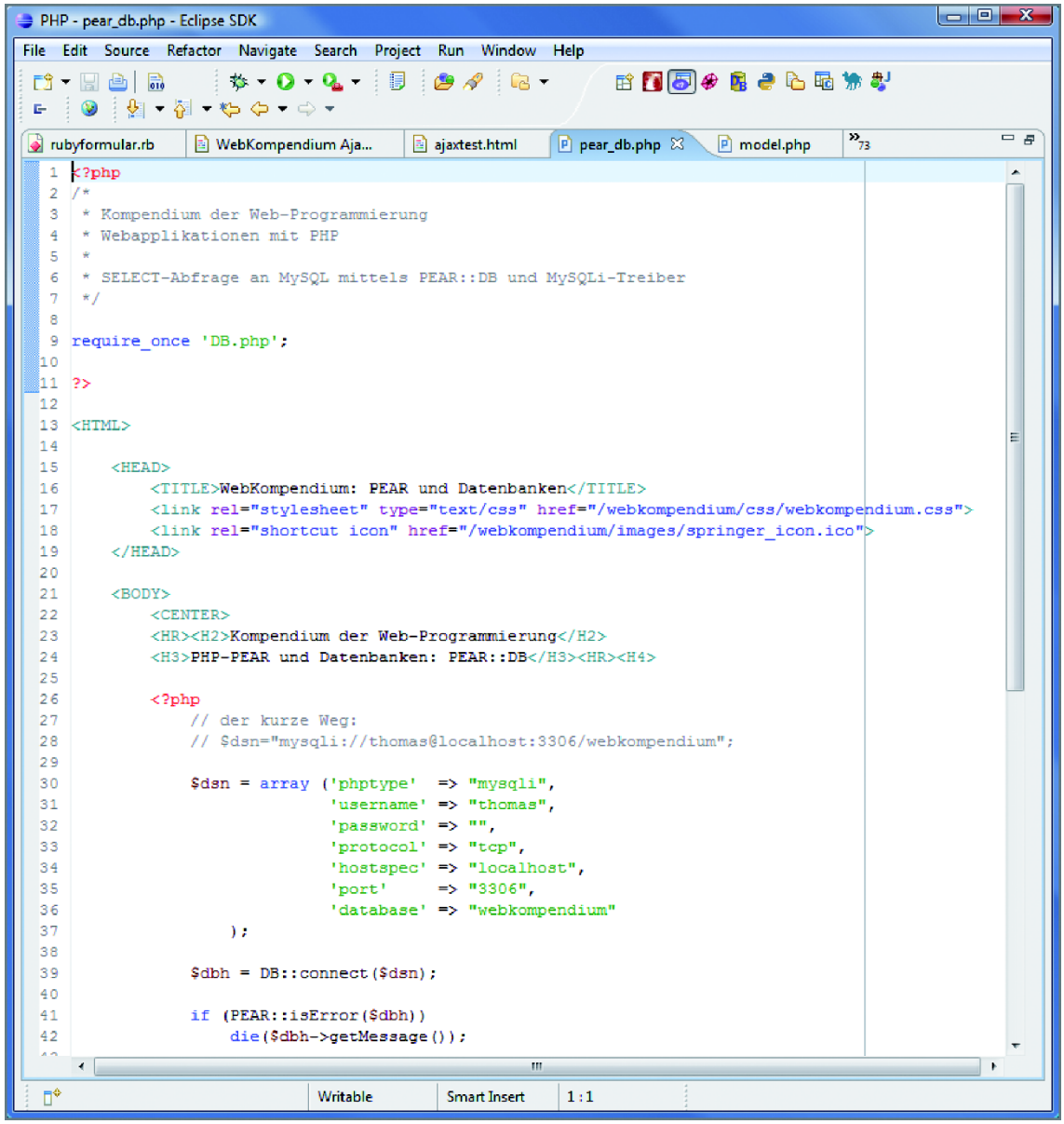
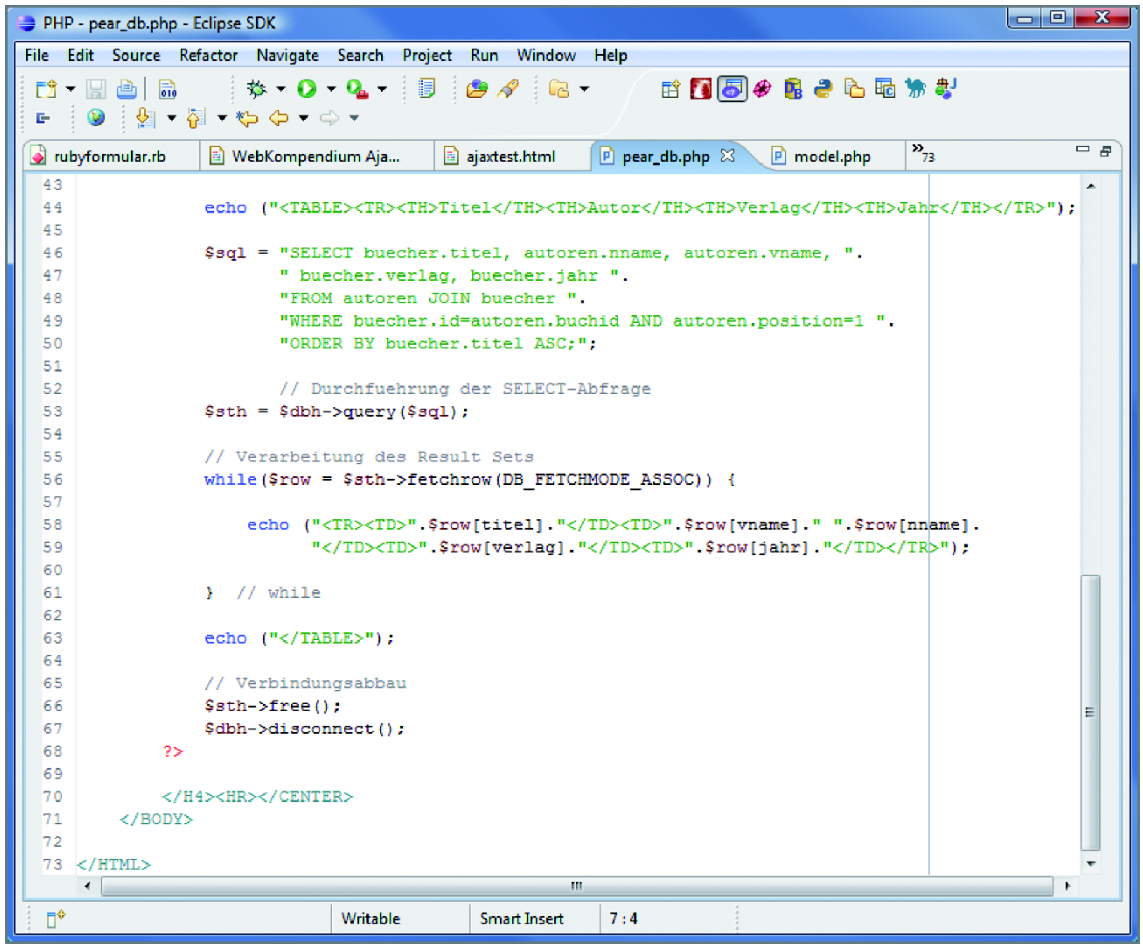



**Abbildung 20.7:**
Das PEAR::DB-Script `pear_db.php`, Teil I




```
43
44              echo ("<TABLE><TR><TH>Titel</TH><TH>Autor</TH><TH>Verlag</TH><TH>Jahr</TH></TR>");
45
46              $sql = "SELECT buecher.titel, autoren.nname, autoren.vname, ".
47                     " buecher.verlag, buecher.jahr ".
48                     "FROM autoren JOIN buecher ".
49                     "WHERE buecher.id=autoren.buchid AND autoren.position=1 ".
50                     "ORDER BY buecher.titel ASC;";
51
52                     // Durchfuehrung der SELECT-Abfrage
53              $sth = $dbh->query($sql);
54
55              // Verarbeitung des Result Sets
56              while($row = $sth->fetchrow(DB_FETCHMODE_ASSOC)) {
57
58                  echo ("<TR><TD>".$row[titel]."</TD><TD>".$row[vname]." ".$row[nname].
59                        "</TD><TD>".$row[verlag]."</TD><TD>".$row[jahr]."</TD></TR>");
60
61              }  // while
62
63              echo ("</TABLE>");
64
65              // Verbindungsabbau
66              $sth->free();
67              $dbh->disconnect();
68          ?>
69
70          </H4><HR></CENTER>
71      </BODY>
72
73  </HTML>
```

**Abbildung 20.8:**
Das PEAR::DB-Script `pear_db.php`, Teil II

**Abbildung 20.9:**
Ausführen von `pear_db.php`

Die übrigen SQL-Anweisungen außer der SELECT-Abfrage sind in PEAR::DB mittels einer Mischung aus `prepare` und `execute` durchzuführen:

```
$sth = $db->prepare
    ('INSERT INTO autoren (buchid) VALUES (?)');

$db->execute($sth, 4711);
```

### 20.5.2 PEAR::MDB2

Das PEAR-Paket *MDB2* ist eine Weiterententwicklung von PEAR::DB, indem es dieses mit dem Metabase-Paket von Manuel Lemos zusammenführt. Die weitere Entwicklung wird zeigen, welcher Weg sich dauerhaft etablieren wird.

# 21

# Template-Engines: Smarty & Co

Für eine strukturierte Web-Programmierung ist es notwendig, sauber in einem konkreten Programmierparadigma, in einem Design-Muster für die Softwarearchitektur, zu arbeiten. Das weitaus verbreitetste Muster ist Model-View-Controller, vgl. 4.3. Eine zentrale Komponente dabei ist die Darstellungsschicht, die View.

Zur Erzeugung einer solchen View gibt es mehrere Techniken. Sehr verbreitet ist die Verwendung von HTML-Templates:

Der große Vorteil von Templates ist die Trennung der eigentlichen Programmierung von der Gestaltung der Sicht, dem Web-Design. So kann etwa bei einem ausgelieferten Produkt der Kunde das Design seiner Web-Anwendung einfach über die Änderung der Templates anpassen, wofür meistens reine HTML-Kenntnisse ausreichen, ohne die eigentliche Programmierung ändern zu müssen. Auch kann bei der Entwicklung derartige Anwendungen der Web-Designer unabhängig vom Programmierer tätig sein.

Es gibt zahlreiche Ansätze für derartige Template-Systeme, die im Kern sehr ähnlich sind. Im Detail wird hier Smarty, eine beliebte Template-Engine für PHP, vorgestellt. Für Java ist Velocity aus dem Apache-Jakarta-Framework ein ähnlicher Ansatz.

## 21.1 Templates

Ein *HTML-Template* ist das Grundgerüst einer HTML-Seite, welche von einer Template-Engine „verarbeitet“ wird. Bei dieser Verarbeitung, auch als Rendering bezeichnet, werden aktive Inhalte verarbeitet und eine reine HTML-Ausgabe erzeugt.

Die Möglichkeiten der aktiven Inhalte in einem HTML-Template schwankt natürlich je nach verwendetem Framework, es gibt aber sehr viele Gemeinsamkeiten zwischen allen Ansätzen. Insbesondere sind hier zu nennen:

- das Anzeigen einfacher (skalarer) Variabler;
- das Ausführen von Schleifen, etwa: „für ein gegebenes Array gebe jedes Element aus“;
- Verzweigungen: „Je nach Wert einer Variablen mache A oder mache B“.

Um dieses zu realisieren, gehört zu jedem Template-System auch eine zugehörige Sprache, eine Template Language. Dies sind immer sehr einfache Sprachen mit typischerweise um die zehn Anweisungen.

## 21.2 Die Template-Engine Smarty

Es gibt zahlreiche Systeme für die Verwendung von Templates, hier wird zunächst *smarty* vorgestellt [web], eine compilierende Template-Engine für PHP. Weitere Template-Engines folgen etwa als Teil der Frameworks django (vgl. 22) und Ruby on Rails (vgl. 23).

Smarty ist eine Template-Engine und bietet die typischen Vorteile solcher Systeme:

- Smarty ist sehr performant;
- ein Template muss nur einmalig verarbeitet (compiliert) werden, wenn es sich nicht ändert;
- Smarty bietet Kontrollstrukturen wie if/elseif/else/endif im Template;
- Smarty unterstützt Caching für eine schnellere Ausgabe;
- Smarty ist erweiterbar durch eine praktische Plugin-Architektur.

**Abbildung 21.1:** Smarty-Logo

Im Web [web] ist auch eine recht umfassende, von Monte Ohrt und Andrei Zmievski erstellte Online-Dokumentation verfügbar.

### 21.2.1 Installation von Smarty

Smarty kann mit jeder PHP-Version ab 4.0.6 betrieben werden; es wird als `tar.gz`-Archiv ausgeliefert und muss lediglich entpackt (Abbildung 21.2) und entsprechend kopiert werden, so dass PHP auf die Smarty-Bibliotheken zugreifen kann.

Nach dem Kopieren ist die Direktive `include_path` der PHP-Konfiguration in der Datei `php.ini` noch so anzupassen, dass Smarty gefunden wird.

**Wird parallel zu Smarty auch PEAR installiert und hierfür das Installationsscript `go-pear` verwendet, so schreibt dieses am Ende der Datei `php.ini` eine eigene Setzung für `include_path`; diese ist dann für Smarty anzupassen.**

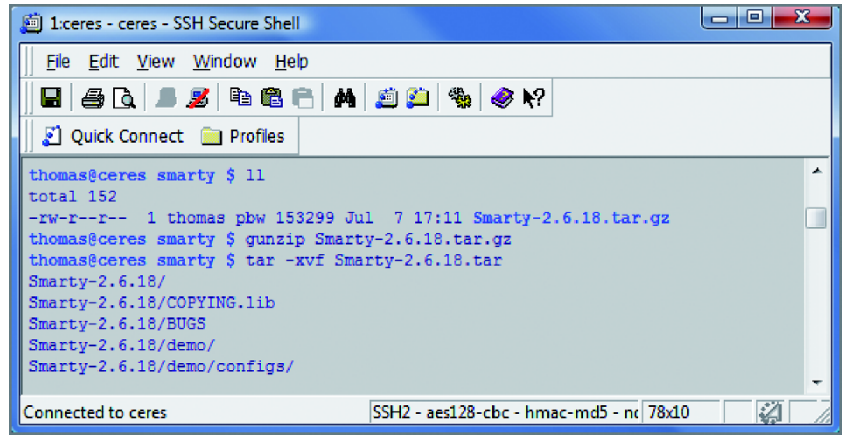



**Abbildung 21.2:** Smarty Installation

### 21.2.2 Ein erstes Beispiel für Smarty

Ein erstes Beispiel soll zunächst eine einfache Textmeldung mit Smarty erzeugen (Abbildung 21.5). Dazu sind zwei Schritte nötig:

- Die eigentliche PHP-Programmierung, im Beispiel die Datei `hellosmarty.php` (Abbildung 21.3). Diese importiert die zentrale Klasse `Smarty.class` (Zeile 8) und erzeugt eine Instanz hiervon (Zeile 10). Anschließend wird definiert, wo Templates abgelegt sind und eine Variable `message` wird „veröffentlicht“; schließlich wird das Template `hellosmarty.tpl` verarbeitet: mittels `display` an den Client ausgeliefert.

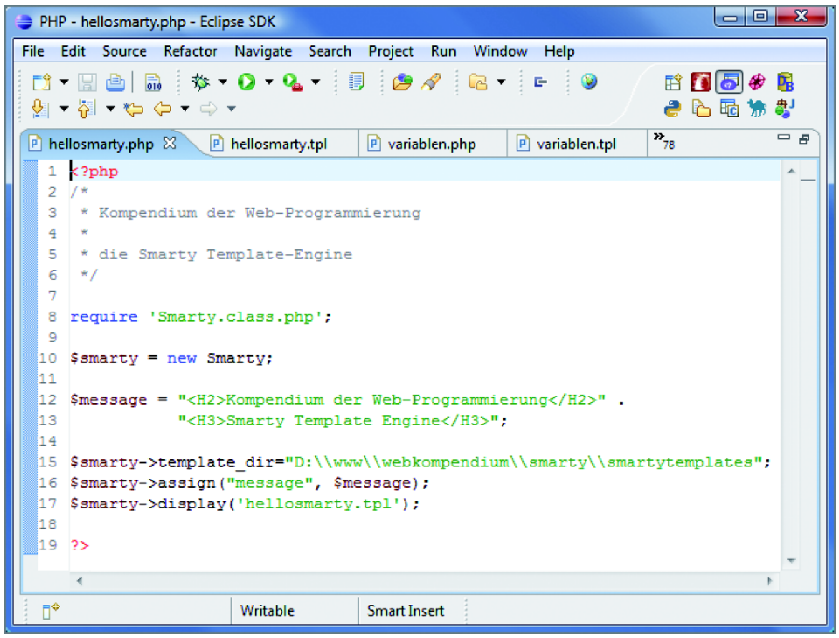



**Abbildung 21.3:** PHP-Datei für Smarty: `hellosmarty.php`

- Im zweiten Schritt ist eine Template-Datei zu erstellen, hier die Datei `hellosmarty.tpl` (Abbildung 21.4). Diese Template-Datei mit der Endung `.tpl` ist im Kern eine HTML-Datei; lediglich Zeile 1 mit einem speziellen Kommentar und insbesondere Zeile 13 mit

  ```
  {$message}
  ```

  ist eine Erweiterung von HTML durch Smarty; hier wird der in der PHP-Datei definierte Wert `message` angezeigt.

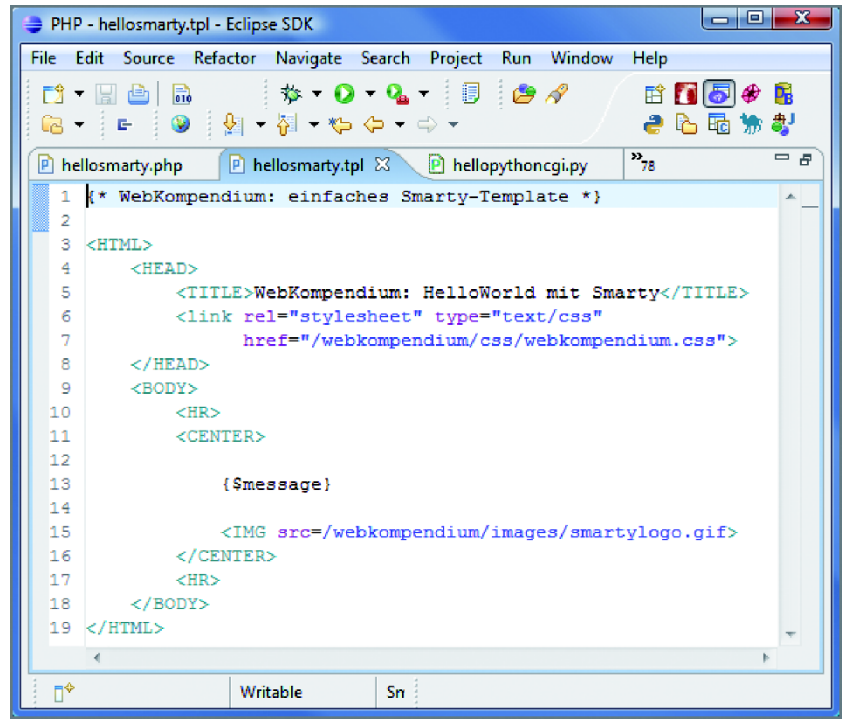



**Abbildung 21.4:**
Smarty Template-Datei `hellosmarty.tpl`

Durch einen einfachen GET-Request an den Webserver geschieht nun folgendes: Smarty erkennt, dass das entsprechende Template angefordert wird und compiliert dieses beim ersten Aufruf; die compilierte Datei wird defaultmäßig in einem Unterordner `templates_c` abgelegt. Diese compilierte Datei ist nichts weiter als eine reine PHP-Datei, welche „normal“ durch PHP verarbeitet wird (Abbildung 21.6).

**Abbildung 21.5:**
Ausführen von `hellosmarty.php`

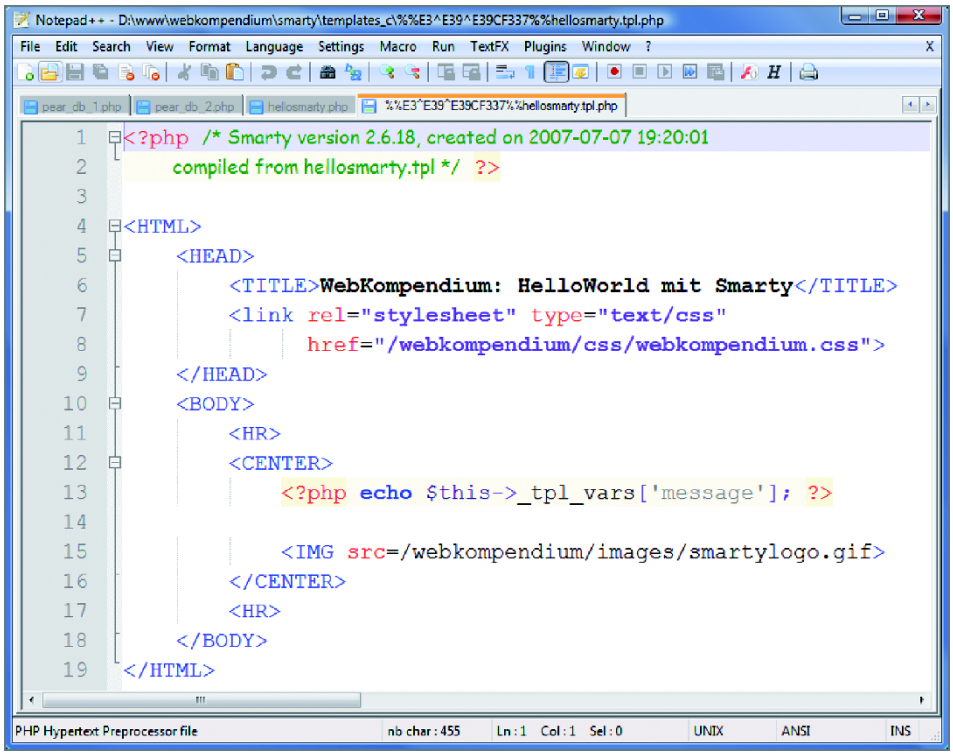



**Abbildung 21.6:**
Die durch das Verarbeiten des Templates entstandene compilierte Datei (eine reine PHP-Datei)

### 21.2.3 Die Smarty Template Language

Bereits im vorherigen Beispiel war das Grundprinzip von Smarty erkennbar: Im Template wird durch die Klammerung `{ ... }` aktiver Inhalt von Smarty integriert:

```
{ Smarty-Code }
```

Smarty-Kommentare im Template werden dabei einfach durch

```
{* Smarty-Kommentar *}
```

geschrieben.

#### 21.2.3.1 Variablen in Smarty

Der Umgang mit Variablen in Smarty ist denkbar unkompliziert, es gibt dafür einige einfache Regeln:

- Variablenbezeichner in Smarty beginnen mit führendem `$`;
- Arrays werden mit `[i]` indiziert;
- assoziative Arrays werden mittels `$assArray.key` indiziert;
- ein Objektattribut wird durch `$referenz->attribut` adressiert;
- eine Objektmethode wird über `$referenz->methode(...)` aufgerufen;
- mittels der Syntax `#PATH#` kann im Smarty-Template auf eine Systemvariable des Servers zugegriffen werden.

Um auf eine Variable im Smarty-Template zugreifen zu können, muss diese im zugrundeliegenden PHP-Script „veröffentlicht“ werden (im Beispiel aus Abbildung 21.3 ist dies die Zeile 16); hierfür dient die PHP-Smarty-Methode `assign`:

```
$smarty->assign(variable,wert);
```

Diese Anweisung veröffentlicht die Variable mit dem Bezeichner `variable` und weist dieser den Wert `wert` zu; innerhalb des Templates kann dann über `$variable` auf diesen Wert zugegriffen werden.

Dieses Prinzip ist ganz analog nicht nur für skalare Variablen, sondern auch für Arrays und assoziative Arrays anzuwenden, wobei dann `wert` eine solche Struktur sein muss.

Das Smarty-Script `variablen.php` veröffentlicht drei solcher Variablen (Abbildung 21.7; mittels des Templates `variablen.tpl` (Abbildung 21.8) erfolgt dann der Zugriff auf einige dieser Variablen (Abbildung 21.9); dabei ist auch der Fall (Zeile 21 im Template), dass der Key des Hashs selbst eine Variable ist.

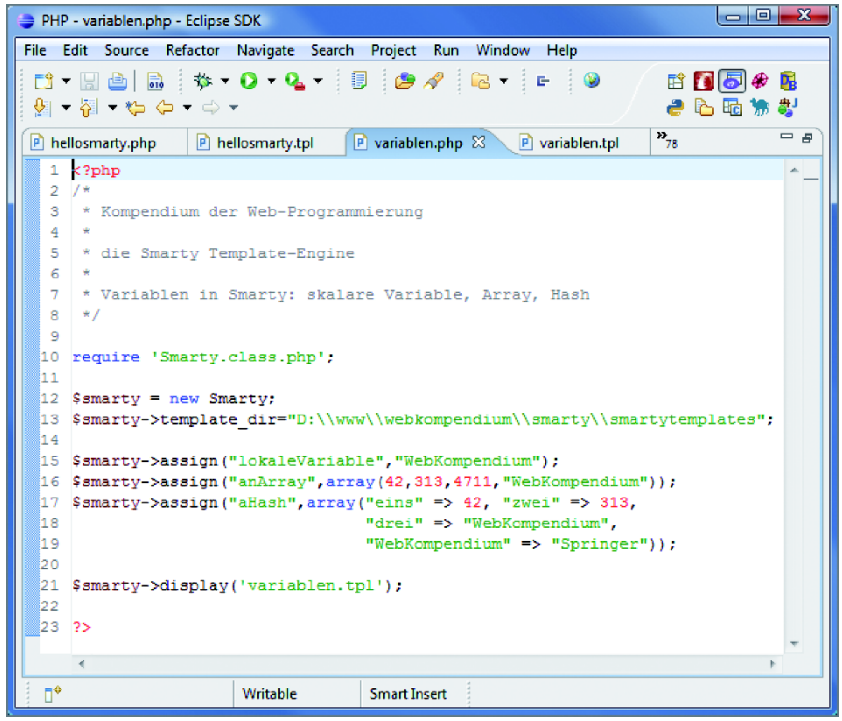



**Abbildung 21.7:** Übergabe von Variablen verschiedenen Typs mittels Smarty `variablen.php`

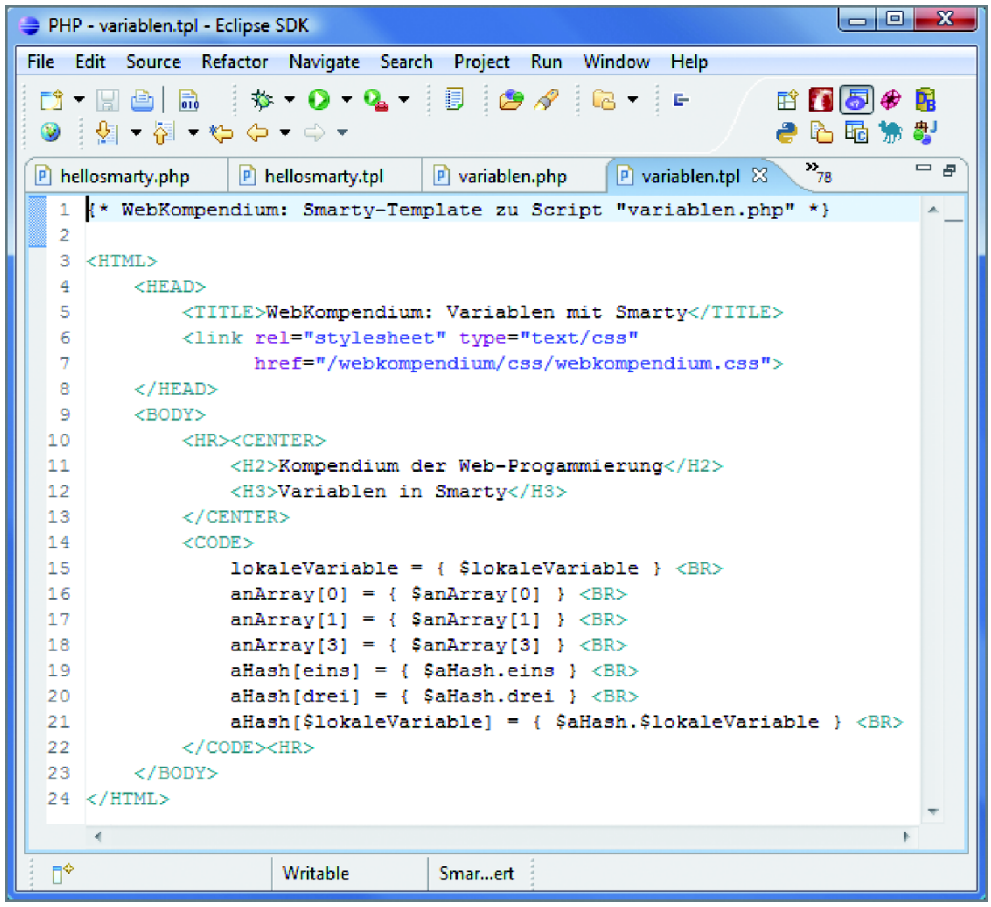



**Abbildung 21.8:**
Template `variablen.tpl` für `variablen.php`

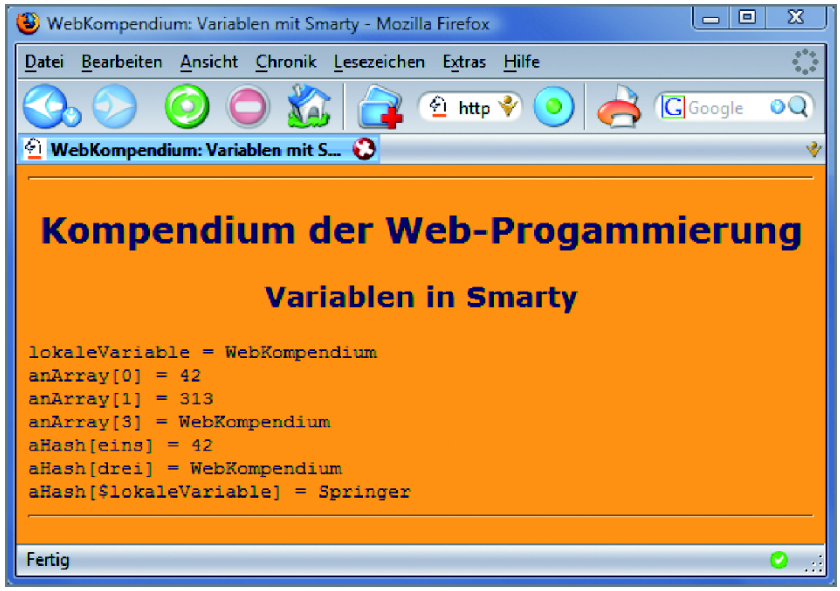



**Abbildung 21.9:**
Ausführen von `variablen.php`

Auch innerhalb eines Templates kann einer Smarty-Variablen direkt ein Wert zugewiesen werden:

```
{ assign var="name" value="value" }
```

#### 21.2.3.2 Die Smarty-Instanz und Smarty-Methoden

Zentral für Smarty ist die *Instanz der Klasse Smarty,* häufig mittels `$smarty` bezeichnet:

```
$smarty = new Smarty;
```

Für dieses Objekt stehen einige nützliche Eigenschaften und Methoden bereit, etwa um Session- und Cookie-Informationen (vgl. Kapitel 25 und 26) zu erhalten:

- `$smarty.server.SERVER_NAME` Name des Servers, der Smarty betreibt;
- Zugriff auf Umgebungsvariablen des Servers wie `$smarty.env.PATH` für die Umgebungsvariable `PATH`;
- Zugriff auf Sessoin-Variablen: `$smarty.session.sessionvar` gibt den Wert der PHP-Sessionvariable ßessionvar“;
- Zugriff auf Cookies: `$smarty.cookies.cooievar`, der Wert des Cookies „cookievar“;
- `$smarty.version` die Version der Smarty-Instanz;
- `$smarty.now` Methode für die aktuelle Serverzeit.

Neben dem eigentlichen Smarty-Objekt stehen weitere Methoden bereit:

- `lower` Umwandlung in Kleinbuchstaben;
- `upper` Umwandlung in Großbuchstaben;
- `nl2br` ersetzt einen normalen Zeilenumbruch durch das HTML-Tag `<BR>`;
- `replace` für Ersetzen;
- `regex_replace` für die Ersetzung mit regulären Ausdrücken (vgl. 10.3.15).

Besondere Bedeutung kommt dem optionalen Laden der Smarty-Konfiguration in PHP zu, welches durch

```
config_load
```

mit der Konfigurationsdatei als zwingendem Argument möglich ist. Die wichtigsten Konfigurationen – alle sind Attributvariablen der Smarty-Instanz – sind dabei

- `SMARTY_DIR` für die Wurzel der Smarty-Installation;
- `$template_dir` die Angabe des Verzeichnisses für die Smarty-Templates; Standard hierfür ist `templates`;
- `compile_dir` gibt das Verzeichnis für die compilierten Smarty-Templates an; Default-Wert ist `templates_c`;
- `$cache_dir` das Verzeichnis für den Cache-Speicher von Smarty, default ist `cache`;
- `$config_dir` (Standard: `configs`) gibt das Verzeichnis für Smarty-Konfigurationsdateien an.

**Zentrale Konfigurationen des Smarty-Systems wie der Ort des Template-Verzeichnisses sollten in eine globale Konfigurationsdatei ausgelagert werden, welche jeweils zu importieren ist. Dadurch wird die Smarty-Instanz deutlich leichter zu pflegen. Ebenso sollte auf eine saubere Aufteilung auf Verzeichnisse für Templates, compilierte Templates etc. geachtet werden.**

#### 21.2.3.3 Kontrollstrukturen in Smarty

Innerhalb der Smarty-Templates stehen einfache *Kontrollstrukturen* bereit, um die durch das Template gesteuerte View variabel zu gestalten; hierzu zählen Verzweigungen und Schleifen.

Die Verzweigung im Smarty-Template wird durch eine Wenn-Dann-Sonst-Struktur implementiert. Die Syntax lautet

```
{if $n == 42}
   HTML-Code: n hat Wert 42
{elseif $n == 313}
   HTML-Code: n hat Wert 313
{else}
   HTML-Code: n hat anderen Wert
{/if}
```

wobei der elseif- und der else-Zweig optional sind.

Im Smarty-Template steht mit der foreach-Schleife eine einfache, für Scriptsprachen typische Iteration über ein Array bereit:

```
{foreach from=$anArray item=i}
   HTML-Code: Wert = <B>{$i}</B>
{/foreach}
```

Das Smarty-Script `schleifen.php` (Abbildung 21.10) mit dem Template `schleifen.tpl` (Abbildung 21.12) zeigt eine solche Iteration über ein Array (Abbildung 21.11)

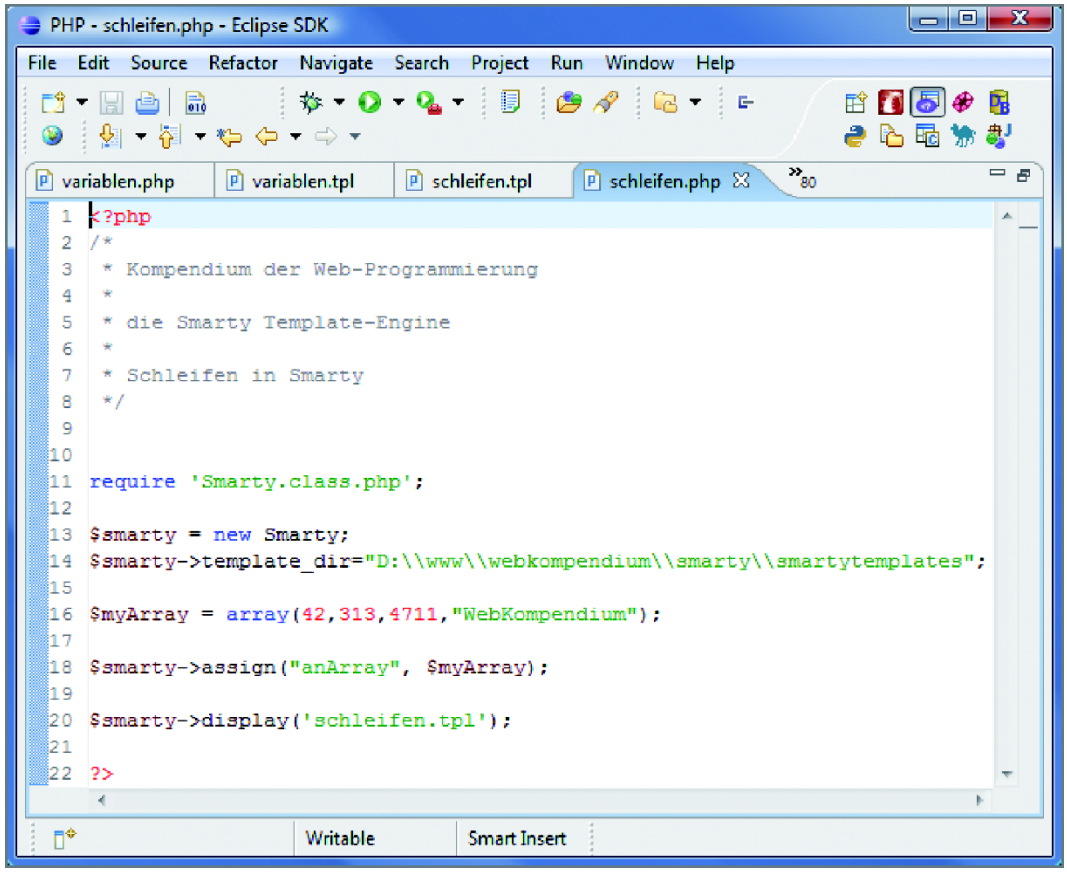



**Abbildung 21.10:**
Schleifen in Smarty `schleifen.php`

**Abbildung 21.11:**
Ausführen von `schleifen.php`

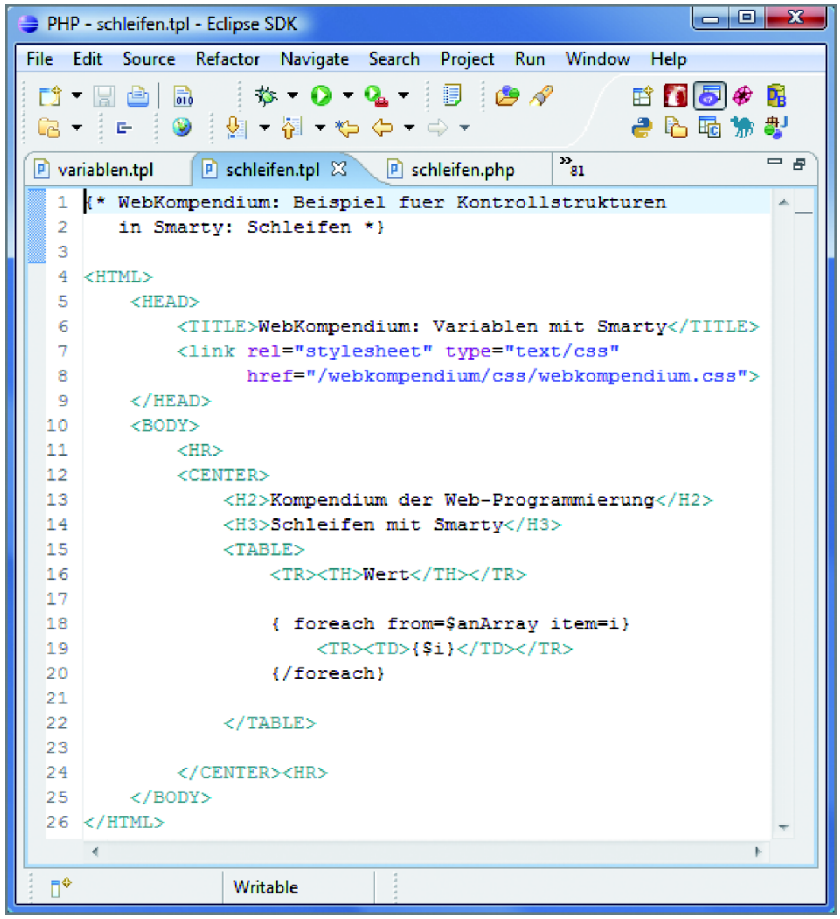



**Abbildung 21.12:**
Template `schleifen.tpl` für `schleifen.php`

### 21.2.3.4 Weitere Möglichkeiten mit Smarty

Häufig nützlich ist der Import von Dateien in die Template-Datei:

```
{  include file="footer.tpl"  }
```

Hierdurch können auch gestufte (hierarchische) Template-Dateien aufgebaut werden.

Neben der `include`-Methode gibt es noch `insert`, nur findet hier kein Caching statt.

Es ist auch möglich, aus einem Smarty-Template eine Internet-Ressource einzubinden. Hierfür steht die Methode `fetch` mit der einfachen Syntax

```
{ fetch file="datei" }
```

bereit, wobei sowohl mittels HTTP als auch mittels FTP über das Netz sowie direkt auf das lokale Filesystem des Webservers zugegriffen werden kann.

Für die Erzeugung von HTML-Code können im Smarty-Template generierende Methoden verwendet werden, ähnlich etwa zu der Codeerzeugung im Perl-Modul CGI.pm (vgl. 10.6). Smarty stellt dafür eine ganze Familie von Methoden der Art

```
html_start(...)
```

bereit, die im Template verwendet werden können.

#### 21.2.3.5 Caching mit Smarty

Die besonders gute Performance von Smarty resultiert aus dem effizienten *Caching* dieser Template-Engine.

## 21.3 Zusammenfassung: Template-Engines und Design Patterns

Mit den modernen Template-Engines ist es einfach möglich, die View von der eigentlichen Programmierung abzuspalten und so zu sehr „sauberem“ Code zu gelangen. Wie eingangs beschrieben ist dies von großem Vorteil, da für die Anpassung der View nun reine HTML-Kenntnisse ausreichen.

Aber diese Template-Engines können keine saubere Softwarearchitektur erzwingen, dies ist weiterhin eine wichtige Aufgabe des Entwicklers. Der folgende Code in einem Smarty-Template soll dies zeigen:

```
{if $smarty.session.user and
      ( $user_type eq "editor" or $user_type eq "admin" )}
   <input type=checkbox name=edit value="y">
   edit <br>
{/if}
```

Hier liegt echte Programmierlogik im Template, welches somit nicht mehr nur die reine Sicht steuert, sondern auch Programmieraufgaben übernimmt, die Trennung der View ist missglückt. Eine saubere Implementierung könnte etwa lauten:

```
{if $edit_flag}
   <input type=checkbox name=edit value="y">
   edit <br>
{/if}
```

Hier stellt die PHP-Programmierung ein Flag `$edit_flag` bereit, die Logik hierfür liegt ganz in der PHP-Programmierung.

**Template-Engines erleichtern die Abtrennung der View, sie wird aber nicht erzwungen; auch mit Template-Engines muss auf eine saubere Architektur der Anwendung geachtet werden.**

# 22

# Das Python-Framework django

Für die fortgeschrittene Web-Programmierung ist das *Framework* ein weit verbreiteter Begriff, der allerdings recht breit verwendet wird. Unter *Framework* wird faktisch eine Sammlung von Softwarekomponenten verstanden, welche letztlich eine Anwendungsarchitektur vorgeben.

Heute liegen zahlreiche Framewokrs für Web-Applikationen vor; zu ihnen gehören die hier vorgestellten django und Ruby on Rails; andere sind etwa PHP-Nuke und ZOPE. Unter dem Dach der Apache Foundation werden ebenfalls Frameworks entwickelt; hier sind beispielsweise Velocity, Struts und das Framework Cocoon zu nennen.

*django* [web] ist ein modernes und rasch wachsendes Framework für Web-Applikationen, welches vollständig in der Scriptsprache Python realisiert ist (vgl. 12).

Aufgrund des Einsatzes von Python ist django plattformunabhängig; das Framework kann darüber hinaus mit mehreren verschiedenen relationalen Datenbankmanagementsystemen eingesetzt werden. django ist von seiner Arbeitsweise und seinem Funktionsumfang dem Framework Ruby on Rails (vgl. 23) nicht unähnlich.

**Abbildung 22.1:** django-Logo

## 22.1 Komponenten und Betrieb

Das django-Framework wird, wie häufig üblich, mit einem eigenen Anwendungsserver ausgeliefert. Dieser ermöglicht den schnellen Einsatz des Frameworks, allerdings ist es für einen stabilen und performanten Einsatz unbedingt ratsam, django in Verbindung mit dem Apache-Webserver zu verwenden.

Für die Anbindung von django an Apache ist ferner das Apache-Modul mod_python sehr wichtig, da hiermit die Performance der Python-Anwendungen wesentlich erhöht werden kann.

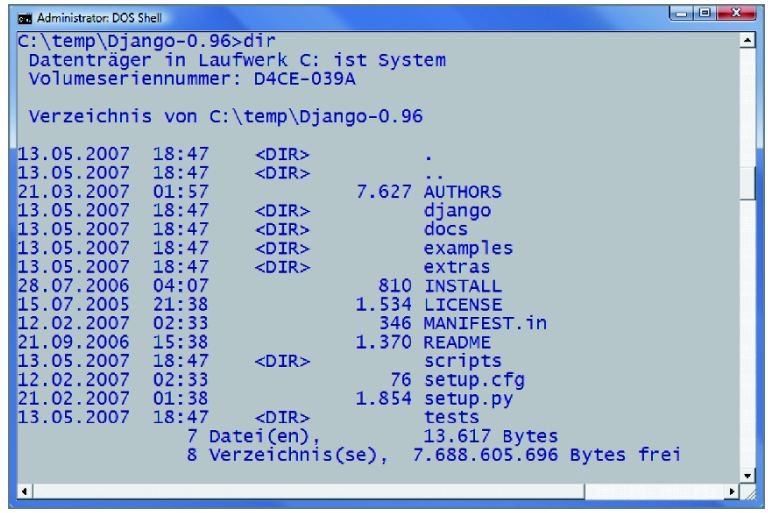



**Abbildung 22.2:** django-Verzeichnis nach dem Entpacken

## 22.2 Installation von django

django wird als Archiv im Format `tar.gz` ausgeliefert, welches mit den üblichen Werkzeugen entpackt werden muss. Danach liegt ein Ordner mit den benötigten Dateien vor (Abbildung 22.2) ; durch

```
python setup.py install
```

wird django installiert, was konkret bedeutet:

- es wird das zentrale Installationsverzeichnis `django` als Unterordner des Python-Systemverzeichnisses `Lib/site-packages` für Python-Ergänzungen angelegt;
- die entsprechenden Unterordner werden angelegt;
- die Python-Dateien werden entsprechend kopiert.

**Bei der Installation ist auf die ausreichenden Rechte zu achten, da die entsprechenden Dateien in das Python-System kopiert werden; unter Unix ist faktisch die root-Kennung dafür notwendig (etwa über Ausführen mittels `sudo python setup.py install`), unter Vista am einfachsten durch Ausführen in einer als Administrator ausgeführten Shell (Abbildung 22.3).**

**Abbildung 22.3:**
Installation von django mittels `python setup.py install`

Nach der Installation ist im Python-Systemordner eine Ordnerstruktur nach Abbildung 22.4 vorhanden.

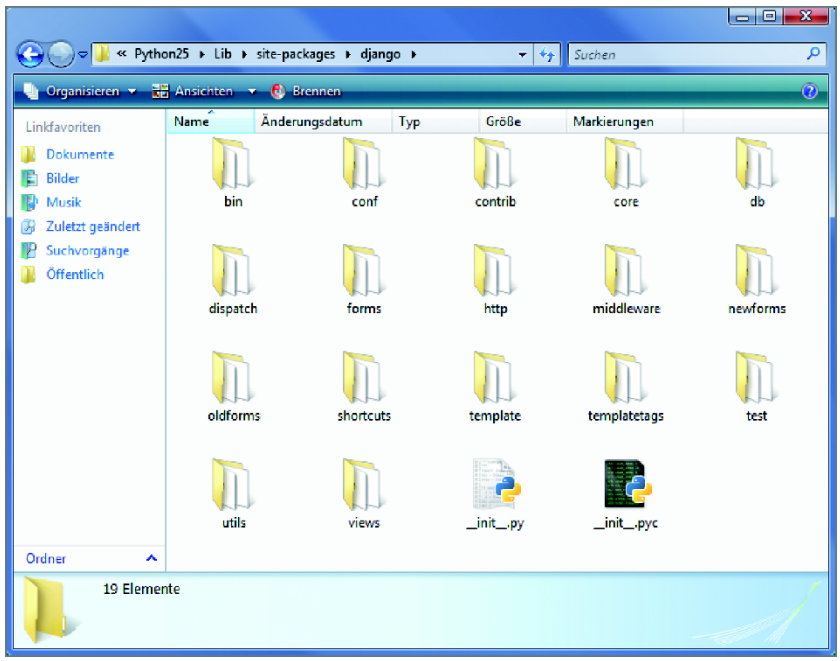



**Abbildung 22.4:**
Erfolgreiche Installation von django

### 22.2.1 Unterstützte Datenbankmanagementsysteme

Praktisch sollte django mit allen Datenbankmanagementsystemen einsetzbar sein, für welche Treiber nach der Python DB-API 2.0 verfügbar sind; die Entwickler verweisen allerdings besonders auf:

- PostgreSQL mit dem psycopg-Treiber;
- MySQL mit dem MySQLdb-Treiber (vgl. 12.5);
- SQLite mit dem pysqlite-Treiber.

### 22.2.2 Test der Installation

Nach der Installation kann im interaktiven Python-Interpreter mittels

```
import django
```

überprüft werden, ob ein funktionsfähiges django zur Verfügung steht.

## 22.3 Ein Beispielprojekt mit django

### 22.3.1 Erzeugen der Beispielapplikation

Nun kann in einem beliebigen Verzeichnis eine django-Anwendung installiert werden; dies geschieht durch die Anweisung

```
django-admin.py startproject <name_des_projekts>
```

Dabei ist das Python-Script `django-admin` im Unterordner `Lib/site-packages/django/bin` zu finden; dieser Ordner sollte zur Vereinfachung zur Umgebungsvariablen `PATH` hinzugefügt werden.

Nach Erzeugen eines solchen Projektes – im Beispiel mit dem Namen `buecher` – werden vier Python-Scripte angelegt (Abbildung 22.5):

- `__init.py__`: leere Datei, durch welche dieser Ordner als Python-Package gekennzeichnet wird und django beim Import entsprechend verfahren kann;
- `manage.py`: Kommandozeilen-Script für den Zugriff auf das django-Framework;
- `settings.py`: Konfigurationsparameter des django-Projektes;
- `urls.py`: Web-Adressen des django-Projektes.

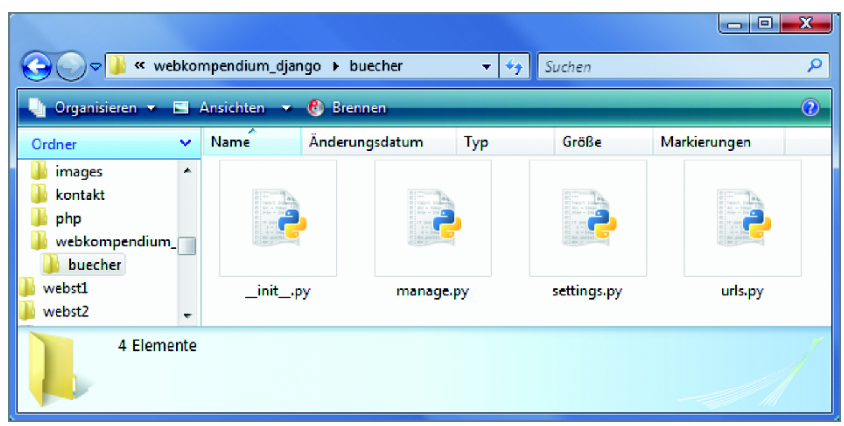



**Abbildung 22.5:** Leere django-Anwendung

Diese Dateien sind zunächst nur im Sourcecode gespeichert; nach dem ersten Starten des Applikationsserver liegen dann auch die compilierten `.pyc`-Dateien vor.

### 22.3.2 Starten des Applikationsservers

Wird django zunächst mit dem integrierten Applikationsserver, welcher ebenfalls vollständig in Python implementiert ist, betrieben, so ist der Server mittels

```
python manage.py runserver <port>
```

zu starten; der Parameter `<port>` gibt dabei den tcp-Port an; wird dieser Parameter nicht angegeben, läuft der Server auf dem tcp-Port 8000.

Abbildung 22.6 zeigt das Starten des Servers auf Port 8086, danach ist über die URL

```
http://<rechneradresse>:8086
```

ein erster Web-Zugriff auf django möglich (Abbildung 22.7).

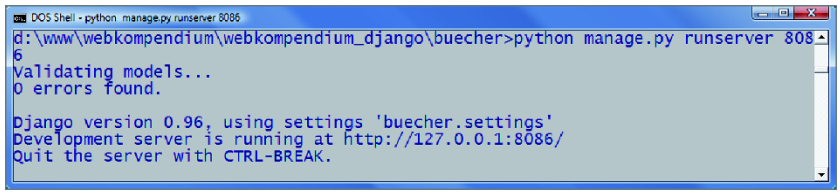



**Abbildung 22.6:** Starten des django-Applikationsservers

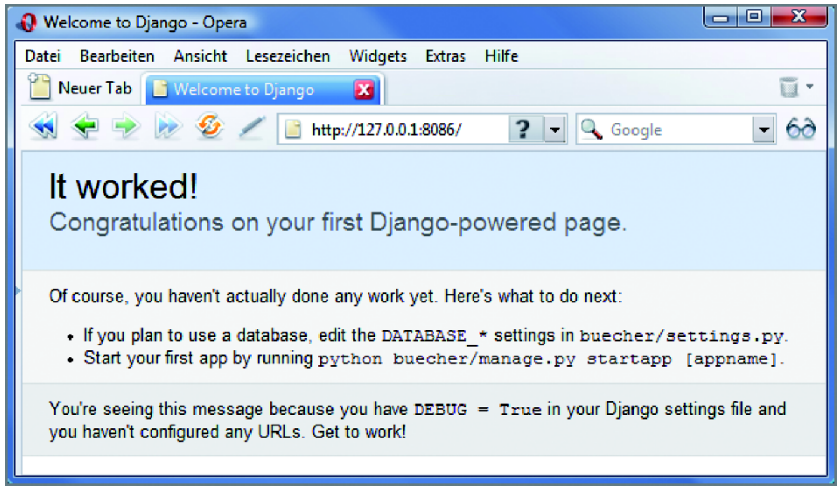



**Abbildung 22.7:** Erster Zugriff auf die django-Anwendung

### 22.3.3 Datenbankkonfiguration

Um die Beispielanwendung fortzuführen, ist im nächsten Schritt die Konfiguration der Beispielanwendung notwendig; dies geschieht durch einfaches Editieren der Datei `settings.py`. Hier sind für das jeweilige DBMS die üblichen Einstellungen (Name des Servers, Port, Name der Datenbank, User/Kennwort) anzugeben (Abbildung 22.8).

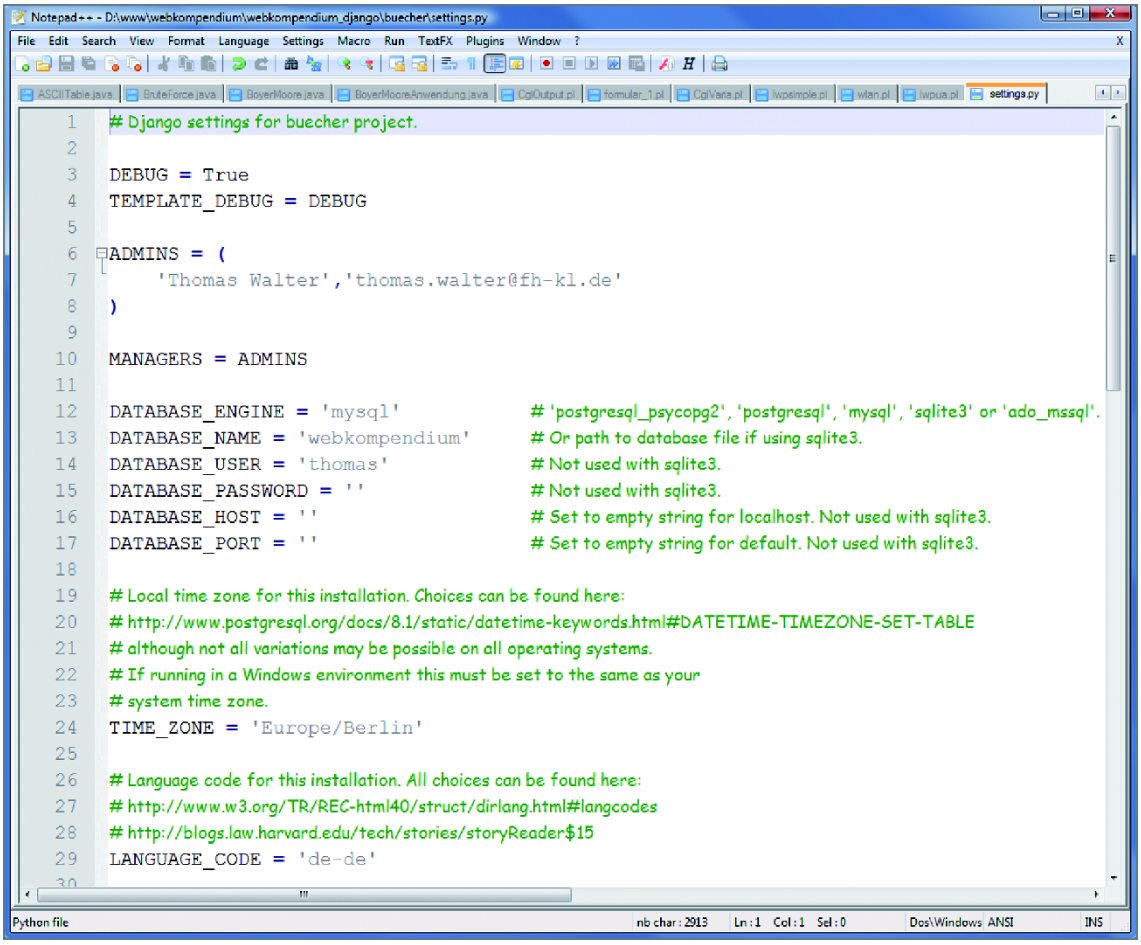



**Abbildung 22.8:** Datenbank-Einstellungen in `settings.py`

### 22.3.4 Vom Projekt zur Anwendung

Ebenfalls in dieser Datei, am Fuß, wird festgelegt, welche Anwendungen von django geladen werden (Direktiven `INSTALLED_APPS`); zur Verfügung stehen dabei

- `django.contrib.auth`: die django-interne Authentifizierung;
- `django.contrib.contenttypes`: ein Framework für verschiedene content-types;
- `django.contrib.sessions` Session-Management mit django;
- `django.contrib.sites`: die Verwaltung verschiedener Sites mit einer django-Instanz.

Diese Anwendungen benötigen in der Datenbank eigene Tabellen, welche mittels

```
python manage.py syncdb
```

automatisiert angelegt werden können.

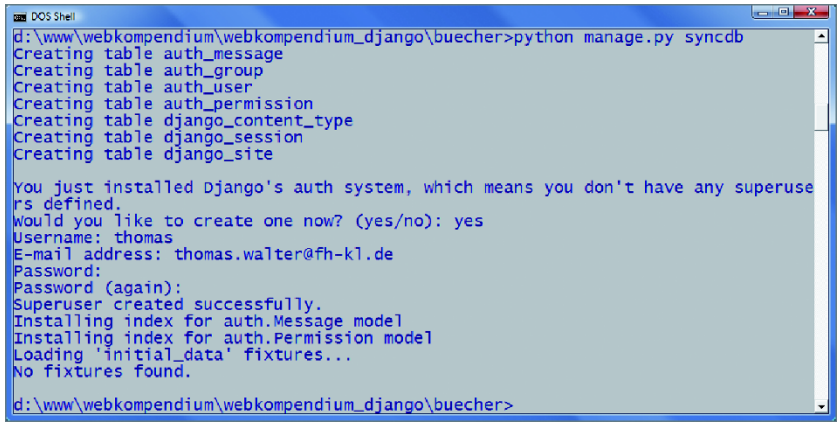



**Abbildung 22.9:** Erzeugung der Datenbanktabellen für django mit allen vier Anwendungen

Damit ist der django-Rahmen für einzelne Anwendungen erzeugt: das django-Projekt.

Als nächstes soll eine einzelne django-Anwendung für den Zugriff auf unsere Datenbanktabellen erzeugt werden; dafür wird im Projekt-Ordner `buecher` mittels

```
python manage.py startapp buecheranwendung
```

eine Anwendung erzeugt. Konkret bedeutet dies, dass ein Unterordner `buecheranwendung` erzeugt wird, in welchem die zwei Dateien

- `models.py`
- `views.py`

neben der leeren Datei `__init.py__` generiert werden.

**Das django-Projekt bündelt django-Anwendungen und Konfigurationen.**

Die hier in dem Beispiel erzeugte Anwendung muss in der Konfigurationsdatei des Projektes, `settings.py`, noch aktiviert werden; dies geschieht ebenfalls im letzten Direktiven-Block durch

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'buecher.buecheranwendung'
)
```

### 22.3.5 Model

django definiert in der Datei `models.py` das Model nach dem MVC-Paradigma (vgl. 4.3). Models in django sind dabei Python-Klassen, die von der django-Klasse `models.Model` abgeleitet sind und welche eine Datenstruktur beschreiben. Abbildung 22.10 zeigt ein Beispiel.

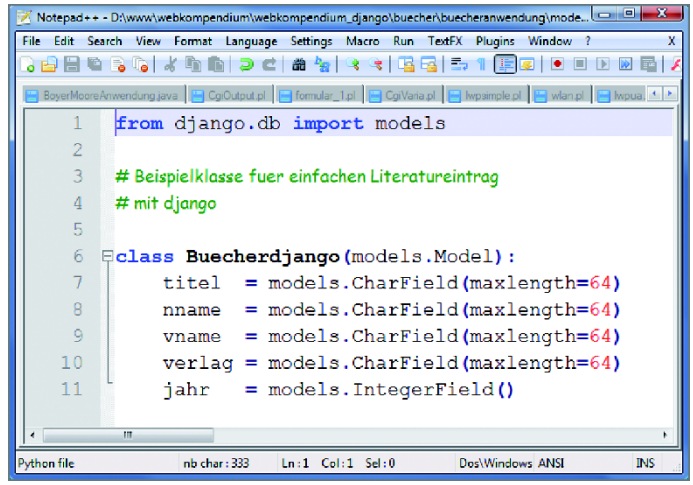



**Abbildung 22.10:** Python-Code des Models

django liefert durch die im Projekt-Ordner ausgeführte Anweisung

```
python manage.py sql <anwendung>
```

den zur Erzeugung der Datenbanktabellen benötigten SQL-Code. Hier im Beispiel:

```
python manage.py sql buecheranwendung

BEGIN;
CREATE TABLE `buecheranwendung_buecherdjango` (
    `id` integer AUTO_INCREMENT NOT NULL PRIMARY KEY,
    `titel` varchar(64) NOT NULL,
    `nname` varchar(64) NOT NULL,
    `vname` varchar(64) NOT NULL,
    `verlag` varchar(64) NOT NULL,
    `jahr` integer NOT NULL
);
COMMIT;
```

**Listing 22.1:** django-generiertes SQL-Script für den Datenbankaufbau

django liefert hier für das jeweilige DBMS spezifischen Code, also durchaus verschiedene SQL-Anweisungen etwa für MySQL und PostgreSQL. Dieser SQL-Code muss über ein Datenbank-Tool ausgeführt werden.

Weitere nützliche Optionen stellt `mamange.py` bereit:

- `python manage.py validate <anwendung>` – Überprüft das Model auf Fehlerfreiheit;
- `python manage.py sqlinitialdata <anwendung>` – Übersicht über notwendige Konfigurationsinformationen;
- `python manage.py sqlclear <anwendung>` – SQL-Anweisungen (`DROP TABLE`) zum Löschen der Datenbanktabellen der Anwendung;
- `python manage.py sqlindexes <anwendung>` – SQL-Anweisungen (`CREATE INDEX`) zum Aufbau von Datenbank-Indices für die Anwendung;
- `python manage.py sqlall <anwendung>` – zusammengefasste SQL-Anweisungen von sql, sqlinitialdata und sqlindexes.

### 22.3.6 Zugriff auf Model

django bietet mittels

```
python manage.py shell
```

die Möglichkeit, den interaktiven Python-Interpreter zu starten und dabei alle Einstellungen des django-Projektes verwenden zu können. Hier kann über (vgl. Abbildung 22.11)

```
from <projekt>.<anwendung>.models import <Modelklasse>
```

die Klasse eines Modells geladen werden, welche anschließend verwendet werden kann, etwa zur Erzeugung einer Instanz. Dieses Objekt kann dann mittels der Methode

```
save()
```

persistent in der Datenbank abgelegt werden. Die Methode

```
all()
```

ermöglicht die Ausgabe aller in der Datenbank zugeordneten Datensätze.

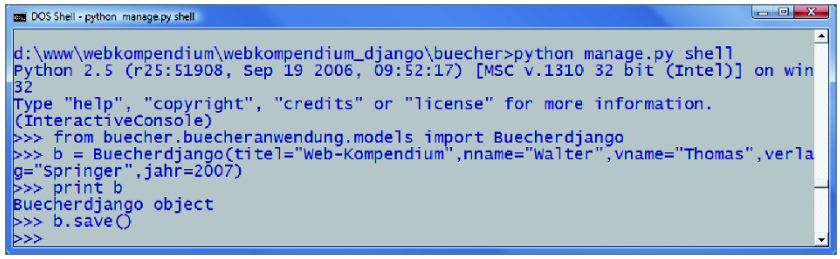



**Abbildung 22.11:** Ausgabe der Elemente der Datenbanktabelle

Um eine sinnvolle Ausgabe zu erhalten, ist es notwendig, die Python-Methode `__str__` mit der Bedeutung der Java-Methode `toString` zu überschreiben; im Beispiel geschieht dies in Abbildung 22.12; das damit erzielte Ergebnis ist in Abbildung 22.13 enthalten.

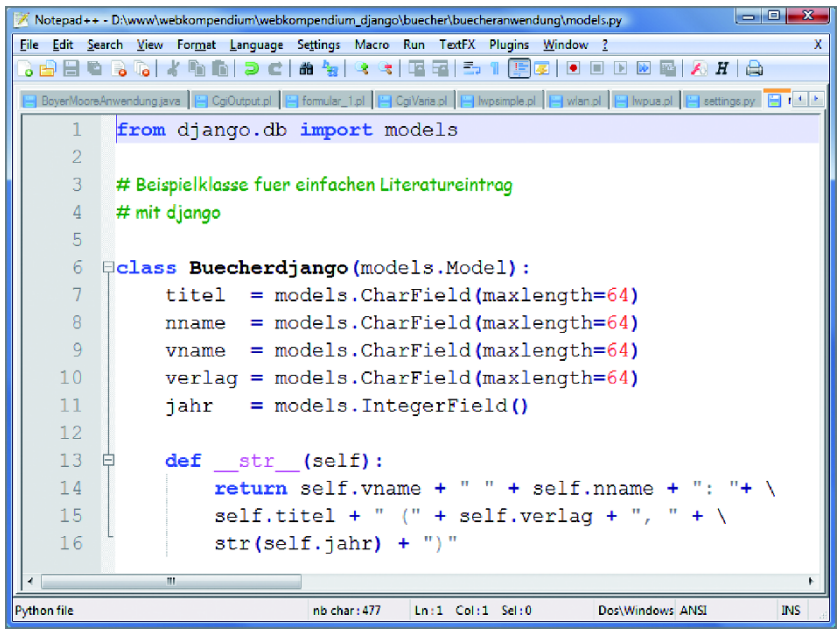



**Abbildung 22.12:** Model-Klasse mit überschriebener Methode `__str__`

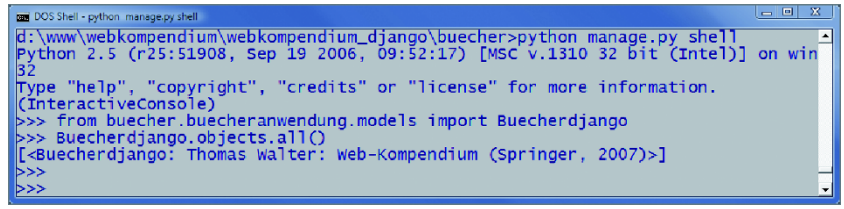



**Abbildung 22.13:** Ausgabe der Elemente nach Überschreiben der Methode `__str__`

### 22.3.7 Mapping der URL

In der zentralen Projekt-Konfigurationsdatei `settings.py` gibt die Direktive

```
ROOT_URLCONF = 'buecher.urls'
```

die Konfiguration des *URL-Mappings* der Anwendungen an, hier die Datei `urls.py`. Diese leitet eingehende HTTP-Requests an die entsprechenden django-Anwendungen weiter; durch den stärkeren Einsatz von regulären Ausdrücken kann diese Weiterleitung umfassender gestaltet werden. Abbildung 22.14 zeigt ein einfaches Beispiel.

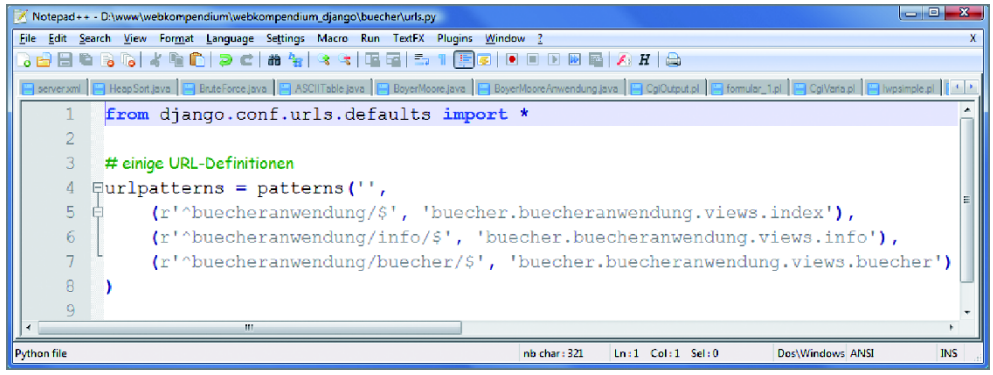



**Abbildung 22.14:** Beispiel für `urls.py`

### 22.3.8 Einfache View

Innerhalb einer django-Anwendung wird die *View* über die Python-Datei `views.py` definiert. In dieser Datei sind für die einzelnen Views nach dem Mapping aus der Datei `urls.py` Methoden zu definieren; im Beispiel nach Abbildung 22.15 sind dies die Methoden `info`, `index` und `buecher`.

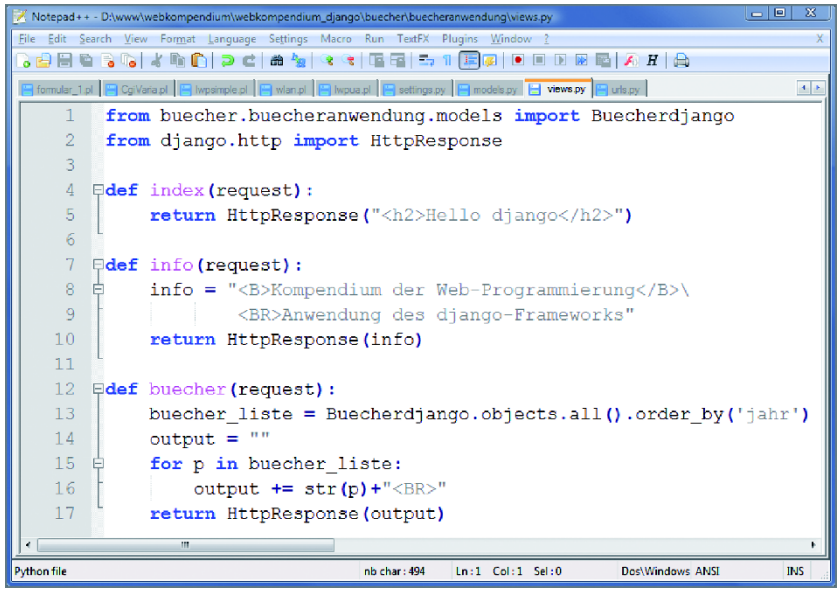



**Abbildung 22.15:**
Beispiel für `views.py`

Während `info` und `index` einfache Textausgabe wie HelloWorld produzieren, liefert `buecher` eine Liste mit allen Einträgen in der Datenbank.

Das Ergebnis über die URL

```
http://<rechnername:port>/buecheranwendungen/buecher/
```

ist in Abbildung 22.16 zu sehen.

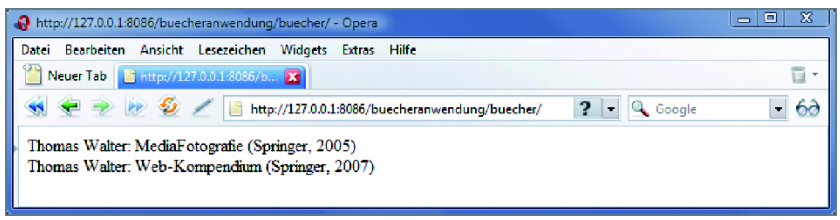



**Abbildung 22.16:**
Einfache View

### 22.3.9 View mit HTML-Templates

Um eine HTML-formatierte View zu erzeugen, könnte man in die entsprechenden Methoden der Python-Datei `views.py` den entsprechenden HTML-Code integrieren – was aber zu sehr unübersichtlichem und schwer zu wartendem Code führen würde. django liefert hierfür ein besseres Verfahren, basierend auf dem verbreiteten Prinzip der *HTML-Templates,* wie es etwa Smarty für PHP (vgl. Kapitel 21) und auch Ruby on Rails (vgl. Abschnitt 23.8.5) umsetzen.

Durch das Template wird die HTML-Sicht von dem Python-Code getrennt. Dies hat viele Vorteile, neben dem klareren Code kann der Web-Designer mit den für ihn vertrauten Dateien arbeiten, ohne tiefer in die Python-Syntax einsteigen zu müssen.

Nun soll zunächst die index-Methode über ein Template ein Design erhalten; hierfür müssen durch

```
from django.template import Context, loader
```

die Klassen `Context` und `loader` geladen werden. Die Index-Methode kann dann etwa folgendermaßen lauten:

```
def index(request):
    t = loader.get_template('buecheranwendung/index.html')
    c = Context({message:'<h2>Hello django</h2>'})
    return HttpResponse(t.render(c))
```

`Context` ist eine key-value-Struktur, welche Python-Variablen – hier ist es nur eine Konstante – unter einem Bezeichner – hier `message` – bereit stellt, damit die Werte der Python-Variablen im Template als Template-Variablen genutzt werden können.

Beim Aufruf der index-Methode in diesem Stadium wird jedoch nur eine Fehlermeldung erzeugt (Abbildung 22.17), da das angegebene Template noch nicht existiert.

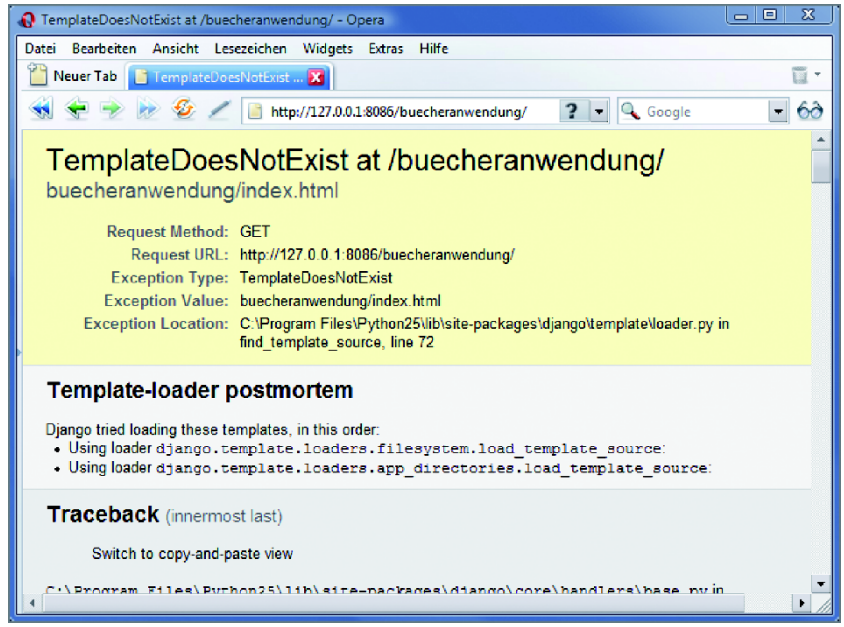



**Abbildung 22.17:** django-Fehlermeldung, da das Template noch nicht existiert

Für das erfolgreiche Einbinden von Templates bedarf es im ersten Schritt eines Verzeichnisbaums der Templates eines Projektes; in dieser Verzeichnisstruktur wird ein entsprechendes Unterverzeichnis für die Templates zur jeweiligen django-Anwendung angelegt.

Im einfachsten Fall ist ein django-Template eine html-Datei, welche lediglich durch

```
{{ context_variable }}
```

den Wert einer Context-Variablen ausgibt; Abbildung 22.18 zeigt ein solches Template (welches die CSS-Datei unabhängig von django von einem Webserver lädt), das damit erzielte Ergebnis ist in Abbildung 22.19 zu sehen.

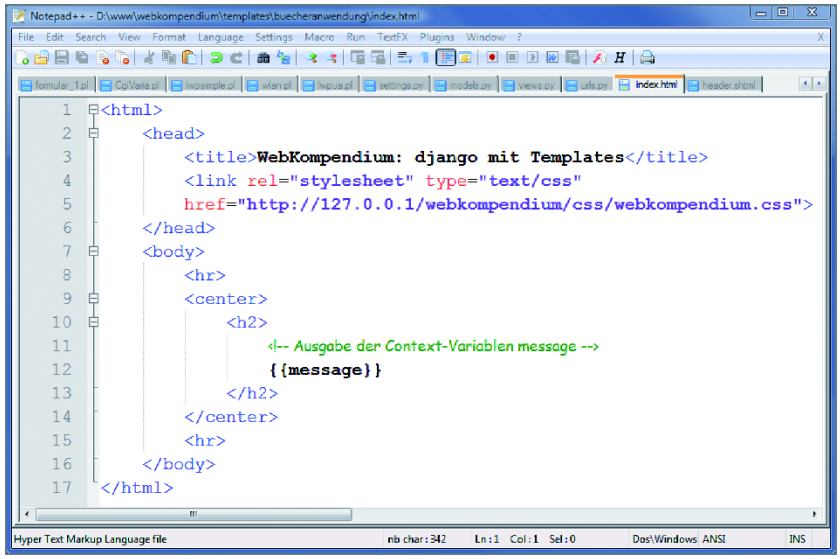



**Abbildung 22.18:**
Einfaches django-Template

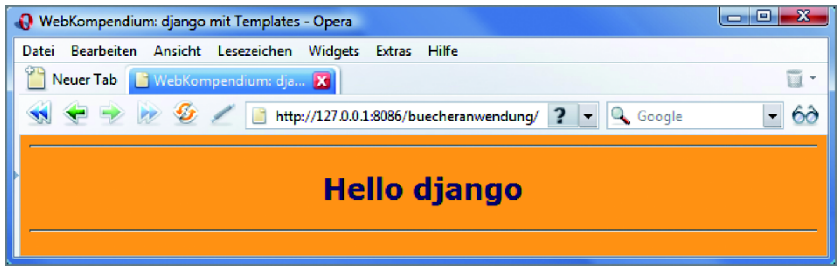



**Abbildung 22.19:**
django-Anwendung mit Template

Natürlich leistet die django-Template-Engine deutlich mehr. Um dies zu zeigen, wird die View für die Anwendung `buecher` noch derart geändert, dass ein Template (`buecher.html`) eingebunden wird und im Context die Trefferliste als Python-Liste veröffentlicht wird. Abbildung 22.20 zeigt die derart geänderte View.

Das zugehörige Template `buecher.html` beinhaltet nun wesentlich mehr Funktionalität als nur die Ausgabe von Context-Variablen: In den Klammern `{% ... %}` können komplexere Anweisungen der django-Template-Sprache, die in ihrer Syntax naheliegenderweise an Python orientiert ist, enthalten sein. In diesem Beispiel erfolgt eine Fallunterscheidung: Wenn es Datenbankeinträge gibt, werden diese in einer for-Schleife ausgegeben (die eigentliche Ausgabe erfolgt wieder über die überschriebene Methode `__str__` des Python-Modells); gibt es keine Ausgabe, erfolgt eine einfache Text-Information. Abbildung 22.21 zeigt das entsprechende Template; das damit erzielte Ergebnis ist in 22.22 zu sehen.

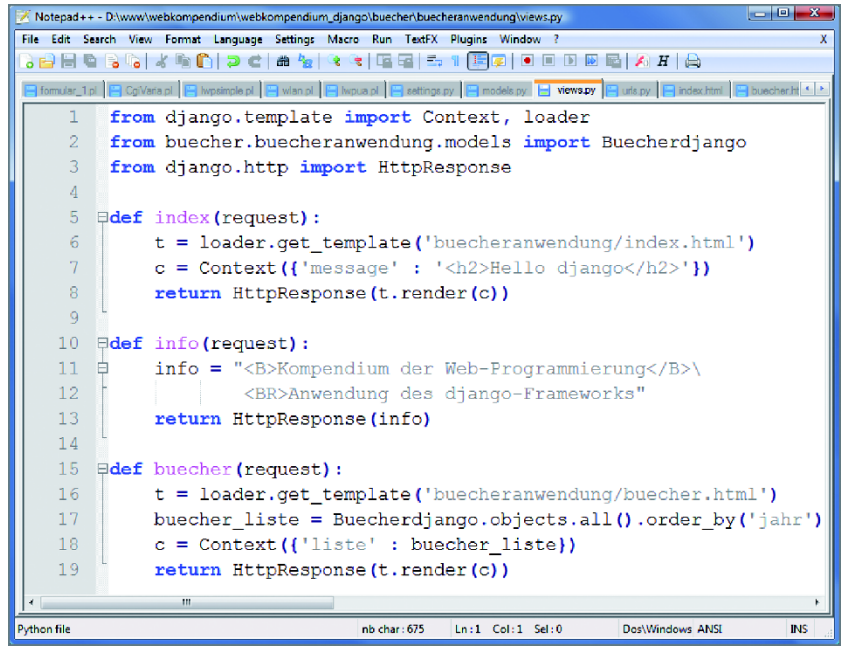



**Abbildung 22.20:** Geänderte django-View in der Methode `buecher`: neuer Context und Einbinden des Templates

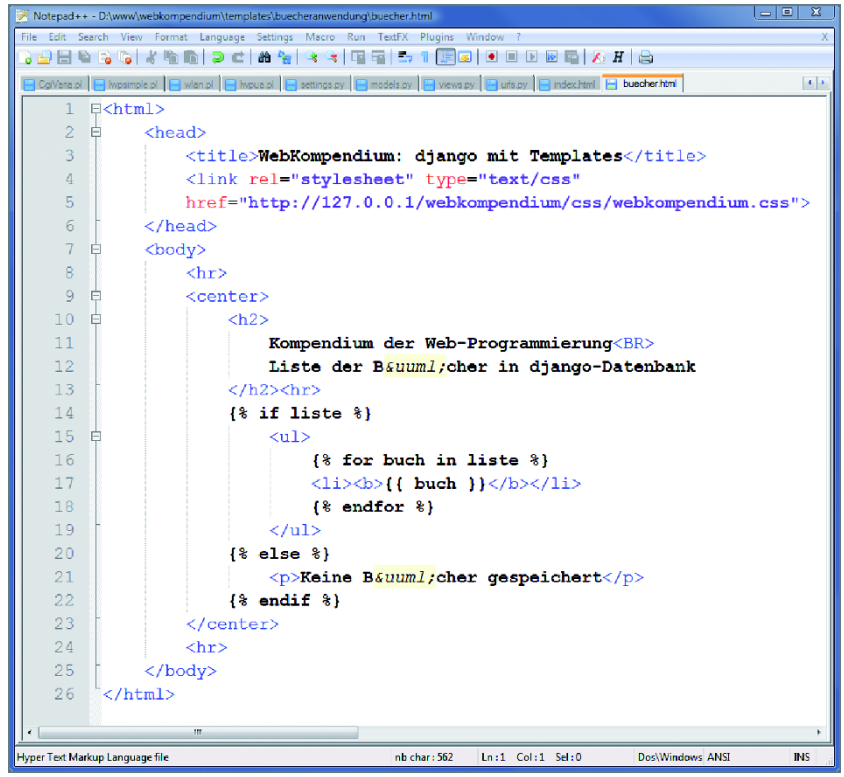



**Abbildung 22.21:** Template für die Ausgabe aller Datenbankeinträge über das Context-Element `liste`

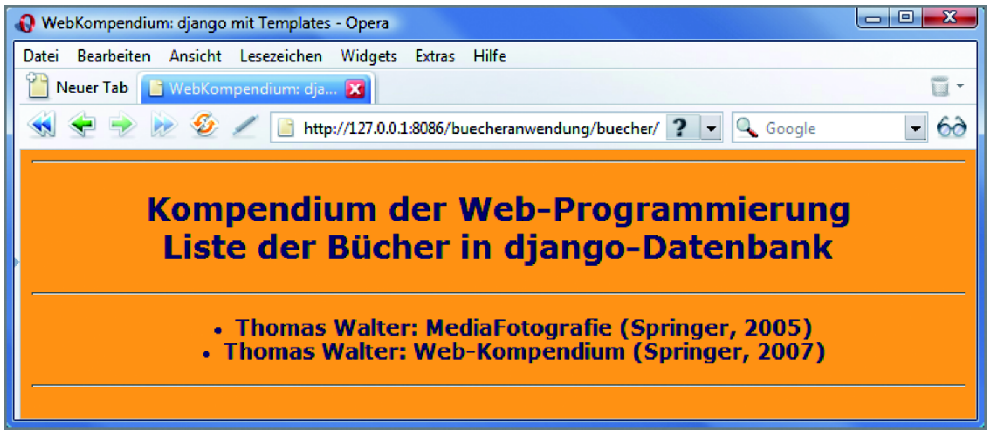



**Abbildung 22.22:** Ausgabe der fertigen django-Anwendung

## 22.4 Das Python-Framework ZOPE

Hier wurde mit django ein „junges“, sich noch stark entwickelndes Framework vorgestellt. Ein anderes Beispiel für ein Framework mit Python ist ZOPE, ein sehr großes und weit entwickeltes Framework [web]. Das Content Management System Plone basiert auf ZOPE.

## 22.5 Zusammenfassung

Mit django steht ein leistungsstarkes Framework für die schnelle Web-Entwicklung mit Python bereit. Wichtig ist die fest integrierte Trennung von Model und View durch die eigenen django-Klassen.

Ein weiterer Bestandteil das django-Frameworks ist eine starke Template-Engine auf der Höhe der Zeit.

django ist im aktuellen Funktionsumfang fast so leistungsstark wie Ruby on Rails (vgl. Kapitel 23). Vor allem für die stärker mit Python vertrauten Entwickler ist django daher sehr interessant. Momentan werden nur wenige DBMS unterstützt, deren Zahl wird aber weiter wachsen, und die wichtigen Open Source-Systeme sind bereits vorhanden.

Ähnlich wie Ruby on Rails liegen die Vorteile von django im schnellen Erstellen eines funktionierenden Prototyps und momentan weniger in der Entwicklung großer Produktionssysteme.

# Das Ruby-Framework Ruby on Rails

Eines der momentan beliebtesten Frameworks für die Entwicklung von Web-Applikationen mit Datenbankbezug ist *Ruby on Rails* (Rails) [web]. Im Juli 2004 stellte David Heinemeier Hansson dieses Framework erstmals vor. Vorteile von Rails sind insbesondere die schnelle Entwicklung serverseitiger Web-Applikationen mit Datenbankanbindung und die durch das Framework praktisch erzwungene Softwarearchitektur im Model-View-Controller-Paradigma MVC (vgl. 4.3).

Insgesamt bietet Rails viele Vorteile, die insbesondere auf eine schnelle Entwicklung lauffähiger Anwendungen zielen. Deshalb liegt der momentan wichtigste Einsatz von Rails auch im Fast Prototyping und nicht in der Entwicklung komplexer Produktionssysteme.

**Abbildung 23.1:**
Ruby on Rails-Logo

**Abbildung 23.2:**
Ruby on Rails Hauptseite

## 23.1 Das Prinzip von Rails

Das schnelle Entwickeln von Web-Applikationen mit Rails resultiert nicht unwesentlich aus dem *Prinzip von Rails:*

**Prinzip von Rails:**
**Convention over Configuration.**

Dahinter verbirgt sich die Idee, bewusst auf komplexe Konfigurationsmöglichkeiten zu verzichten, indem strikte Namenskonventionen eingehalten werden und man schon durch die Namensgebung vieles festlegt, was nicht nochmals definiert werden muss:
**Don't repeat yourself (DRY).**

Dieses Prinzip wird etwa dadurch deutlich, dass zu einer Model-Klasse `Book` eine Datenbanktabelle `books` (Plural) gehört. Ein weiteres Beispiel ist etwa ein durch die Namenskonvention fest vorgegebenes integriertes Mapping der URL auf die vorhandenen Controller-Methoden.

## 23.2 Scaffolding

Unter *Scaffolding* (Bauprinzip) versteht das Rails-Framework eine weitere Besonderheit zur schnellen Entwicklung: Mittels einfacher Bausteine wird eine Web-Applikation „on the fly" direkt erzeugt. Die wichtigsten dieser Bausteine werden unter CRUD zusammengefasst:

- **C**: Create
- **R**: Read
- **U**: Update
- **D**: Delete

Dahinter verbergen sich die wichtigsten Datenbank-Operationen zur Veränderung der Datenbestände der Anwendung, die somit nicht einzeln implementiert werden müssen, sondern von dem Rails-Framework generiert werden.

## 23.3 Webserver für Rails

Für den Betrieb einer Rails-Applikation wird natürlich ein Webserver benötigt. Hier stehen zwei prinzipielle Wege bereit:

- Auch zum Rails-Framework gehört ein einfacher Webserver, welcher mit dem Framework ausgeliefert wird: WEBrick. Dieser ist direkt lauffähig und einfach zu starten.
- Natürlich kann Rails auch mit dem Apache-Webserver betrieben werden. Um Rails-Anwendungen performant auszuführen, sollten aber unbedingt sowohl das fastCGI-Modul (vgl. Kapitel 19) als auch das Ruby-Modul mod_ruby (vgl. 13.5.3) verwendet werden.

## 23.4 Unterstützte Datenbankmanagementsysteme

Rails beinhaltet den Datenbankzugriff für einige Datenbankmanagementsysteme, so dass keine spezielle Ruby-Datenbank-Anbindung installiert werden muss (vgl. 13.6).

Momentan werden die folgenden Systeme unterstützt:

- MySQL;
- PostgreSQL;
- IBM DB2;
- Oracle;
- Microsoft SQL Server.

Allerdings wird sich diese Liste beim aktuell raschen Fortschritt von Rails zügig erweitern.

## 23.5 Rails-Module und das MVC-Pattern

Rails basiert zunächst auf fünf *Modulen,* welche bereits das *MVC-Pattern* (vgl. 4.3) grundlegend anlegen:

- Model: Active Record;
- View: Action Pack (Request-/Response-Behandlung);
- Action-Mailer (Email);
- Action Web-Service;
- Active Support (Ruby-Erweiterungen).

Das Modul Active Record, welches für das Model zuständig ist, verwaltet dabei auch die Datenbankanbindung zu einem der Datenbanksysteme aus Abschnitt 23.4.

Die eigentliche View wird von einer Klasse ActionView abgeleitet und kann verschiedene Sichten wie HTML und XML mittels spezieller Rails-Templates erzeugen.

**Abbildung 23.3:** Ruby on Rails API-Dokumentation im Web

## 23.6 Installation von Rails

Die *Installation des Rails-Frameworks* ist denkbar einfach, da hier, ähnlich zu ppm für Perl, ein eigener Paketmanager verwendet wird: Ruby gem. Die eigentliche Installation von Rails wird dann über gem gesteuert – eine funktionierende und aktuelle Ruby-Version wird natürlich bereits vorausgesetzt.

Ruby gem wird als Archiv (zip oder tar.gz) ausgeliefert; nach Entpacken steht ein Ordner bereit, welcher das Installationsscript `setup.rb` enthält; dieses wird plattformunabhängig über

```
ruby setup.rb
```

installiert, wobei auf ausreichende Rechte zu achten ist.

Nach der Installation kann mittels

```
gem
```

der Paketmanager genutzt werden (Abbildung 23.4).

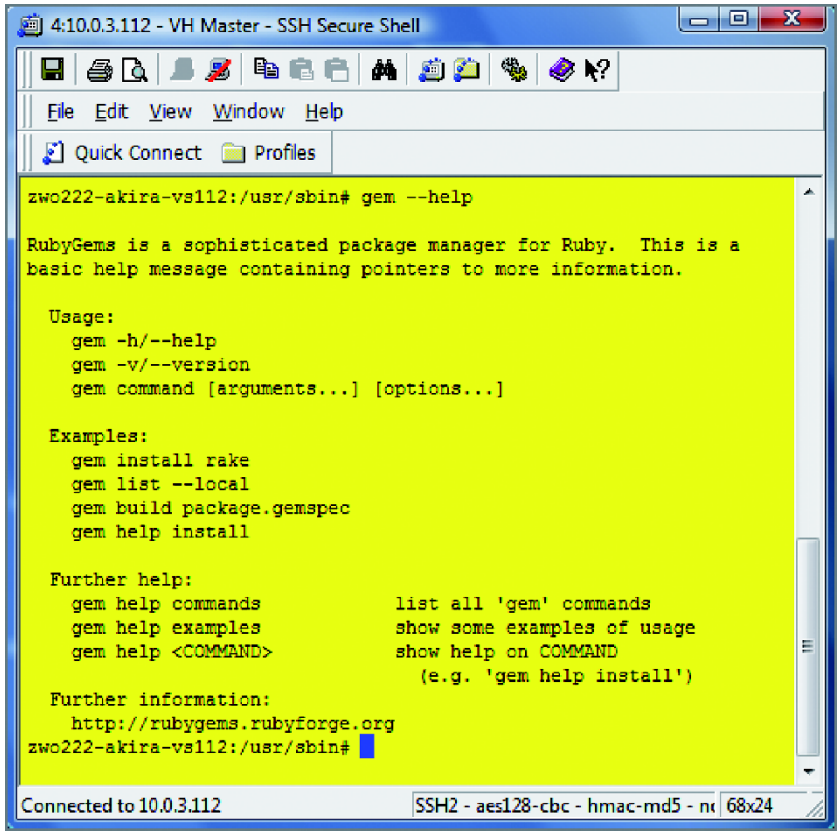



**Abbildung 23.4:** Starten von gem

### Die eigentliche Rails-Installation

Nachdem Ruby und gem vorhanden sind, erfolgt die *Rails-Installation* über den Paketmanager:

```
gem install rails --include-dependencies
```

Dies installiert direkt die aktuelle Rails-Version sowie die zugehörige Dokumentation.

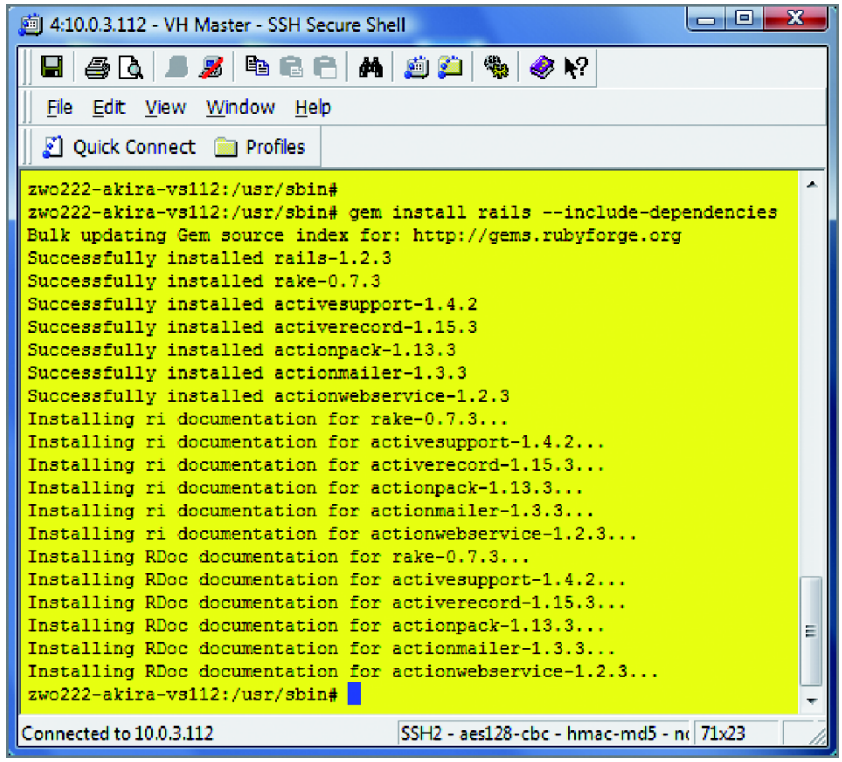



**Abbildung 23.5:**
Rails-Installation mit gem

radrails

**Abbildung 23.6:**
radrails-Logo

## 23.7 Entwicklungsumgebung für Rails

Prinzipiell genügt für die Softwareentwicklung mit Rails ein Ruby-fähiger Editor, es stehen aber sehr leistungsfähige Systeme bereit.

Hervorzuheben ist hier wieder die Entwicklungsumgebung Eclipse, für welche ein leistungsfähiges Plugin speziell für die Anwendungsentwicklung mit Ruby on Rails zur Verfügung steht: radrails [web] (Rapid Application Development IDE). Aktuell wird radrails in die weiterhin frei verfügbare Aptana-Entwicklungsumgebung integriert. Das radrails-Plugin selbst basiert auf einem weiteren Eclipse-Plugin, welches aktuell ebenfalls zu Aptana migriert: RDT, das Ruby Development Tool.

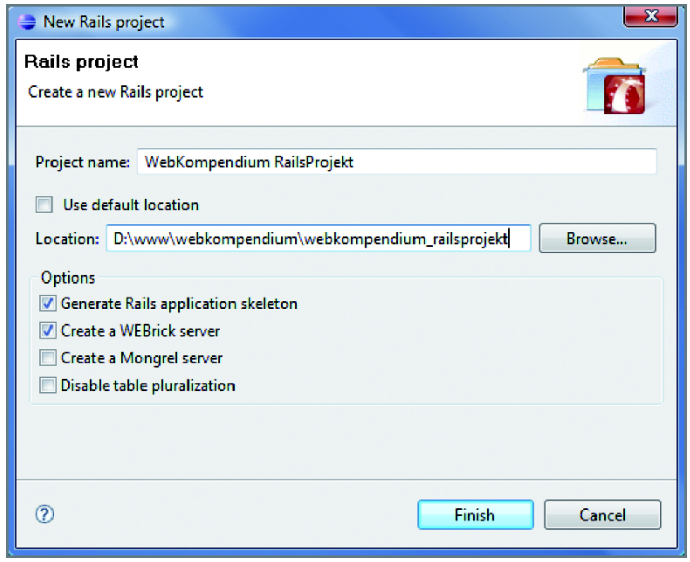



**Abbildung 23.7:**
Anlegen eines Rails-Projekts in Eclipse

Mit radrails können in Eclipse direkt Rails-Projekte angelegt werden (Abbildung 23.7), und es stehen umfangreiche Werkzeuge zur Verfügung (Abbildung 23.8).

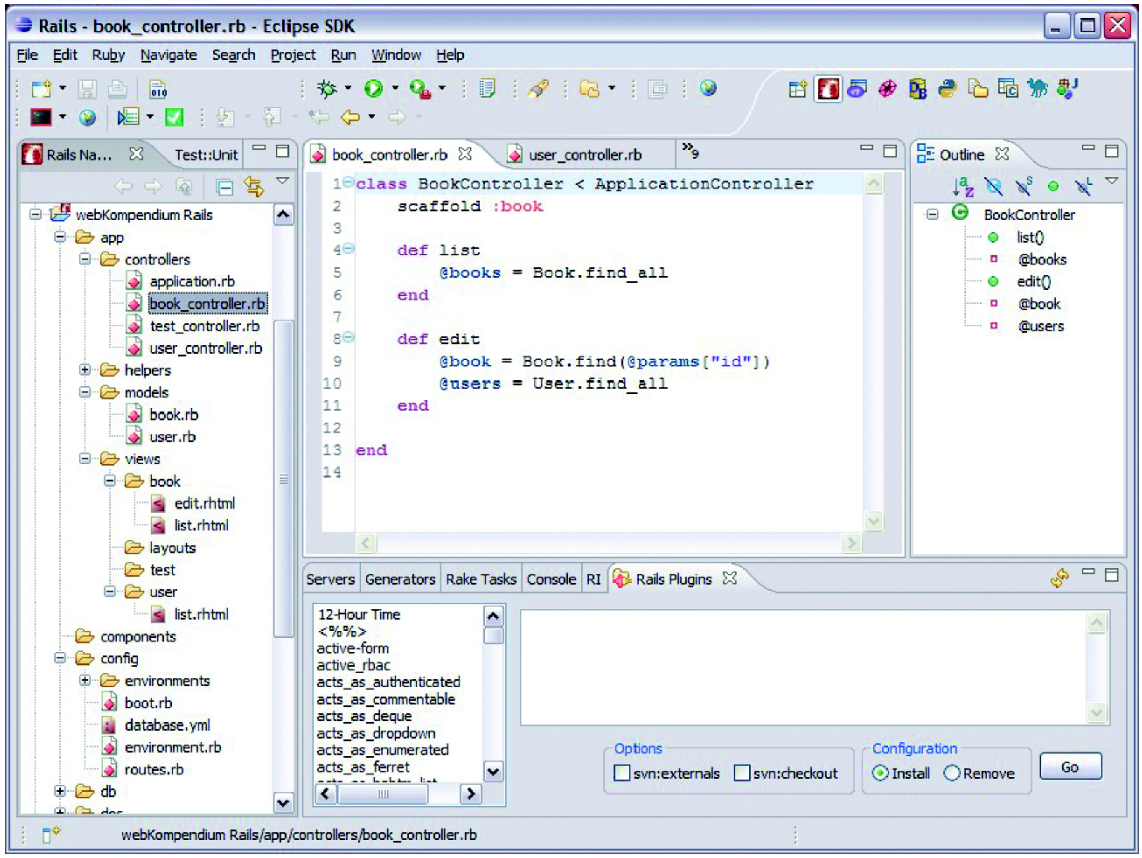



**Abbildung 23.8:**
radrails in Eclipse

## 23.8 Eine Beispielanwendung mit Rails

Um die Funktionsweise von Rails zu verdeutlichen, soll eine Beispielanwendung mit Rails entworfen werden, welche sich an dem Beispiel nach Kapitel 7 orientiert. Es soll dabei wieder die MySQL-Datenbank verwendet werden, welche nach 23.4 von Rails unterstützt wird.

Dabei wird ein Datenmodell für Bücher entworfen, welches als Model in Rails angelegt wird und mit verschiedenen Views angezeigt werden kann; die Views werden später über Templates gesteuert. Dabei kommt das bereits erwähnte Scaffolding intensiv zum Einsatz, welches viele Methoden automatisch erzeugt und etwa die Controller-Funktionalität abdeckt.

### 23.8.1 Erzeugen der Rails-Anwendung

Im ersten Schritt ist eine Rails-Anwendung zu erzeugen. Dies geschieht mittels des nach der Installation vorhandenen `rails`-Scripts (vgl. Abbildung 23.10) durch die Syntax

```
rails <pfad_zur_rails_anwendung>
```

Im konkreten Beispiel wird im Ordner `www/webkompendium` die Anwendung `webkompendium_rails` angelegt (Abbildung 23.9).

**Abbildung 23.9:**
Erzeugen der Rails-Anwendung `webkompendium_rails`

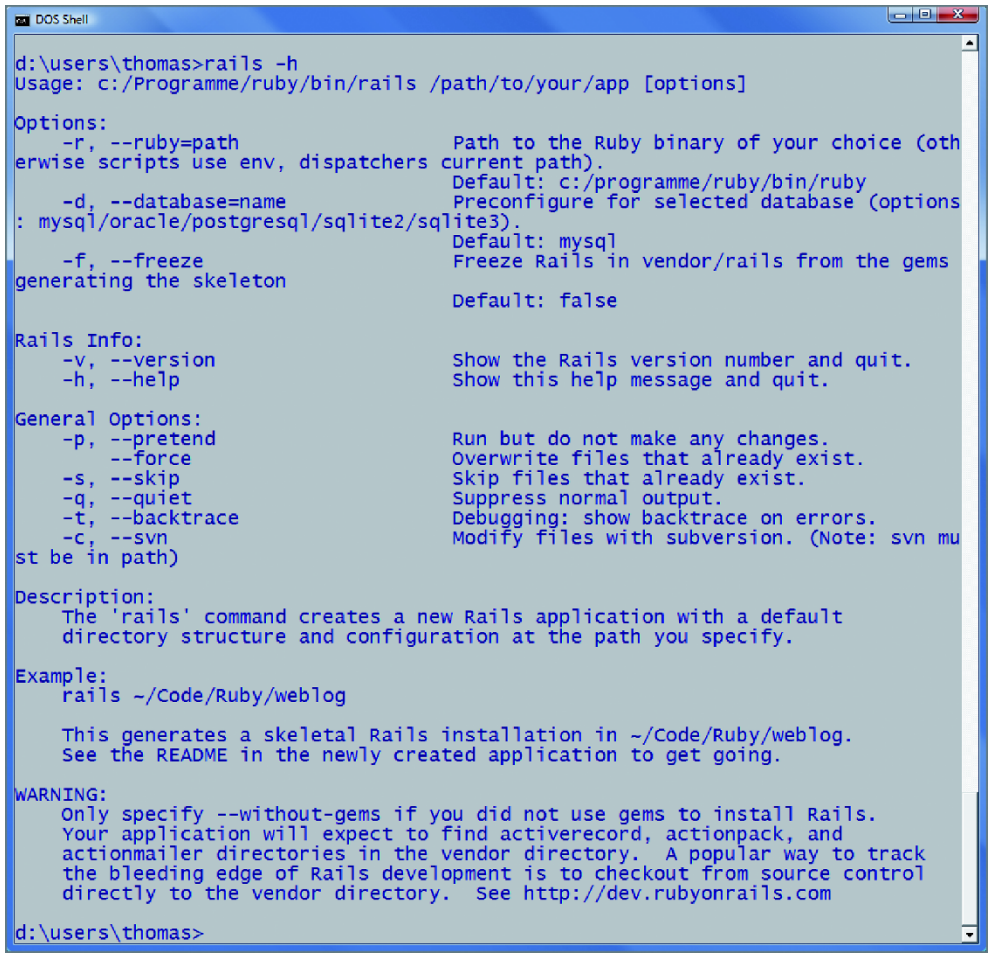



**Abbildung 23.10:**
Optionen für `rails`

Durch das Anlegen der Rails-Anwendung werden zahlreiche Unterordner und Dateien in dem angegebenen Projektordner angelegt (Abbildung 23.11). Hierdurch unterscheiden sich die internen Strukturen von Ruby on Rails und django, da django zu einem Projekt einzelne Anwendungen in eigenen Unterordnern zuordnet.

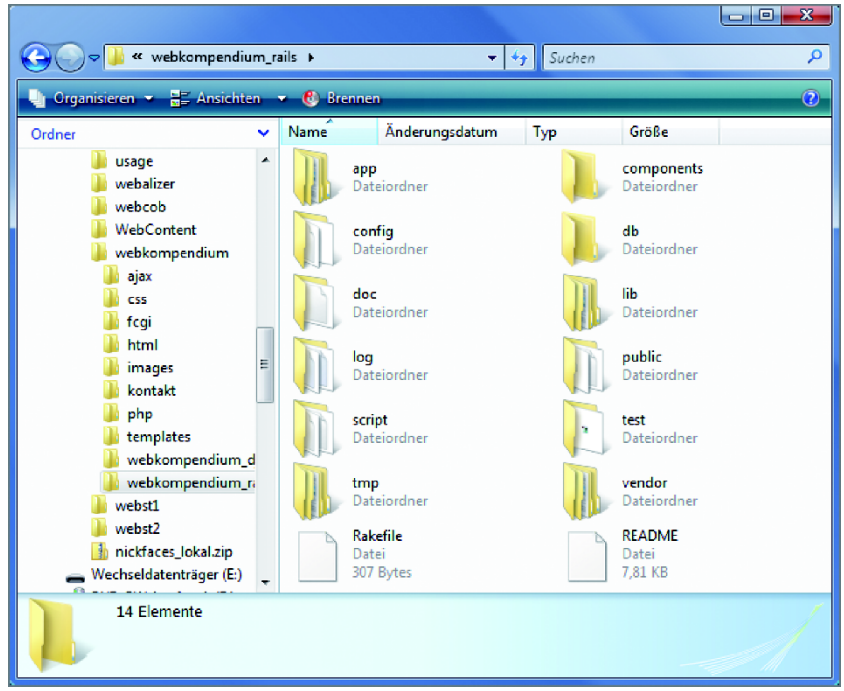



**Abbildung 23.11:** Verzeichnisstruktur des neuen Rails-Projektes

### Starten der Anwendung

Nach diesem Anlegen einer leeren Rails-Anwendung soll diese zunächst gestartet werden; dafür ist ein Webserver-Zugriff notwendig. Es kann, wie bereits erwähnt, hierfür sowohl der Rails-eigene Server WEBrick als auch der Apache verwendet werden:

- WEBrick: Im Projektordner der Rails-Anwendung wird durch

  ```
  ruby script/server
  ```

  das Ruby-Script `server` im Unterordner `script` ausgeführt; dadurch wird WEBrick auf dem Standard-Port 3000 gestartet (Abbildung 23.12).

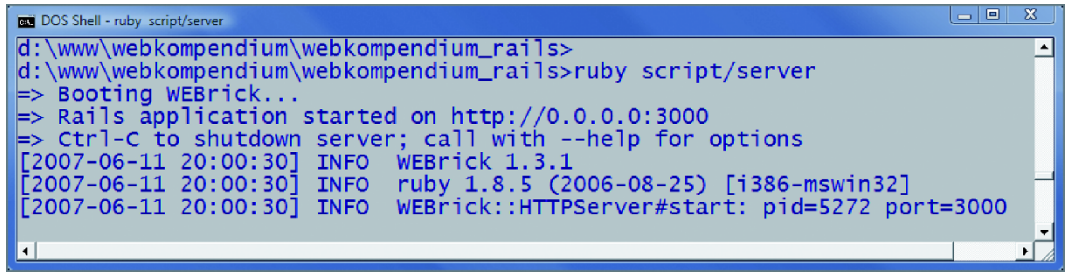



**Abbildung 23.12:** Starten von WEBrick

Der WEBrick-Server kann über Parameter beim Starten grundlegend konfiguriert werden; etwa ist der zu verwendende tcp-Port einstellbar. Abbildung 23.13 zeigt die verfügbaren Parameter.

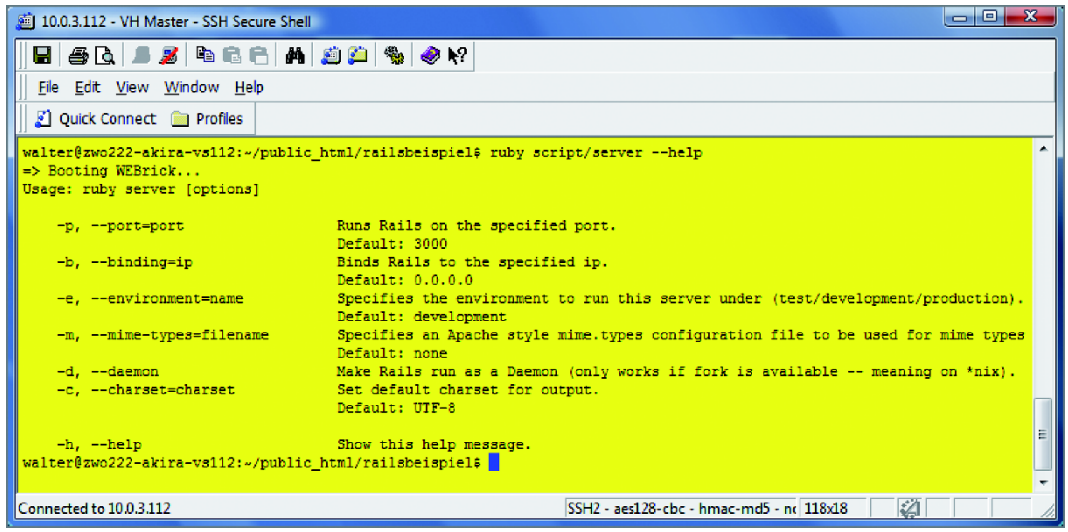



**Abbildung 23.13:** Startparameter des WEBrick-Servers

- Um eine Rails-Anwendung mit dem Apache-Webserver zu nutzen, ist der Weg über die Definition eines virtuellen Hosts sinnvoll; hierbei wird eine virtuelle Apache-Instanz auf einem speziellen tcp-Port eingerichtet, für welche insbesondere die DocumentRoot-Direktive und einige weitere Regeln zu beachten sind (die Abbildung 23.14 zeigt den konkreten Fall des vorgestellten Beispiels); dies geschieht entweder direkt in der `httpd.conf` oder in einer von dieser importierten Konfigurationsdatei wie `httpd-vhosts.conf`.

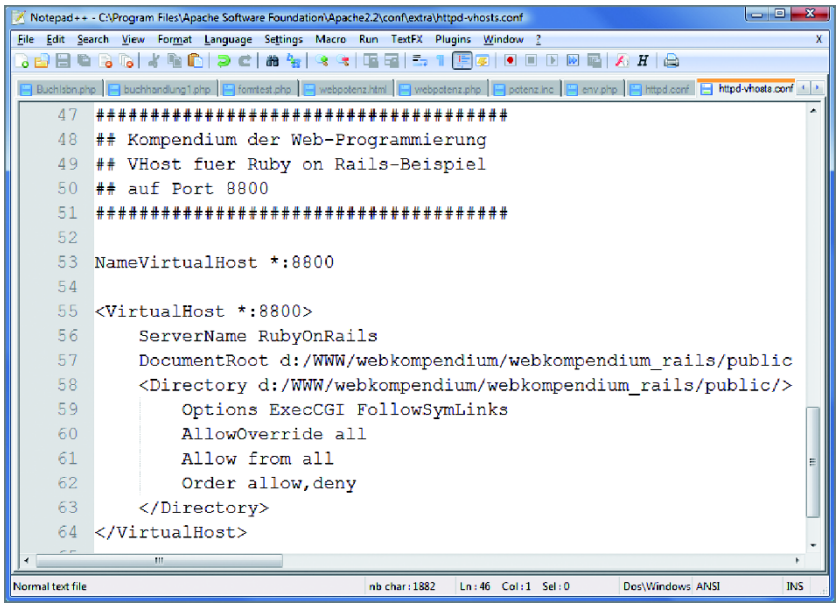



**Abbildung 23.14:** Definition des VHosts von Apache für Rails

Wenn ein Webserver für die Rails-Anwendung verfügbar ist, kann die Default-Startseite für das Projekt abgefragt werden (Abbildung 23.15).

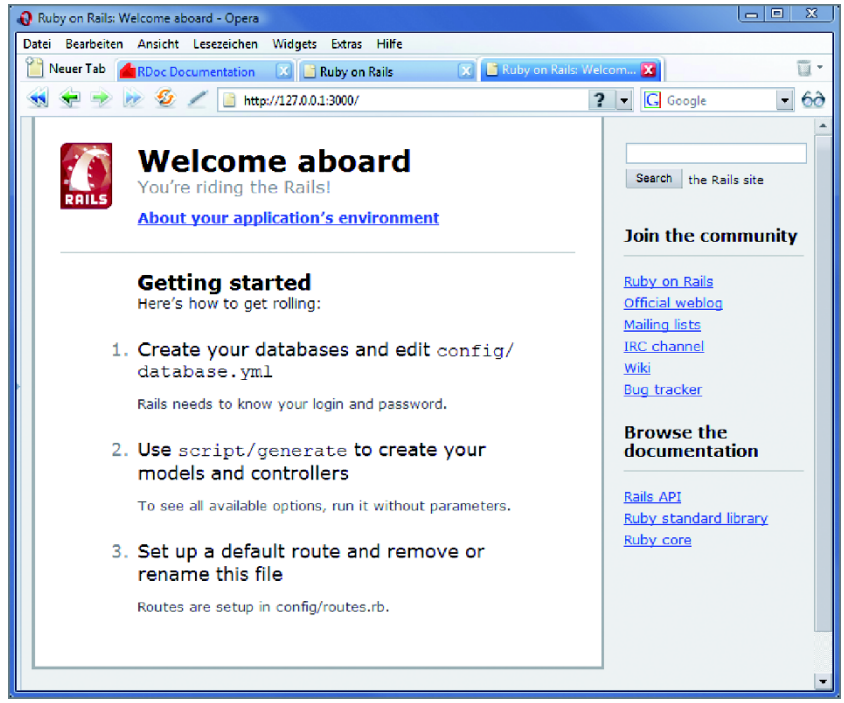



**Abbildung 23.15:** Default-Startseite des Rails-Projektes

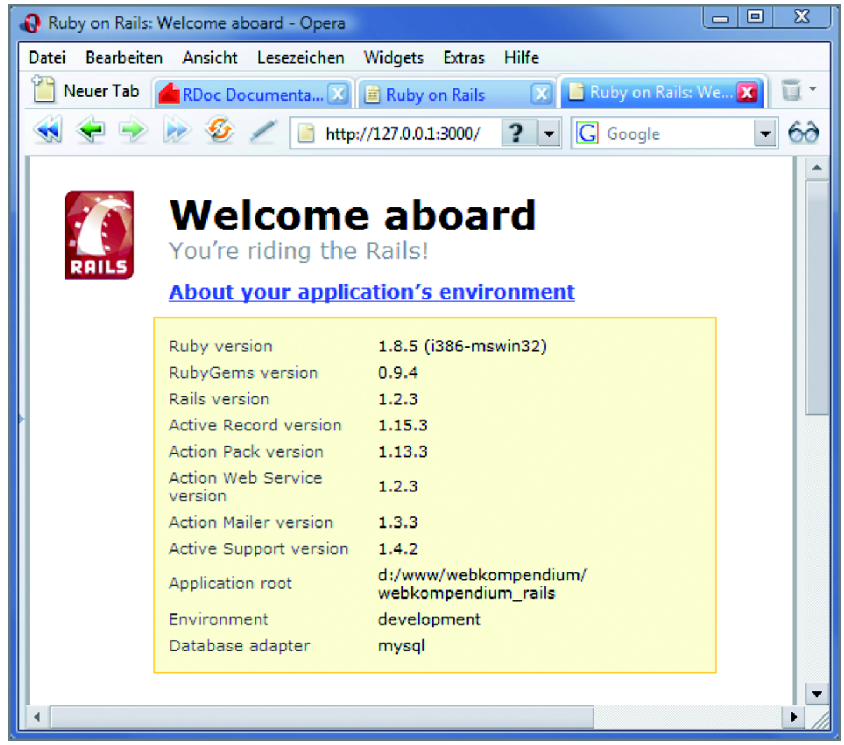


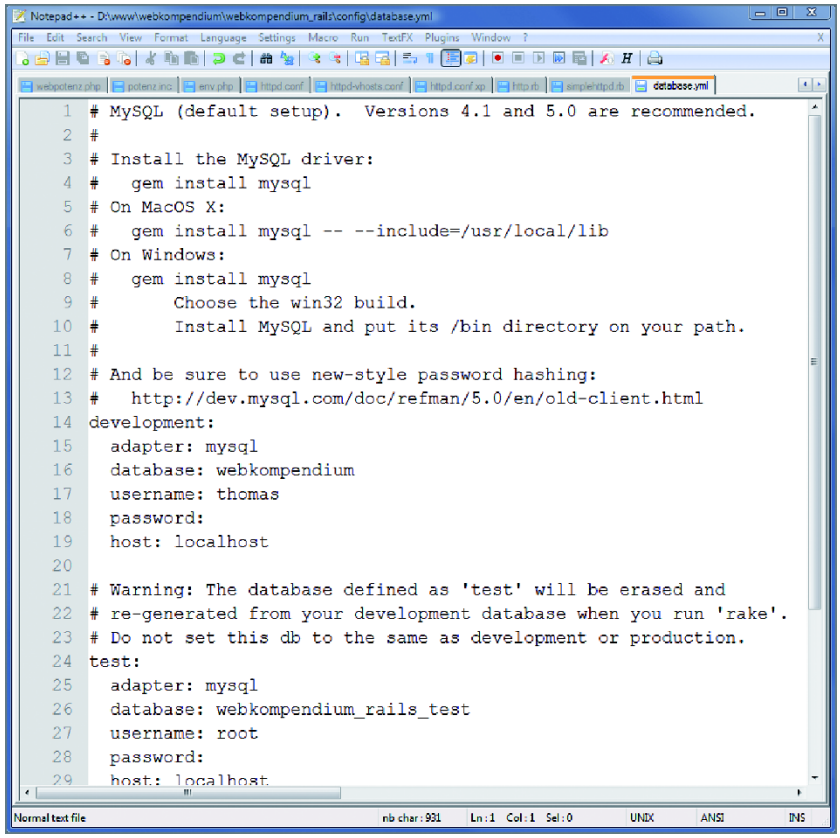

**Abbildung 23.16:** Konfigurationsinformation des Rails-Projektes

### 23.8.2 Konfiguration der Datenbank-Anbindung

Der erste Schritt beim weiteren Umgang mit Rails ist die Konfiguration der Anbindung an ein Datenbankmanagementsystem (die aktuell unterstützten Systeme sind in 23.4 aufgeführt). Dies geschieht in der Konfigurationsdatei `database.yml` im Unterordner `config` des Rails-Projektes.

Die Datenbank-Konfiguration erlaubt den Zugriff auf drei verschiedene Datenbanken für Development, Test und Production.

**Nach der Datenbankkonfiguration ist der Webserver neu zu starten, damit die Änderungen wirksam werden.**




```
# MySQL (default setup).  Versions 4.1 and 5.0 are recommended.
#
# Install the MySQL driver:
#   gem install mysql
# On MacOS X:
#   gem install mysql -- --include=/usr/local/lib
# On Windows:
#   gem install mysql
#       Choose the win32 build.
#       Install MySQL and put its /bin directory on your path.
#
# And be sure to use new-style password hashing:
#   http://dev.mysql.com/doc/refman/5.0/en/old-client.html
development:
  adapter: mysql
  database: webkompendium
  username: thomas
  password:
  host: localhost

# Warning: The database defined as 'test' will be erased and
# re-generated from your development database when you run 'rake'.
# Do not set this db to the same as development or production.
test:
  adapter: mysql
  database: webkompendium_rails_test
  username: root
  password:
  host: localhost
```

**Abbildung 23.17:** Rails-Datenbankkonfiguration in der Datei `database.yml`

### 23.8.3 Erzeugung von Model und Controller

Zu den „Bequemlichkeiten“ von Rails gehört die Unterstützung beim Schreiben der Grundgerüste der benötigten Ruby-Klassen durch Generatoren, insbesondere durch das Metascript `generate` im Unterordner `script` der Rails-Anwendung (Abbildung 23.18).

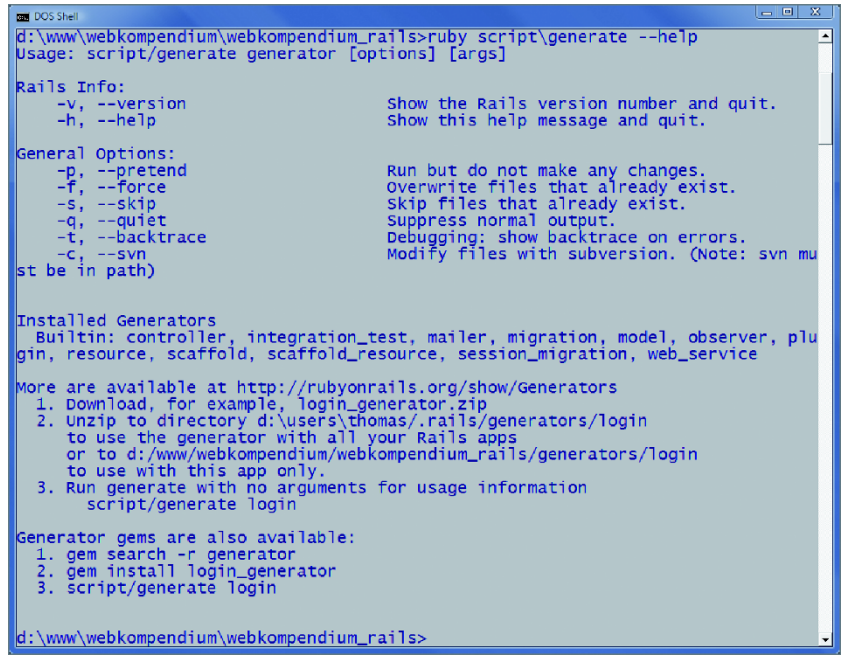



**Abbildung 23.18:**
Das Meta-Script `generate`

### Controller

Im ersten Schritt soll über das `generate`-Script eine Controller-Klasse mit dem Bezeichner `test` erzeugt werden; dies geschieht durch die Syntax

```
ruby script/generate controller test
```

Es wird damit eine Ruby-Klasse generiert, die von der `ApplicationController`-Klasse abgeleitet ist; ein Mapping der URL über in diesem Fall `/test` wird ebenfalls automatisiert eingerichtet (Abbildung 23.19).

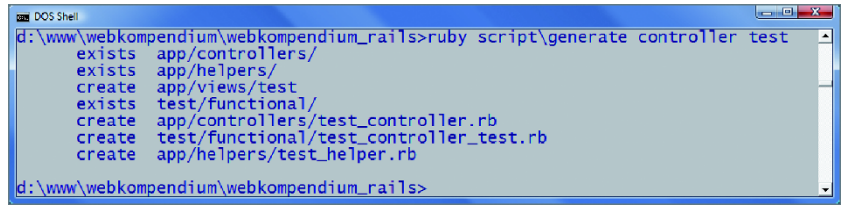



**Abbildung 23.19:**
Erzeugung des Controllers `test`

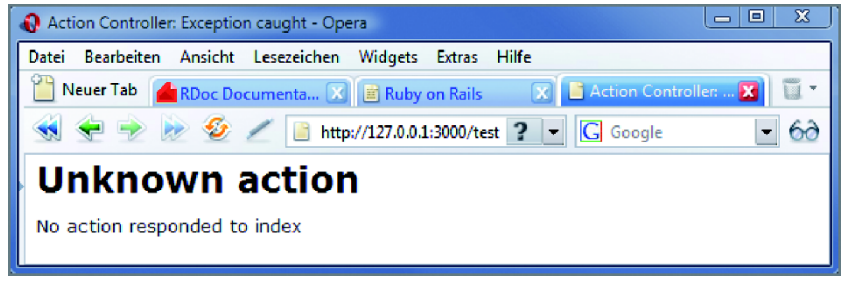



**Abbildung 23.20:**
Direkter Aufruf der URL `/test`

Bei Aufruf der damit verfügbaren URL `/test` wird allerdings nur eine Fehlermeldung erzeugt (Abbildung 23.20), was daran liegt, dass in der erzeugten

Controller-Klasse `TestController` noch keine Methoden implementiert sind (Abbildung 23.21).

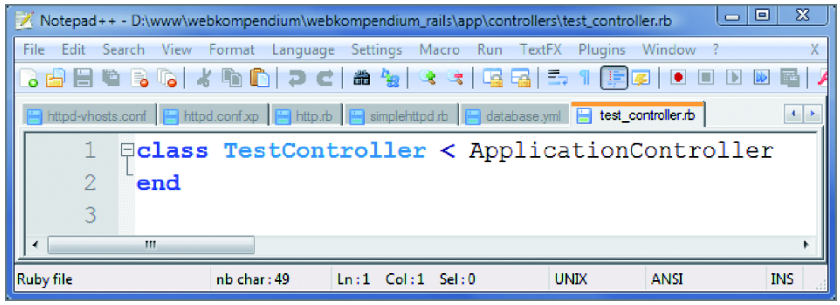



**Abbildung 23.21:** Die erzeugte Ruby-Klasse `TestController`

Um eine Ausgabe zu erzeugen, muss in dieser Ruby-Klasse eine Methode `index` implementiert werden; Abbildung 23.22 zeigt eine entsprechende Implementierung, Abbildung 23.23 zeigt das damit erzielte Ergebnis.

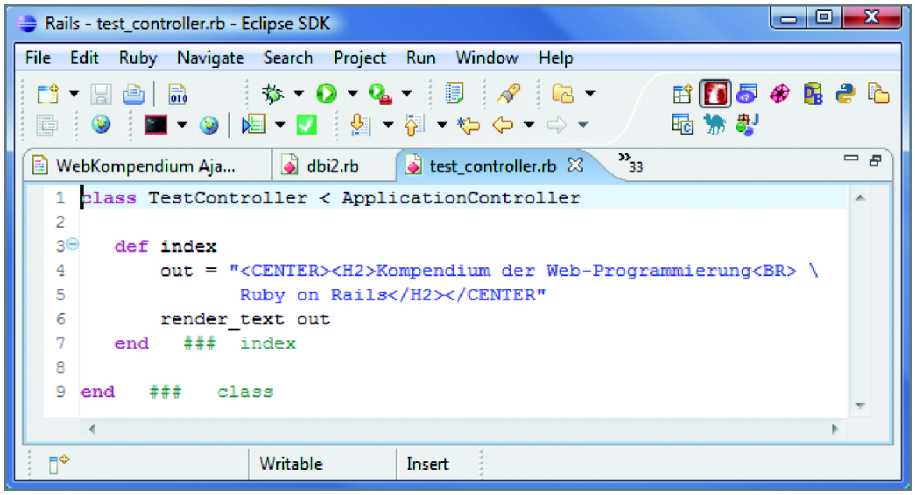



**Abbildung 23.22:** Einfügen der Methode `index`

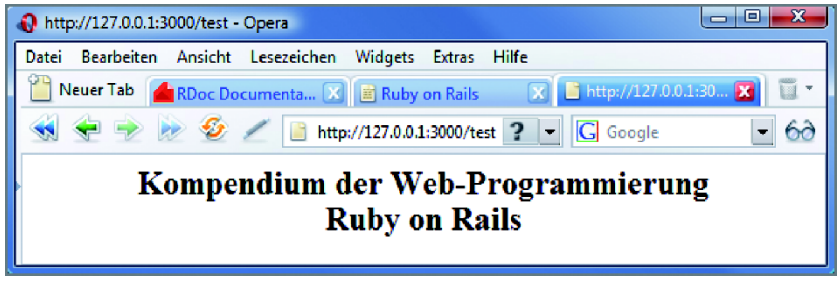



**Abbildung 23.23:** Aufruf der URL `/test` nach Implementierung von `index`

In diesen Controller-Klassen können nun weitere Methoden integriert werden, welche dann über die URL

```
<rails-server>/controllerbezeichner/methodenbezeichner
```

abgerufen werden können (Abbildungen 23.24 und 23.25).

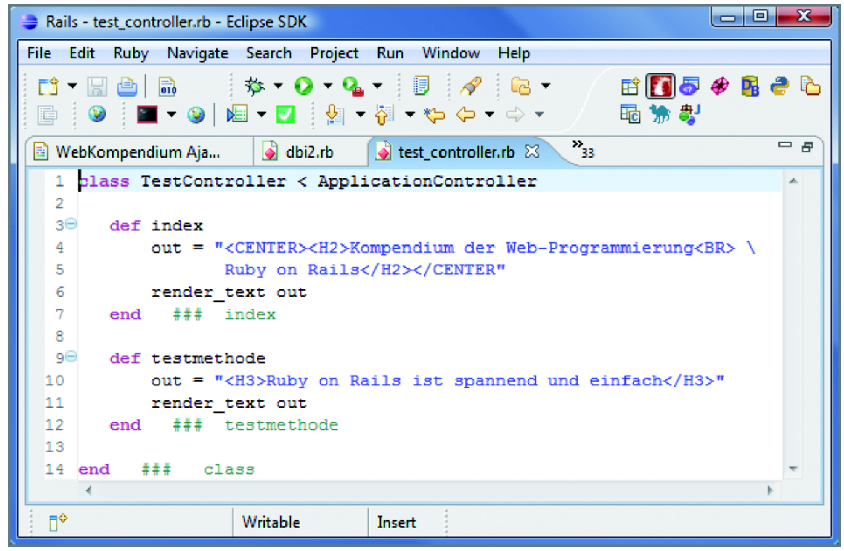



**Abbildung 23.24:** Einfügen einer weiteren Methode `testmethode`

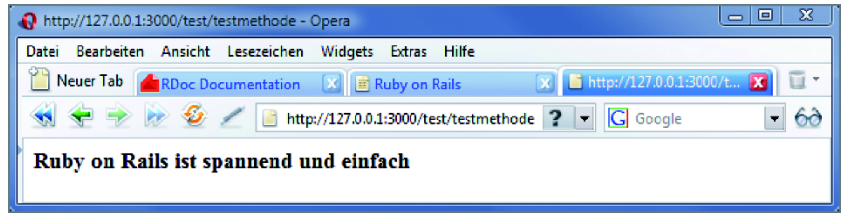



**Abbildung 23.25:** Aufruf der URL `/test/testmethode` nach Implementierung von `testmethode`

### Model

Für das Model ist zunächst eine Datenbanktabelle notwendig; in diesem Beispiel erzeugen wir die Tabelle `book` durch das Script `book.sql` (23.26), welche einfach fünf Attribute und einen Primary Key für den Datensatz eines Buches verwendet. Anschließend werden einige Datensätze in die Tabelle eingefügt.

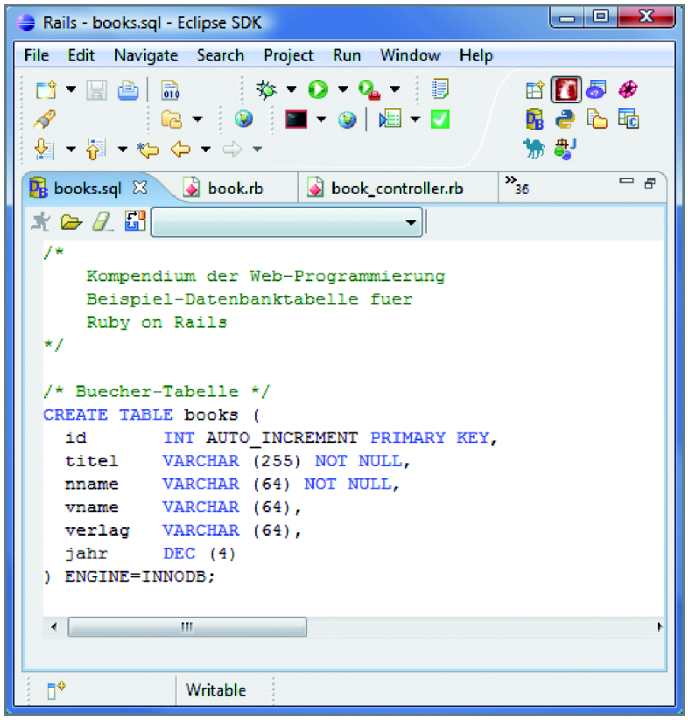



**Abbildung 23.26:**
Erzeugung der Datenbanktabelle `book.sql`

Ebenso wie die Controller-Klasse lässt sich auch die Model-Klasse über das `generate`-Script erzeugen. Hier kommen nun aber die Prinzipien von Rails aktiv zur Anwendung:

- Bezeichnungskonvention: Zur Datenbanktabelle „books“ (mit englischem Plural-S) gehört die Model-Klasse „Book“; allerdings ist Rails durchaus der englischen Sprache mächtig:
  „Rails is smart about English pluralization rules, so Company maps to companies, Person maps to people, and so forth.“;
- durch das schon vorgestellte Scaffolding werden die CRUD-Methoden automatisch verfügbar gemacht.

Dies setzt jeweils voraus, dass der Datenbankzugriff für das Rails-Projekt korrekt eingerichtet ist.

Im Beispiel erzeugen wir für die Datenbanktabelle `books` mittels

```
ruby script/generate model Book
```

(Abbildung 23.27) die zugehörige Model-Klasse `Book`.

Es entsteht damit das Skeleton für die Ruby-Klasse `Book`, welche von `ActiveRecord::Base` abgeleitet ist (Abbildung 23.28).

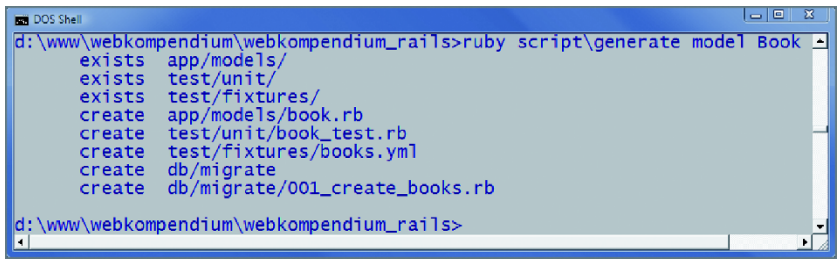



**Abbildung 23.27:**
Erzeugung der Model-Klasse `Book` zur Datenbanktabelle `books`

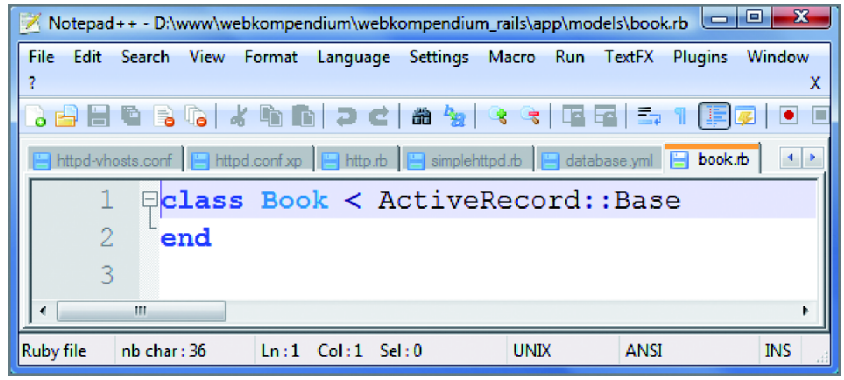



**Abbildung 23.28:** Skeleton der Model-Klasse `Book`

Nach dem Gerüst für die Model-Klasse, welches zunächst schon ausreichend ist, soll als nächstes ein Standard-Controller für das Model erzeugt werden; dies geschieht über `generate` durch (Abbildung 23.29)

```
ruby script/generate controller Book
```

Hierdurch wird die Ruby-Klasse `BookController` erzeugt – zunächst wieder ein reines Skeleton, welches lediglich von der Klasse `ApplicationController` abgeleitet ist.

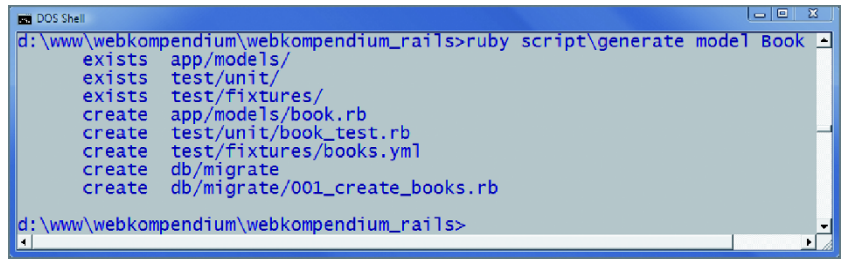



**Abbildung 23.29:** Erzeugung des Controllers für das Model `Book`

Das Meta-Script `generate` ist sehr leistungsfähig, so können mittels

```
ruby script/generate scaffold Book
```

sowohl Controller als auch Model für `Book` erzeugt werden.

### 23.8.4 Anwendung des Scaffolding mit CRUD

Nun soll das „Baukastenprinzip“ von Rails zur Anwendung kommen, das *Scaffolding*. Um dieses zu aktivieren, ist nun lediglich noch im Controller `BookController` die Anweisung

```
scaffold :book
```

zu ergänzen (Abbildung 23.30).

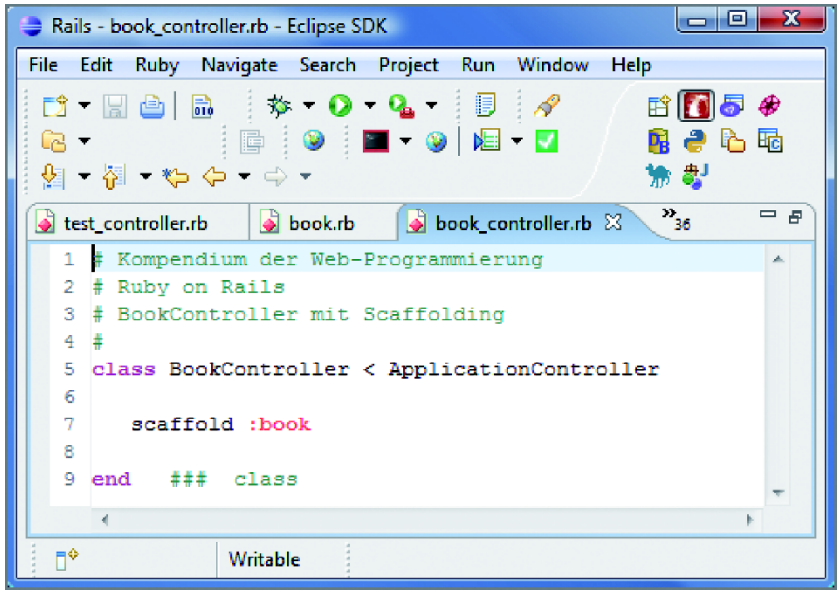



**Abbildung 23.30:**
`BookController` mit Scaffolding

Diese unscheinbare Anweisung hat in Rails sehr weitreichende Konsequenzen, da nun CRUD zur Verfügung steht: Auf die Datenbanktabelle kann vielfach zugegriffen werden.

**list**

Mittels der URL

```
<rails-server>/book/list
```

wird nun vom `BookController` die Methode `list` aufgerufen; diese wird durch das Scaffolding erzeugt und listet alle Einträge der Datenbanktabelle `book` auf (Abbildung 23.31).

**Abbildung 23.31:**
Scaffolding-Methode `list`

**show**

Neben der gesamten Tabelle kann mittels der URL

```
<rails-server>/book/show/id-nummer
```

ein einzelner Datensatz angezeigt werden (Abbildung 23.32), wobei standardmäßig nach einem Tabellenattribut `id` selektiert wird.

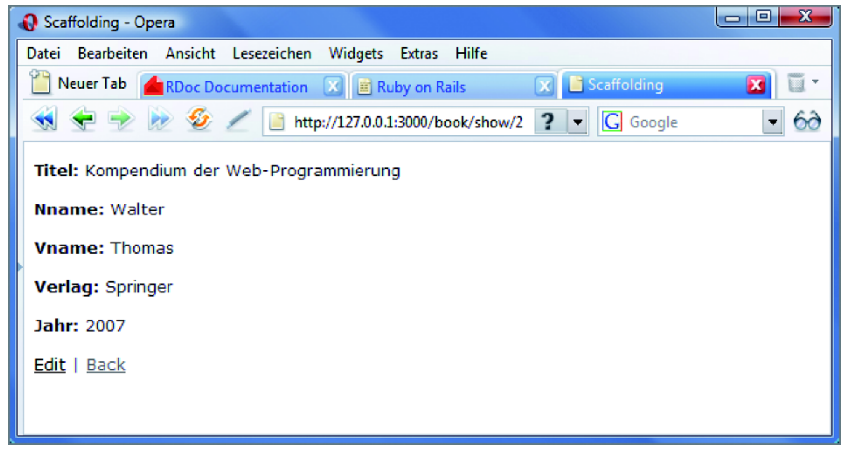



**Abbildung 23.32:** Scaffolding-Methode `show`

**edit**

Das Ändern eines Datensatzes (UPDATE) ist ebenfalls mittels Scaffolding über die URL

```
<rails-server>/book/edit/id-nummer
```

möglich; der Browser liefert ein vorausgefülltes HTML-Formular mit den ursprünglichen Werten des Datensatzes (Abbildung 23.33).

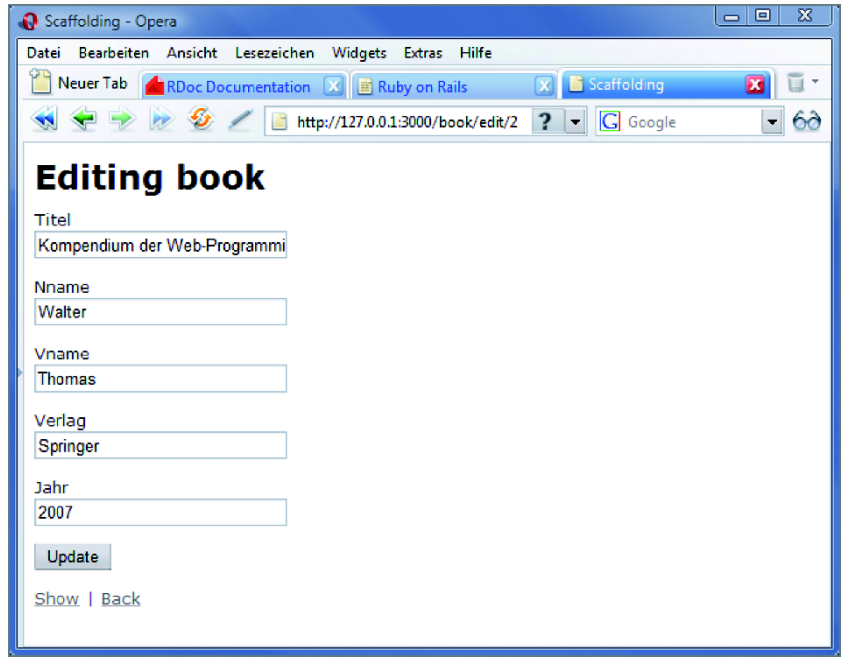



**Abbildung 23.33:** Scaffolding-Methode `edit`

**new**

Ähnlich zu `edit` liefert `new` über die URL

```
<rails-server>/book/new
```

ein HTML-Formular passend zu den Attributen der korrespondierenden Datenbanktabelle (Abbildung 23.34); nach Absenden über den „Create“-Button werden über die `list`-Methode alle Datensätze angezeigt (Abbildung 23.35).

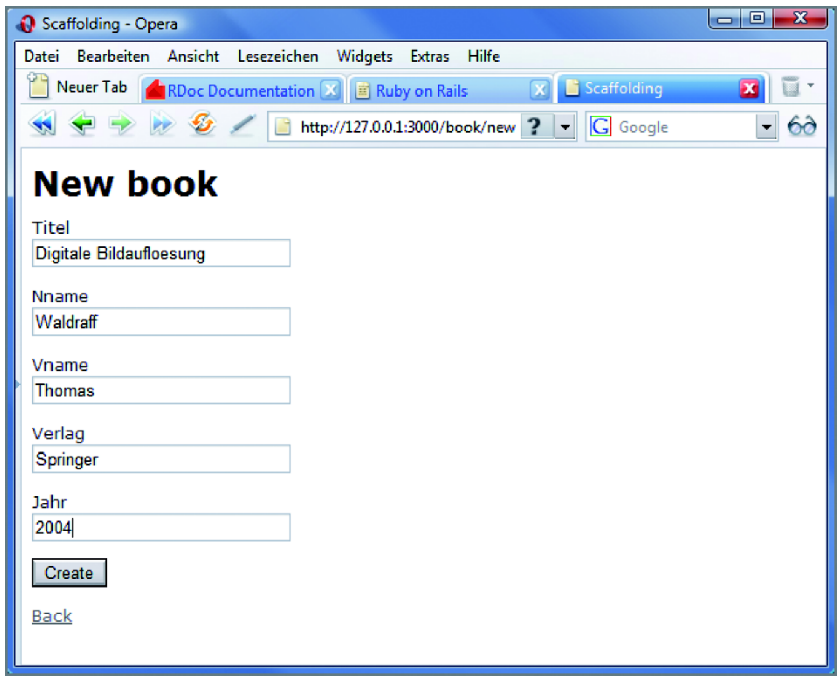



**Abbildung 23.34:**
Scaffolding-Methode `new`

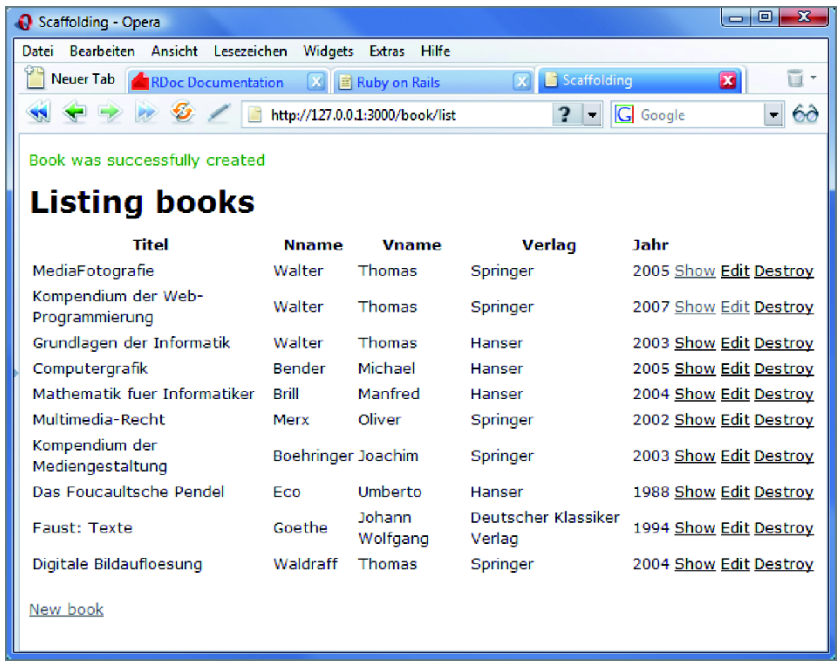



**Abbildung 23.35:**
nach Anlegen eines neuen Eintrags

**delete**

Umgekehrt kann über die URL

```
<rails-server>/book/delete/id-nummer
```

ein einzelner Datensatz gelöscht werden; hier erfolgt lediglich eine JavaScript-Abfrage zur Bestätigung des Löschens (Abbildung 23.36).

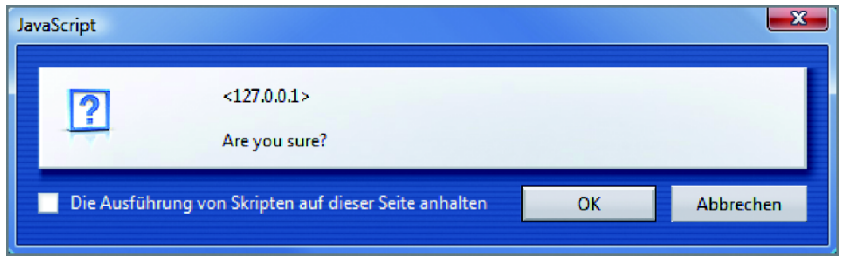



**Abbildung 23.36:** Scaffolding-Methode `delete`

**Überschreiben von Scaffolding-Methoden**

Über das Rails-Framework stehen somit direkt die leistungsstarken Scaffolding-Methoden zur Verfügung. Diese können im Controller natürlich bei Bedarf auch *überschrieben* werden, um eine abweichende Implementierung zu ermöglichen. So wird durch

```
def list
   ...
end
```

in der Controller-Klasse die `list`-Methode des Scaffolding überschrieben.

Durch dieses Überschreiben stehen damit natürlich alle denkbaren Funktionalitäten zur Verfügung.

### 23.8.5 Die View in Rails

Die *View des Rails-Frameworks* verwenden das Template-Konzept, also die Steuerung der Anzeige über eine HTML-artige Maske. Dies ist ein weitverbreitetes Vorgehen, wie es auch beim django-Framework (vgl. 22.3.9) und bei PHP durch die Template-Engine Smarty (vgl. Kapitel 21) angewendet wird.

#### 23.8.5.1 Veröffentlichung von Variablen

Im ersten Schritt muss ein Rails-Template den Zugriff auf die Werte von Ruby-Variablen der Rails-Anwendung erhalten. Die Übergabe von Variablen aus der Rails-Anwendung an ein Template wird dadurch ermöglicht, dass der zuständige Controller eine entsprechende Ruby-Instanzvariable an das Template übergibt. Wie in Absschnitt 13.4.3 erklärt, werden Bezeichner von Ruby-Instanzvariablen mittels beginnendem `@` gekennzeichnet. Meistens sind hierfür Instanzvariablen vom Typ des Models relevant; im behandelten Beispiel wird durch

```
@books = Book.find_all
```

eine Instanzvariable `@books` definiert, welche intern eine Ruby-Liste mit allen Instanzen des Models `Book` speichert, also letztlich alle Einträge in der Datenbanktabelle `book` verwaltet (Abbildung 23.37).

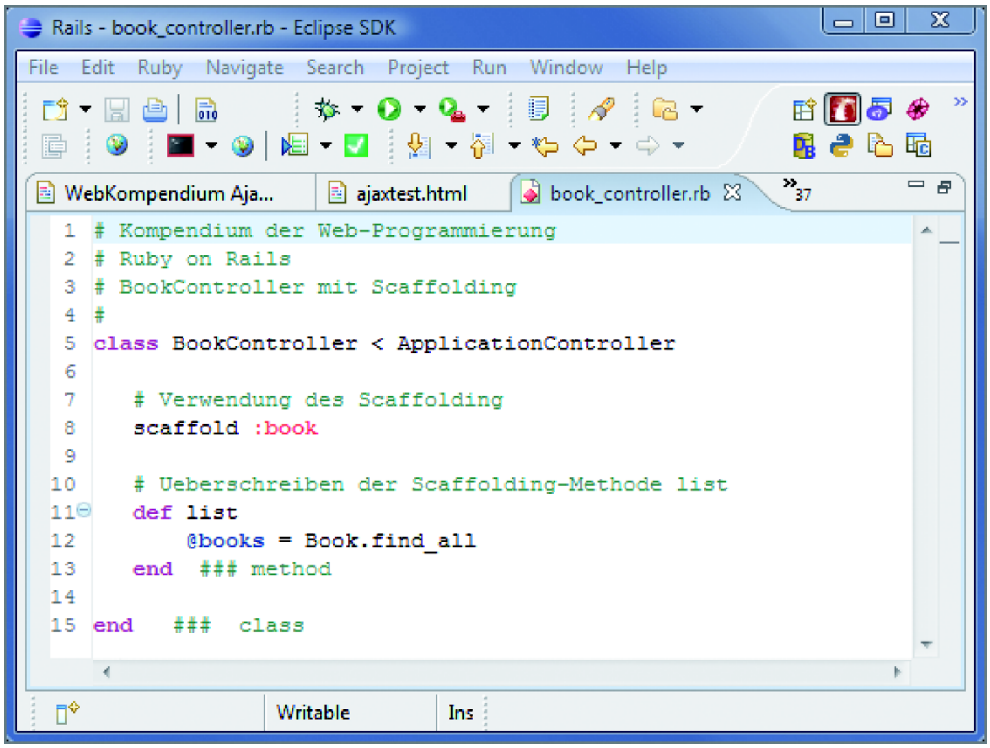



**Abbildung 23.37:** Überschriebene Methode `list` und Veröffentlichung des Models `Book`

Allerdings führt nun der Zugriff auf die Methode `list` zu einer Fehlermeldung (Abbildung 23.38): Nach dem Überschreiben der `list`-Methode wird ein Template für die neue Methode benötigt – und dieses ist nicht vorhanden.

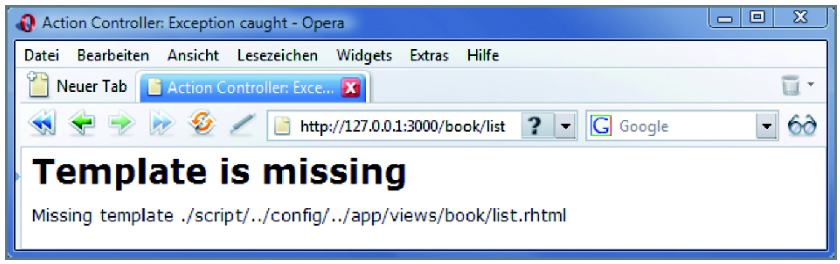



**Abbildung 23.38:** Fehlermeldung nach Überschreiben der `list`-Methode

### 23.8.5.2 Rails Template-Dateien

*Rails Template-Dateien* haben standardmäßig die Dateiendung `.rhtml` und sind im Verzeichnis

```
<rails-projekt>/app/views/<controllername>
```

mit dem Dateinamen `<methoden-name>.rhtml` abgelegt.

Vom Aufbau her sind Rails-Templates – wie allgemein alle Web-Templates – HTML-Dateien mit den üblichen HTML-Elementen der Version 4. Wesentlich ist nun aber die Integration von Ruby-Programmcode in diesen Templates:

- Über die Syntax

  ```
  <%=  rubyVariable  %>
  ```

  kann der Wert einer Ruby-Variablen angezeigt werden.

- Leistungsstärker ist das allgemeine Tag

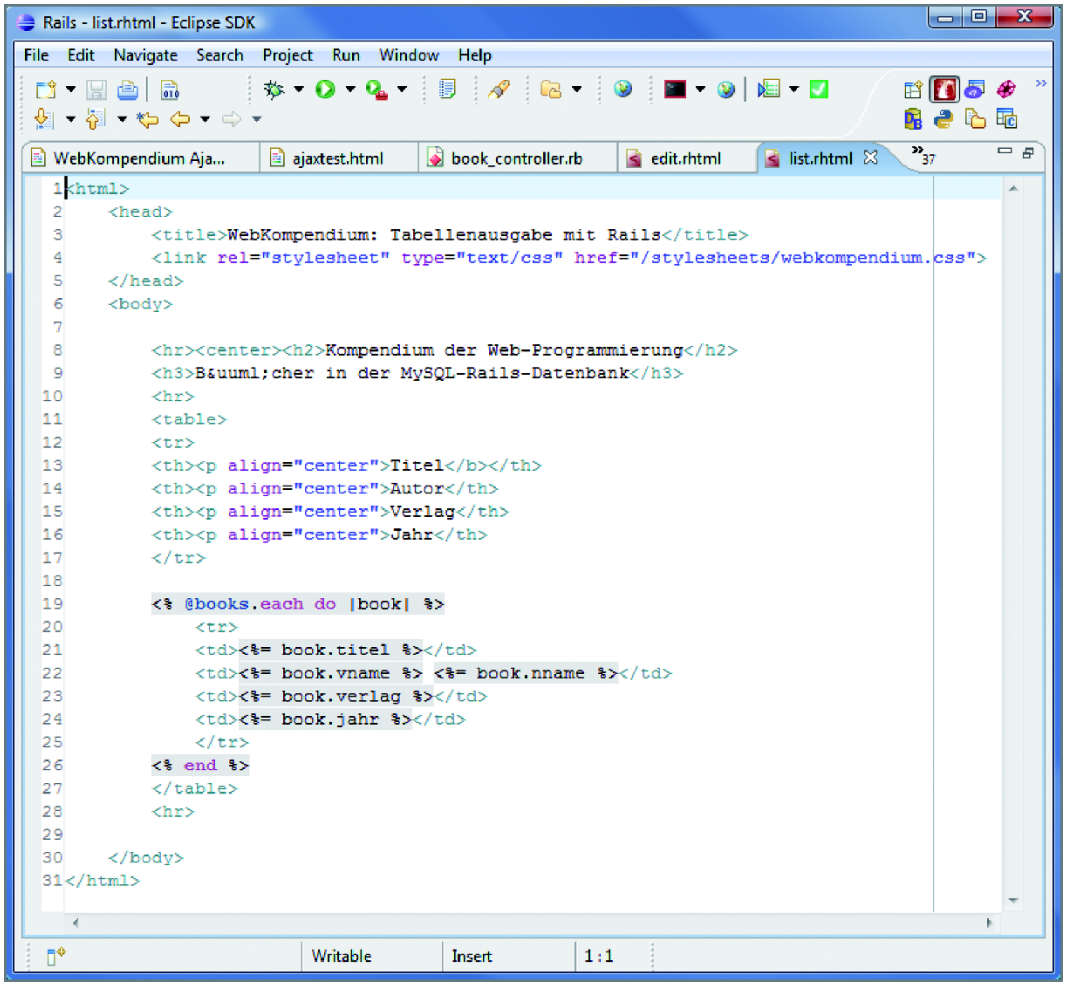
  ```
  <%  ruby_code  %>
  ```

  durch welche beliebiger Ruby-Code innerhalb des Templates integriert werden kann.

Ein erstes Template für die Methode `list` soll nun die Elemente der Instanz-Variablen `books` aus der geänderten Controller-Methode `list` nach Abbildung 23.37 ausgeben. Dies geschieht in einer `each`-Schleife in Ruby innerhalb des Templates `list.rhtml`:

```
<% @books.each do |book| %>
      <tr>
      <td><%= book.titel %></td>
      ...
      </tr>
<% end %>
```

Das vollständige Template ist in Abbildung 23.39 zu sehen, das damit gewonnene Ergebnis in Abbildung 23.40.




```
<html>
    <head>
        <title>WebKompendium: Tabellenausgabe mit Rails</title>
        <link rel="stylesheet" type="text/css" href="/stylesheets/webkompendium.css">
    </head>
    <body>

        <hr><center><h2>Kompendium der Web-Programmierung</h2>
        <h3>B&uuml;cher in der MySQL-Rails-Datenbank</h3>
        <hr>
        <table>
        <tr>
        <th><p align="center">Titel</b></th>
        <th><p align="center">Autor</th>
        <th><p align="center">Verlag</th>
        <th><p align="center">Jahr</th>
        </tr>

        <% @books.each do |book| %>
            <tr>
            <td><%= book.titel %></td>
            <td><%= book.vname %> <%= book.nname %></td>
            <td><%= book.verlag %></td>
            <td><%= book.jahr %></td>
            </tr>
        <% end %>
        </table>
        <hr>

    </body>
</html>
```

**Abbildung 23.39:**
Fertiges Template für die `list`-Methode: Datei `list.rhtml`

| Titel | Autor | Verlag | Jahr |
|---|---|---|---|
| MediaFotografie | Thomas Walter | Springer | 2005 |
| Kompendium der Web-Programmierung | Thomas Walter | Springer | 2007 |
| Grundlagen der Informatik | Thomas Walter | Hanser | 2003 |
| Computergrafik | Michael Bender | Hanser | 2005 |
| Mathematik fuer Informatiker | Manfred Brill | Hanser | 2004 |
| Multimedia-Recht | Oliver Merx | Springer | 2002 |
| Kompendium der Mediengestaltung | Joachim Boehringer | Springer | 2003 |
| Das Foucaultsche Pendel | Umberto Eco | Hanser | 1988 |
| Faust: Texte | Johann Wolfgang Goethe | Deutscher Klassiker Verlag | 1994 |

**Abbildung 23.40:**
Ausgabe von `list` mit Template

#### 23.8.5.3 Template-Datei für Update

Ein weiteres Template soll nun das Ändern eines Datensatzes ermöglichen. Die hierfür verwendete Methode `edit` (Abbildung 23.33) wird dazu nicht überschrieben, sie soll aber mittels eines Templates `edit.rhtml` (Abbildung 23.41) besser gestaltet werden.

Beim Standard-Mapping (vgl. 23.8.7) wird dann über die URL

```
<rails-server>/edit/id
```

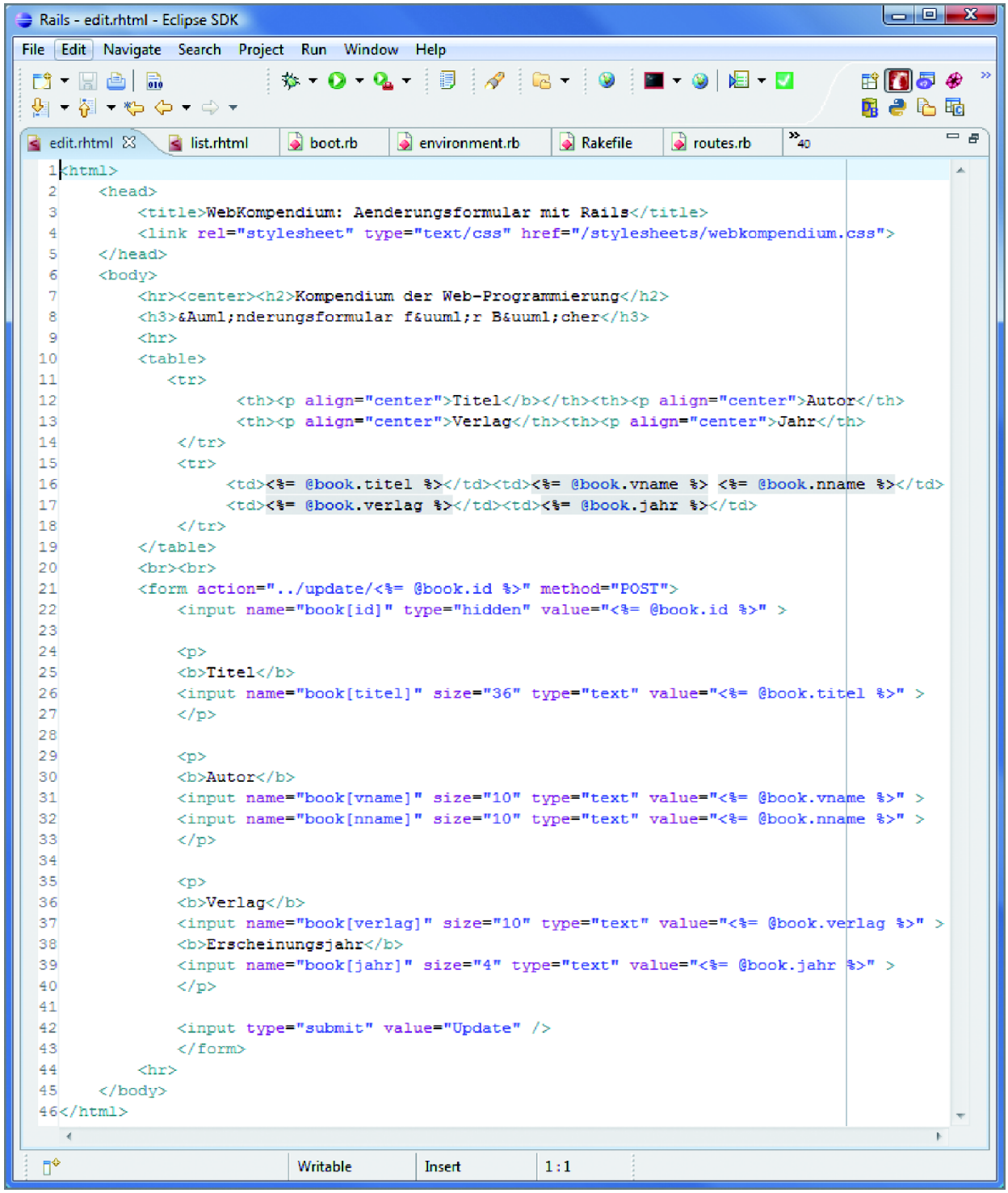
der Datensatz mit der Nummer `id` aufgerufen. Innerhalb des Templates haben wir dann folgende Situation:

- Die Instanzvariable `book` ist mit den Werten des zu editierenden Datensatzes vorbelegt; `@book.verlag` ist beispielsweise der Verlag.
- Das Editierformular baut Eingabefelder auf; diese haben einen Bezeichner der Art `datenbanktabelle[attribut]`, also etwa für den Titel

  ```
  <input name="book[titel]" value="<%= @book.titel %>" >
  ```

- Das ausgefüllte Formular wird an die Scaffolding-Methode `update/id` gesendet, welche die Datenbankänderung durchführt.

Abbildung 23.41 zeigt den Sourcecode des Templates `edit.rhtml`, Abbildung 23.42 die `edit`-Methode des Scaffoldings, nun aber mit einem eigenen Template; dies korrespondiert zur Abbildung 23.33 ohne Template.




```
<html>
    <head>
        <title>WebKompendium: Aenderungsformular mit Rails</title>
        <link rel="stylesheet" type="text/css" href="/stylesheets/webkompendium.css">
    </head>
    <body>
        <hr><center><h2>Kompendium der Web-Programmierung</h2>
        <h3>&Auml;nderungsformular f&uuml;r B&uuml;cher</h3>
        <hr>
        <table>
           <tr>
                 <th><p align="center">Titel</b></th><th><p align="center">Autor</th>
                 <th><p align="center">Verlag</th><th><p align="center">Jahr</th>
            </tr>
            <tr>
                <td><%= @book.titel %></td><td><%= @book.vname %> <%= @book.nname %></td>
                <td><%= @book.verlag %></td><td><%= @book.jahr %></td>
            </tr>
        </table>
        <br><br>
        <form action="../update/<%= @book.id %>" method="POST">
            <input name="book[id]" type="hidden" value="<%= @book.id %>" >

            <p>
            <b>Titel</b>
            <input name="book[titel]" size="36" type="text" value="<%= @book.titel %>" >
            </p>

            <p>
            <b>Autor</b>
            <input name="book[vname]" size="10" type="text" value="<%= @book.vname %>" >
            <input name="book[nname]" size="10" type="text" value="<%= @book.nname %>" >
            </p>

            <p>
            <b>Verlag</b>
            <input name="book[verlag]" size="10" type="text" value="<%= @book.verlag %>" >
            <b>Erscheinungsjahr</b>
            <input name="book[jahr]" size="4" type="text" value="<%= @book.jahr %>" >
            </p>

            <input type="submit" value="Update" />
            </form>
        <hr>
    </body>
</html>
```

**Abbildung 23.41:**
Template für die `edit`-Methode: Datei `edit.rhtml`

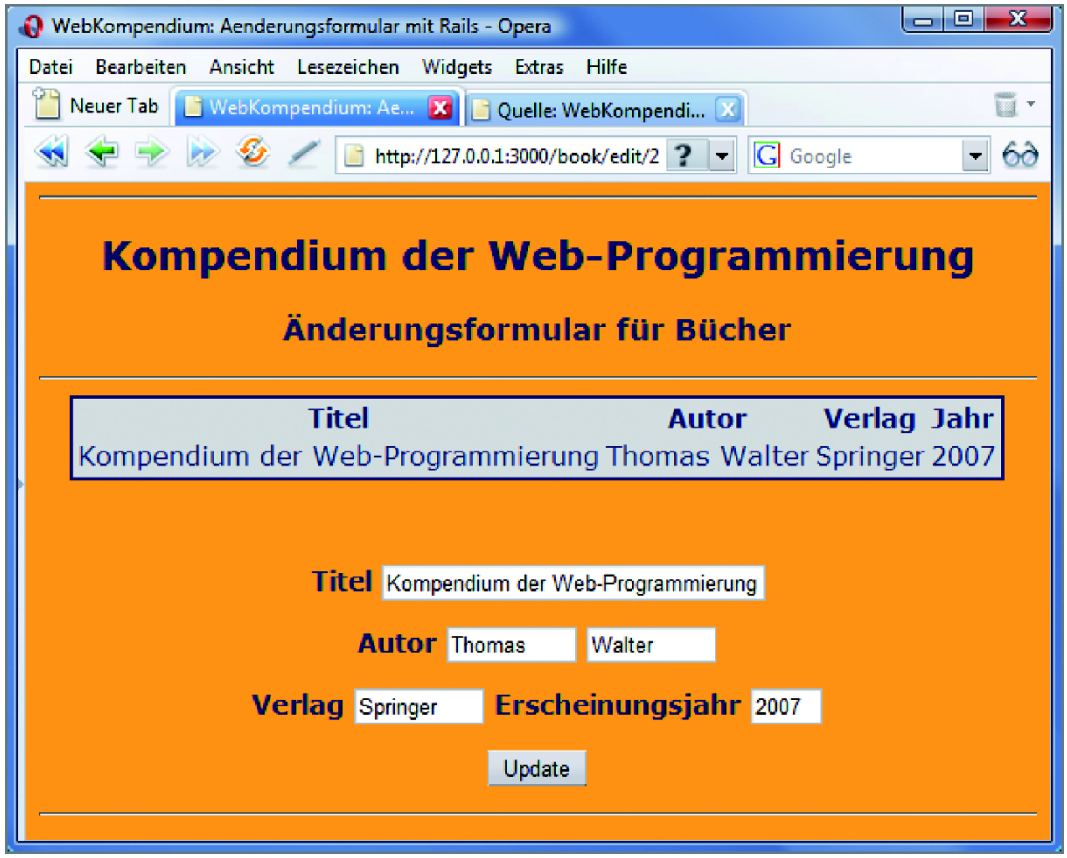



**Abbildung 23.42:** Ausgabe von `edit` mit Template

#### 23.8.5.4 Ausgabe von statischen Informationen

Ein Rails-Projekt kann natürlich auch statische Ausgaben enthalten; hierfür gibt es im Projekt-Ordner das Verzeichnis `public`. Dieser Wurzelordner enthält die statischen Ausgabedateien, etwa die Datei `index.html`, welche die Default-Ausgabe eines leeren Rails-Projektes nach Abbildung 23.15 steuert.

Dieser Ordner bietet sich auch für die Ablage der Stylesheets an. Das Template aus Abbildung 23.39 nutzt dieses zum Einbinden des Stylesheets mit der URL

```
<link rel="stylesheet" type="text/css"
      href="/stylesheets/webkompendium.css">
```

### 23.8.6 Beziehungen über mehrere Datenbanktabellen

Bisher arbeitet das Beispiel nur mit einer Tabelle (`books`) in der MySQL-Datenbank. Rails bietet eine einfache Methode in den Model-Klassen, Beziehungen zwischen verschiedenen Datenbanktabellen abzubilden. Dies geschieht beispielsweise mit der Syntax `has_many` für eine 1:n-Beziehung und `belongs_to` für den umgekehrten Fall. Eine 1:1-Beziehung wird mittels `has_one` in der Model-Klasse beschrieben.

### 23.8.7 URL-Mapping in Rails

Bis hierher wurde strikt das Standard-Mapping von Rails verwendet – was natürlich die Namensgebung eingeschränkt hat, dafür konnte aber völlig auf jegliche Konfiguration von Zuordnungen verzichtet werden.

Natürlich ist Rails wesentlich flexibler. Im Unterordner `config` des Rails-Projektes findet sich neben der schon vorgestellten Datei `database.yml` die Konfiguration des Mappings: `routes.rb` (Abbildung 23.43).

`routes.rb` definiert das *Mapping einer URL* standardmäßig durch die Weiterleitung an

```
Controller-Name/Methoden-Name/ID
```

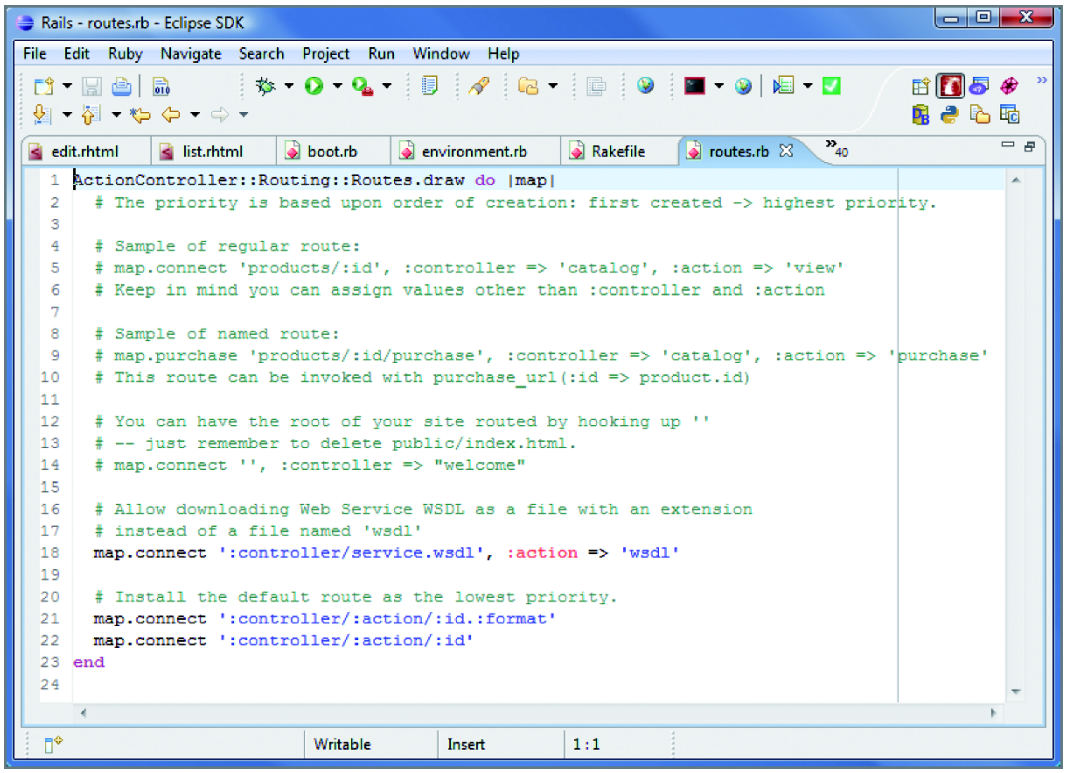



**Abbildung 23.43:**
Die Datei `routes.rb` zur Definition des URL-Mappings mit Rails

## 23.9 Zusammenfassung

Wie haben in diesem Kapitel das Grundprinzip von Ruby on Rails kennengelernt. Dabei wurde insbesondere der Vorteil des Rails-Frameworks sichtbar: Die schnelle Entwicklung funktionsfähiger Prototypen. Dieser Vorteil resultiert aus zwei Gründen: Zum einen bietet das Rails-eigene Scaffolding bereits zahlreiche komplexe Funktionen wie Update und Change, zum anderen umgeht Rails durch eine vorgegebene Namenskonvention vielfältige Konfigurationen.

Die Prinzipien von Rails werden sich weiter entwickeln: Zum einen entstehen ähnliche Frameworks auch für andere Sprachen, so dass die enge Bindung an Ruby aufgelöst wird. Zum anderen kann das Framework auch für weitere Gebiete als nur das Fast Prototyping eingesetzt werden. Die Zukunft wird zeigen, welche Wege Rails erfolgreich beschreitet.

# 24

# Serverseitiges Java

Die plattformunabhängige Programmiersprache Java nimmt seit Mitte der neunziger Jahre eine zentrale Rolle in vielen Bereichen der Softwareentwicklung ein. Insbesondere im Bereich der Web-Programmierung ist Java inzwischen ein vorherrschender Standard.

In diesem Kapitel soll ein Einstieg in die serverseite Web-Programmierung mit Java gegeben werden.

## 24.1 J2EE

Die Programmiersprache Java (vgl. Abschnitt 1.5.2) war direkt seit ihrer Vorstellung intensiv mit dem Internet verbunden. Dies lag zum einen an dem erfolgreichen Konzept der Plattformunabhängigkeit über die Java Virtual Machine, zum anderen aber ebenso an dem Applet, das in Kapitel 18 vorgestellt wurde: Der Möglichkeit, praktisch vollständige Clientprogramme in die Webseite zu integrieren und weltweit auf jedem Browser ausführen zu können. Trotz dieser neuen Vorteile hat das Konzept des Applets auch viele Probleme, wie in 18.4 ausgeführt, weshalb es sich nicht durchgesetzt hat.

Seine Stärke für die Web-Programmierung gründet Java inzwischen auf den serverseitigen Einsatz: Java kann als Nachfolger der CGI-Techniken sehr performant eingesetzt werden.

Java bietet dabei Vorteile:

- Web-Applikationen in Java profitieren von den Eigenschaften von Java als moderne, objektorientierte Hochsprache. So stehen ausgefeilte und saubere Konzepte der Objektorientierung genauso wie ein Compiler und eine strenge Typisierung bereit: Java ist keine Scriptsprache.
- Für Java sind zahlreiche Bibliotheken und Frameworks verfügbar; der Bereich der Ergänzungen direkt durch das eigentliche Java-Framework bis hin zu zahllosen Open Source-Projekten ist grenzenlos.

Diese Vorteile, insbesondere der erste, kann aber auch als Nachteil gesehen werden: Java ist – insbesondere beim Einsatz auf dem Webserver – vergleichsweise sehr komplex, weshalb sich der Aufwand für Java erst bei größeren Webprojekten auszahlt. Abbildung 24.1 verdeutlicht den Vergleich der Java-Techniken zum einfachen CGI.

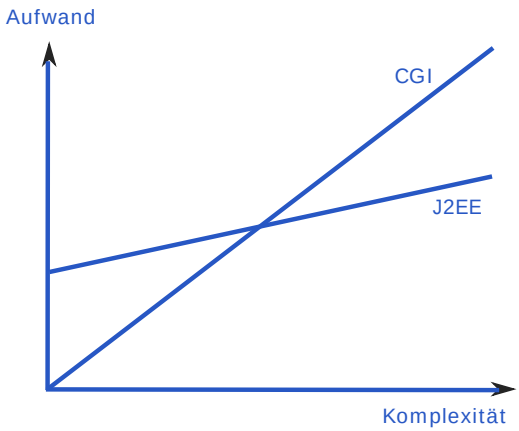



**Abbildung 24.1:** Vergleich des Entwicklungsaufwands CGI gegenüber J2EE

Java wird seit langem in drei verschiedenen Paketen angeboten: das „normale“ Java ist die Standard Edition, J2SE; für mobile Endgeräte gibt es die Micro Editon J2ME und für Serveranwendungen die Enterprise Edition *J2EE;* diese ist für dieses Kapitel notwendig, wobei sie auf der J2SE aufbaut, welche also auch vorhanden sein muss.[1]

Insgesamt versteht man unter J2EE eine Familie von Java-APIs, welche alle für den Einsatz im Geschäftsumfeld vorgesehen sind. Hierzu zählen

- Servlets
- JavaServer Pages (JSP)
- Enterprise Java Beans (EJB)
- Java Database Connectivity (JDBC)
- Java Naming and Directory Interface (JNDI)
- Webservices;
- Java Server Faces (JSF).

und darüber hinaus viele mehr.
Für den Betrieb einer J2EE-Anwendung genügt weder ein Webserver, noch kann dieser durch ein einfaches Modul erweitert werden; es wird ein Java-EE-Server benötigt. Zu diesen zählen etwa JBoss und Geronimo; der Tomcat-Server deckt nur einen Teil der J2EE-Funktionalitäten ab (beispielsweise keine EJBs).

Typisch für eine J2EE-Entwicklung sind eine mehrschichtige Software (vgl. 4.2.4) und transaktionssichere Datenbankoperationen.

Im Folgenden werden zwei Kerntechniken von J2EE vorgestellt: Java Servlets und JavaServer Pages.

## 24.2 Java Servlets

Das *Servlet* ist ein Kunstwort aus „Server“ und „Applet“. Die Idee hinter dem Servlet ist zunächst die Optimierung des Prozessmanagements des Servers:

- Bei CGI (Kapitel 9) wird für jeden Request ein neuer Prozess gestartet; dieser arbeitet die Anfrage ab und endet dann.

[1] beispielsweise ist der Java-Compiler nur Bestandteil von J2SE.

- Bei fastCGI (Kapitel 19) wird für jede mögliche Anfrage (also für die verfügbaren fastCGI-Scripte) ein Prozess einmalig gestartet – entweder beim Starten des Webservers durch diesen oder beim ersten Request. Eine Parallelisierung für den einzelnen Prozess ist nicht vorgesehen.
- Das Servlet verwendet eine wesentlich modernere Architektur: Der „Servlet-Server“ betreibt die JVM, er erzeugt eine Instanz der Servlet-Klasse; für jeden Request an edieses Servlet wird nur ein Thread auf der JVM gestartet, kein eigener Prozess, was die hohe Performance und gut Skalierbarkeit dieser Technik begründet.

Gemäß diesen Modells sind Servlets als Threads auf der JVM unabhängig vom Betriebssystem und skalieren ausgesprochen gut.

Eine umfassende Darstellung von Servlets und JavaServer Pages ist in [Hal00] zu finden.

### 24.2.1 Voraussetzungen von Servlets

Für den Betrieb von Servlets wird zunächst ein entsprechender Server benötigt; dies kann nicht der Webserver leisten, ein dedizierter Server ist notwendig. Für Servlets ist hierfür der Tomcat-Server aus dem Apache-Jakarta-Projekt ausreichend (vgl. 24.2.2), es kann aber auch ein vollwertiger Java-EE-Server wie JBoss verwendet werden.

Dieser Server, die Servlet-Engine, kann alleine betrieben oder an einen Apache Webserver gebunden werden: Im zweiten Fall sendet der Client vorzugsweise auf Port 80 den HTTP-Request an den Apache, welcher diesen an die Servlet-Engine über eine tcp-Verbindung weiterleitet; dies ist der stabilere, sicherere und flexiblere Weg als der Betrieb der Servlet-Engine alleine. Für die Kopplung des Apache mit Tomcat steht hierfür das Modul mod_jk bereit (jk steht für Jakarta, das Apache-Java-Projekt).

Auf dieser Servlet-Engine ist dann ein Projekt, eine „Web-Applikation“ einzurichten (vgl. Abschnitt 24.2.2).

Eine weitere Voraussetzung für Servlets sind die Java-Pakete

- `javax.servlet`
- `javax.servlet.http`

welche die Servlet-Klassen des J2EE enthalten; sie sind Bestandteil von J2EE, es genügt aber das Java-Archiv `servlet-api.jar` für die Entwicklung und den Betrieb von Servlets aus.

### 24.2.2 Der Servlet-Container Tomcat

**Abbildung 24.2:** Tomcat-Logo

*Tomcat* ist die klassische Servlet-Engine, inzwischen auch die offizielle Referenz-Implementierung des Servlet-Containers. Tomcat selbst ist in Java geschrieben, was bedeutet, dass die Installation plattformunabhängig im Prinzip nur aus dem Entpacken besteht. Voraussetzung ist, dass der Server eine aktuelle Java-Version vorhält: Für Servlets genügt eine JVM, für JavaServer Pages muss nach 24.4 eine Java-Compiler vorhanden sein, also das komplette JDK.

Nach dieser Installation bietet Tomcat folgende Verzeichnisstruktur:

```
bin      Programme
conf     Konfigurationsdateien
lib      Bibliotheken für alle Web-Applikationen
logs     Logfiles
webapps  Default-Stammordner der Web-Applikationen
```

In `conf` liegt die zentrale Konfigurationsdatei `server.xml` des Tomcat, welche u.a. den tcp-Port festlegt; `lib` – in älteren Versionen `common/lib` – enthält die Java-Bibliotheken im jar-Format, welche für alle Web-Applikationen bereitgestellt werden (etwa die erwähnte `servlet-api.jar`). In `bin` finden sich insbesondere die Scripte zum Starten und Beenden des Tomcat:

```
startup
shutdown
```

Tomcat wird direkt mit zugehörigen Web-Applikationen ausgeliefert, insbesondere „examples“ für Beispiele und „manager“ (Abbildung 24.3 ) zum Administrieren des Servers; beide sind nützlich, so kann mittels der Beispiele die korrekte Arbeitsweise geprüft werden; für einen Produktionsbetrieb sollten diese aber entfernt werden.

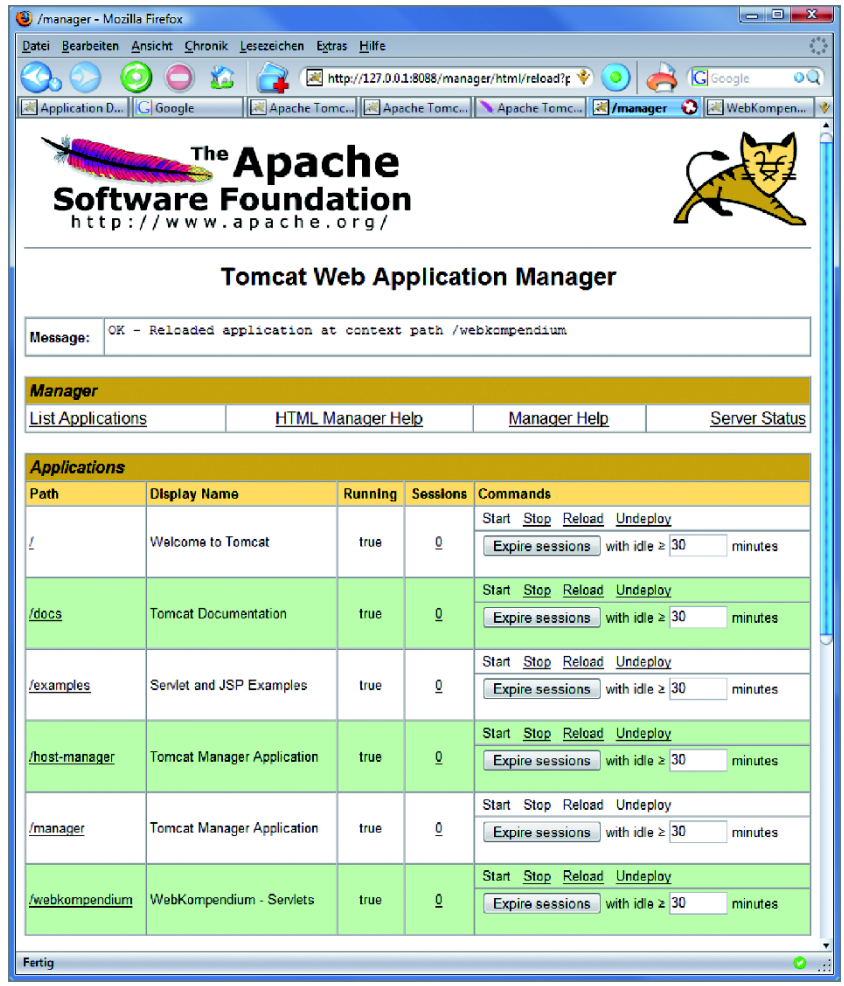



**Abbildung 24.3:**
Das Tomcat Manager-Tool

Über das Sysdeo-Plugin für Eclipse [web] kann Tomcat auch direkt aus der Entwicklungsumgebung gestartet werden.

### 24.2.3 Deployment einer Web-Applikation

Nach der Tomcat-Installation ist das *Deployment der Web-Applikation* der zentrale Schritt zum ersten Servlet. Dies geschieht klassisch in zwei Stufen:

- Erzeugung eines Unterordners von `webapps` für die neue Web-Applikation; dieser muss wiederum einen Unterordner `WEB-INF` haben; hier liegt die Konfigurationsdatei der Web-Applikation `web.xml` (Abbildung 24.4) sowie der Unterordner `classes` für die class-Dateien des Servlets.
- Über das Manager-Tool (Abbildung 24.3) ist die neue Web-Applikation zu starten.

Ein einfacherer Weg besteht darin, in einer Entwicklungsumgebung, etwa Eclipse mit den WTP, ein Web-Archiv zu erstellen, eine war-Datei, und diese in den `webapps`-Ordner zu legen bzw. über den Manager zu installieren.

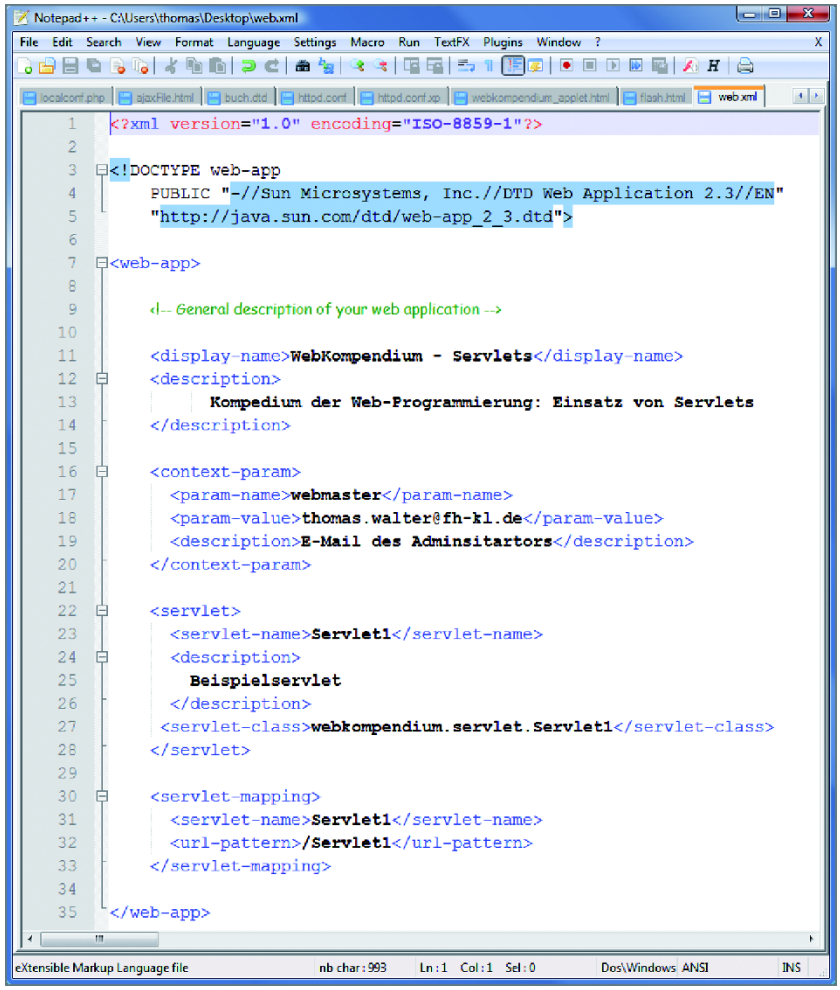



**Abbildung 24.4:**
Beispiel einer `web.xml`

Die Konfigurationsdatei `web.xml` macht insbesondere die einzelnen Servlets für den Tomcat bekannt und legt ein Mapping fest, welcher Request zu welchem Servlet gesendet wird.

Üblicherweise sollte hier nach dem MVC-Designpattern (vgl. 4.3) ein Controller-Servlet diese Aufgabe übernehmen, wobei in gewisser Weise der Tomcat selbst durch dieses Mapping übernimmt, ähnlich wie es auch im django-Framework nach Kapitel 22 geschieht.

### 24.2.4 Java Servlet-Klassen

Ein Servlet ist Instanz einer Java-Klasse, welche von `javax.servlet.http.HttpServlet` abgeleitet ist; genauer ist dies ein Servlet für das HTTP-Protokoll, es gibt auch die generische Servlet-Klasse `javax.servlet.GenericServlet`.

Eine Anfrage an die Servlet-Engine führt dann zu einem Aufruf der Methode `service(...)`, welche bereits aus dem `GenericServlet` verügbar, oft aber in den Kindklassen überschrieben ist. `service` hat zwei wichtige Argumente, das request-Objekt und das response-Objekt für die Verwaltung der Anfrage und der Antwort.

Speziell für das `HttpServlet` ist die Situation etwas anders: Anstelle von `service` werden zwei Methoden für den GET- und für den POST-Request erklärt:

- `doGet` für den GET-Request;
- `doPost` für den POST-Request.

Diese Methoden haben als Argumente jeweils eine Instanz der Klassen

- `HttpServletRequest` für die Anfrage;
- `HttpServletResponse` für die Antwort.

Zwei weitere Methoden in der Klasse `HttpServlet` sind `doTrace` und `doHead`.

Eine einfache, exemplarische Klasse für ein Servlet ist in Abbildung 24.5 zu sehen, die Klasse `webkompendium.servlet.Servlet1`. Diese implementiert nur die `doGet`-Methode.

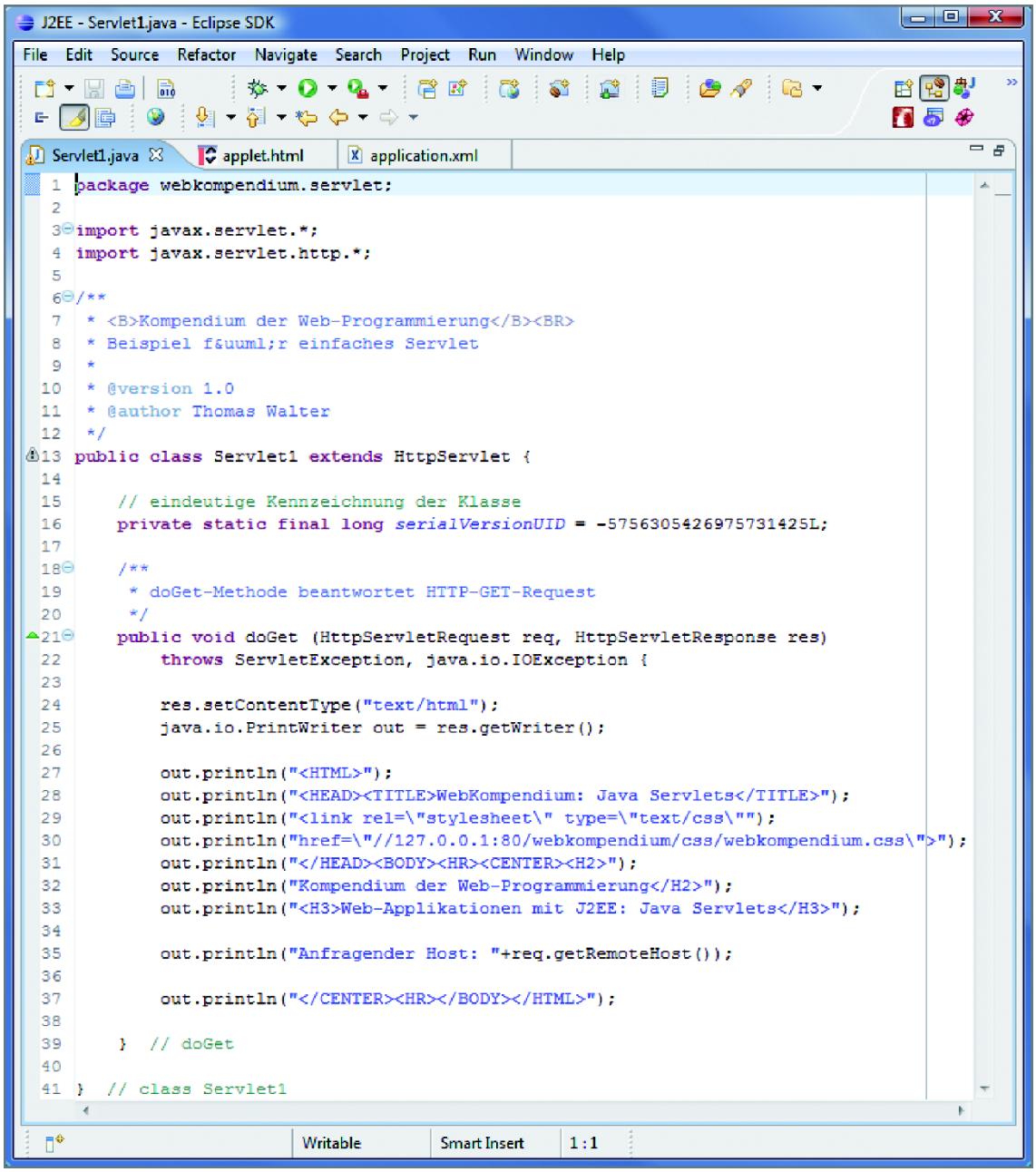



**Abbildung 24.5:**
Die Klasse `Servlet1`

Nach dem Compilieren ist die entstandene class-Datei in den Unterordner `WEB-INF/classes` der Web-Applikation zu kopieren (vgl. 24.2.2), wobei hier wiederum die Paketstruktur zu beachten ist; im konkreten Beispiel liegt die class-Datei also im Ordner

```
<tomcat_home>/webapps/<app_name>/WEB-INF/classes/webkompendium/servlet
```

Dann kann über die URL

```
http://<rechner:port>/webkompendium/Servlet1
```

das Servlet ausgeführt werden, wobei die Datei `web.xml` aus Abbildung 24.4 mit dem dort angegebenen Servlet-Mapping vorausgesetzt wird. Das Ergebnis ist in Abbildung 24.6 zu sehen.

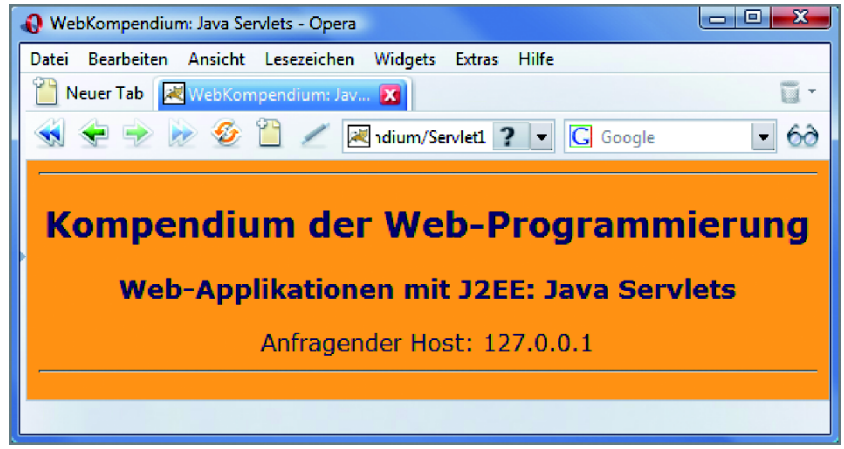



**Abbildung 24.6:** Ausführen von `Servlet1`

Das Attribut `serialVersionUID` kennzeichnet die Version der Java-Klasse und wird benötigt, da Servlets das Interface `Serializable` implementieren; sein Wert kann mittels des Java-Dienstprogramms `serialver` generiert werden. Ohne diese Konstante ist das Servlet prinzipiell lauffähig, es kann aber je nach Compilereinstellungen zu Warnmeldungen kommen.

#### 24.2.4.1 Tomcat als Webserver

Der derartig konfigurierte Tomcat kann auch direkt als Webserver eingesetzt werden; hierzu sind nur geeignete Verzeichnisse in den Applikationsordner zu legen.

Dies ist für einige Fälle ausreichend, im Allgemeinen sollte aber ein separater Apache wie schon beschrieben vor dem Tomcat betrieben werden.

Im Beispiel aus Abbildung 24.5 wird die CSS-Datei auch von Apache auf Port 80 ausgeliefert, während der Tomcat auf einem anderen Port (hier 8088) läuft; Default-Port für die Kommunikation nach außen ist 8080, auf 8005 kommuniziert der Tomcat mit dem Webserver und dem Betriebssystem.

#### 24.2.4.2 Servlets: Aufwand

An dieser Stelle sei nochmals auf den Entwicklungsaufwand und Abbildung 24.1 verwiesen: Es ist ein nicht unerheblicher Aufwand notwendig, um ein minimales Servlet zu erzeugen. Dieser Ansatz, wie allgemein die J2EE-Techniken, ist erst für größere Webprojekte sinnvoll.

### 24.2.5 Formularverarbeitung mit Servlets

Innerhalb eines Servlets – in der `doGet`- oder `doPost`-Methode – kann durch die Methode

```
public java.lang.String
           getParameter(java.lang.String name)
```

welche auf das `HttpServletRequest`-Objekt angewendet wird, der Wert des Formularfeldes `name` gewonnen werden.

Das Servlet `webkompendium.servlet.Servlet2` (Abbildungen 24.7 und 24.8) gibt auf diese Weise den Wert des Formularfeldes „`name`“ aus, wobei

die Übertragungsmethode beliebig ist. Ferner bindet dieses Servlet im Gegensatz zu `Servlet1` die CSS-Datei vom Tomcat-Server ein und erzeugt ein JavaScript, welches `titel`, `URL` und `lastModified` des Dokuments ausgibt: Da das Dokument die dynamische Ausgabe des vom Servlet-Container Tomcat ausgeführten Servlets ist, und Tomcat die HTTP-Headerinformationen sauber setzt, kann der Browser das JavaScript problemlos ausgeben; es wird der Zeitpunkt der Ausführung des Servlets, der Pfad zum Servlet sowie der von diesem erzeugte Titel ausgegeben (Abbildung 24.9), wobei zuvor wieder die Projektkonfiguration in der Datei `web.xml` (Abbildung 24.4) angepasst werden muss.

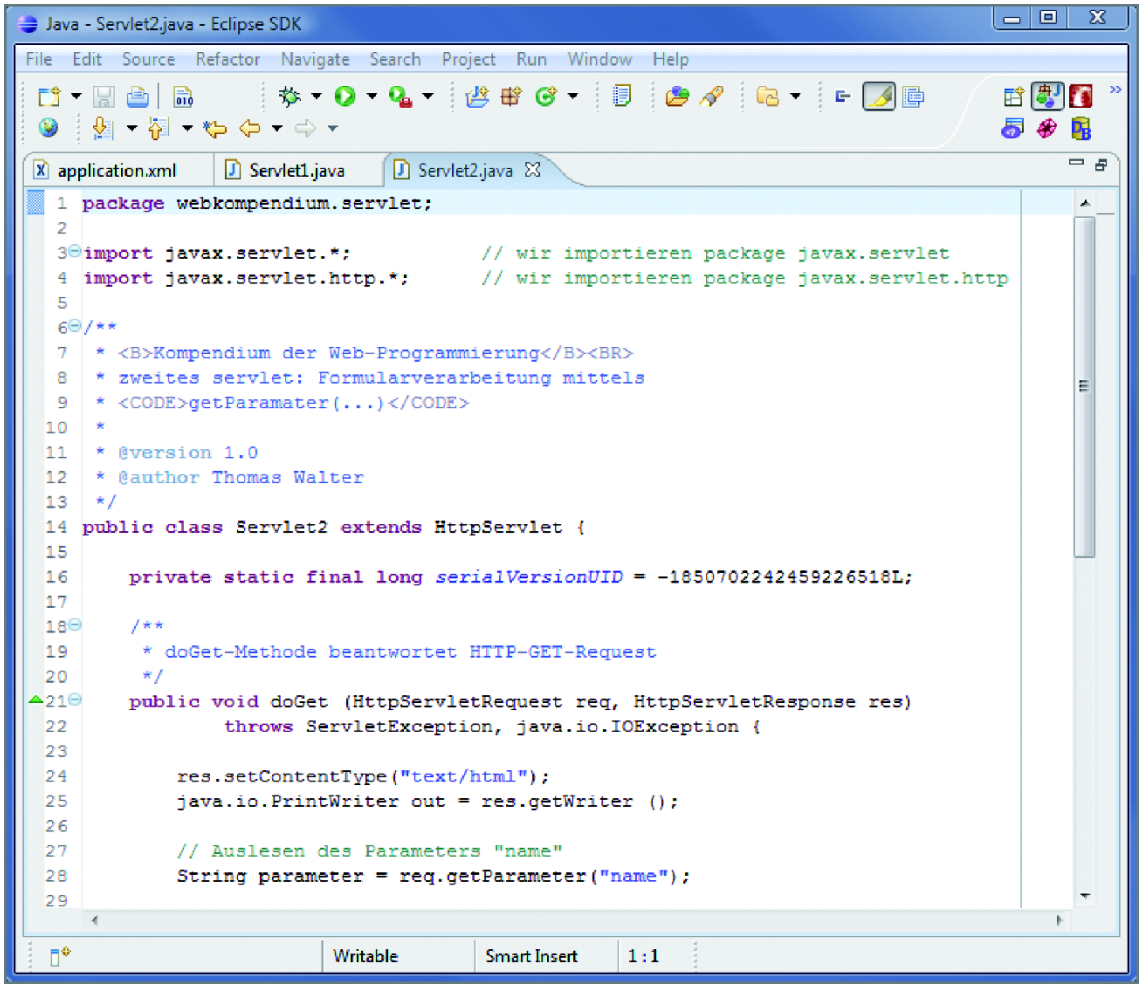



**Abbildung 24.7:**
Die Klasse `Servlet2`, Teil I

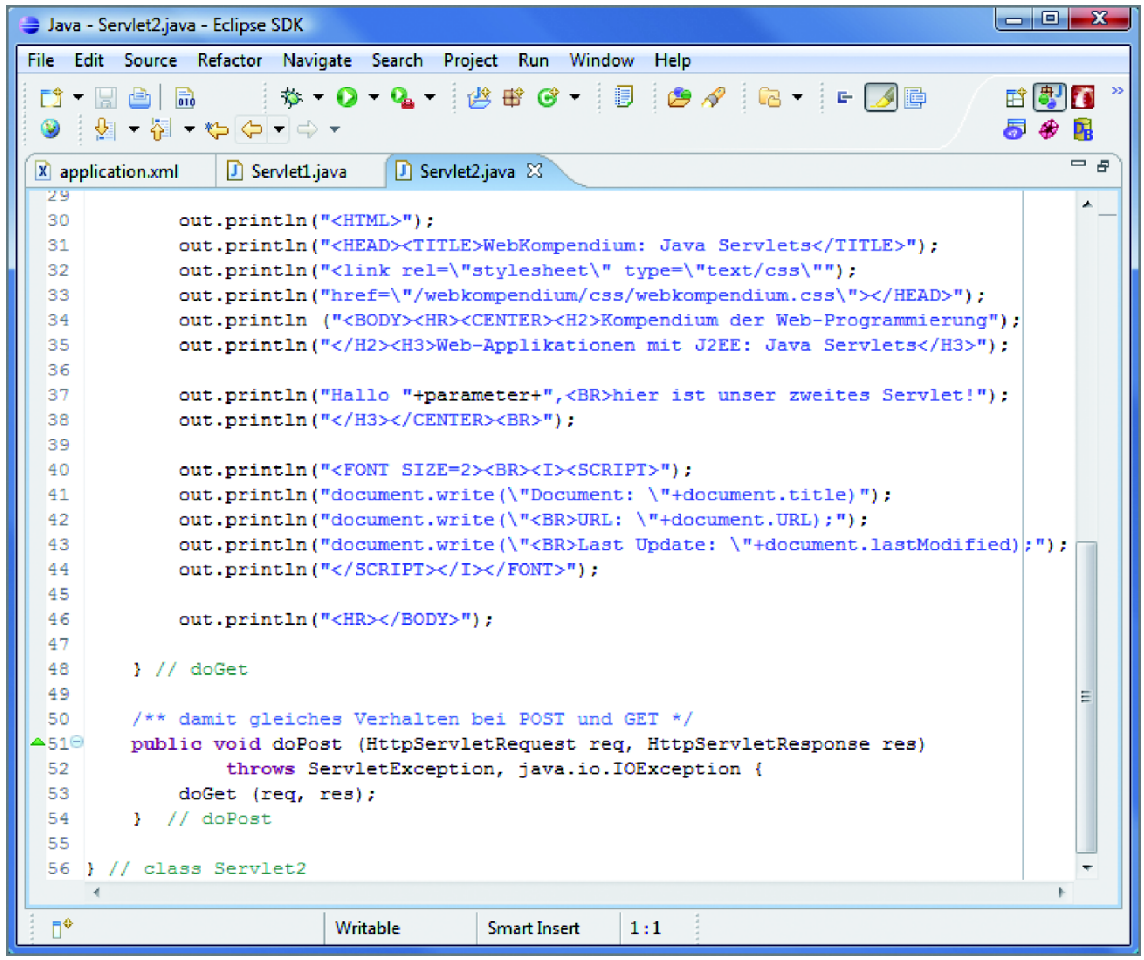



**Abbildung 24.8:**
Die Klasse `Servlet2`, Teil II

**Abbildung 24.9:**
Ausführen von `Servlet2`

#### 24.2.5.1 Weitere Request-Methoden

Auf Instanzen der Klasse `HttpServletRequest` können etliche weitere Methoden angewendet werden; nützlich sind etwa:

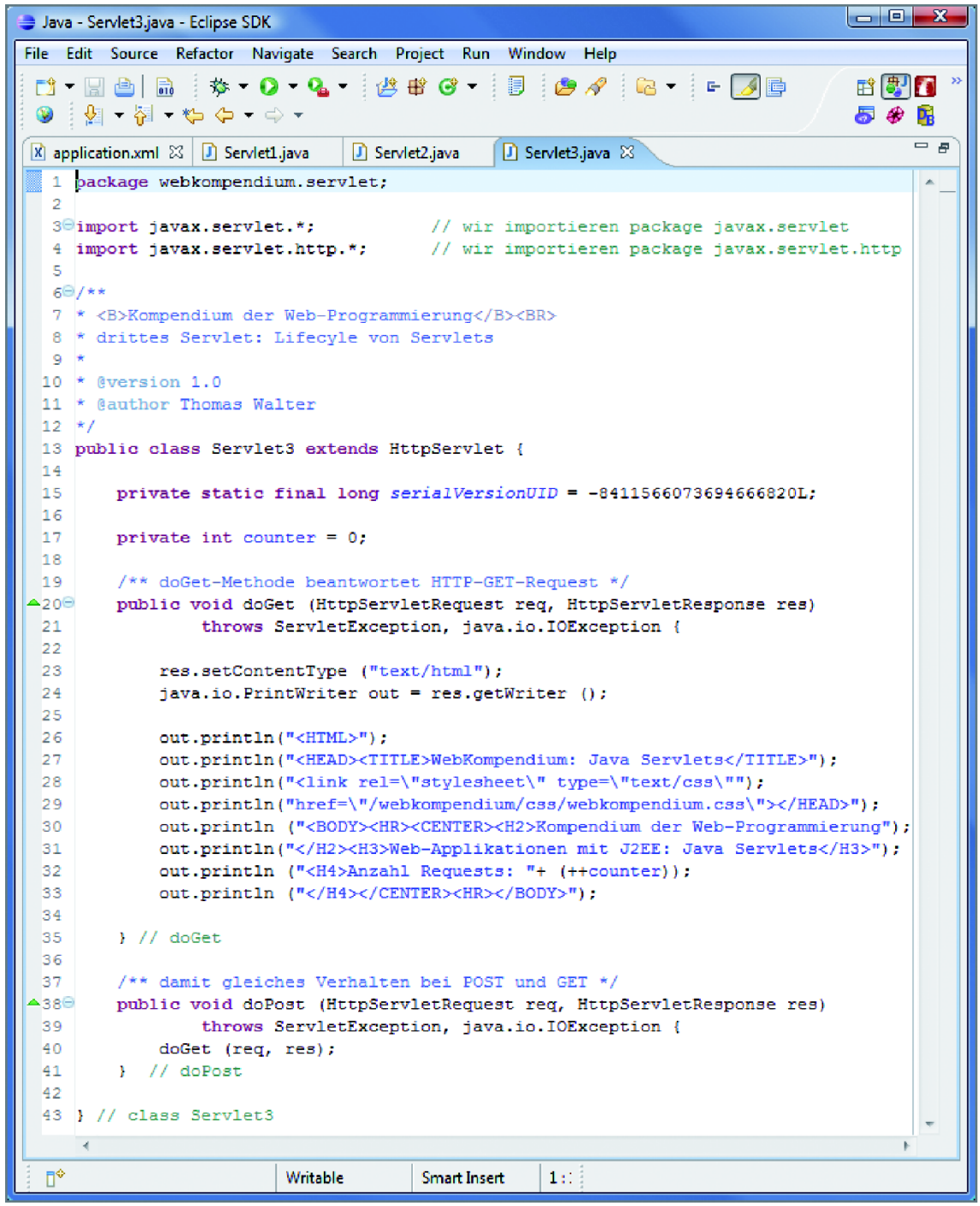
- `getMethod()` zur Abfrage der HTTP-Übertragungsmethode (vgl. 1.6.5.1);
- `getRequest()` für eine String-Darstellung der Anfrage;
- `getProtocol()` gibt das Protokoll an;
- `getPathInfo()` für den vollständigen Pfad der Anfrage;
- `getRemoteAdr()` für die ip-Adresse des anfragenden Clients; dies wurde schon in der Klasse `Servlet1` (Abbildung 24.5) angewendet.

### 24.2.6 Lebenszyklen von Servlets

Wir haben nun das Servlet als Instanz der `HttpServlet`-Klasse kennengelernt. Der Servlet-Container (Tomcat) übernimmt die Verwaltung: Er instanziiert das Servlet, leitet die HTTP-Requests weiter und zerstört ggf. das Servlet. Das Servlet kann, ganz ähnlich zum fastCGI-Prozess, bereits beim Starten des Servlet-Containers bzw. beim Laden der Web-Applikation in den Server, oder erst beim ersten Request an das Servlet erzeugt werden; danach lebt das Servlet aber weiter. Beispiel `webkompendium.servlet.Servlet3` (Abbildung 24.10) zeigt ein Servlet, welches einfach die Requests zählt; dabei lebt das Servlet, solange die Web-Applikation im Container vorhanden ist (Abbildung 24.11). Parallele Zugriffe auf das gleiche Servlet können dadurch sicher gestartet werden, dass die Methode `doGet` als `synchronized` deklariert wird:

```
public synchronized void doGet (...)
```

Die Servlet-Methode `init()` kann im Sinne eines Konstruktors genutzt werden, denn sie wird bei Instanziierung des Servlets automatisch ausgeführt. Umgekehrt wird die Methode `destroy()` bei der Vernichtung eines Servlets ausgeführt. Die gleichen Methoden haben wir schon beim Applet kennengelernt (vgl. 18.3).




```java
package webkompendium.servlet;

import javax.servlet.*;           // wir importieren package javax.servlet
import javax.servlet.http.*;      // wir importieren package javax.servlet.http

/**
* <B>Kompendium der Web-Programmierung</B><BR>
* drittes Servlet: Lifecyle von Servlets
*
* @version 1.0
* @author Thomas Walter
*/
public class Servlet3 extends HttpServlet {

    private static final long serialVersionUID = -8411566073694666820L;

    private int counter = 0;

    /** doGet-Methode beantwortet HTTP-GET-Request */
    public void doGet (HttpServletRequest req, HttpServletResponse res)
            throws ServletException, java.io.IOException {

        res.setContentType ("text/html");
        java.io.PrintWriter out = res.getWriter ();

        out.println("<HTML>");
        out.println("<HEAD><TITLE>WebKompendium: Java Servlets</TITLE>");
        out.println("<link rel=\"stylesheet\" type=\"text/css\"");
        out.println("href=\"/webkompendium/css/webkompendium.css\"></HEAD>");
        out.println ("<BODY><HR><CENTER><H2>Kompendium der Web-Programmierung");
        out.println("</H2><H3>Web-Applikationen mit J2EE: Java Servlets</H3>");
        out.println ("<H4>Anzahl Requests: "+ (++counter));
        out.println ("</H4></CENTER><HR></BODY>");

    } // doGet

    /** damit gleiches Verhalten bei POST und GET */
    public void doPost (HttpServletRequest req, HttpServletResponse res)
            throws ServletException, java.io.IOException {
        doGet (req, res);
    }  // doPost

} // class Servlet3
```

**Abbildung 24.10:**
Die Klasse `Servlet3`

**Abbildung 24.11:** Ausführen von `Servlet3`: Das Servlet zählt weiter

#### 24.2.6.1 Übergabe von Startwerten

Eine weitere Parallele des Servlets zum Applet ist die Möglichkeit der Übergabe von Startwerten: In der Konfigurationsdatei der Web-Applikation, `web.xml`, ist hierfür im `<servlet>`-Block ein entsprechender Teil zu ergänzen:

```
<servlet>
      <servlet-name>Servlet1</servlet-name>
      <description>erstes Beispielservlet</description>
      <servlet-class>
               webkompendium.servlet.Servlet1
      </servlet-class>

      <init-param>
              <param-name>n</param-name>
              <param-value>42</param-value>
      </init-param>

</servlet>
```

Nun kann das Servlet mit der Methode `getInitParam(name)` auf diesen Wert zugreifen, etwa in der `init`-Methode beim Starten:

```
private int n;
public void init() {
    n = getInitParameter("n");
} // init
```

#### 24.2.6.2 Servlets und CGI

Viele Möglichkeiten hat die CGI-Programmierung durch die CGI-Umgebungsvariablen eröffnet. Die gleiche Information kann auch von einem Servlet genutzt werden, allerdings werden hier anstelle von Umgebungsvariablen Methoden des `HttpServletRequest`-Objektes verwendet; Beispiele hierfür sind

- anstelle von `SERVER_NAME` die Methode `req.getServerName();`
- anstelle von `SERVER_PROTOCOL` die Methode `req.getServerProtocol();`
- anstelle von `SERVER_PORT` die Methode `req.getServerPort();`
- anstelle von `REMOTE_HOST` die Methode `req.getRemoteHost ()`.

#### 24.2.6.3 HTML-Ausgabe von Servlets

Das Interface `ServletResponse` deklariert Methoden für die *HTML-Ausgabe eines Servlets:*

- `public void setContentType(java.lang.String type)` setzt zunächst den MIME-Typ der Response, üblicherweise „text/html“;
- `public java.io.PrintWriter getWriter()` liefert einen `PrintWriter` für Zeichendaten

Über den `PrintWriter` haben die bisherigen Beispiele ihre Ausgabe gesteuert; wichtig für die Optimierung ist hier die Methode

```
setBuffersize(int size)
```

welche auf das Response-Objekt angewendet wird: Hierdurch wird die Puffergröße für die Ausgabe gesetzt, was die Performance deutlich erhöhen kann.

Ebenfalls in der Klasse `HttpServletResponse` finden sich Konstanten für die HTTP-Antwortcodes nach Tabelle 1.3 etwa `SC_OK` für 200 (ok) und `SC_NOT_FOUND` für 404 (not found). Diese Antwortcodes können mit den Methoden

```
public void setStatus(int sc)
public void sendError(int sc)
```

von `HttpServletResponse` zum Client gesendet werden.

## 24.3 Datenbankanbindung mit Java

Entsprechend der übrigen Beispiele in diesem Buch soll nun ein Servlet für die *Datenbankabfrage* vorgestellt werden. Hierfür muss Java um die Möglichkeit der Datenbankkommunikation erweitert werden, ähnlich wie es DBI für Perl leistet (vgl. 10.7). Dies ist bei Java *JDBC:* Java Database Connectivity [web].

Das wesentliche Paket für JDBC ist `java.sql`, und dieses ist bereits im Umfang von J2SE enthalten; dennoch wird der JDBC-Standard zu den J2EE-Techniken gezählt.

### 24.3.1 Arbeitsweise von JDBC

Die Arbeitsweise von JDBC entspricht konzeptionell dem Vorgehen etwa von Perl-DBI:

1. Aufbau der Verbindung zu einer Datenbank in einem DBMS auf einem Datenbankhost;
2. Absenden einer SQL-Anweisung;
3. im Falle einer SELECT-Abfrage: Abarbeiten des zurückgegebenen Resultsets;
4. Speicherbereinigung: Löschen des Resultsets und der Datenbankverbindung.

Dabei ist JDBC selbst wieder eine Abstraktionsschicht: Die JDBC-Syntax ist unabhängig vom jeweiligen DBMS; nur das Laden des Treibers ist spezifisch. Die Darstellung von Abbildung 10.33 gilt somit auch für Java mit JDBC, und ähnlich wie bei Perl wird ein spezifischer Treiber für das jeweilige DBMS benötigt.

Die JDBC-Treiber werden in vier Arten klassifiziert; die erste simuliert über ODBC einen JDBC-Treiber (Bridge Driver), ein einfacher Weg, wenn kein JDBC-Treiber verfügbar ist. Für die meisten Datenbanksysteme besteht aber inzwischen ein reiner Java-Treiber für das native Protokoll des jeweiligen Systems, ein Treiber der vierten Stufe. Diese Treiber werden als jar-Archiv verteilt; dieses jar-Archiv muss Teil des Klassenpfades sein, oder hier für Servlets im `lib`-Ordner des gesamten Tomcats oder der Web-Applikation liegen.

Im Detail läuft die JDBC-Abfrage folgendermaßen ab:

1. Laden des DBMS-spezifischen Treibers; für MySQL bedeutet dies:

   ```
   Class.forName("com.mysql.jdbc.Driver").newInstance();
   ```

2. Aufbau der Verbindung (Erzeugung des Database-Handles):

   ```
   java.sql.Connection dbh;
   String dsn = "jdbc:mysql://rechner:port/datenbank";
   dbh = DriverManager.getConnection(dsn,user,kennwort);
   ```

   Java bezeichnet im Sinne des Webs den „Data-Source-Name“ als URL und verwendet einen Bezeichner `url`, da der Aufbau formal tatsächliche eine URL nach 1.6.9 ist.
3. Erzeugen des Statement-Handles:

   ```
   Statement sth = dbh.createStatement();
   ```

4. Durchführen einer Abfrage, welche ein Resultset liefert:

   ```
   ResultSet rs = sth.executeQuery(sql);
   ```

   Für andere SQL-Anweisungen stehen entsprechende Methoden, etwa `executeUpdate`, bereit.

   ```
   Statement sth = dbh.createStatement();
   ```

5. Bei SELECT-Abfragen: Zeilenweises Auswerten des Resultsets mittels

   ```
   rs.next()
   ```

   Dann kann eine Zeichenkettendarstellung des ersten Elementes durch

   ```
   rs.getString(1)
   ```

   gewonnen werden. Vorsicht, der numerische Index beginnt tatsächlich bei 1 und nicht bei 0.
6. Abbau des Resultsets und der Verbindung durch

   ```
   rs.close();
   dbh.close();
   ```

Das Servlet `webkompendium.servlet.Servlet4` (Abbildungen 24.12 und 24.13) wendet dieses Prinzip für die Auswertung der Literaturtabelle nach Kapitel 7 an; das Ergebnis zeigt Abbildung 24.14.

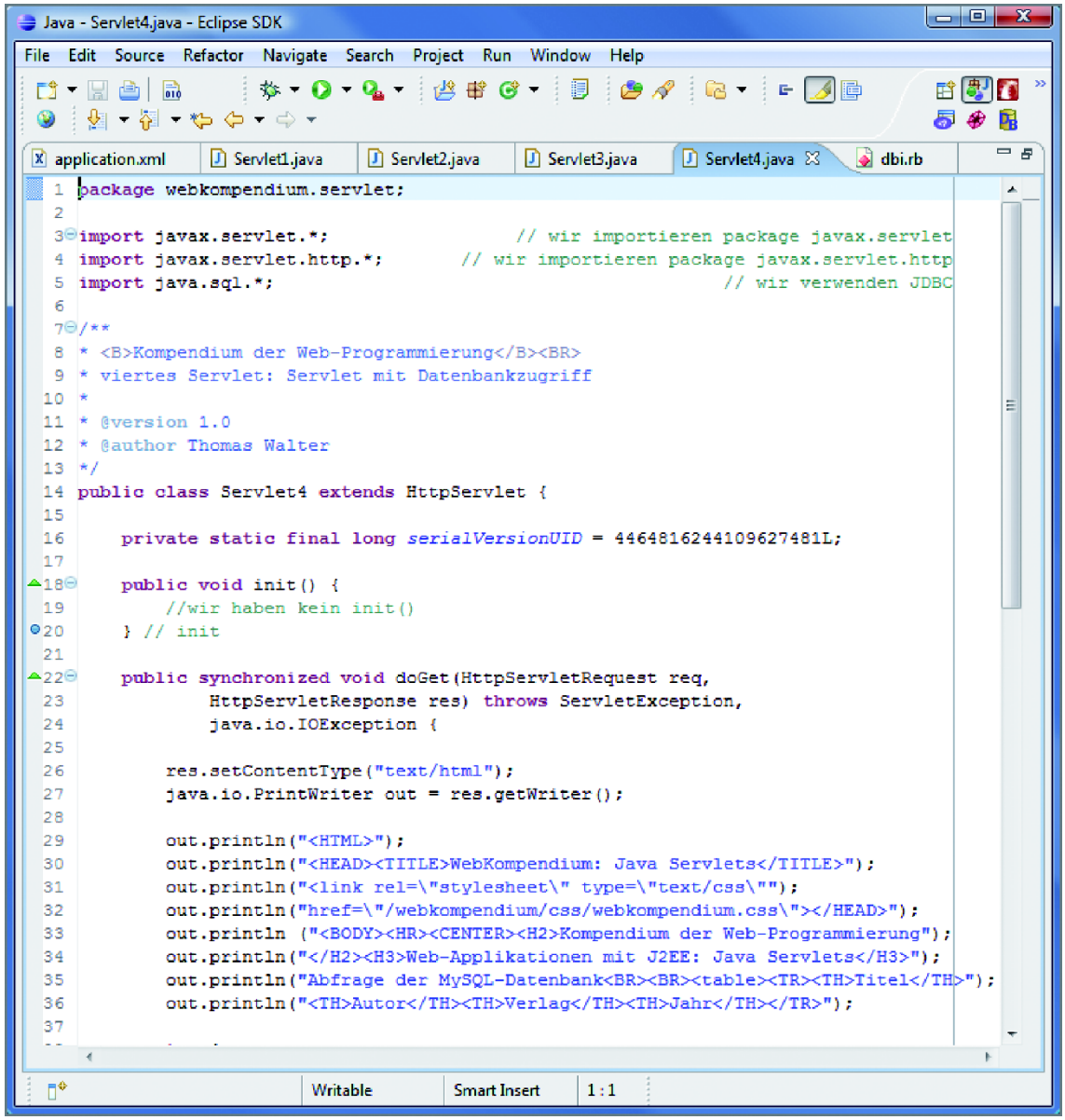



**Abbildung 24.12:**
Die Klasse `Servlet4`, Teil I

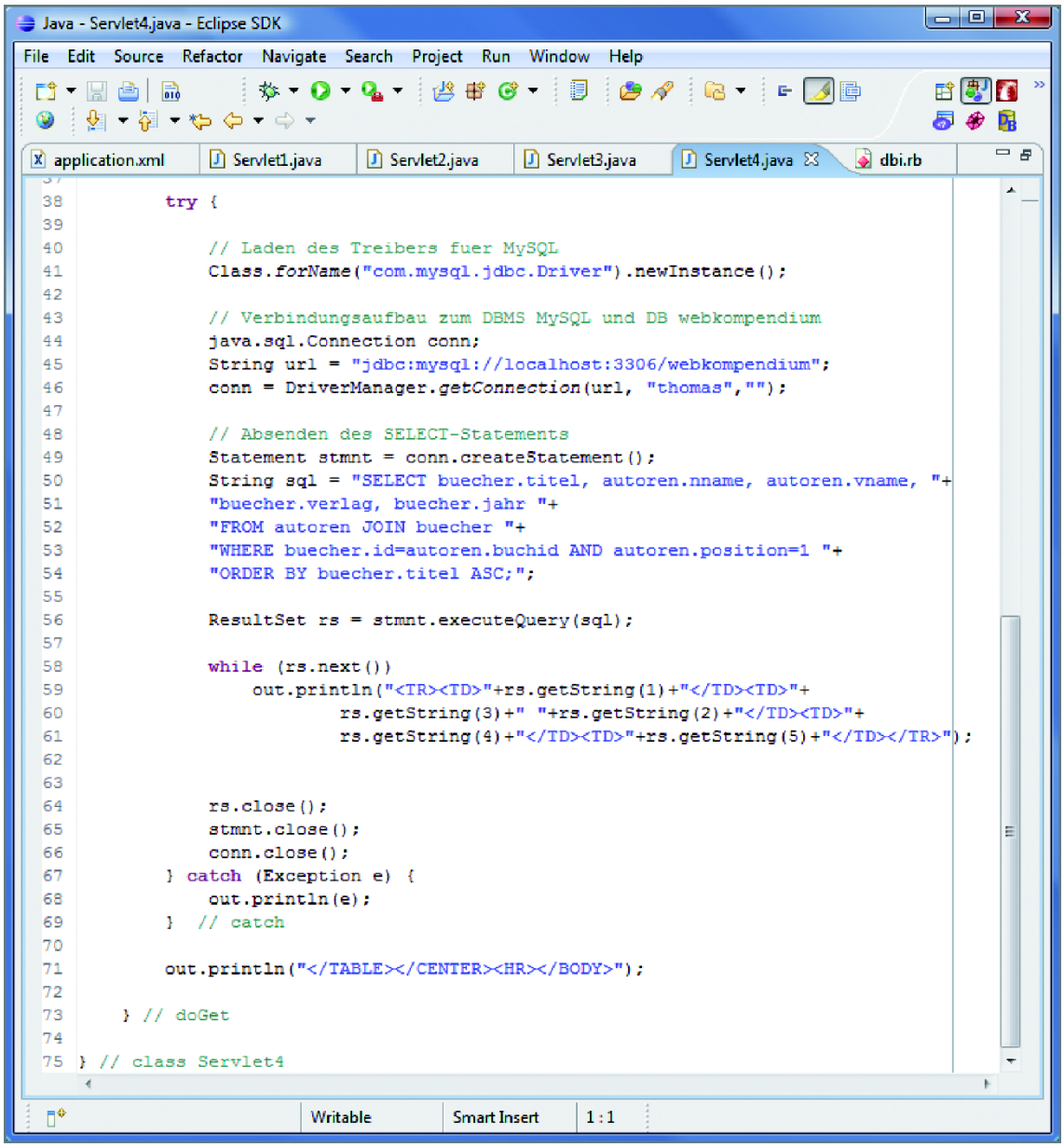



**Abbildung 24.13:**
Die Klasse `Servlet4`, Teil II

**Abbildung 24.14:** Ausführen von `Servlet4` (mit URL-Mapping auf `/db` in `web.xml`

#### 24.3.1.1 Connection-Pool und JNDI

Der Apache-Webserver verwendet seit langem eine einfache Strategie zur Performance-Optimierung: Bereits beim Starten werden einige Prozesse (oder bei moderneren Versionen: einige Threads) „auf Vorrat“ erzeugt, welche auf künftige kommende Anfragen warten und diese dann schneller bedienen können. Dieses Prinzip ist auch für Datenbankanwendungen möglich: Ein *Connection-Pool* verwaltet mehrere Datenbankanbindungen und erzeugt einige freie „auf Vorrat“.

Derartige Connection-Pools, eine klassische Technik für Datenbankapplikationen, können Anwendungen wie die hier gezeigte insbesondere für mehrere parallele Anfragen wesentlich beschleunigen. Im Netz sind Vorlagen für gute Connection-Manager zu finden; J2EE bietet hierfür beispielsweise das Interface

```
public interface ConnectionPoolDataSource
```

welches grundlegend das Verhalten von Connection-Pools festlegt; es sind zahlreiche gute Implementierungen hierfür zu finden.

Java selbst bietet mit dem Java Naming and Directory Interface (JNDI) [web] darauf aufbauend einen universellen Weg zur Nutzung beliebiger Datenquellen – sei es nun ein Connection-Pool oder nur ein einfacher Dateizugriff.

Darüber hinaus erweitert JNDI die Möglichkeiten, weitere Datenquellen wie Verzeichnis- und Namensdienste – etwa LDAP oder AcitveDirectory – einfach einzubinden.

## 24.4 JSP: JavaServer Pages

Warum ist für den Einsatz im Web PHP deutlich beliebter als Perl? Ein Grund hierfür liegt sicherlich in der Vorgehensweise: Während bei Perl der HTML-Code eher umständlich aus der Programmierung generiert wird, besteht PHP aus einem harmlosen HTML-Gerüst, in welches einige Code-Anteile integriert sind.

Diesen Unterschied hat auch Sun erkannt, und nach den Servlets, welche eher den Perl-Weg gehen, *JavaServer Pages* (JSP) vorgestellt: Ein HTML-Rahmen, in welchen aktive Anteile integriert sind.

Letztlich sind JSP nur ein Ansatz, ein HTML-Template mit Programmiercode zu verbinden. Hier wurden einige ähnliche Ansätze bereits vorgestellt: Smarty für PHP sowie die Template-basierte View von django und Ruby on Rails. Für Java ist etwa Velocity, ein Framework im Jakarta-Projekt, ein weiterer Ansatz zu diesem Ziel.

Hier sollen das Prinzip und die Arbeitsweise von JSP kurz vorgestellt werden.

### 24.4.1 Aufbau von JSP

Wie beschrieben handelt es sich bei einer JSP-Datei um ein HTML-Gerüst, in welches Java-Codeanteile integriert werden können; Abbildung 24.16 zeigt `webkompendiumJSP.jsp`.

Diese Datei ist einfach in ein geeignetes Verzeichnis – nicht unterhalb von `WEB-INF` – auf dem Tomcat-Server abzulegen, anschließend genügt ein Aufruf durch den Browser mit der direkten URL (Abbildung 24.15). Hier ist keine Anpassung in der `server.xml` oder anderes notwendig.

**Abbildung 24.15:** Ausführen von `webkompendiumJSP.jsp`

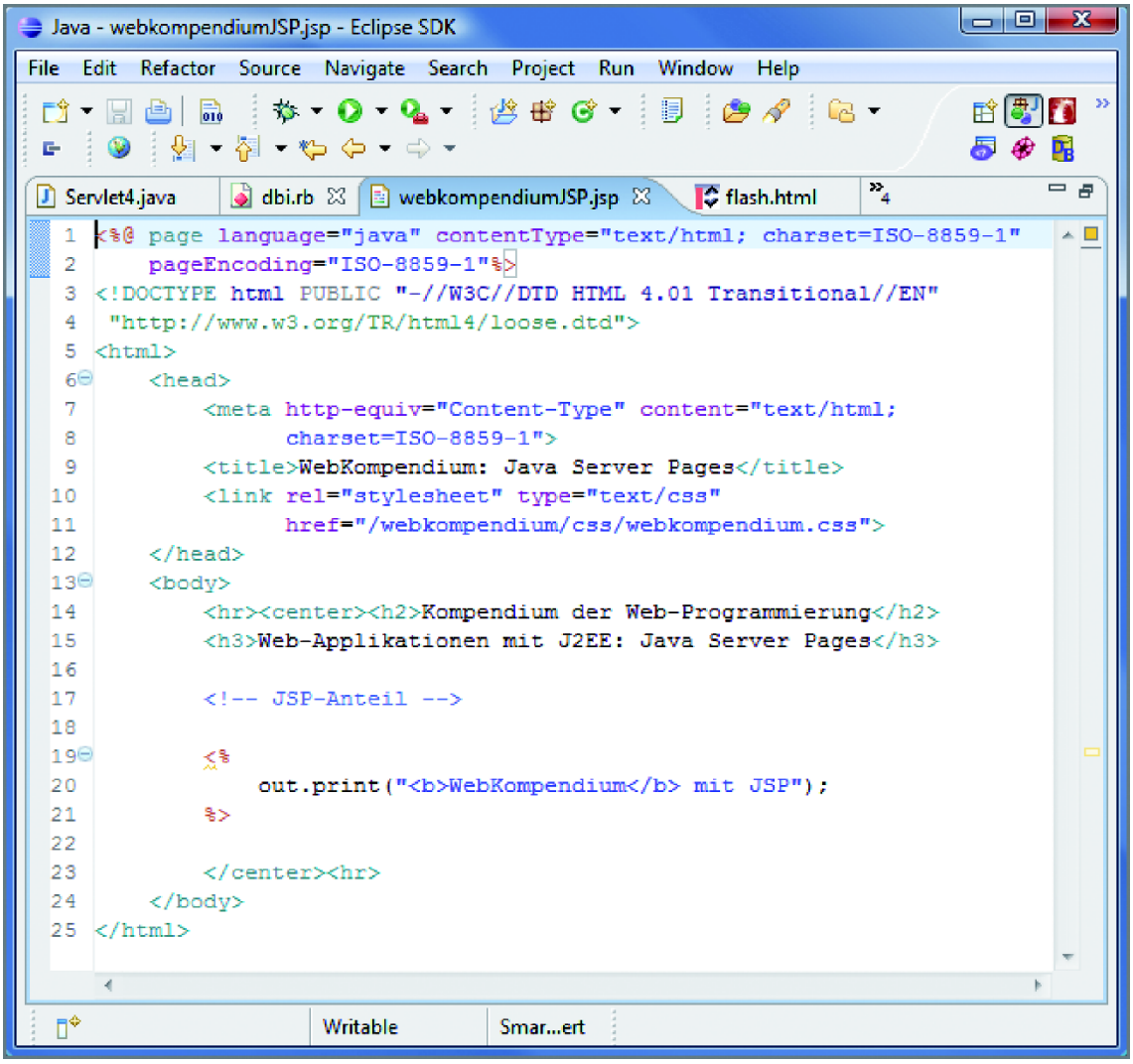



**Abbildung 24.16:** Die JSP-Datei `webkompendiumJSP.jsp`

Zentral für die weitere Nutzung von JSP sind die speziellen JSP-Tags:

- `<% java-code %>` führt die Java-Anweisungen aus wie im Beispiel `webkompendiumJSP.jsp`;
- `<%= java-code %>` wertet den Java-Ausdruck aus und wandelt das Ergebnis in eine Zeichenkette um, die dann angezeigt wird;
- `<%! java-code %>` dient zur Deklaration von statischen Variablen oder Instanzvariablen und von Methoden.

### 24.4.2 Arbeitsweise von JSP

Der erste Aufruf einer JSP-Seite dauert im Normalfall lange, die späteren gehen deutlich schneller – warum? Hintergrund ist die Arbeitsweise von JSP:

**Intern wird eine JSP-Datei in ein Servlet umgewandelt, dieses wird compiliert und ausgeführt.**

Aus diesem Grund benötigt der Tomcat-Server zur Ausführung von JSP auch einen Java-Compiler, während für Servlets im Prinzip eine JVM ausreicht.

Tomcat speichert das generierte Servlet unterhalb seines Verzeichnisses `<tomcat_home>/work` ab; die Abbildungen 24.17 und 24.18 zeigen den aus der JSP-Datei `webkompendiumJSP.jsp` generierten Sourcecode für das Servlet.

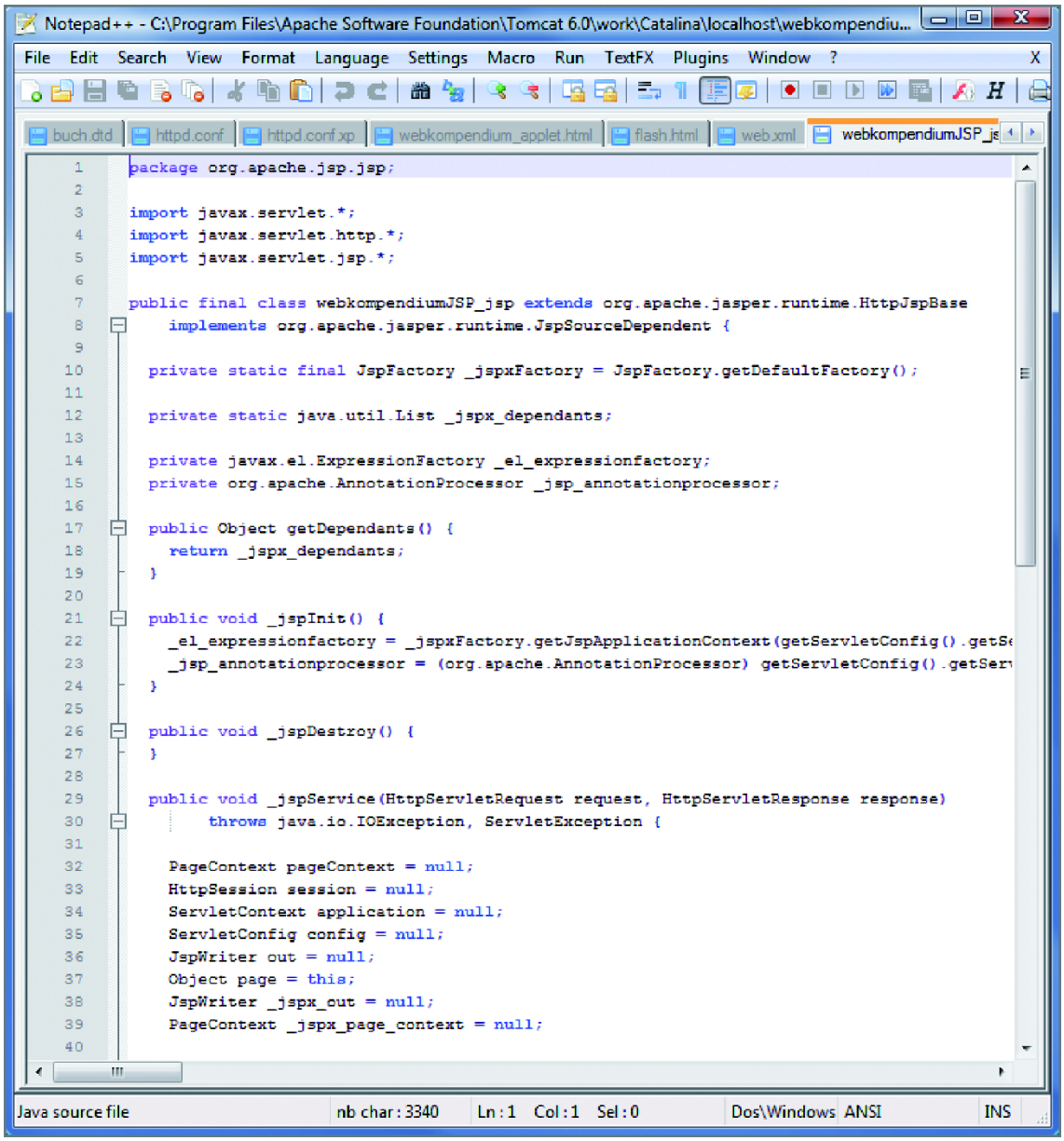
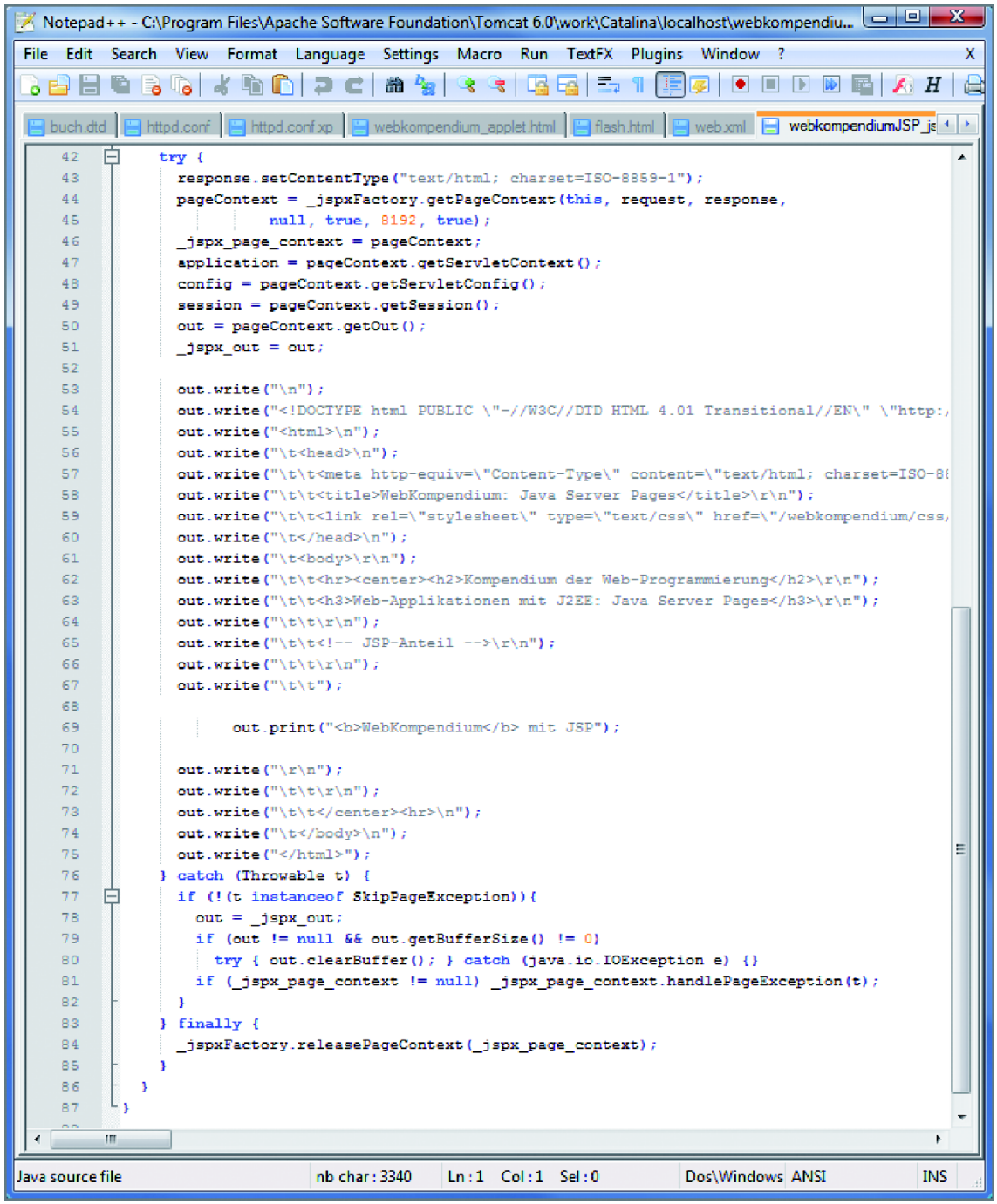



**Abbildung 24.17:**
Das aus `webkompendiumJSP.jsp` generierte Servlet, Teil I




```
42      try {
43        response.setContentType("text/html; charset=ISO-8859-1");
44        pageContext = _jspxFactory.getPageContext(this, request, response,
45                    null, true, 8192, true);
46        _jspx_page_context = pageContext;
47        application = pageContext.getServletContext();
48        config = pageContext.getServletConfig();
49        session = pageContext.getSession();
50        out = pageContext.getOut();
51        _jspx_out = out;
52
53        out.write("\n");
54        out.write("<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\" \"http:/
55        out.write("<html>\n");
56        out.write("\t<head>\n");
57        out.write("\t\t<meta http-equiv=\"Content-Type\" content=\"text/html; charset=ISO-8
58        out.write("\t\t<title>WebKompendium: Java Server Pages</title>\r\n");
59        out.write("\t\t<link rel=\"stylesheet\" type=\"text/css\" href=\"/webkompendium/css/
60        out.write("\t</head>\n");
61        out.write("\t<body>\r\n");
62        out.write("\t\t<hr><center><h2>Kompendium der Web-Programmierung</h2>\r\n");
63        out.write("\t\t<h3>Web-Applikationen mit J2EE: Java Server Pages</h3>\r\n");
64        out.write("\t\t\r\n");
65        out.write("\t\t<!-- JSP-Anteil -->\r\n");
66        out.write("\t\t\r\n");
67        out.write("\t\t");
68
69            out.print("<b>WebKompendium</b> mit JSP");
70
71        out.write("\r\n");
72        out.write("\t\t\r\n");
73        out.write("\t\t</center><hr>\n");
74        out.write("\t</body>\n");
75        out.write("</html>");
76      } catch (Throwable t) {
77        if (!(t instanceof SkipPageException)){
78          out = _jspx_out;
79          if (out != null && out.getBufferSize() != 0)
80            try { out.clearBuffer(); } catch (java.io.IOException e) {}
81          if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
82        }
83      } finally {
84        _jspxFactory.releasePageContext(_jspx_page_context);
85      }
86    }
87  }
```

**Abbildung 24.18:** Das aus `webkompendiumJSP.jsp` generierte Servlet, Teil II

### 24.4.3 JSP-Direktiven

JSP verfügen über ein Set eigener Direktiven, welche durch die Syntax <% eingeleitet werden; das Beipsiel `webkompendiumJSP.jsp` verwendet dies bereits in der ersten Zeile.

So kann etwa durch

```
<%@ page import="java.sql.*"  %>
```

das JDBC-Paket für eine Datenbankauswertung mit JSP importiert werden.

Per Default importiert jede JSP-Datei die vier Pakete

- `java.lang`;
- `javax.servlet`;
- `javax.servlet.http`;
- `javax.servlet.jsp`.

### 24.4.4 Ein typisches JSP-Problem

So attraktiv JSP auf den ersten Blick erscheinen mögen, haben sie doch einen nicht unerheblichen Nachteil: Sie verleiten dazu, jegliches Porgrammiermuster wie MVC zu verletzen, da die eigentlich durch die JSP gebildete View zu leicht auch Aufgaben eines Controllers übernehmen kann.

Die vielen Alternativen zu JSP, etwa Velocity, sind hier strenger.

## 24.5 Einige weitere J2EE-Begriffe

Das J2EE-Framework besteht aus einer großen Anzahl weiterer Techniken, deren genaue Vorstellung den Rahmen dieses Buches gänzlich sprengen würden. Deshalb werden hier nur noch die Bedeutung einiger besonders wichtiger Begriffe vorgestellt.

### 24.5.1 Enterprise JavaBeans

*Enterprise JavaBeans* erweitern das Modell der JavaBean für modularisierte Softwarekomponenten auf Unternehmensanwendungen. EJBs ermöglichen etwa Datenbank-Transaktionen, was Servlets nicht ohne weiteres können.

Für den Betrieb von EJBs reicht allerdings der Servlet-Container Tomcat nicht mehr aus, es wird ein voller Java EE-Server wie JBoss oder Geronimo benötigt.

### 24.5.2 JavaServer Faces

*Java ServerFaces* sind ein Framework für die Verwendung von Komponenten für Benutzerschnittstellen in Webseiten; sie ermöglichen eine strenge Umsetzung von MVC (vgl. 4.3) bei der Entwicklung von Web-Applikationen.

### 24.5.3 Struts

*Struts* gehört nicht direkt zu J2EE, sondern ist eines der Jakarta-Projekte der Apache Group. Struts bündelt bewährte Einzeltechniken wie Servlets und JSP, um nach einem Baukastenprinzip komplexe Web-Applikationen zu erzeugen.

# Teil V

# Ergänzungen zur Web-Programmierung

# 25

# Was sind Cookies, warum braucht man sie und warum sie keiner will

Das *Cookie* ist eine alte und einfache Technik der Web-Programmierung, welche häufig im Ruf steht, sicherheitstechnisch kritisch zu sein. Das ist allerdings bei genauer Betrachtung stark einzuschränken, was in diesem Kapitel genauer erläutert wird. Außerdem bilden Cookies eine wichtige Grundlage für das folgende Kapitel 26, das Session-Management.

## 25.1 Was sind Cookies?

Cookies sind eine spezielle Technik des HTTP-Protokolls. Der Name kommt vom „Plätzchen“ und bedeutet die Speicherung von Information auf dem Webclient: Der Webserver ist in der Lage, dauerhaft (persistent) Informationen auf dem Client abzulegen und beispielsweise beim nächsten Besuch der Seite darauf wieder zuzugreifen.

Der Vorteil von Cookies liegt in der Möglichkeit, Benutzerinformationen zu speichern: Etwa bei mehrsprachigen Seiten kann die ausgewählte Sprache auf dem Client hinterlegt werden und beim nächsten Aufruf wird dann diese Sprache direkt verwendet.

Wie schon erwähnt sind Cookies wichtig für das Session-Management.[1]

Ein typisches Beispiel für den Einsatz von Cookies liefert www.amazon.de: Ruft man diese Seite auf und sucht nach einem Artikel, bekommt man bei einem späteren Besuch diesen und ähnliche Artikel gleich auf der Startseite angeboten: Der Diensteanbieter kann über das Cookie den vorherigen Besuch rekonstruieren und darauf geschickt reagieren.

**Die Cookie-Information bezieht sich auf den Browser und ist somit in allen Browser-Fenstern und Tabs identisch.**

JavaScript/DOM würde von der Gültigkeit des Cookies im gesamten navigator-Objekt nach Abschnitt 15.5.2 sprechen.

## 25.2 Cookies im Browser kontrollieren

Alle modernen Browser bieten eine Möglichkeit, die auf dem Client abgelegten Cookies zu kontrollieren. Die Abbildungen 25.1 und 25.2 zeigen zunächst für zwei Browser diese Übersicht über alle abgelegten Cookies.

[1] es gibt aber Alternativen für ein Session-Management auch ohne Cookies, siehe Kapitel 26.

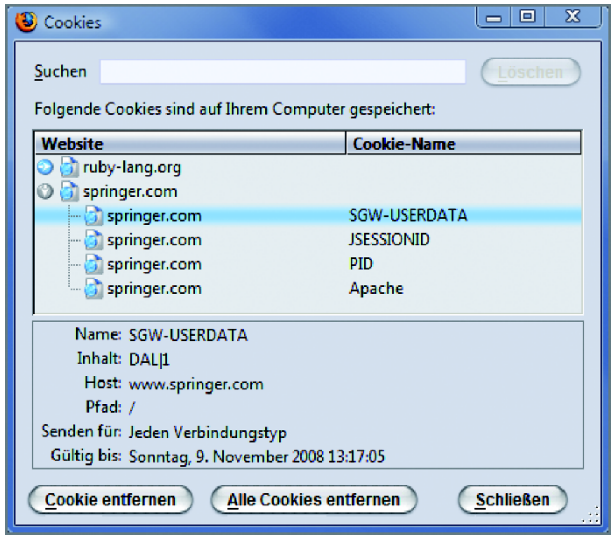



**Abbildung 25.1:** Übersicht über die gespeicherten Cookies im Firefox-Browser unter Extras|Einstellungen|Datenschutz

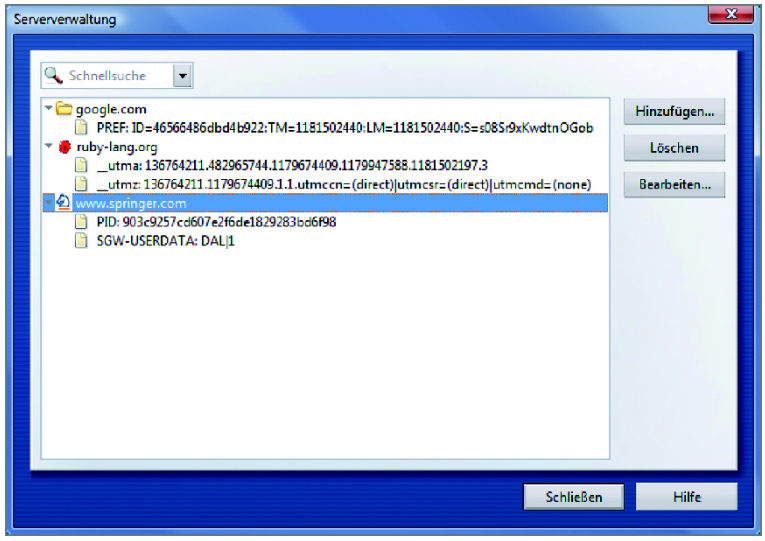



**Abbildung 25.2:** Übersicht über die gespeicherten Cookies im Opera-Browser unter Extras|Einstellungen|Erweitert

Zum einzelnen Cookie sind dabei genauere Informationen abrufbar (bei Firefox im unteren Teil des Fensters von Abbildung 25.1, für Opera in einem weiteren Fenster (Abbildung 25.3). Dabei können auch einfach der Name und der Wert des Cookies verändert werden (Abbildung 25.4).

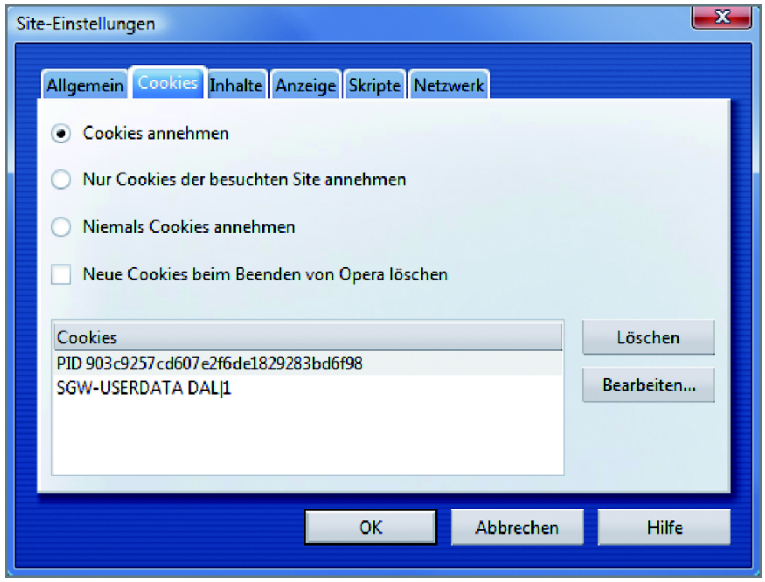



**Abbildung 25.3:** Detailinformation über ein gespeichertes Cookie im Opera-Browser

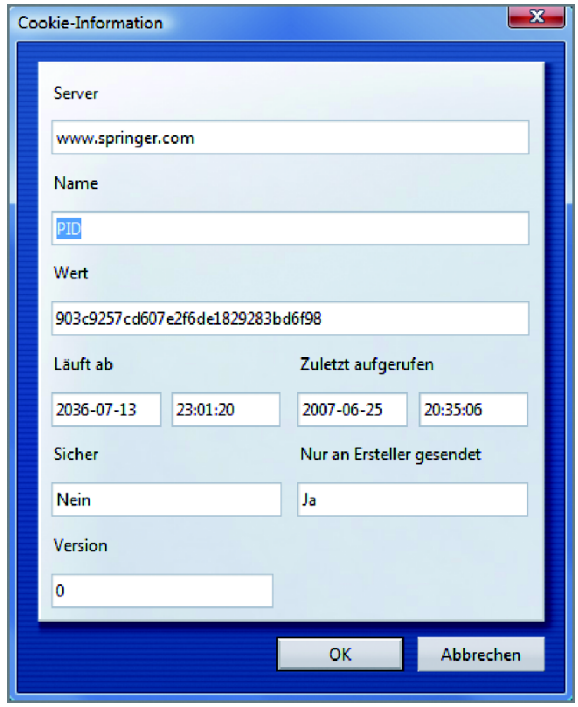



**Abbildung 25.4:** Ändern eines gespeicherten Cookies im Opera-Browser

Browserseitig kann der Umgang mit Cookies konfiguriert werden; dabei bieten die verschiedenen Browser unterschiedliche Möglichkeiten des Umgangs mit Cookies an. Ausführlich ist die Konfiguration etwa in Opera, hier kann gewählt werden, Cookies ganz zu verbieten, nur bestimmte Cookies – die von der aktuellen Seite – zu akzeptieren oder alle zuzulassen (Abbildung 25.5).

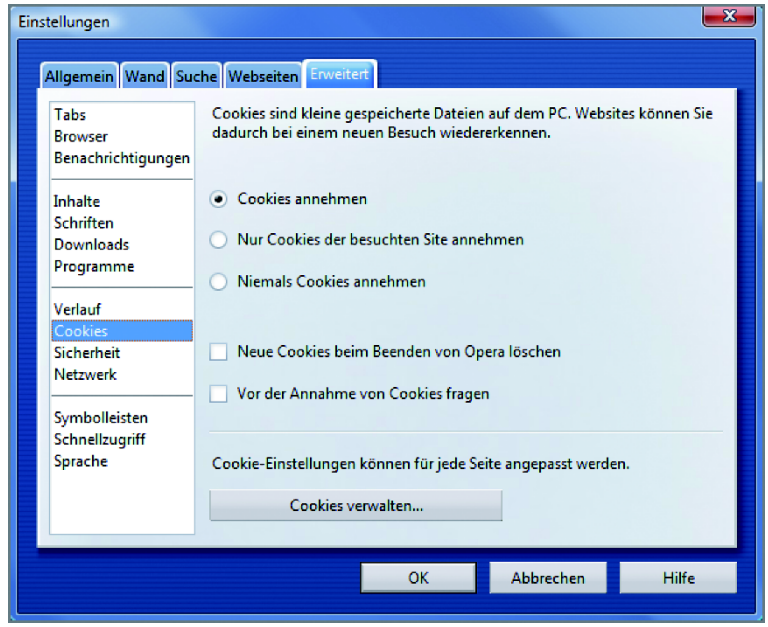



**Abbildung 25.5:** Konfiguration des Umgangs mit Cookies im Opera-Browser

## 25.3 Arbeitsweise von Cookies

Cookies sind eine Technik des HTTP-Protokolls (vgl. 1.6.5). Sie werden im HTTP-Header durch die Anweisung `Set-Cookie` gesetzt. Abbildung 25.6 zeigt den Anfang einer einfachen GET-Anfrage an die Site www.amazon.de; dabei sieht man, wie der Webserver drei Cookies setzt.

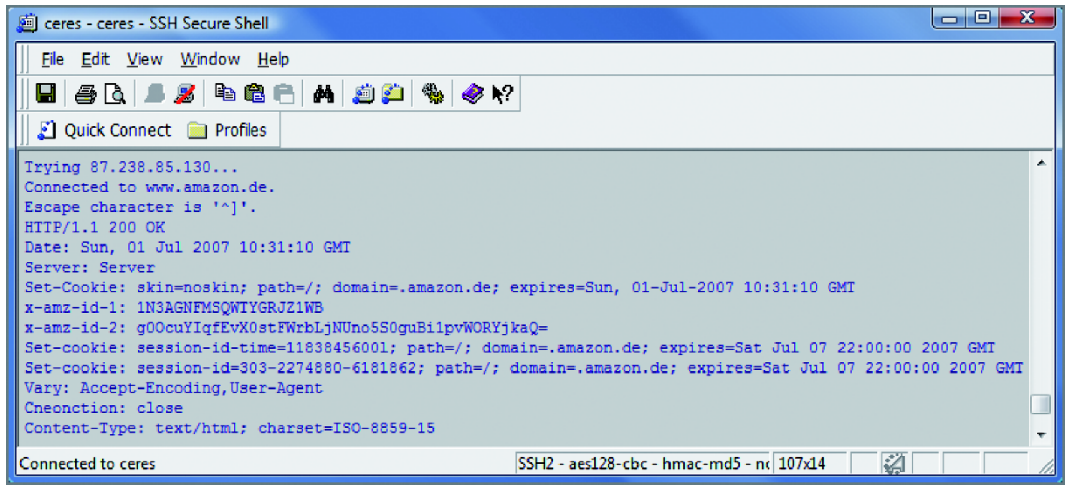



**Abbildung 25.6:** Setzen von Cookies im HTTP-Header von www.amazon.de (Echtbeispiel); die vorletzte Zeile ist ein Geheimnis des Anbieters

Nachdem mittels dieser HTTP-Methode die Cookie-Informationen vom Server zum Client übertragen wurden, legt der Client diese in Text-Dateien ab, die in einem einfachen Editor verändert werden können entsprechend der Browser-Aktion aus Abbildung 25.4.

## 25.4 Die Datenstruktur der Cookies

Der Kern eines Cookies ist die persistente Speicherung eines einfachen Key-Value-Paares wie etwa der später wichtige Key `PHPSESSID` und ein zugehöriger Wert.

Neben dieser Kerninformation gehören zahlreiche weitere Informationen zu einem Cookie, etwa dessen Gültigkeit: (Expires) Eine Datumsangabe, nach deren Ablauf das Cookie automatisch gelöscht wird (vgl. Abbildungen 25.2 und 25.1). Im Einzelnen stehen die Attribute nach Tabelle 25.1 für ein Cookie bereit.

| Attribut | Bedeutung |
|---|---|
| Name und Wert | die Kerninformation des Cookies: Key und Value, mit „=“ getrennt |
| `version` | Information für das Cookie-Management (Dezimalzahl) |
| `expires` | Gültigkeit des Cookies (Zeitpunkt der automatischen Löschung in UTC) |
| `max-age` | Ablaufzeit in Sekunden |
| `domain` | Domain oder Bestandteil des Domainennamens, für den das Cookie gilt |
| `path` | Pfad-Angabe, um die Gültigkeit des Cookies auf einen bestimmten Pfad zu beschränken |
| `port` | Beschränkung des Ports |
| `comment` | Kommentar zur Beschreibung des Cookies |
| `commenturl` | URL-Angabe für die weitere Beschreibung des Cookies |
| `secure` | erzwingt die Rücksendung des Cookie über eine mindestens ebenso sichere Verbindung, faktisch in https |
| `discard` | erzwingt Löschung des Cookies bei Beendigung des User-Agents |

**Tabelle 25.1:** Cookie-Attribute in HTTP 1.1

Im Beispiel aus Abbildung 25.6 werden die Cookies mit den Namen `skin`, `session-id` und `session-id-time` gesetzt.

## 25.5 Cookies und Sicherheit

Wie eingangs beschrieben haben *Cookies unter Sicherheitsaspekten* einen sehr schlechten Ruf. Dies lag zunächst an einer missverständlichen Publikation zu diesem Thema: Eine aktive Gefährdung des Clientrechners dadurch, dass dort eine Speicherung stattfindet, existiert nicht.

Die reale Gefahr durch Cookies liegt in einem anderen Bereich: Cookies erlauben zahlreiche Möglichkeiten der Verfolgung des Benutzers und damit des „Ausspionierens“ des Anwenders. Dies kann für Diensteanbieter ausgesprochen attraktiv und aufschlussreich sein. Etwa das Apache-Modul mod_user_track arbeitet so und verwendet Cookies, auch Google verwendet umfangreich Cookies für verschiedene Zwecke. Einige Web-Shops sind ohne Cookies gar nicht mehr bedienbar.

Ein zweiter Problemkreis liegt bei Rechnern, welche von mehreren Benutzern verwendet werden. Da die Cookie-Informationen offen in Textdateien abgelegt sind, besteht hier die Gefahr, dass andere Nutzer diese Auslesen können oder gar über bestehende Browser-Sessions oder zu langlebige Cookies auf fremde Sessions zugreifen können.

Ähnlich wie bei JavaScript liegt hier also eine nicht einfach zu bewertende Situation vor. In vielen Fällen wird man nicht auf Cookies verzichten wollen, man muss sich aber der damit verbundenen Gefahren bewusst sein.

Als Diensteanbieter sollte man sich klar sein, dass nicht jeder Kunde beliebig Cookies zulassen wird. Insbesondere Cookies mit unsinnig langer Gültigkeitsdauer, die man häufig antrifft, zeugen von einem unachtsamen Einsatz von Cookies durch den Diensteanbieter.

Und schließlich stellt das Cookie nicht nur für den Client, sondern insbesondere für den Diensteanbieter ein Sicherheitsproblem dar: Da sich Cookie-Informationen beliebig einfach ändern lassen (Abbildung 25.4), kann der Anbieter sich keinesfalls auf diese verlassen und sollte dies bei seinen Konzeptionen stets berücksichtigen.

## 25.6 Cookies in PHP

Abschließend zeigt dieses Kapitel noch, wie einfach Cookies in der Web-Programmierung eingesetzt werden können. Hierfür wird als Beispiel PHP gewählt.

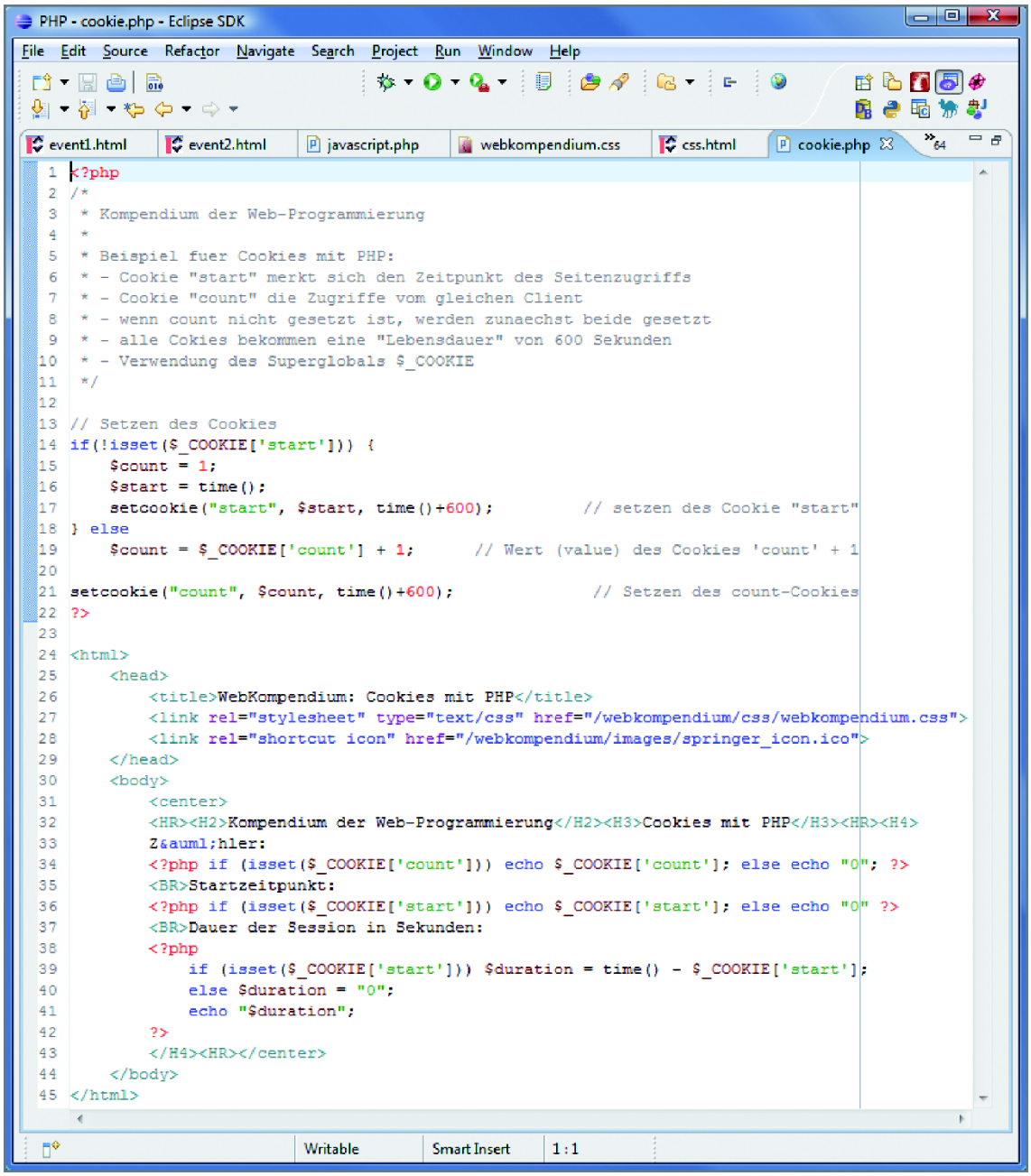
PHP bietet mehrere nützliche Methoden für den Einsatz von Cookies an.

## 25.7 Das Beispiel

Exemplarisch wird eine einfache Web-Anwendung vorgestellt, die drei Dinge leistet:

- beim ersten Aufruf merkt sich die Anwendung den Zeitpunkt und gibt bei späteren Aufrufen die vergangene Zeit und den Startzeitpunkt an;
- die Anzahl der Aufrufe dieser Seite durch den Client wird angegeben;
- diese Funktionalität ist auf eine Dauer von 600 Sekunden beschränkt.

Das PHP-Script `cookie.php` (Abbildung 25.7) leistet dies (Abbildung 25.8) und soll nun erklärt werden.




```php
<?php
/*
 * Kompendium der Web-Programmierung
 *
 * Beispiel fuer Cookies mit PHP:
 * - Cookie "start" merkt sich den Zeitpunkt des Seitenzugriffs
 * - Cookie "count" die Zugriffe vom gleichen Client
 * - wenn count nicht gesetzt ist, werden zunaechst beide gesetzt
 * - alle Cokies bekommen eine "Lebensdauer" von 600 Sekunden
 * - Verwendung des Superglobals $_COOKIE
 */

// Setzen des Cookies
if(!isset($_COOKIE['start'])) {
    $count = 1;
    $start = time();
    setcookie("start", $start, time()+600);         // setzen des Cookie "start"
} else
    $count = $_COOKIE['count'] + 1;      // Wert (value) des Cookies 'count' + 1

setcookie("count", $count, time()+600);              // Setzen des count-Cookies
?>

<html>
    <head>
        <title>WebKompendium: Cookies mit PHP</title>
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
        <link rel="shortcut icon" href="/webkompendium/images/springer_icon.ico">
    </head>
    <body>
        <center>
        <HR><H2>Kompendium der Web-Programmierung</H2><H3>Cookies mit PHP</H3><HR><H4>
        Z&auml;hler:
        <?php if (isset($_COOKIE['count'])) echo $_COOKIE['count']; else echo "0"; ?>
        <BR>Startzeitpunkt:
        <?php if (isset($_COOKIE['start'])) echo $_COOKIE['start']; else echo "0" ?>
        <BR>Dauer der Session in Sekunden:
        <?php
            if (isset($_COOKIE['start'])) $duration = time() - $_COOKIE['start'];
            else $duration = "0";
            echo "$duration";
        ?>
        </H4><HR></center>
    </body>
</html>
```

**Abbildung 25.7:**
Das PHP-Script `cookie.php` (Abbildung 26.7 zeigt eine analoge Lösung mit einer Session)

- Das Script arbeitet mit zwei Cookies, „start“ für den Anfangszeitpunkt und „count“ für die Anzahl der Seitenaufrufe des Clients.
- Der Zugriff auf ein Cookie kann sowohl im HTTP-Header als auch im Dokument erfolgen; entsprechend finden wir in dem Beispielscript Cookie-Operationen sowohl innerhalb des Dokuments als auch vor dem Dokumentenanfang `<html>` – allerdings kann das Setzen eines Cookies nur vor dem Dokumentenanfang erfolgen.

- Der Zugriff auf ein Cookie erfolgt durch das Superglobal `$_COOKIE`; durch

```
$value = $_COOKIE['key']
```

  erhalten wir den Wert des Cookies mit dem Namen `key`. Damit überprüft das PHP-Script in Zeile 14, ob das Cookie „start“ schon gesetzt ist; entweder wird es neu gesetzt mit dem aktuellen Zeitpunkt, oder es wird bei Bedarf über das Superglobal ausgelesen.
- Ein Cookie wird in PHP durch die Methode `setcookie` gesetzt; diese Methode hat die vollständige Syntax

```
bool setcookie(string name
               [, string value
               [, int expire
               [, string path
               [, string domain
               [, bool secure]]]]] )
```

  Diese Parameter erklären sich direkt aus den Parametern von Cookies nach der Tabelle 25.1.

**Abbildung 25.8:** Ausführen von `cookie.php`

## 25.8 Cookies in den anderen Sprachen

Ähnliche Mechanismen für den Einsatz von Cookies bieten die anderen Sprachen der Web-Programmierung in ähnlicher Form, häufig in Verbindung mit speziellen Modulen:

- für Perl sind im wichtigen Modul `CGI::Cookie` (vgl. 10.6) die entsprechenden Funktionalitäten verfügbar; ein Cookie in Perl wird dann etwa durch

```
use CGI;

my $cgi = CGI->new();
my $cookie = $cgi->cookie(-name=>'cookie1',
                          -value=>'WebKompendium',);

print $cgi->header(-cookie => $cookie);
```

gesetzt; der Zugriff auf ein Cookie ist dann über

```
use CGI;

my $cgi = CGI->new();
my %cookie = $cgi->cookie('cookie1');

print $cgi->header();
for my $key(sort keys %cookie){
  print $key," -> ",$cookie{$key},"\n";
}
```

möglich;

- Python verfügt über das einschlägige Modul `Cookie`;
- Ruby verhält sich wieder ähnlich zu Perl: auch in Ruby werden Cookies über das Modul `cgi` ermöglicht.

Cookies können in allen Sprachen der Web-Programmierung eingesetzt werden – nur ist es in PHP vergleichsweise am einfachsten.

# 26

# Sessionmanagement

In Kapitel 25 wurde das Cookie vorgestellt: Die Möglichkeit, Informationen in Form von Key-Value-Paaren auf dem Webclient abzuspeichern. Einen wesentlichen Schritt weiter geht die *Session*. Sie dient ebenso dem personalifizierten Webangebot, nur werden hier die Daten nicht mehr auf dem Client, sondern auf dem Server abgelegt. Die Anwendungen der Session sind sehr umfassend.

## 26.1 Vom Cookie zur Session

Das Prinzip des Cookies wurde in Kapitel 25 vorgestellt. Probleme des Cookies waren insbesondere, dass die gespeicherte Information einfach verändert werden kann (Abbildung 25.4), also eine potentielle Gefährdung des Diensteanbieters, und dass Anwender dem Einsatz von Cookies nicht zustimmen könnten.

Die *Session* findet auf diese Probleme eine einfache Lösung: Hier greift der Client auf den Webserver zu, dieser, also der Server, speichert die Daten und übermittelt nur eine große Zufallszahl – die Session-ID – zum Client. Alles, was der Client speichert, ist diese Session-ID; alle Daten liegen sicher auf dem Server. Diese Speicherung auf dem Client erfolgt im einfachen Fall in einem Cookie, kann aber auch als GET-Parameter übertragen werden, falls Cookies nicht erlaubt sind.

Der Server legt im einfachsten Fall die Session-Informationen als Textdatei ab; es kann aber auch performanter eine Ablage in einem Datenbanksystem gewählt werden.

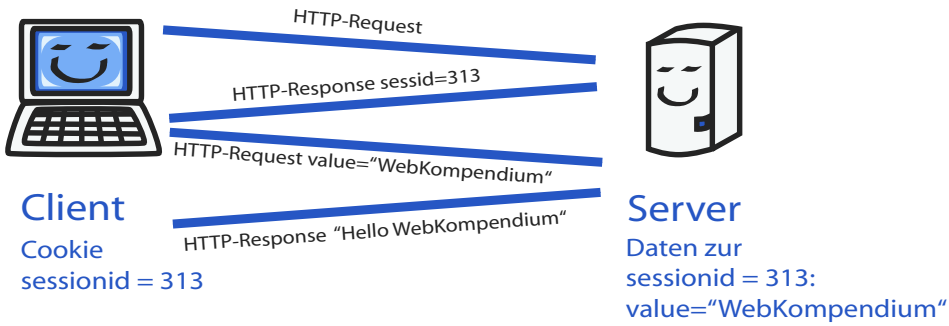



**Abbildung 26.1:**
Prinzip der Session

## 26.2 Sessionmanagement in PHP

Wie schon bei den Cookies bietet PHP auch bei den Sessions eine ausgesprochen einfache Art der Umsetzung an.

Die eigentlichen Daten zur jeweiligen Session liegen auf dem Webserver, typischerweise im Verzeichnis `/tmp`; dies ist in der php.ini-Datei konfigurierbar über die Direktive `session.save_path`. Die dort abgelegten Daten sind dann natürlich vor einer Manipulation durch den Client geschützt.

Die PHP-Session muss, gleich dem Cookie, vor der ersten Ausgabe angelegt werden; nach dem Dokumentenanfang ist dies nicht mehr möglich. Eine Session wird dabei in PHP begonnen durch

```
session_start()
```

Die zu einer Session gehörenden Daten werden wieder in einem Superglobal verwaltet, konkret in:

```
$_SESSION['key']
```

Dies registriert den Schlüssel `key` zur aktuellen Session, entsprechend kann das Superglobal verwendet werden.

Ältere PHP-Versionen verwendeten hier `$HTTP_SESSION_VARS` und mussten die Variablen einzeln mittels `session_register(...)` zur Session hinzufügen; mit dem Superglobal entfällt dies. Wie bei den cgi-Umgebungsvariablen kann über den Schalter `register_globals` auch ein abweichender Weg gewählt werden, was aus Sicherheitsgründen heute aber nicht mehr relevant ist.

Mittels

```
session_destroy()
```

wird weitergehend die ganze Session beendet.

Das PHP-Script `session1.php` (Abbildung 26.2) startet eine Session und registriert zwei Variablen; diesen wird dann ein Wert zugewiesen. Das Script endet, ohne die Session zu beenden. Als Resultat wird auf dem Client ein Cookie `PHPSESSID` gesetzt (Abbildung 26.3). Anschließend greift das Script `session2.php` vom gleichen Browser auf den Server zu (Abbildung 26.4); da dieser Browser über das Cookie die Session-ID abgespeichert hat, verwendet der zweite Aufruf die gleiche Session und kann somit als Ergebnis die Werte der Session-Variablen des ersten Aufrufs nutzen (Abbildung 26.4).

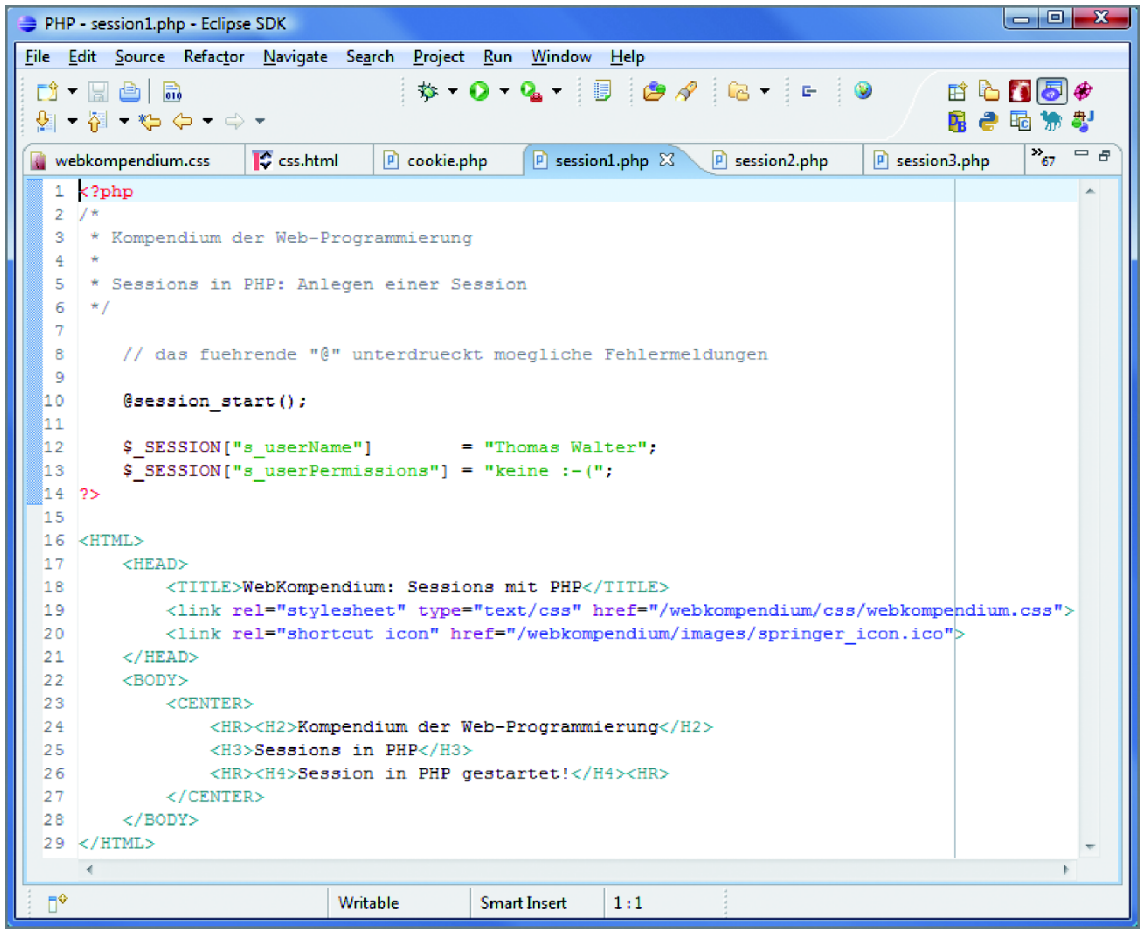



**Abbildung 26.2:**
Starten der Session und registrieren von zwei Variablen im Script `session1.php`

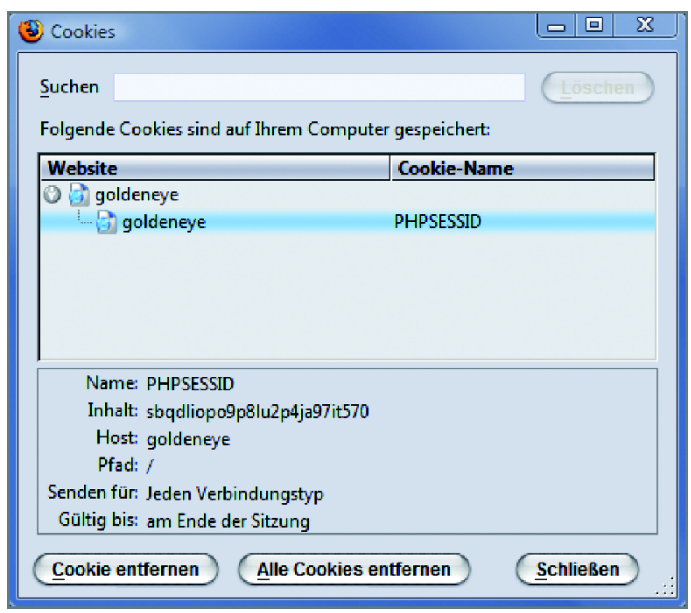



**Abbildung 26.3:**
Nach Ausführen von Script `session1.php`: Cookie `PHPSESSID` ist gesetzt

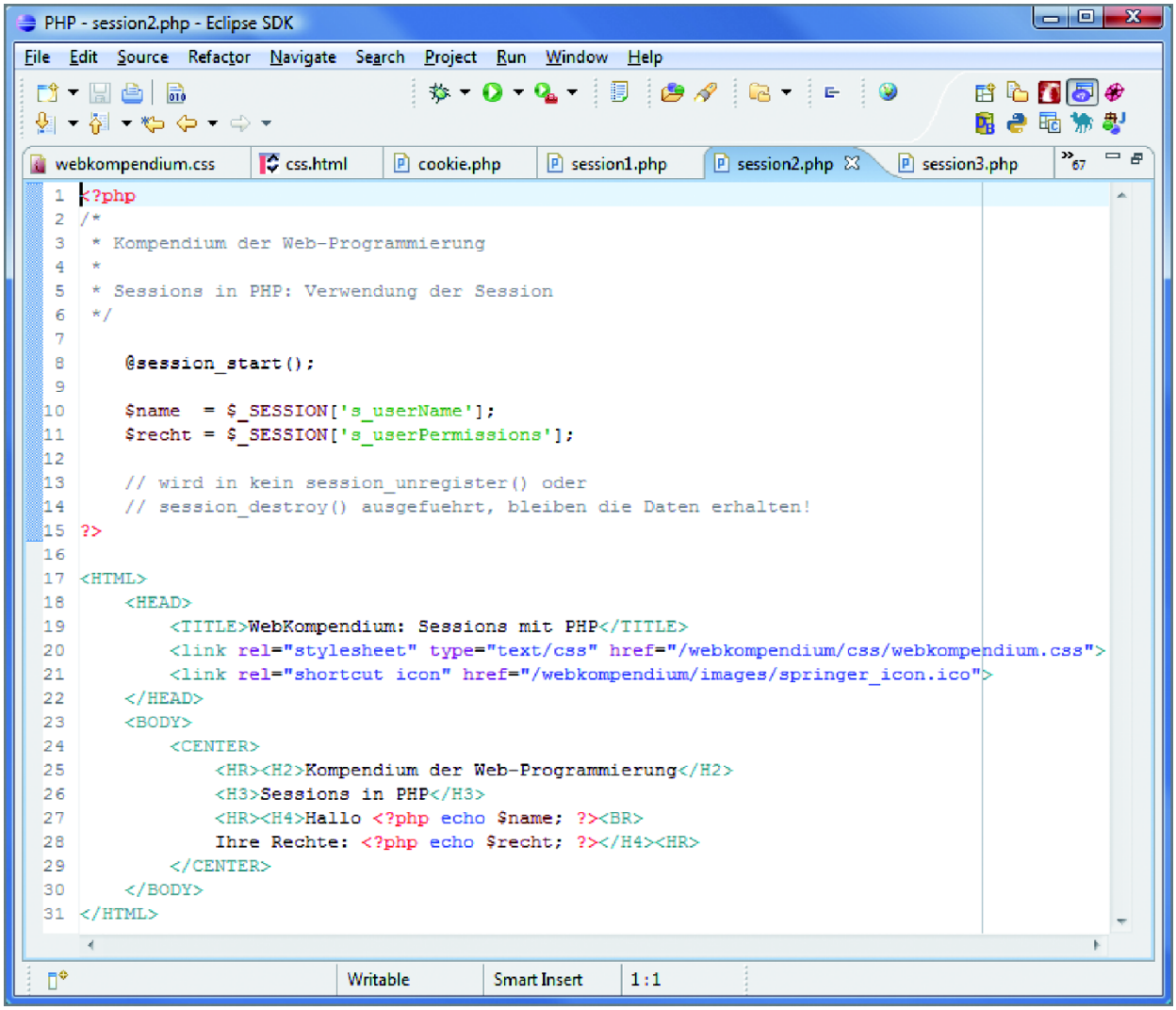



**Abbildung 26.4:** Zugriff auf die Session-Variablen (Script `session2.php`)

Ein zweites Beispiel soll analog zum Cookie-Script `cookie.php` (Abbildung 25.7) die Zugriffe vom jeweiligen Client zählen und die Dauer der Session angeben; dies wird nun in `session3.php` mittels einer Session, die die Variablen `start` und `count` zur Session hinzufügt, geleistet (Abbildungen 26.7 und 26.6).

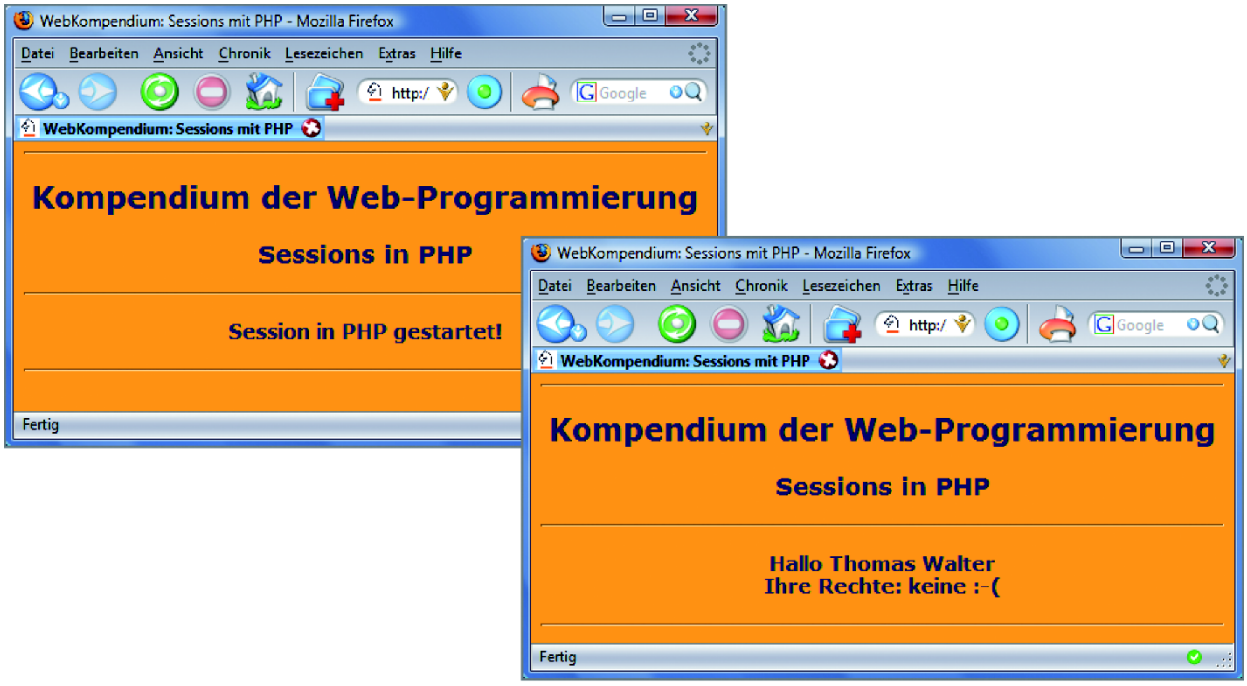



**Abbildung 26.5:**
Ausführen von `session1.php` und anschließend `session2.php`

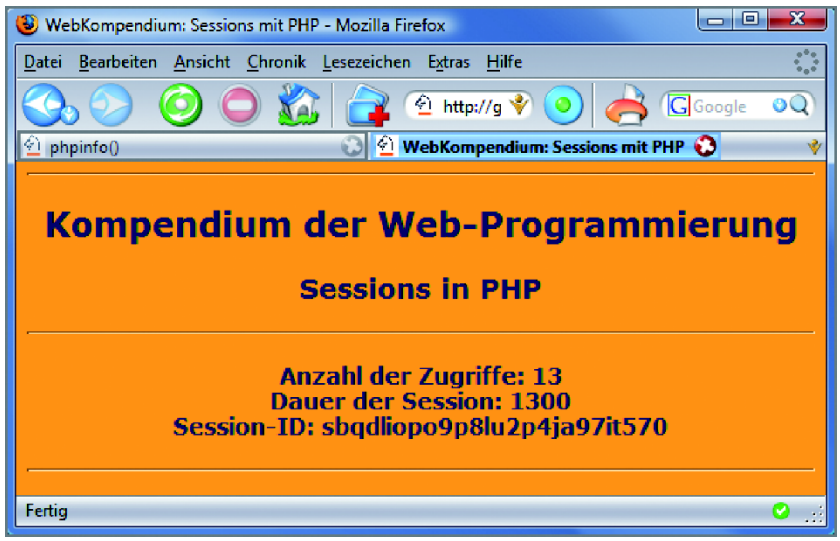



**Abbildung 26.6:**
Ausführen von `session3.php`

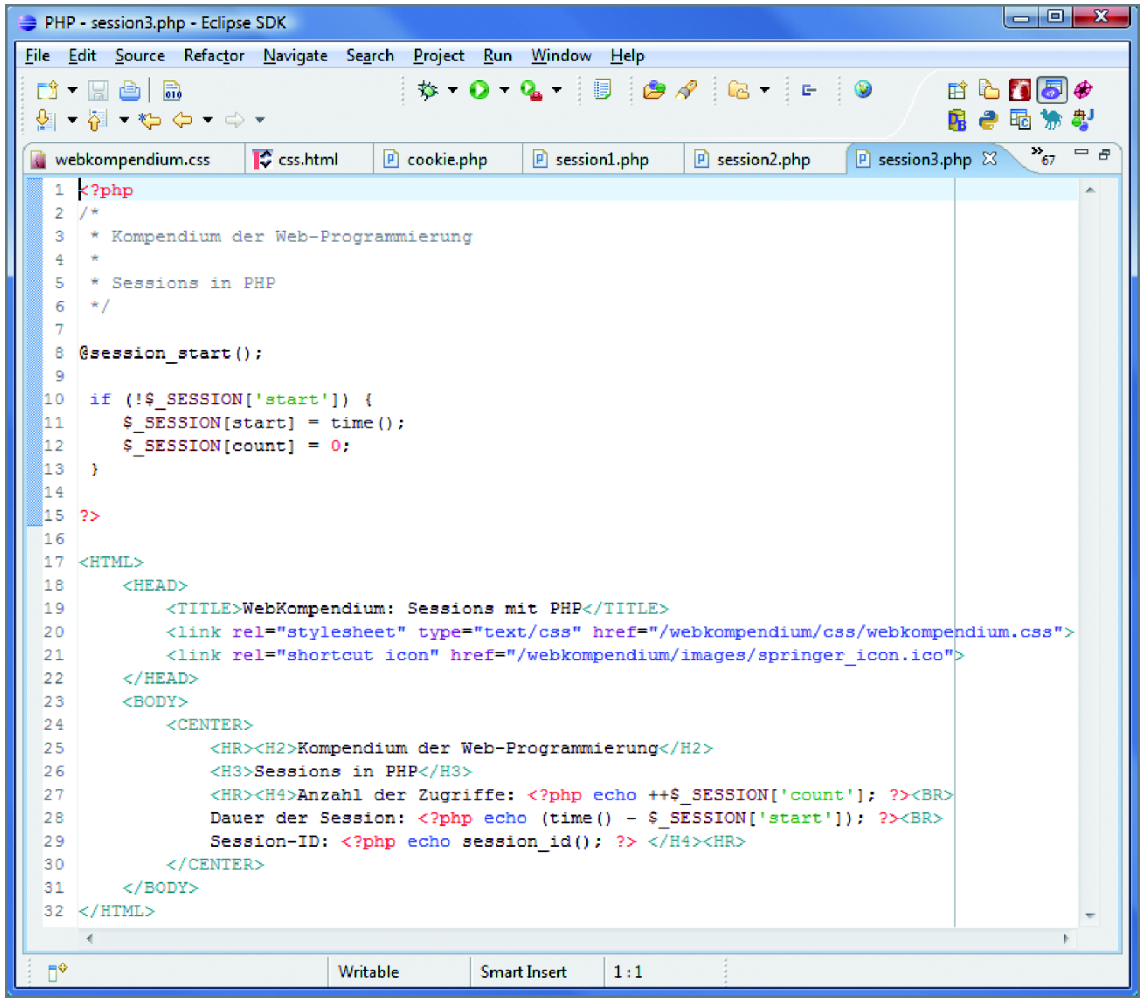



**Abbildung 26.7:**
Session mit zwei Variablen: Startzeit und Anzahl der Aufrufe (Script `session3.php` analog zu Abbildung 25.7)

### 26.2.1 PHP-Sessions ohne Cookies

Die vorgestellten Beispiele verwenden jeweils Cookies zur persistenten Speicherung der Session-ID; es geht aber prinzipiell auch ohne.

Zwei Möglichkeiten bestehen, um Sessions ohne Cookies verwenden zu können:

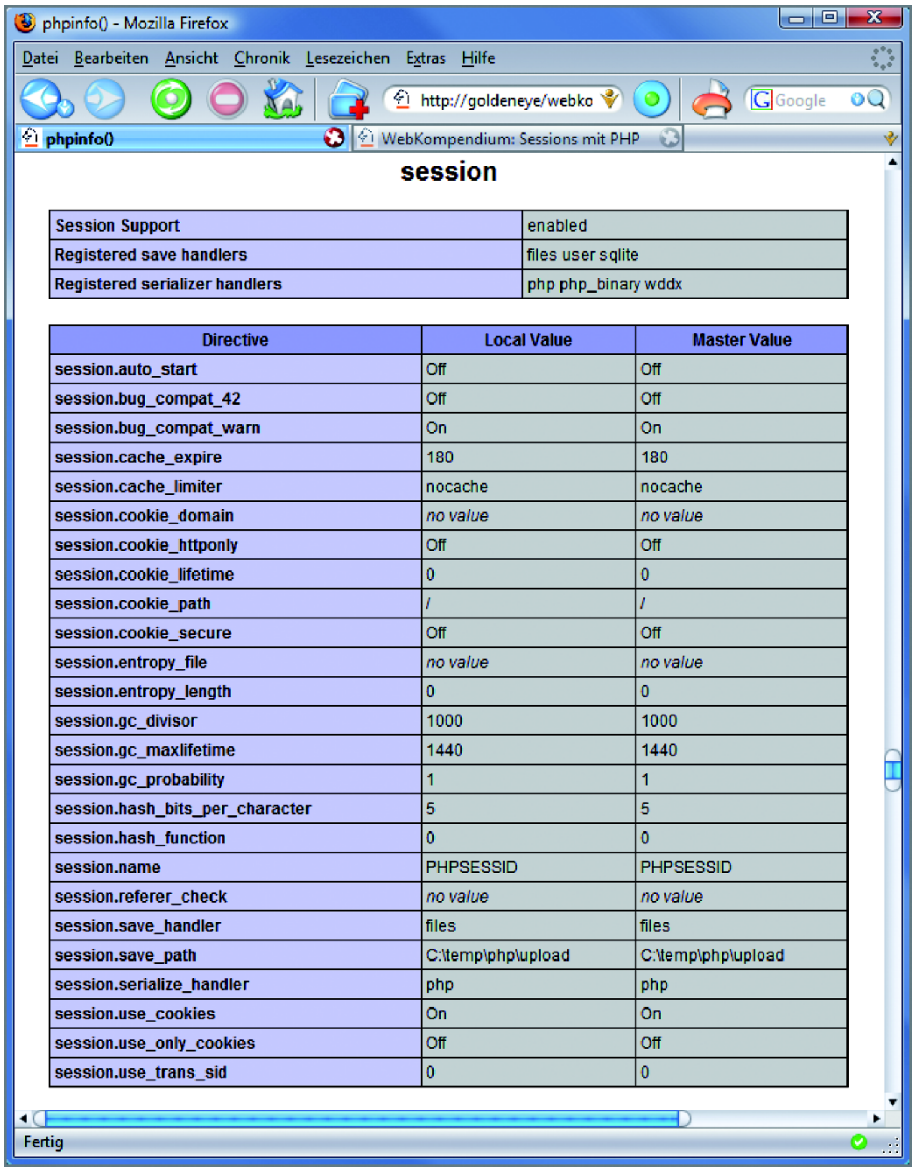
- die Codierung der Session-ID über die URL; dies geschieht in PHP mittels der Methode `Session->url()`;
- die Übermittlung der Session-ID als verstecktes Formular-Feld (hidden field); dies geschieht in PHP mittels der Methode `Session->hidden()`.

### 26.2.2 PHP-Konfigurationen für Sessions

Das Session-Management von PHP kann sehr umfassend über die Konfigurationsdatei php.ini gesteuert werden. Die bekannte Methode `phpinfo()` gibt eine ausführliche Übersicht über die aktuelle Konfiguration.

**Unter Windows, insbesondere Vista, können die Pfade zur Ablage der Session-Informationen in der Datei `php.ini` Schwierigkeiten bereiten und müssen angepasst werden, insbesondere `session.save_path`.**

Abbildung 26.8 zeigt den entsprechenden Ausschnitt zur Sessionmanagement-Konfiguration aus `phpinfo()`.




session

| | |
|---|---|
| Session Support | enabled |
| Registered save handlers | files user sqlite |
| Registered serializer handlers | php php_binary wddx |

| Directive | Local Value | Master Value |
|---|---|---|
| session.auto_start | Off | Off |
| session.bug_compat_42 | Off | Off |
| session.bug_compat_warn | On | On |
| session.cache_expire | 180 | 180 |
| session.cache_limiter | nocache | nocache |
| session.cookie_domain | *no value* | *no value* |
| session.cookie_httponly | Off | Off |
| session.cookie_lifetime | 0 | 0 |
| session.cookie_path | / | / |
| session.cookie_secure | Off | Off |
| session.entropy_file | *no value* | *no value* |
| session.entropy_length | 0 | 0 |
| session.gc_divisor | 1000 | 1000 |
| session.gc_maxlifetime | 1440 | 1440 |
| session.gc_probability | 1 | 1 |
| session.hash_bits_per_character | 5 | 5 |
| session.hash_function | 0 | 0 |
| session.name | PHPSESSID | PHPSESSID |
| session.referer_check | *no value* | *no value* |
| session.save_handler | files | files |
| session.save_path | C:\temp\php\upload | C:\temp\php\upload |
| session.serialize_handler | php | php |
| session.use_cookies | On | On |
| session.use_only_cookies | Off | Off |
| session.use_trans_sid | 0 | 0 |

**Abbildung 26.8:** Konfiguration des Session-Managements

### 26.2.3 Beispiel für den Einsatz von Sessions

Ein fortgeschritteneres Beispiel soll den Einsatz von Sessions zeigen. In Kapitel 7 wird das Beispiel der Literaturdaten in zwei Tabellen vorgestellt, wobei zu jedem Buch beliebig viele Autoren gehören können. Bisher haben wir diese Daten ausgelesen, nun sollen mittels Sessions auch Einträge in diese Tabellen erfolgen.

Dazu fragt ein erstes Formular die Grunddaten (Titel, Verlag, Jahr und zwingender Erstautor) ab; anschließend können beliebig viele weitere Autoren angegeben werden. Alle diese Daten werden serverseitig in Session gespeichert und erst im letzten Schritt in die Datenbank geschrieben.

Diese Anwendung ist im Folgenden auf mehrere Scripte aufgeteilt:

- `s_eintrag.html` ist das Ausgangsformular für die Abfrage der notwendigen Werte (Abbildung 26.9);
- `s_eintrag_a.php` (Abbildung 26.10) übernimmt die Eingabewerte des Ausgangsformulars und schreibt diese in eine Session; man kann verzweigen zur Eingabe weiterer Autoren (`s_eintrag_b.php`) oder den Eintrag in die Datenbank vornehmen (`s_eintrag_c.php`);
- `s_eintrag_b.php` (Abbildung 26.11) übernimmt die Angaben des neuen Autors und fügte diese zur Session hinzu; danach werden alle Daten der Session angezeigt, man kann wieder verzweigen zur Eingabe weiterer Autoren (`s_eintrag_b.php`) oder den Eintrag in die Datenbank vornehmen (`s_eintrag_c.php`);
- `s_eintrag_c.php` (Abbildungen 26.12 und 26.13) schreibt alle Werte der Session in die Datenbank.

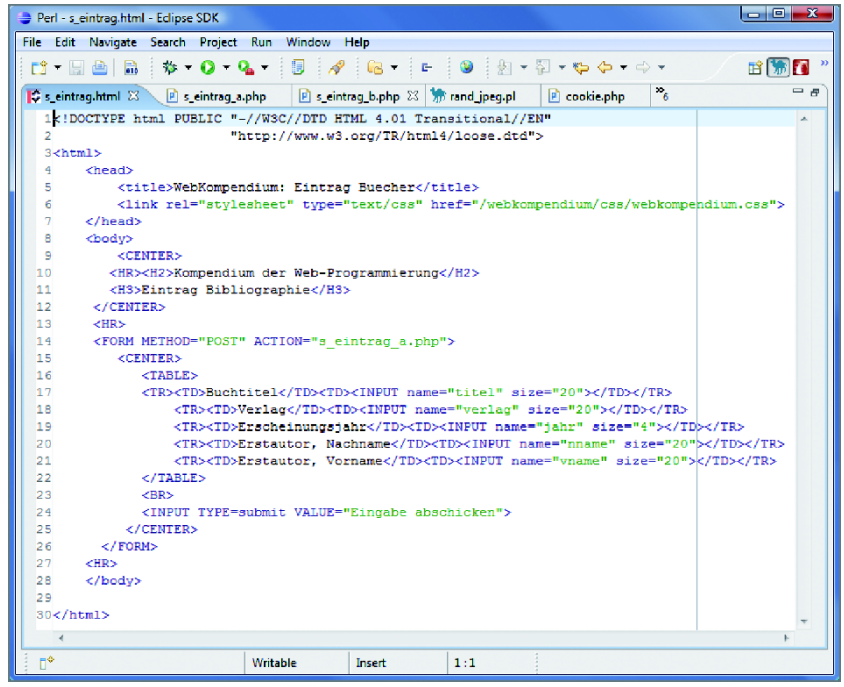



**Abbildung 26.9:** HTML-Formular für die Eingabe der notwendigen Daten (`s_eintrag.html`)

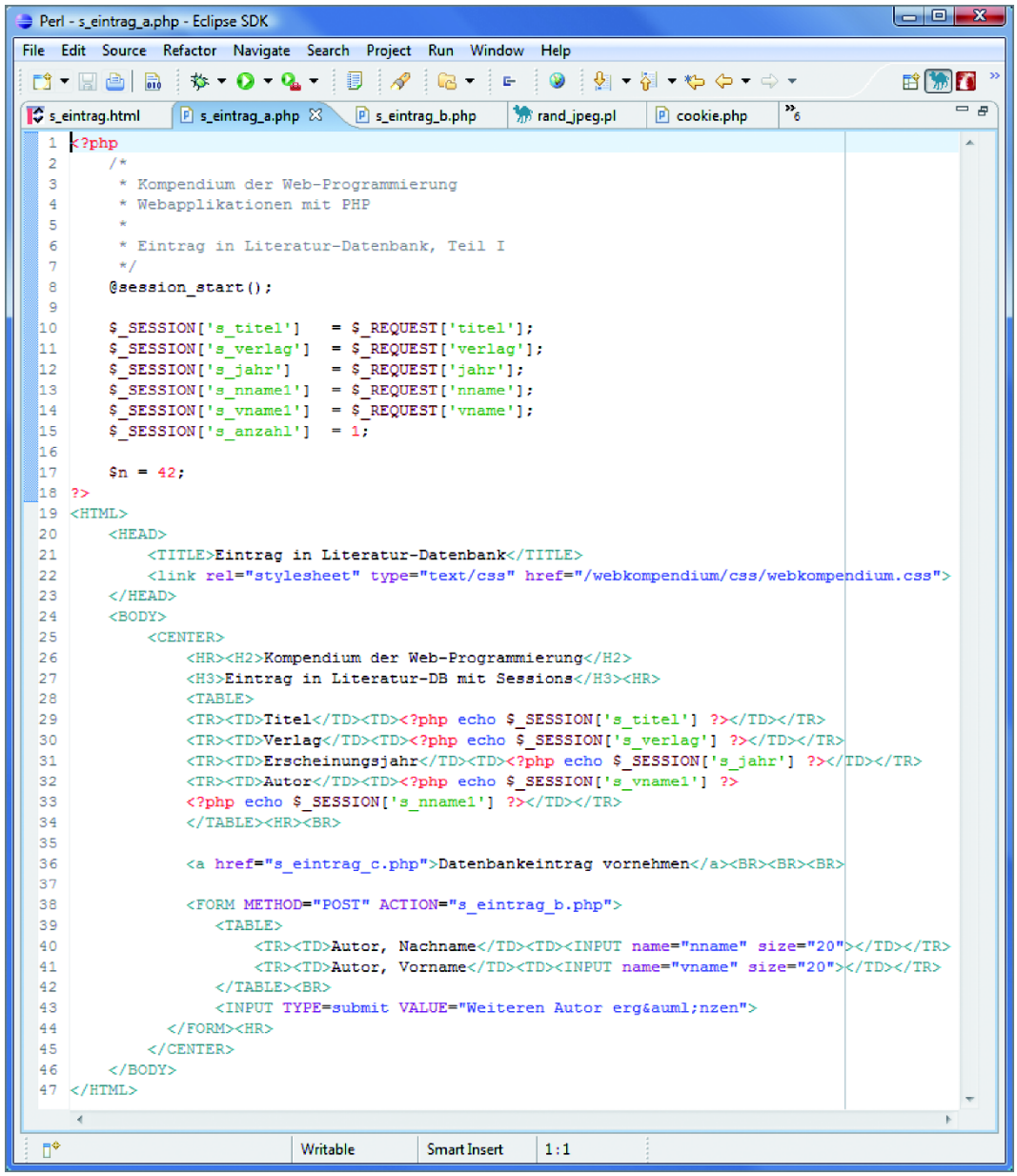
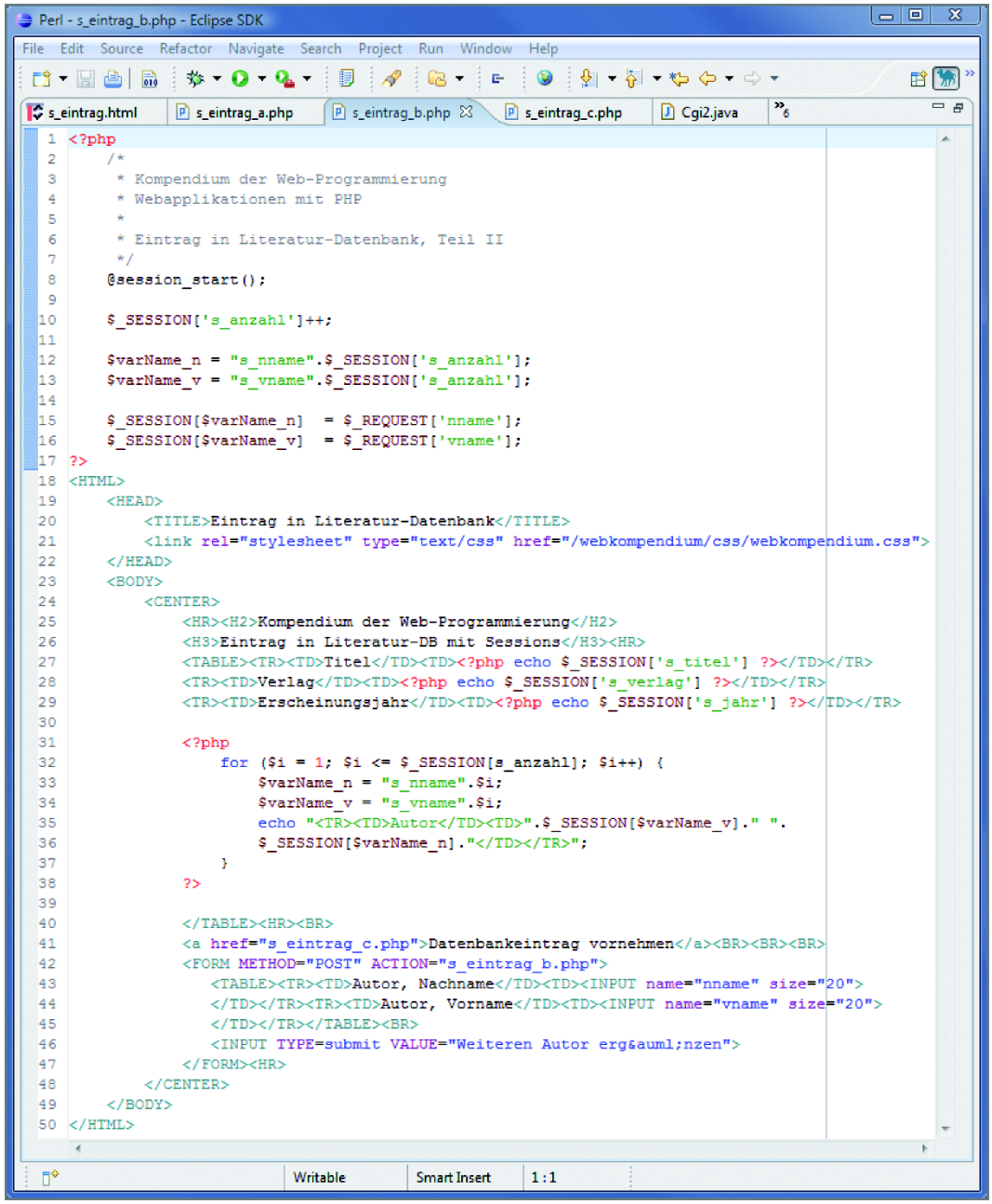



**Abbildung 26.10:**
PHP-Script (`s_eintrag_a.php`)




```
<?php
    /*
     * Kompendium der Web-Programmierung
     * Webapplikationen mit PHP
     *
     * Eintrag in Literatur-Datenbank, Teil II
     */
    @session_start();

    $_SESSION['s_anzahl']++;

    $varName_n = "s_nname".$_SESSION['s_anzahl'];
    $varName_v = "s_vname".$_SESSION['s_anzahl'];

    $_SESSION[$varName_n]  = $_REQUEST['nname'];
    $_SESSION[$varName_v]  = $_REQUEST['vname'];
?>
<HTML>
    <HEAD>
        <TITLE>Eintrag in Literatur-Datenbank</TITLE>
        <link rel="stylesheet" type="text/css" href="/webkompendium/css/webkompendium.css">
    </HEAD>
    <BODY>
        <CENTER>
            <HR><H2>Kompendium der Web-Programmierung</H2>
            <H3>Eintrag in Literatur-DB mit Sessions</H3><HR>
            <TABLE><TR><TD>Titel</TD><TD><?php echo $_SESSION['s_titel'] ?></TD></TR>
            <TR><TD>Verlag</TD><TD><?php echo $_SESSION['s_verlag'] ?></TD></TR>
            <TR><TD>Erscheinungsjahr</TD><TD><?php echo $_SESSION['s_jahr'] ?></TD></TR>

            <?php
                for ($i = 1; $i <= $_SESSION[s_anzahl]; $i++) {
                    $varName_n = "s_nname".$i;
                    $varName_v = "s_vname".$i;
                    echo "<TR><TD>Autor</TD><TD>".$_SESSION[$varName_v]." ".
                    $_SESSION[$varName_n]."</TD></TR>";
                }
            ?>

            </TABLE><HR><BR>
            <a href="s_eintrag_c.php">Datenbankeintrag vornehmen</a><BR><BR><BR>
            <FORM METHOD="POST" ACTION="s_eintrag_b.php">
               <TABLE><TR><TD>Autor, Nachname</TD><TD><INPUT name="nname" size="20">
               </TD></TR><TR><TD>Autor, Vorname</TD><TD><INPUT name="vname" size="20">
               </TD></TR></TABLE><BR>
               <INPUT TYPE=submit VALUE="Weiteren Autor erg&auml;nzen">
            </FORM><HR>
        </CENTER>
    </BODY>
</HTML>
```

**Abbildung 26.11:**
PHP-Script (`s_eintrag_b.php`)

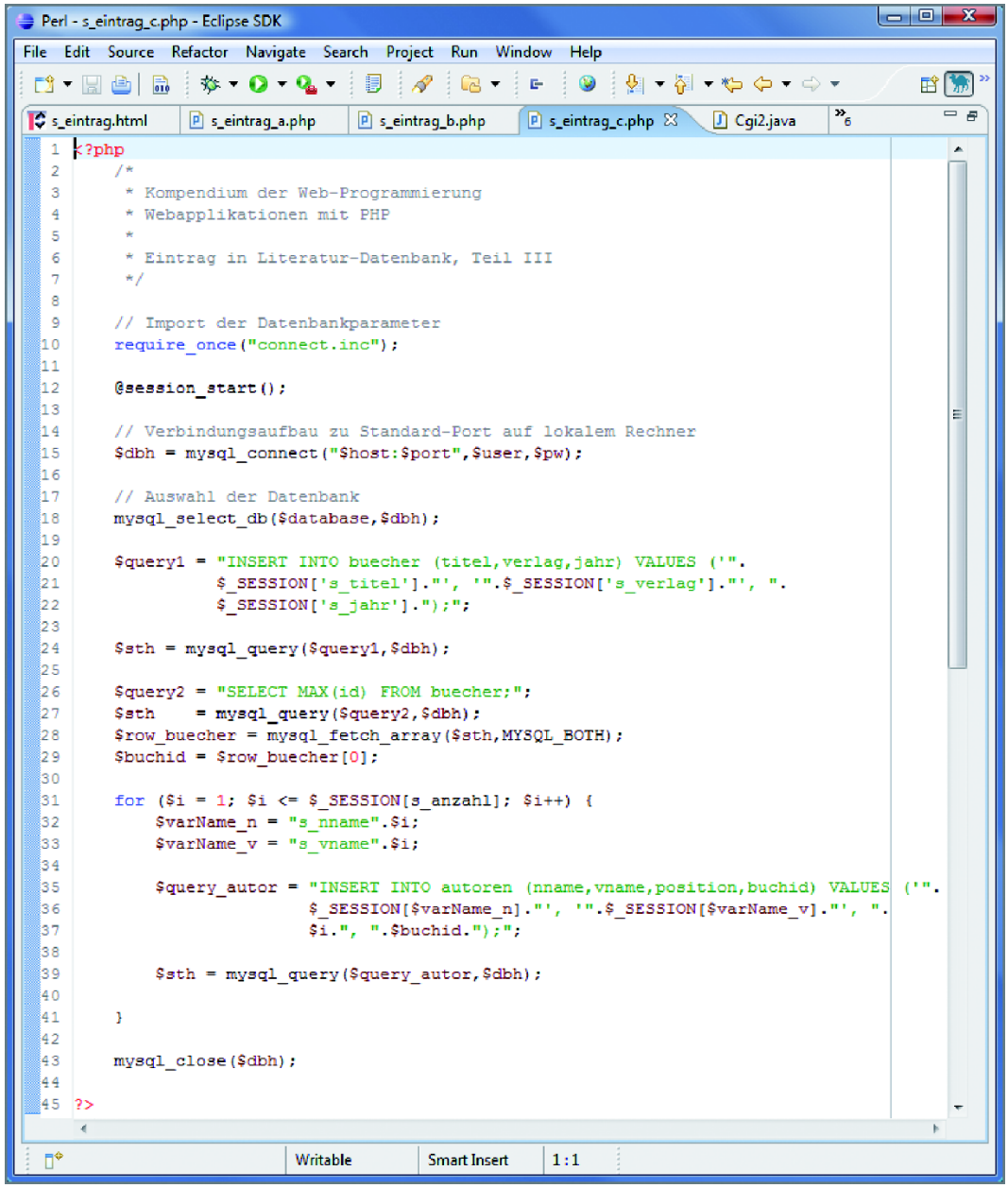
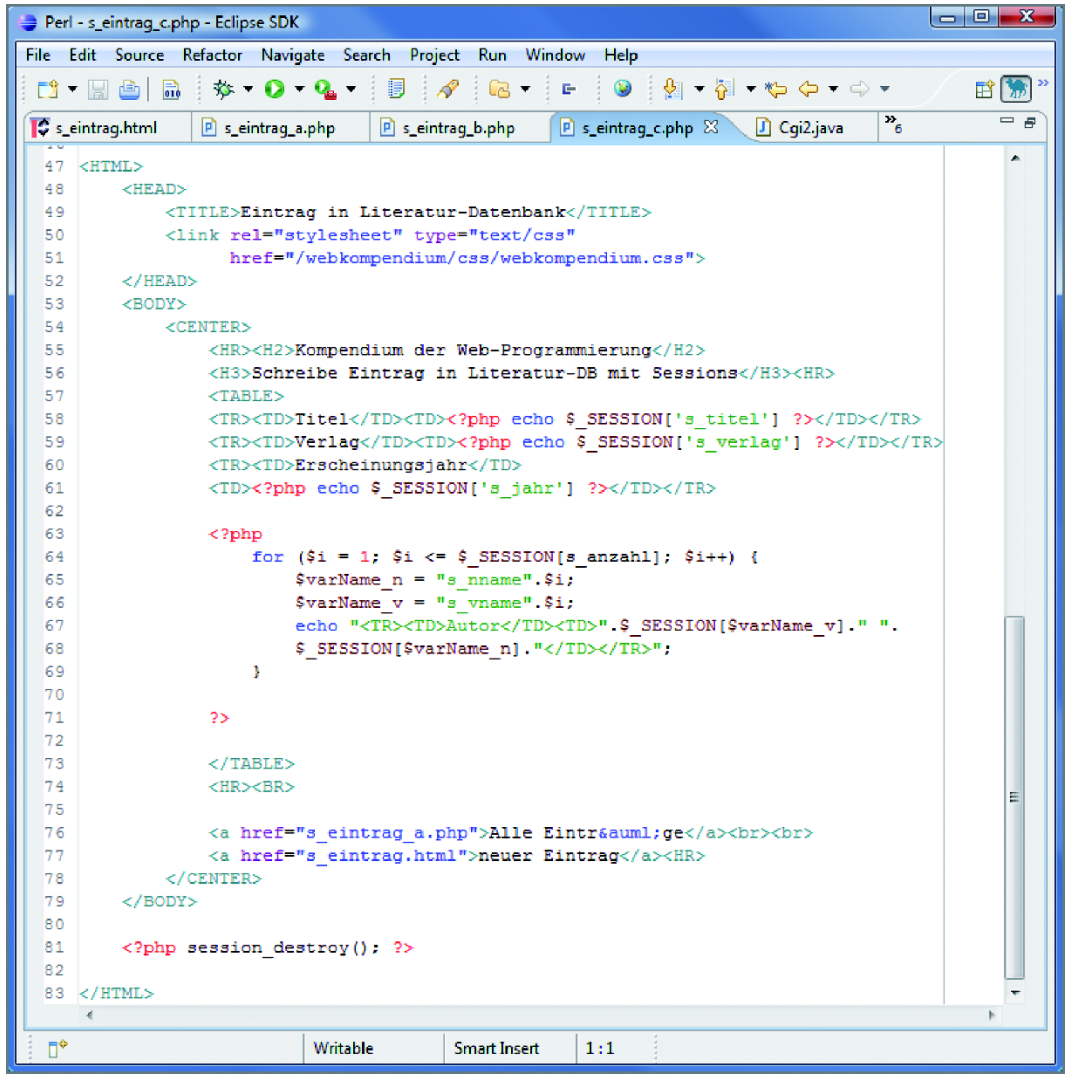



**Abbildung 26.12:**
PHP-Script (`s_eintrag_c.php`), Teil I




```
<HTML>
    <HEAD>
        <TITLE>Eintrag in Literatur-Datenbank</TITLE>
        <link rel="stylesheet" type="text/css"
              href="/webkompendium/css/webkompendium.css">
    </HEAD>
    <BODY>
        <CENTER>
            <HR><H2>Kompendium der Web-Programmierung</H2>
            <H3>Schreibe Eintrag in Literatur-DB mit Sessions</H3><HR>
            <TABLE>
            <TR><TD>Titel</TD><TD><?php echo $_SESSION['s_titel'] ?></TD></TR>
            <TR><TD>Verlag</TD><TD><?php echo $_SESSION['s_verlag'] ?></TD></TR>
            <TR><TD>Erscheinungsjahr</TD>
            <TD><?php echo $_SESSION['s_jahr'] ?></TD></TR>

            <?php
                for ($i = 1; $i <= $_SESSION[s_anzahl]; $i++) {
                    $varName_n = "s_nname".$i;
                    $varName_v = "s_vname".$i;
                    echo "<TR><TD>Autor</TD><TD>".$_SESSION[$varName_v]." ".
                    $_SESSION[$varName_n]."</TD></TR>";
                }

            ?>

            </TABLE>
            <HR><BR>

            <a href="s_eintrag_a.php">Alle Eintr&auml;ge</a><br><br>
            <a href="s_eintrag.html">neuer Eintrag</a><HR>
        </CENTER>
    </BODY>

    <?php session_destroy(); ?>

</HTML>
```

**Abbildung 26.13:** PHP-Script (`s_eintrag_c.php`), Teil II

Nun können Datenbankeinträge nach dem Datenmodell von Kapitel 7 mit beliebig vielen Autoren vorgenommen werden; Abbildung 26.14 zeigt die Neuanlage mit zwei Autoren.

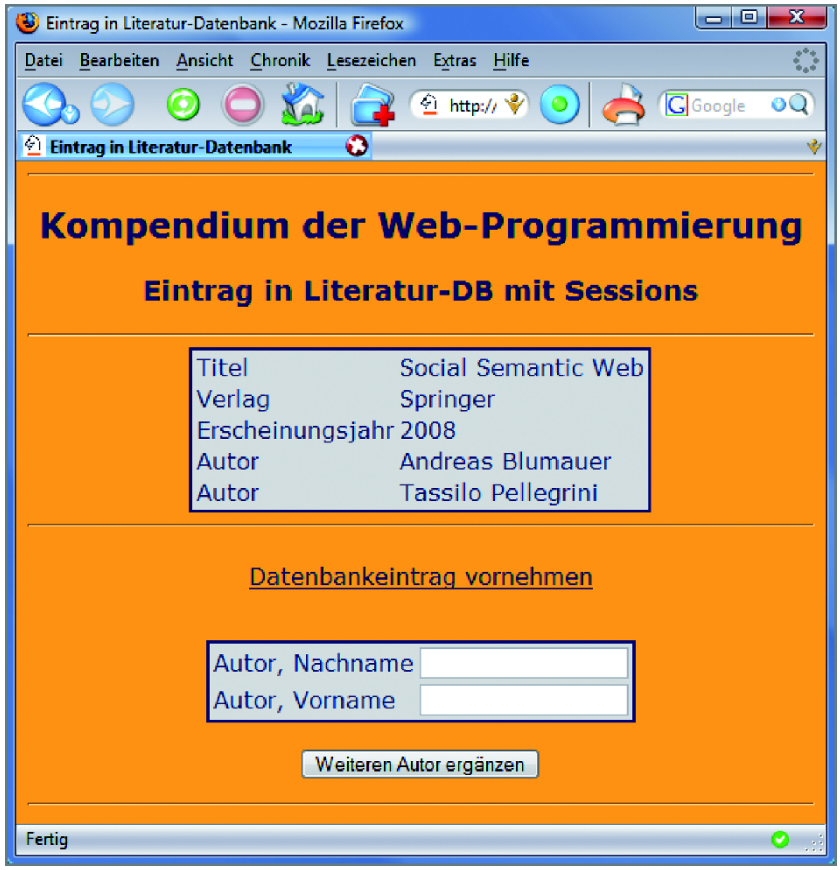



**Abbildung 26.14:**
Neuanlage eines Datenbakeintrags mit zwei Autoren

### 26.2.4 Sessions mit anderen Sprachen

Ganz analog zur vorgestellten Umsetzung mit PHP können auch die anderen Sprachen der Web-Programmierung das Session-Management umsetzen. Wie bei den Cookies (Abschnitt 25.8) benötigen sie aber wieder entsprechende Zusatzmodule.

# 27

# Media-Formate

Eine der wesentlichen Neuerungen des World Wide Web war die Unterstützung von Medienformaten, in der ersten Stufe insbesondere von Grafiken.

Heute nehmen die verschiedenen Grafikformate eine zentrale Rolle bei der Gestaltung von Web-Sites ein; dieses Kapitel setzt sich damit kurz auseinander.

## 27.1 Der MIME-Typ

MIME, „Multipurpose Internet Mail Extensions“ ist eine Kennzeichung der Medienart, die ursprünglich für E-Mails gedacht war. Inzwischen wird dieses Verfahren im Web sehr vielfältig eingesetzt:

- Die erste „Antwort“ eines CGI-Scriptes (vgl. Kapitel 9) ist der MIME-Typ, also die Angabe, welche Medienart folgt.
- Einzelne HTML-Elemente haben einen MIME-Typ, etwa das in Kapitel 15 vorgestellte `<script>`-Tag.

Der Aufbau der MIME-Typen erfolgt nach dem Schema

```
haupttyp/untertyp
```

wie etwa `text/html`, `text/plain` oder `image/gif`.

Im Header der HTTP-Kommunikation wird der MIME-Typ mit angegeben; Abbildung 1.10 zeigt einen solchen Fall. Der Browser entscheidet dann anhand des übermittelten MIME-Typs, was er mit der Antwort des Servers machen soll: Anzeigen, Speichern oder andere Optionen stehen damit offen.

## 27.2 Die verschiedenen MIME-Typen

Inzwischen liegen weit über hundert MIME-Typen vor; im Internet [web] gibt es eine vollständige Auflistung, Tabelle 27.1 gibt einige wichtige Typen an.

**Tabelle 27.1:** Einige MIME-Typen

| MIME-Typ | Dateiendung | Bedeutung |
|---|---|---|
| application/pdf | pdf | Adobe pdf |
| application/xml | xml | XML |
| application/zip | zip | zip-Archiv |
| audio/x-mpeg | mp2 | MPEG-Datei |
| image/jpeg | jpeg/jpg | jpeg-Bild |
| image/png | png | png-Bild |
| text/css | css | CSS-Datei |
| text/html | html/htm | HTML-Dokument |
| text/javascript | js | JavaScript |
| text/plain | txt | reiner Text |

## 27.3 Grafik-Formate: Bilddateien im Web

Vorab eine zentrale Bemerkung: Das Web ist, was die technische Bildqualität angeht, denkbar schlecht. Dafür gibt es mehrere Gründe, die in den folgenden Abschnitten deutlich werden.

Die Konsequenz hieraus ist, dass mit zwei Versionen eines Bildes gearbeitet werden muss: mit der hochwertigen Ausgangsversion in hoher Auflösung und einem speziellen Farbraum und einer für das Web optimierten, also effektiv verschlechterten Version.

Die Kriterien an eine entsprechende Webvariante werden im Folgenden erläutert.

### 27.3.1 Das Bild auf der Web-Site

Das Einbinden von Bilddateien in eine Web-Site geschieht in HTML mittels des Image-Tags. Diese zentrale Anweisung hat folgende Grundsyntax:

```
<IMG src="bild.jpg" alt="Beispielbild">
```

Durch diese Anweisung wird die Bilddatei `bild.jpg`, welche im gleichen Verzeichnis wie die eigentliche HTML-Datei abgespeichert sein muss, in ihrer Originalgröße in die Seite eingebunden.

| Attribut | mögliche Werte | Beschreibung |
|---|---|---|
| `align` | top, middle, bottom, left, center | Ausrichtung der Grafik |
| `alt` | | Alternativtext, wenn Bilddatei nicht angezeigt werden kann |
| `border` | Dicke in Pixel oder Prozent | Rahmen um das Bild |
| `height` | Höhe in Pixel oder Prozent | Höhe des Bildes |
| `hspace` | Abstand in Pixel oder Prozent | horizontaler Abstand zwischen Bild und umgebendem Text |
| `ismap` | | serverseitiges Handling für bildbereichsensitive Links |
| `longdesc` | | verlinkte Langbeschreibung des Bildes |
| `name` | | Name des Bildes |
| `src` | | Dateiname |
| `usemap` | | Verlinkung auf ein zugeordnetes map-Element |
| `vspace` | Abstand in Pixel oder Prozent | vertikaler Abstand zwischen Bild und umgebendem Text |
| `width` | Breite in Pixel oder Prozent | Breite des Bildes |

**Tabelle 27.2:** Attribute zum `<IMG>`-Tag

Natürlich hat diese HTML-Anweisung zahlreiche wichtige Attribute, welche die Anzeige der Bilddatei grundlegend steuern können. Das angegebene Beispiel verwendet schon zwei dieser Attribute, `src` und `alt`. Insgesamt stehen die Attribute aus Tabelle 27.2 zur Verfügung. Nur die Attribute `src` und `alt` sind dabei zwingend, alle anderen sind optional.

Häufig zur Anwendung kommen die Attribute zur Vorgabe der Höhe und der Breite. Dabei ist die Skalierung des Bildes zu beachten: Wird nur ein Attribut, also entweder Höhe oder Breite, vorgegeben, so wird das ganze Bild proportional verändert, es bleibt also das Seitenverhältnis unverändert. Werden beide Attribute gesetzt, kann es zu einer beliebigen Veränderung dieses Verhältnisses kommen.

Moderne Webeditoren wie Adobes GoLive CS oder Macromedias Dreamweaver (vgl. 8.2) stellen praktische Dialoge zum Bearbeiten der Attribute des `<IMG>`-Tags bereit (Abbildungen 27.1 und 27.2).

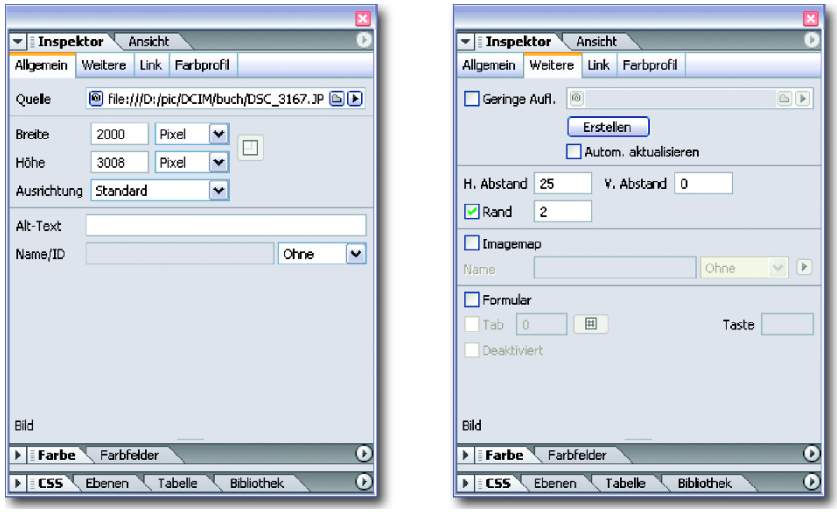



**Abbildung 27.1:**
Bearbeiten der Attribute des `<IMG>`-Tags mit GoLive

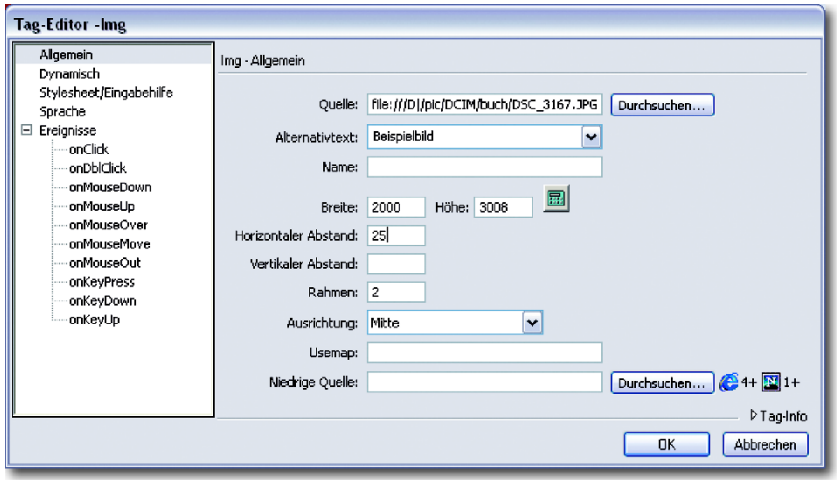



**Abbildung 27.2:**
Bearbeiten der Attribute des `<IMG>`-Tags mit Dreamweaver

### 27.3.2 Auflösung im Web

Die Frage der *Auflösung,* die eine Bilddatei für die Präsentation im Web haben sollte, ist recht einfach zu beantworten: Es genügt die Anzahl von Bildpunkten,

die vom Browser angezeigt werden.

Für die Entwicklung einer guten Web-Site ist von Anfang an die angestrebte Monitorauflösung von wesentlicher Bedeutung. Heute werden Seiten für eine Auflösung von 800 × 600 oder 1.024 × 768 Bildpunkten entwickelt (wer eine höhere Auflösung verwendet, wird kaum seinen Browser in einem Vollbildmodus betreiben). Geht man somit von einer maximalen Anzahl von 1.024 horizontalen Pixeln aus und verwendet ein Bild, welches ein Drittel des Monitors füllen soll, so ist eine Bildbreite von 300 Pixeln ausreichend – also in praktisch allen Fällen eine deutliche Verkleinerung des Bildes.

Mehr hierzu ist in [Wal04] zu finden.

### 27.3.3 Farben im Web: Indizierte Farben

Web-Anwendungen verwenden häufig keinen echten RGB-Farbraum, sondern beschränken sich auf *indizierte Farben*. Hierbei wird ein Index, eine Referenztabelle von 256 Farben, verwendet, auf welche die Bilddarstellung – und Übermittlung im Internet – reduziert wird.

Bei diesem Verfahren sind zwei prinzipiell unterschiedliche Vorgehensweisen möglich: die Verwendung einer unabhängigen, möglichst großen, also einen großen Farbbereich abdeckenden Farbtabelle oder die Erstellung einer individuellen Tabelle für das Ausgangsbild, so dass möglichst viele der Ausgangsfarben wiedergegeben werden. Das erste Vorgehen führt zu einer standardisierten Farbtabelle mit 216 Farben, siehe Abbildung 27.3.

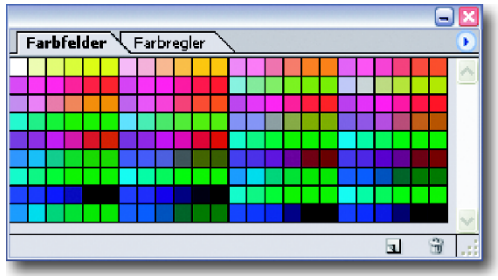



**Abbildung 27.3:** Die 216 Webfarben

Diese 216 von Netscape eingeführten websicheren Farben sind genau diejenigen, welche auf den Betriebssystemen Windows und Mac OS gleich sind, weshalb sie für die Gestaltung von Web-Sites besonders wichtig sind.

Eine besondere Untermenge dieser websicheren Farben bildet die VGA-Farbpalette, deren Farben innerhalb von HTML über ihren Namen angesprochen werden können. Diese Farben sind einheitlich für alle VGA-Grafikkarten und -Bildschirme.

**Tabelle 27.3:** VGA-Farben

| Farbname | Farbwert | Farbname | Farbwert |
|---|---|---|---|
| black | #000000 | gray | #808080 |
| maroon | #800000 | red | #FF0000 |
| green | #008000 | lime | #00FF00 |
| olive | #808000 | yellow | #FFFF00 |
| navy | #000080 | blue | #0000FF |
| purple | #800080 | fuchsia | #FF00FF |
| teal | #008080 | aqua | #00FFFF |
| silver | #C0C0C0 | white | #FFFFFF |

Netscape hat 120 weitere Farben über Namen definiert, die die 216 websicheren Farben ergänzen und von den meisten Browsern über ihren Namen korrekt interpretiert werden.

#### 27.3.3.1 Die hexadezimale Farbdarstellung

Für den Einsatz im Internet ist eine *hexadezimale Farbcodierung* sehr gebräuchlich; viele HTML-Attribute verwenden diese Farbdarstellung. Das Vorgehen hierbei ist einfach: Ausgehend von den drei RGB-Werten wird jeder einzelne Farbwert auf einer Skala von 0 bis 255 codiert, also mit einer Auflösung von einem Byte oder acht bit – oder genau zwei Hexziffern. Somit wird der Farbwert durch $2 \times 3 = 8$ Hexziffern dargestellt. Die Nomenklatur sieht vor, dass diese Werte mit einem führenden # dargestellt werden.

So entspricht etwa `#000000` reinem Schwarz, `#FF0000` ist Rot, `#00FF00` Grün und `#0000FF` Blau.

Es gibt zahlreiche gute Werkzeuge, um den Hexwert einer Farbe einfach zu bestimmen, etwa der Farbwähler von Photoshop.

### 27.3.4 Dateiformate für das Web

Im Web sind heute drei *Dateiformate* gebräuchlich:

- gif: Das Graphics Interchange Format verwendet indizierte Farben mit Dithering. Eine Besonderheit sind animierte Bilder, die mit gif möglich sind und im Web gerne eingesetzt werden. Transparenz ist ebenfalls mit gif möglich. gif verwendet eine verlustbehaftete Komprimierung (LZM).
- png: Portable Network Graphics ist eine Weiterentwicklung von gif und bietet mehr Farben (PNG-8 mit 8 bit Farbtiefe, PNG-24 mit 24 bit Farbtiefe). Für die Transparenz steht ein Alpha-Kanal bei PNG-24 zur Verfügung. Farbbilder im Format PNG-8 verlieren wie bei gif Farbinformation, wobei wieder durch Dithering eine Verbesserung des Ergebnisses erzielt werden kann.
- jpeg: Das Format der Joint Photographics Experts Group ist ein im Internet sehr verbreitetes Format. Es bietet eine Farbtiefe von 24 bit und verlustbehaftete starke Komprimierung, aber keine Transparenz.

Bilder, die eigens für das Ziel der Veröffentlichung im Web abgespeichert werden, sollten nicht nur bezüglich der Auflösung und der Komprimierung optimiert werden, sondern können durch weitere Maßnahmen verschlankt werden.

Da meistens das Colorprofil nicht ausgewertet wird, kann dieses entfallen, braucht also nicht in die Bilddatei integriert zu werden. Allerdings sieht die neueste Version des Standards für Bilddateien, Exif 2.21, vor, nicht das ganze Farbprofil zu integrieren, sondern nur auf das jeweilige Profil zu verweisen, wodurch kleine Bilddateien mit Colormanagement resultieren. Durch die Aktion Für Web speichern von Adobes ImageReady kann das ICC-Profil entfernt werden; der entsprechende Dialog bietet dazu eine Auswahl an.

Möglicherweise im Exif-File enthaltene Thumbnails sind für den Einsatz im Internet unnötig und können ebenfalls zur Reduzierung der Dateigröße entfernt werden.

### 27.3.5 Vektorgrafiken

Für den Einsatz im Web ist die Vektorgrafik von besonderer Bedeutung, da hier nur eine kleine Datenmenge zu übertragen ist.

In Kapitel 17 wurde bereits erwähnt, dass das Flash-Format swf ein Vektorgrafik-Format ist. Die freie Alternative hierzu ist das SVG Scalable Vector Graphics.

PHP ist – etwa in Verbindung mit der GD Library – in der Lage, Vektorgrafiken in SVG zu erzeugen.

### 27.3.6 PHP und Bilder im Web

Viele Grafik-Anwendungen wie Webgalerien sind in der Programmiersprache PHP realisiert – warum?

Aufgrund der Lizenzprobleme für das gif-Format wurde die Unterstützung für dieses Format zugunsten von png aufgegeben, die aktuelle Version enthält aber wieder die gif-Unterstützung, da die Rechte an LZW inzwischen ausgelaufen sind.

Eine weitere, hier wichtige Stärke von PHP ist die einfache Möglichkeit, Dateien – wie Bilddateien – aus dem Client auf den Webserver „hochzuladen“. Hierzu wird nicht die eher selten eingesetzte HTTP-Methode PUT verwendet, sondern eine (in der Norm RFC 1867 definierte) Erweiterung der POST-Methode. Hierbei wird ein spezielles Attribut eines HTML-Formulars zur Auswahl der Datei verwendet, wodurch letztlich eine Ziel-URL zum Server übertragen wird, unter welcher das Bild abgelegt wird. PHP kann dabei so konfiguriert werden, dass eine Schranke für die maximale Dateigröße den Server vor zu großen Datenmengen schützt.

PHP bietet auch direkt die Möglichkeit, Metadaten nach dem Exif-Standard mittels der Methode `exif_read_data` auszulesen.

### 27.3.7 ImageMagick und GD

Eine weitere Stärke von PHP liegt in der häufig integrierten GD-Bibliothek von Thomas Boutell; diese stellt sehr zahlreiche Funktionen zur serverseitigen Bildbearbeitung zur Verfügung.

Einige typische Funktionen, die GD bereitstellt, sind etwa:

- `imagecreate`: Erzeugt ein Bild;
- `imagejpg`, `imagepng` und `imagegif`: Sendet ein Bild im jpeg-, png- oder gif-Format an den Client;
- `getimagesize`: Bestimmung der Bildgröße;
- `imagesetpixel`: Setzen eines Pixels.

Damit können einfach Bilder serverseitig erzeugt werden, etwa für einen grafischen Seitenzähler.

Ähnliche Funktionen wie mit GD lassen sich auch mit ImageMagick erzielen, welches auch direkt auf der Kommandozeile anwendbar ist. ImageMagick beinhaltet beispielsweise leistungsstarke und schnelle Algorithmen zum Ändern der Bildgröße, weshalb es bei Webgalerien sehr beliebt ist.

Sehr viele Bildgalerien im Web sind aus diesen Gründen heute mit PHP realisiert.

## 27.4 Das pdf-Format

Das pdf-Format erfreut sich insbesondere im Bereich der Internetanwendungen einer großen Beliebtheit. Ausschlaggebend hierfür sind zwei Gründe:

- pdf-Dokumente bieten eine mächtige Formatierung unabhängig vom Betriebssystem;
- pdf-Dokumente sind kompakt: Sie benötigen vergleichsweise wenig Speicherplatz. Sind Grafiken in pdf integriert, kann sogar der Grad der Komprimierung gewählt werden.

pdf selbst ist ein Vektorgrafik-Format. Das Hauptwerkzeug zur Erzeugung und Bearbeitung dieses Formats ist die Acrobat-Produktfamilie der Firma Adobe. Allerdings ist der Einsatz der Adobe-Produkte zur pdf-Generierung bei Web-Applikationen aus lizenzrechtlichen Gründen heute die Ausnahme, Open Source-Projekte haben eine sehr große Bedeutung.

### 27.4.1 pdf mit PHP

PHP bietet zahlreiche Methoden zur Erzeugung von pdf-Dokumenten bis hin zur Integration von Grafiken. Es ist damit ein einfacher und lizenzfreier Weg, um pdf im Web anbieten zu können.

### 27.4.2 pdf mit Java

Auch mit Java kann pdf erzeugt werden, allerdings reicht hierfür nicht das Java Standardpaket, es werden Erweiterungen benötigt.

#### 27.4.2.1 Jakarta FOP

Eine attraktive Java-Erweiterung ist *FOP:* Formatting Objects Processor. Herausgegeben von der Apache Group, kann FOP aufgrund von Formatierungsvorgaben in XSL-FO (Extensible Stylesheet Language-Formatting Objects, eine XML-Anwendung) verschiedene Ausgabeformate erzeugen, darunter neben pdf auch Postsript und RTF.

# 28

# Content Management Systeme: TYPO3

## 28.1 Content Management Systeme

An vielen Stellen in diesem Buch wird die Trennung der Formatierung von den eigentlichen Inhalten und der Programmierung behandelt. Ein sehr einfacher Weg, um dies zu erreichen, ist ein *Content Management System* (CMS), auch als Redaktionssystem bezeichnet. Hier werden nur Systeme für das Web betrachtet, man spricht dann auch vom Web Content Management System (WCMS). Im Kern besteht es aus Templates für die Formatierung, aus einer Datenbank für die persistente Ablage des Contents, und der eigentlichen Programmlogik.

Ziel eines CMS ist die benutzerfreundliche und komfortable Verwaltung der Inhalte (Content) für große Sites, welche nicht mehr sinnvoll mit den klassischen Methoden des Web-Publishings verwaltet werden können. Dabei implementiert das CMS stets eine Benutzer- und Rollenverwaltung: Inhalte werden von Redakteuren eingepflegt, welche keine Kenntnisse der „Programmierung“ etwa in HTML benötigen, sondern über einfache Eingabemasken ihre Inhalte pflegen können.

Die Vorteile eines CMS sind somit vielfältig:

- Die Pflege der Inhalte erfordert keine HTML-Kenntisse.
- Das „richtige“ Layout wird automatisch erzeugt, das CMS liefert keine Seiten in einem falschen Design.
- Die Inhaltsdaten liegen wiederverwertbar in Datenbanken; dies kann Cross Media Publishing unterstützen: Die gleichen Ausgangsdaten werden verschieden aufbereitet, beispielsweise für das Web und für eine klassische Print-Ausgabe.
- Ein CMS verwaltet die Struktur der gesamten Sites; so werden Links automatisch erzeugt und bei Verschiebung des Targets direkt angepasst.
- Das Benutzer- und Rollenkonzept ermöglicht eine effiziente Aufgabenverteilung.
- Für die Inhalte können etwa Zeiten für die Veröffentlichung oder Erscheinungszeiten vorgegeben werden.
- Eine Versionsverwaltung, welche verschiedene Versionen eines Inhaltes verwaltet und verfügbar hält, wird von den größeren CMS geboten.
- Ein CMS kann Inhalte in verschiedenen Sprachversionen verwalten; dabei kann beim Fehlen eines Inhalts in der gewählten Sprache entweder auf die Default-Sprache zugegriffen werden, oder der Inhaltspunkt wird nicht angeboten.

**Abbildung 28.1:**
Ein großes CMS, aber nicht TYPO3

Ein populäres Beispiel für ein großes CMS ist etwa die Site „spiegelonline“ `spiegel.de`; hier kann man direkt die Veröffentlichung der Artikel beobachten. Der Redakteur braucht hierfür keine expliziten HTML-Kenntnisse, er kann für die Artikel zwischen verschiedenen Templates wählen, etwa mit/ohne Bild(er), eine Zusammenfassung ist anzugeben und der vollständige Artikel wird automatisch korrekt verlinkt. Das CMS ergänzt dann das komplette Layout einschließlich der Werbebanner.

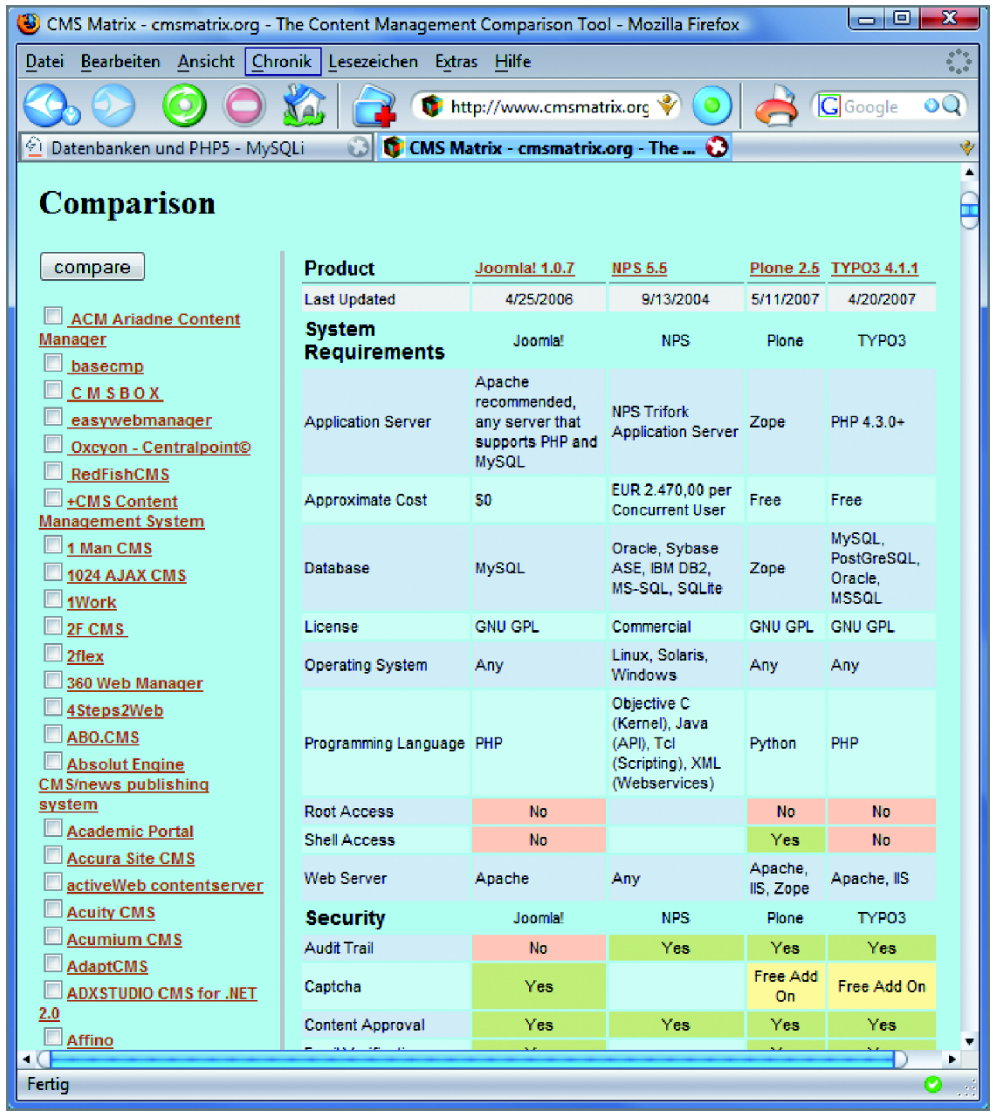



**Abbildung 28.2:** Vergleich verschiedener CMS bei `cmsmatrix.org`

### 28.1.1 Beispiele für Content Management Systeme

Content Management Systeme erfreuen sich einer starken und weiter wachsenden Beliebtheit; immer mehr Seiten werden auf CMS umgestellt. Entsprechend groß ist das Angebot von CMS sowohl im kommerziellen als auch im Open Source Bereich. Die Site `cmsmatrix.org` gibt einen guten Vergleich zwischen den einzelnen Systemen, die detailliert gegenübergestellt werden; ein guter Einstand für die Auswahl eines CMS.

### 28.1.2 Joomla! und Mambo

*Mambo* ist ein eher kleines, einfaches und übersichtliches Open Source CMS.[1] Sein Vorteil liegt gerade im geringen Komplexitätsgrad des CMS, vgl. [Deu05]: ein einfach zu konfigurierendes und dennoch leistungsfähiges CMS. Aufgrund einer Aufspaltung der Entwicklergruppe findet die Fortentwicklung im Content Management System *Joomla!* statt [web].

Joomla! (und genauso Mambo) basiert, wie viele CMS, auf PHP, als Datenbanksystem kommt MySQL zum Einsatz.

### 28.1.3 Plone

*Plone* ist ein CMS, welches wesentlich auf dem Python-Framework ZOPE basiert (vgl. 22.4); es ist praktisch die Reduzierung von ZOPE auf die Anforderungen eines CMS [web].

Da Plone auf ZOPE beruht, wird standardmäßig zur Datenhaltung das ZOPE-eigene Datenbanksystem verwendet.

### 28.1.4 NPS

*NPS* ist ein verbreitetes kommerzielles CMS des Anbieters Infopark [web]. Es ist weitgehend in C realisiert und ist Teil eines umfassenden kommerziellen Angebotes bis hin zu einem vollständigen Portal-Server.

### 28.1.5 TYPO3

Das Open Source CMS *TYPO3* gilt als eines der leistungsfähigsten CMS und erfreut sich einer sehr großen Verbreitung. Deshalb wird es hier in 28.2 ausführlicher vorgestellt.

### 28.1.6 ILIAS

*ILIAS* ist kein Content Management System, es soll hier aber dennoch kurz vorgestellt werden; ILIAS ist ein „Learning Management System“ (LMS). Diese Systeme sind spezielle Formen eines CMS für die Zwecke E-Learning und Blended Learning, also für netzgestütztes Lehren und Lernen. Auch hier, deshalb die Parallele zum CMS, veröffentlicht der Redakteur entsprechende Inhalte. Diese Systeme ergänzen allerdings weitere Funktionalitäten, etwa ein spezielles Benutzermanagement für den Frontend-Benutzer, also den Lernenden, integrierte Prüfungssysteme und vieles mehr.

ILIAS steht für „Integriertes Lern-, Informations- und Arbeitskooperations-System“ und ist eine von der Universität zu Köln ausgehende Open Source Entwicklung [web]. Neben diesem gibt es einige weitere Learning Management Systeme, kommerzielle wie freie. Innerhalb der freien Systeme nimmt ILIAS eine ähnlich starke Rolle ein wie TYPO3 für die Content Management Systeme.

Technisch basiert ILIAS ebenfalls auf PHP, als Datenbanksystem kommt MySQL zum Einsatz.

[1] Mambo liegt inzwischen in zwei Versionen, einer freien und einer kommerziellen, vor.

## 28.2 Das CMS TYPO3

*TYPO3* ist momentan das populärste Content Management System [web]. Dieses Open Source System findet in allen Bereichen einen breiten Einsatz.

**Abbildung 28.3:** TYPO3-Logo

Die Literatur zu TYPO3 ist umfassend; hier soll anhand dieses CMS die wesentliche Arbeitsweise von Content Management Systemen erläutert und dabei ein erster Einstieg in TYPO3 gegeben werden. Vertiefende Literatur ist etwa [SB06].

### 28.2.1 Genese von TYPO3

TYPO3 gilt als „Enterprise Content Management System“, da es ausgesprochen mächtig ist. Es wird auch im kommerziellen Umfeld intensiv genutzt, zahlreiche DAX-Unternehmen betreiben ihre Site mit TYPO3.

Der Schwede Kasper Skårhø ist der Initiator von TYPO3; er begann 1997 mit der Entwicklung des CMS. Die erste offiziell stabile Version wurde 2002 veröffentlicht, obwohl schon zahlreiche Sites mit den Vorversionen erfolgreich arbeiteten.

Das neue Major-Release 4 von TYPO3 hat u.a. den XHTML-Support und CSS_styled_content als Teil des Standardsystems eingeführt und ermöglicht die saubere Ersetzung aller TABLE-Darstellungen von HTML. In der Version 4.1 kam insbesondere Ajax zum Einsatz (vgl. Kapitel 16); damit wird im Backend bei Änderungen an der Struktur asynchron der Seitenbaum neu geladen, aber nur dieser: Ein kompletter Neuaufbau der gesamten Administrations-Seite ist nicht mehr erforderlich, was den Umgang mit TYPO3 wesentlich angenehmer gestaltet.

Neu in TYPO3 4.1 ist eine von Oliver Hader entwickelte Technologie: Inline Relational Record Editing (IRRE). Diese ermöglicht es, untergeordnete Datensätze zu einem übergeordneten Datensatz direkt im TYPO3-Backend-Formular hinzuzufügen.

### 28.2.2 Architektur von TYPO3

Die *Architektur* von TYPO3 wird in Abbildung 28.4 verdeutlicht: Serverseitig handelt es sich um PHP-Code, welcher mit einem relationalen DBMS arbeitet, häufig MySQL, und durch den Apache-Webserver ausgeführt wird. Das eigentliche TYPO3 ist die Zwischenschicht, welche durch Extensions erweitert werden kann (vgl. 28.2.5). Der Administrator und der Redakteur benutzen das Backend (BE) zur Administration und zur Pflege der Inhalte, der normale Besucher der Web-Site sieht das Frontend (FE).

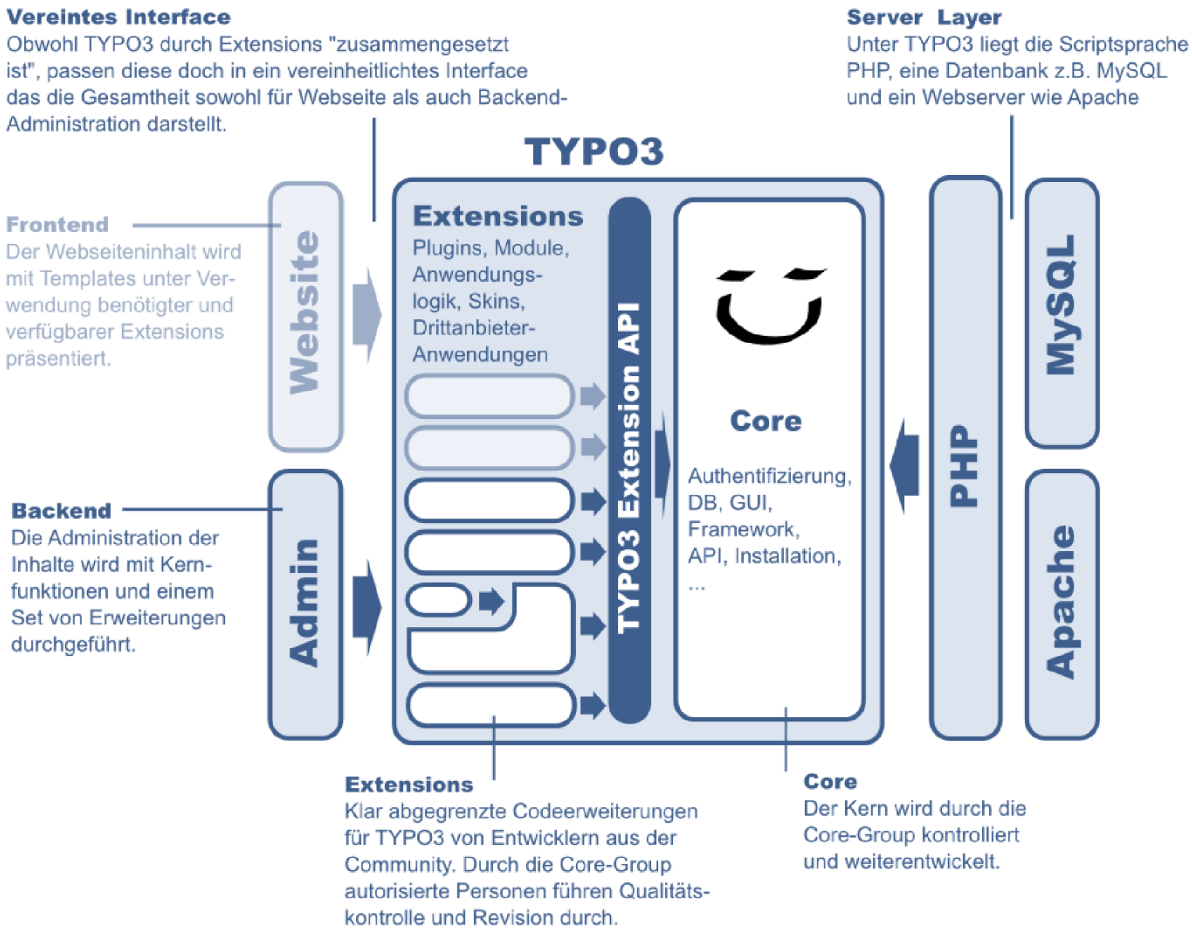



**Abbildung 28.4:**
Architektur von TYPO3

### 28.2.3 Installation von TYPO3

TYPO3 ist zunächst – wie viele andere CMS auch – ein umfassendes PHP-Framework. Das bedeutet, zur Installation sind eigentlich nur PHP-Archive entsprechend zu entpacken und dann als wesentlicher Schritt die Anbindung an ein Datenbanksystem zu konfigurieren. TYPO3 unterstützt dabei viele verschiedene DBMS, häufig wird es mit MySQL eingesetzt.

Voraussetzung für den Einsatz von TYPO3 sind somit lediglich:[2]

- ein funktionierender Apache-Webserver[3] mit einer aktuellen PHP-Version[4];
- ein relationales DBMS wie MySQL, PostgreSQL oder Oracle;
- einige Erweiterungen wie die Bildbearbeitung ImageMagick und die Grafik-Library GD für PHP.

Das Server-Betriebssystem ist dabei im Prinzip wieder beliebig, da PHP betriebssystemunabhängig ist.

[2] auf der TYPO3-Site gibt es auch ein Gesamtbundle einschließlich Apache-Webserver und MySQL-Datenbank, welches eine einfache Gesamtinstallation ermöglicht.

[3] der Einsatz von Microsft IIS soll auch möglich sein.

[4] formal genügt momentan die Version 4.3 von PHP, ab der Version 4.2 von TYPO3 wird PHP ab Version 5 vorausgesetzt.

Für die Installation werden dann zwei Archiv-Pakete benötigt:

- das eigentliche Source-Paket
- das Paket für das Anlegen einer leeren TYPO3-Site `dummy`.

Beide Pakete sind im Documentroot des Webservers oder einem geeigneten Unterordner zu entpacken.

Die Leistungsfähigkeit kann man schon daran sehen, dass eine TYPO3-Installation mehrere Domains verwalten kann; dann werden mehrere Dummy-Pakete in verschiedenen Unterordnern benötigt.

Nach diesen Schritten ist die Datei `index.php` im Startverzeichnis der TYPO3-Site vorhanden; mit dem Browseraufruf auf diese Datei beginnt die eigentliche Installation zunächst mit dem 1-2-3-Tool, daran schließt sich die Konfiguration des CMS an.

#### 28.2.3.1 Installation von TYPO3: das 1-2-3-Tool

Durch den Browseraufruf der `index.php` startet zunächst das *1-2-3 Install Tool* von TYPO3. Aufgabe hiervon ist lediglich die Konfiguration des Datenbankzugriffs von TYPO3.

Im ersten Schritt ist deshalb die Kennung (Benutzer/Passwort/Host) für das DBMS anzugeben; im zweiten Schritt ist die Datenbank auszuwählen oder eine neue anzulegen (Abbildung 28.5); danach werden die notwendigen rund 40 Datenbanktabellen für TYPO3 angelegt und für ein „leeres“ CMS gefüllt.

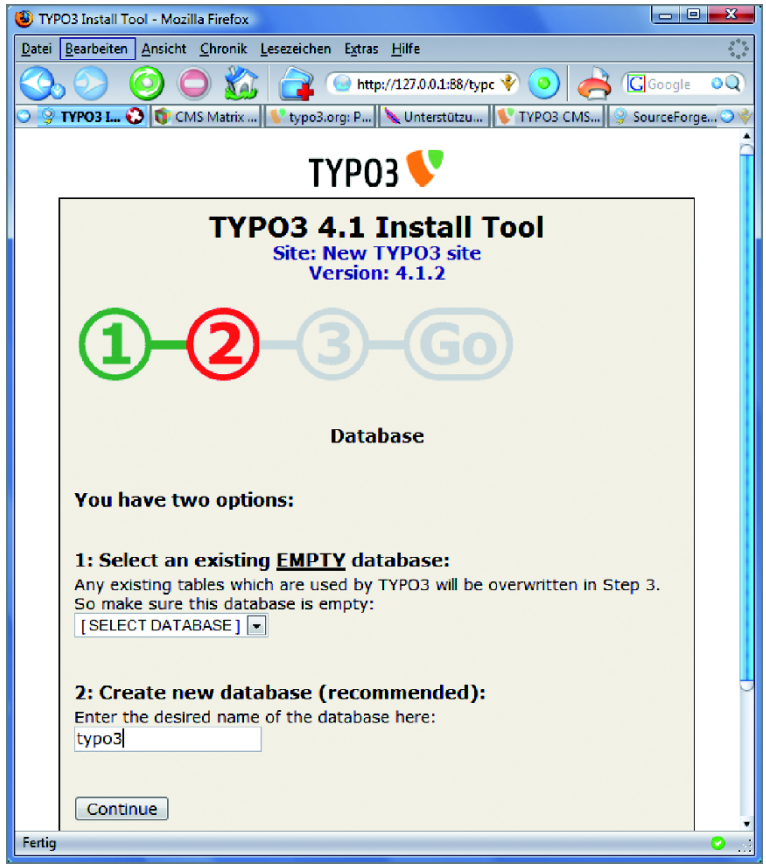



**Abbildung 28.5:** Auswahl der Datenbank im 1-2-3-Tool

Nach diesem Schritt ist die Datenbankanbindung von TYPO3 konfiguriert und eine Datenbank angelegt. Als nächstes ist das System selbst grundlegend zu konfigurieren.

#### 28.2.3.2 Installation von TYPO3: Anpassen der localconf.php

Der nächste Schritt der Installation kann aus dem Abschluss des 1-2-3-Tools oder nachträglich aus dem Backend über den Menüpunkt „Install“ gestartet werden. Hier wird die eigentliche Konfiguration von TYPO3, die Datei

```
typo3conf/localconf.php
```

im Installationsverzeichnis von TYPO3, verändert.

Diese Datei kann auch direkt editiert werden, was aber nicht empfehlenswert ist, da TYPO3 diese auch wieder überschreibt; die Konfiguration erfolgt natürlich Web-basiert über das Konfigurationsmenü nach Abbildung 28.6.

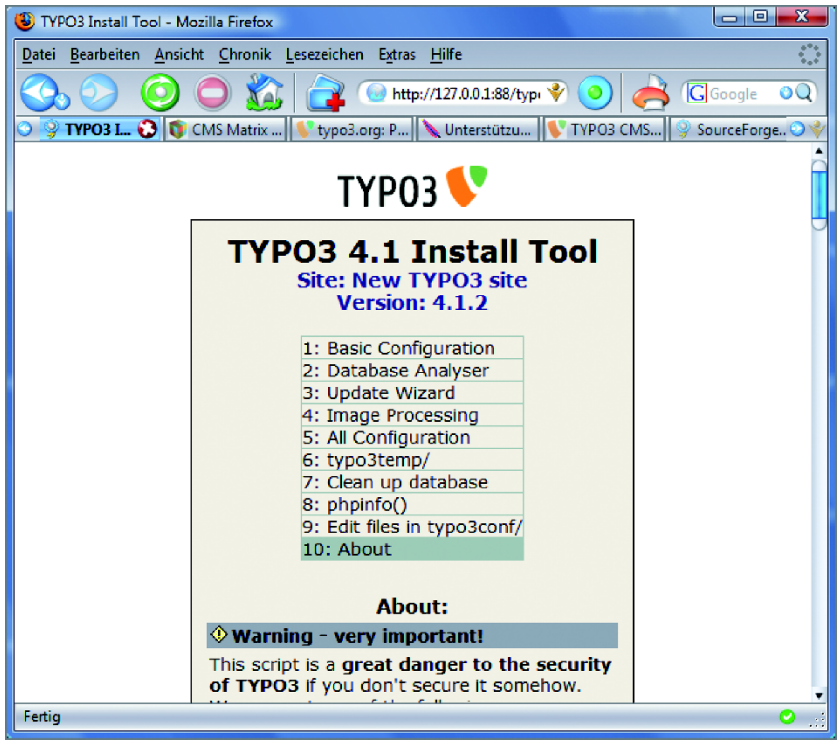



**Abbildung 28.6:** Konfigurationsmenü

Dieses Konfigurationsmenü bietet zahlreiche Unterpunkte; zentral sind insbesondere

- die „Basic Configuration“ für zentrale Einstellungen wie Kennwortverwaltung;
- „Database Analyser“ und „Image Processing“;
- „All Configuration“: hier kann die `localconf.php` wesentlich verändert werden.

Alle Konfiguratiosparameter werden intern in einem PHP-Array

```
$TYPO3_CONF_VARS
```

vorgehalten.

Wesentlich ist hierbei der Menüpunkt „All Configuration“. Dieser selbst gliedert sich in fünf Abschnitte für die Systemkonfiguration des CMS:

- GFX für die Grafikverarbeitung (Pfad zu ImageMagick etc.);
- SYS für die Systemkern-Konfiguration (Name der verwalteten Site etc.);
- EXT für die Konfiguration der TYPO3-Erweiterungen durch Extensions (vgl. 28.2.5);
- BE für die Backend-Konfiguration;
- FE für die Frontend-Konfiguration.

Nach abgeschlossener Konfiguration steht die Anmeldung am Backend für die eingerichteten Administratoren bereit (Abbildung 28.7).

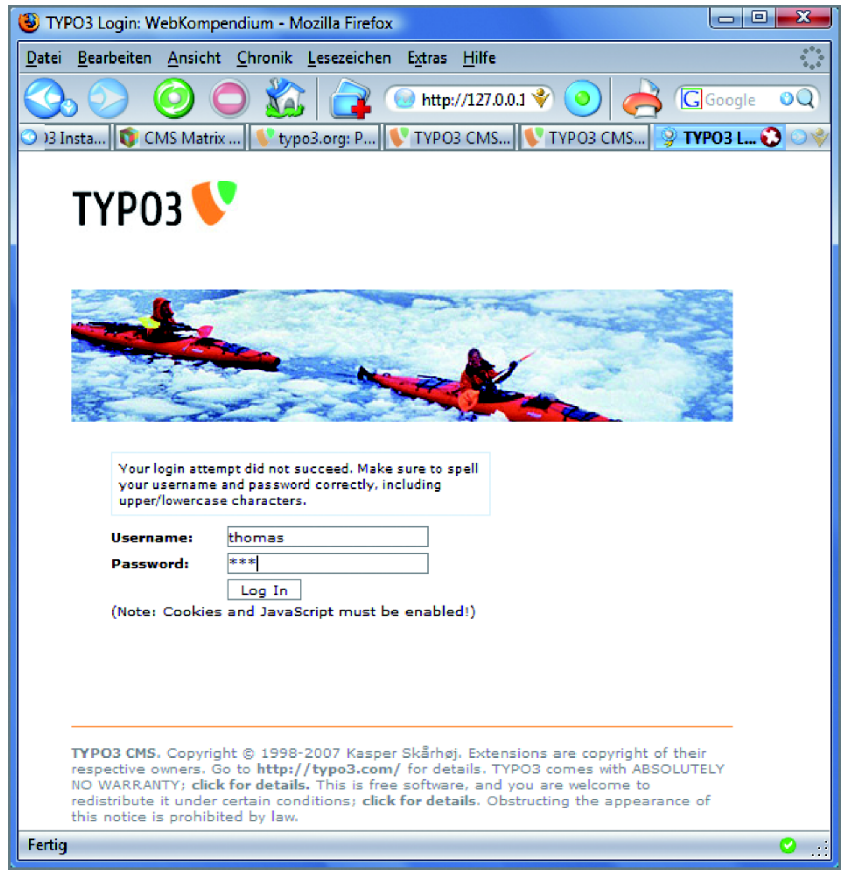



**Abbildung 28.7:** Anmeldung am Backend

### 28.2.4 Das Backend von TYPO3

Das *Backend* (BE) ist der Bereich für Administatoren und Redakteure zum Verwalten des CMS und der Site; üblicherweise wird es über den Link

```
<typo3instanz>/typo3
```

erreicht. Innerhalb des Backends stehen einzelne Aktivitäten als auswählbare Module bereit, die standardmäßig wiederum für mehr Übersichtlichkeit in fünf Modulgruppen geordnet sind:[5]

- Modulgruppe Web: Funktionen zur Bearbeitung einer Site mit dem Seitenbaum (Page), einer Vorschau (View), der Listen-Darstellung (List) für die

[5] natürlich ist auch diese Anordnung in TYPO3 veränderbar, außerdem ändert sich diese Auflistung je nach den Rechten und Rollen das Benutzers.

Darstellung möglichst genauer Informationen, der Bearbeitung von Templates (Template), der Versionierung (Versioning), der Zugriffsregelung (Access) und vielem mehr;

- Modulgruppe File mit der Filelist;
- Modulgruppe User für die Benutzerverwaltung (Abbildung 28.8);
- Modulgruppe Tools mit wichtigen Funktionalitäten wie dem Aufruf des Install-Tools (Install), der Extensions-Verwaltung (Extension Manager), der Verwaltung der Backend-Adminsitratoren (User Admin), der Auswertung von Logfiles (Log) und noch einigen mehr.

Jede einzelne dieser Aktivitäten wird durch ein eigenes TYPO3-Modul ermöglicht.

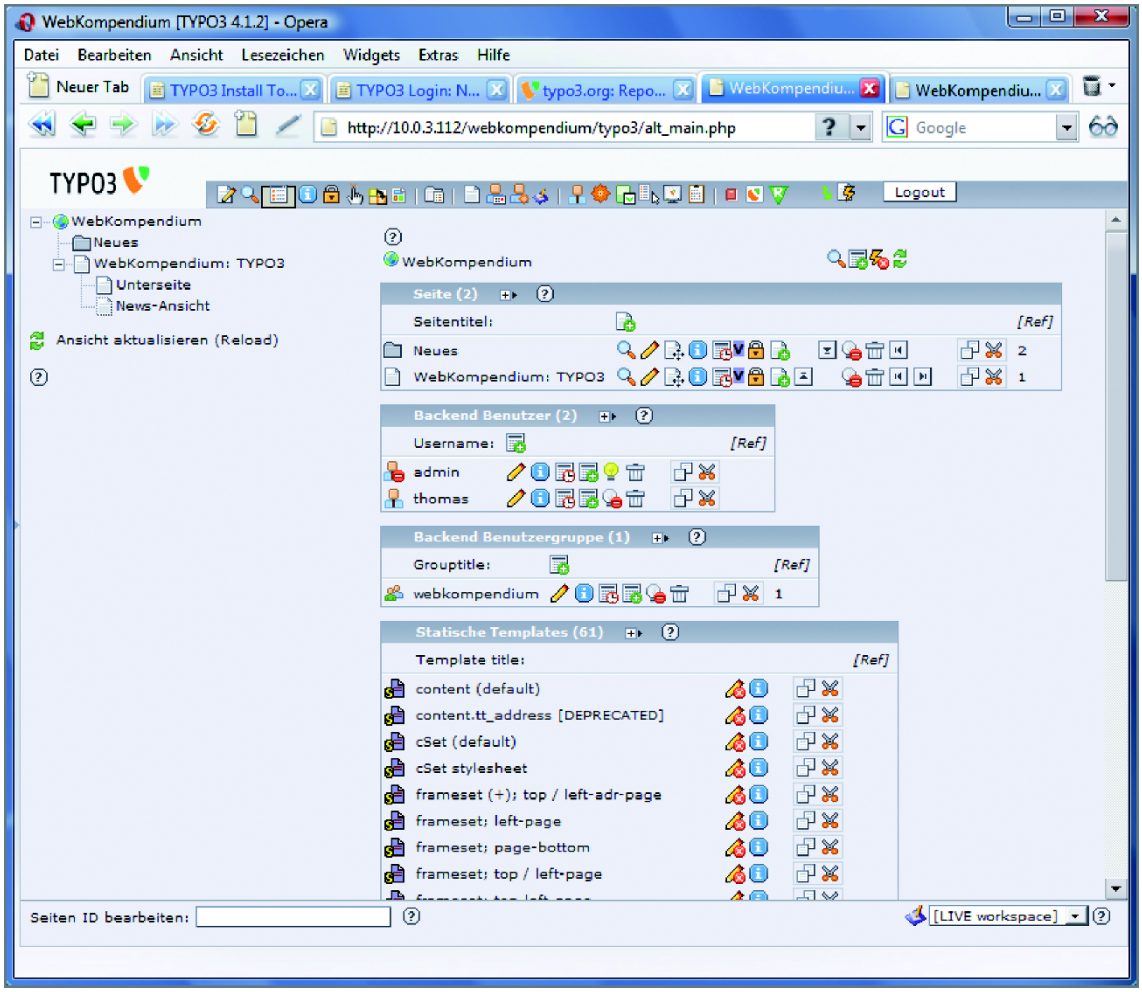



**Abbildung 28.8:** Benutzerverwaltung im TYPO3-Backend

### 28.2.5 TYPO3-Erweiterungen: Extensions

TYPO3 bietet eine umfassende und gut dokumentierte API, die das Entwickeln von Erweiterungen, sogenannten *Extensions,* erlaubt. Durch diese kann das TYPO3-System wesentlich in seinem Funktionsumfang erweitert werden.

Es gibt inzwischen zahlreiche Extensions für TYPO3; die TYPO3-Site enthält das TYPO3 Extensions Repository (TER), welches auch eine Suche nach Extensions anbietet (Abbildung 28.9).

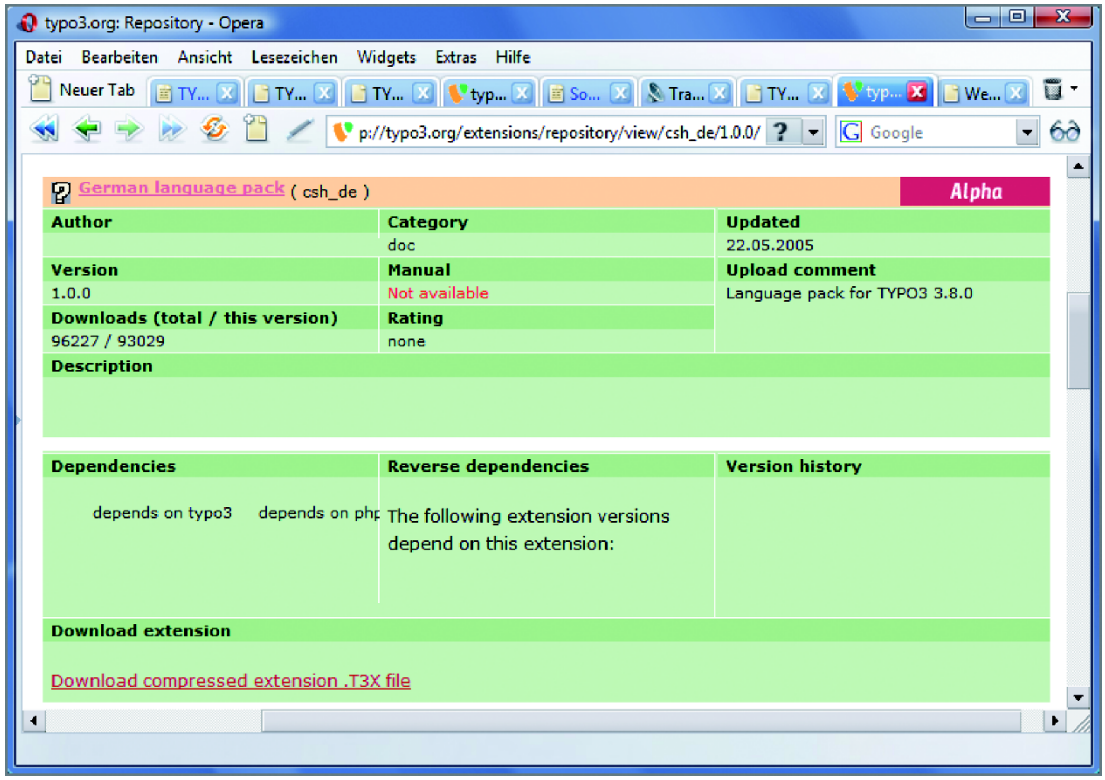



**Abbildung 28.9:** TYPO3-Extension für deutsche Sprachunterstützung

Eine solche Extension ist ein spezielles gepacktes Archiv in Form einer `T3X`-Datei. Diese kann von der TYPO3-Seite heruntergeladen werden und ist dann vom Administrator im Extension Manager des Backends hinzuzufügen (Abbildung 28.10).

Im hier vorgestellten Beispiel werden zwei solcher Extensions hinzugefügt: chs_de für die deutsche Sprachunterstützung und tt_news für die Verwaltung von News.

Es gibt zahlreiche weitere wichtige Extensions, etwa das beliebte TemplaVoila! für eine freiere Gestaltung der TYPO3-Sites mit vorgefertigten Templates (Flexforms).

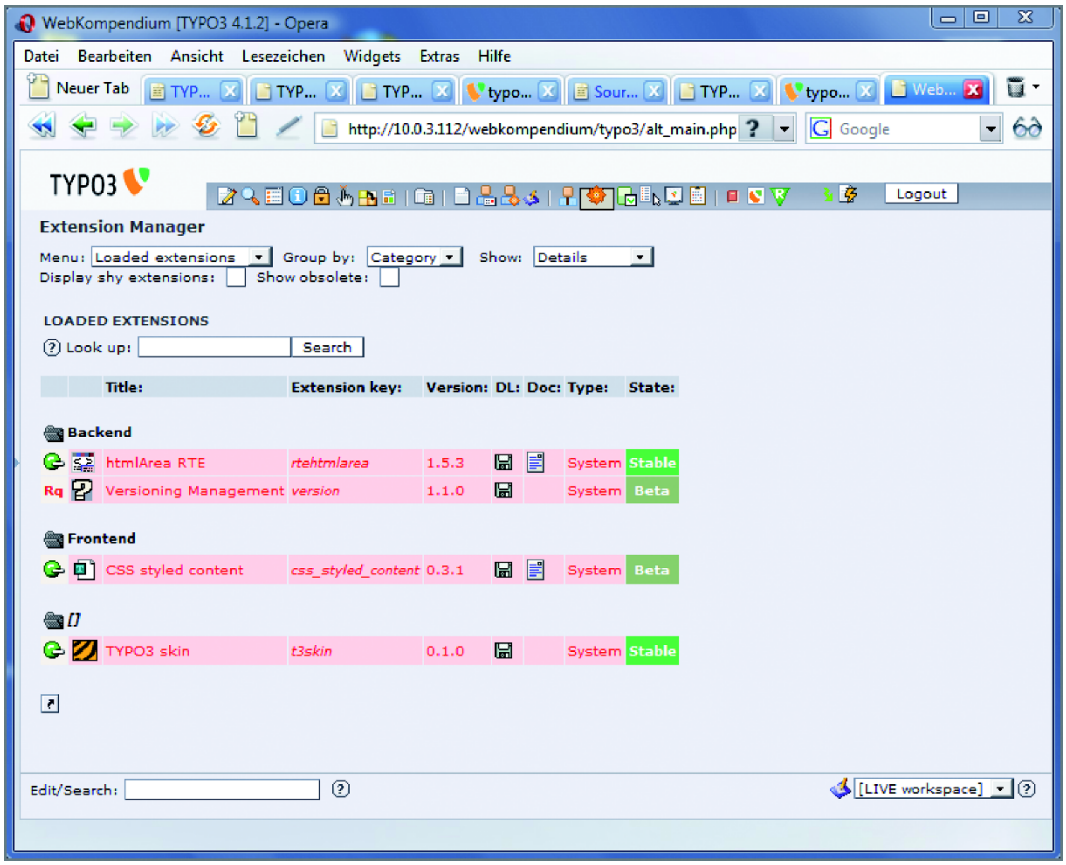



**Abbildung 28.10:**
TYPO3 Extension Manager im Backend

### 28.2.6 Ein TYPO3-Projekt

Zur Verdeutlichung der TYPO3-Arbeitsweise soll hier ein minimalistisches Projekt „WebKompendium“ angelegt werden. Hierzu wird im Seitenbaum unterhalb dem Wurzelelement (Weltkugel) eine neue Einstiegsseite (Rechteck) erzeugt (Abbildung 28.11). Dieses hat verschiedene Eigenschaften wie einen Seitentitel.

Diesem Element wird ein Template zugeordnet, welches die Inhalte des Elements verwaltet.

Ein TYPO3-Template unterteilt die Seite in vier Spalten: links, normal (die eigentliche Inhaltsspalte), rechts, Rand. Durch spezielle Extensions, etwa das bereits angesprochene TemplaVoila!, kann hier auch ein flexibleres Layout implementiert werden.

Im Beispiel soll nur die mittlere Spalte gefüllt werden. Folgender Inhalt soll auf der Startseite unseres TYPO3-Projektes enthalten sein:

- ein Überschriftentext
- ein Bild
- ein weiterer Text
- eine Liste von News
- ein Link auf eine nachfolgende Seite

Ferner soll das exemplarische CSS aus Kapitel 2.5 eingebunden werden.

Um dies zu leisten, ist das Template für das TYPO3-Element entsprechend zu gestalten (Abbildung 28.12).

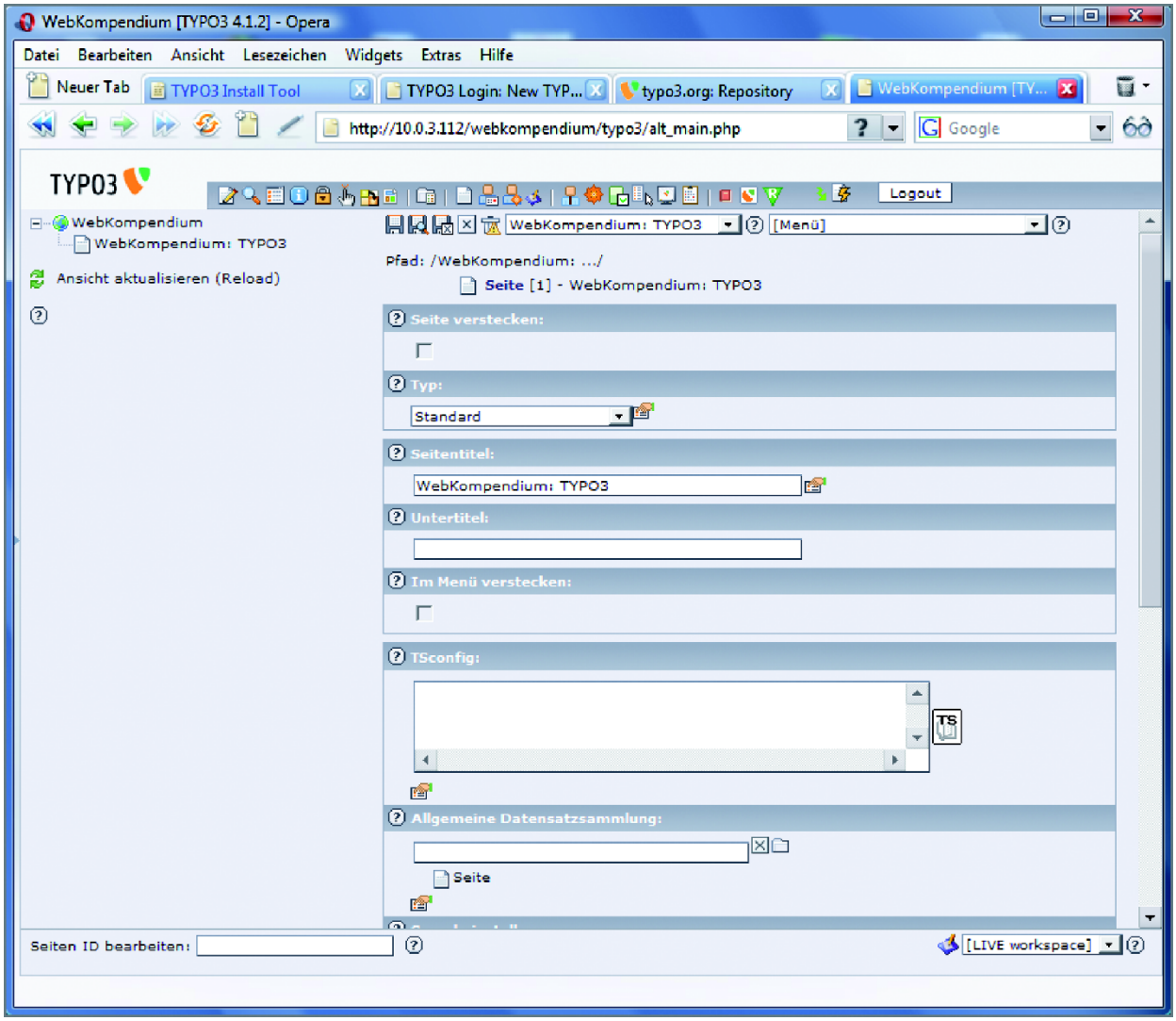



**Abbildung 28.11:**
Neues TYPO3-Element für die erste Projekt-Seite

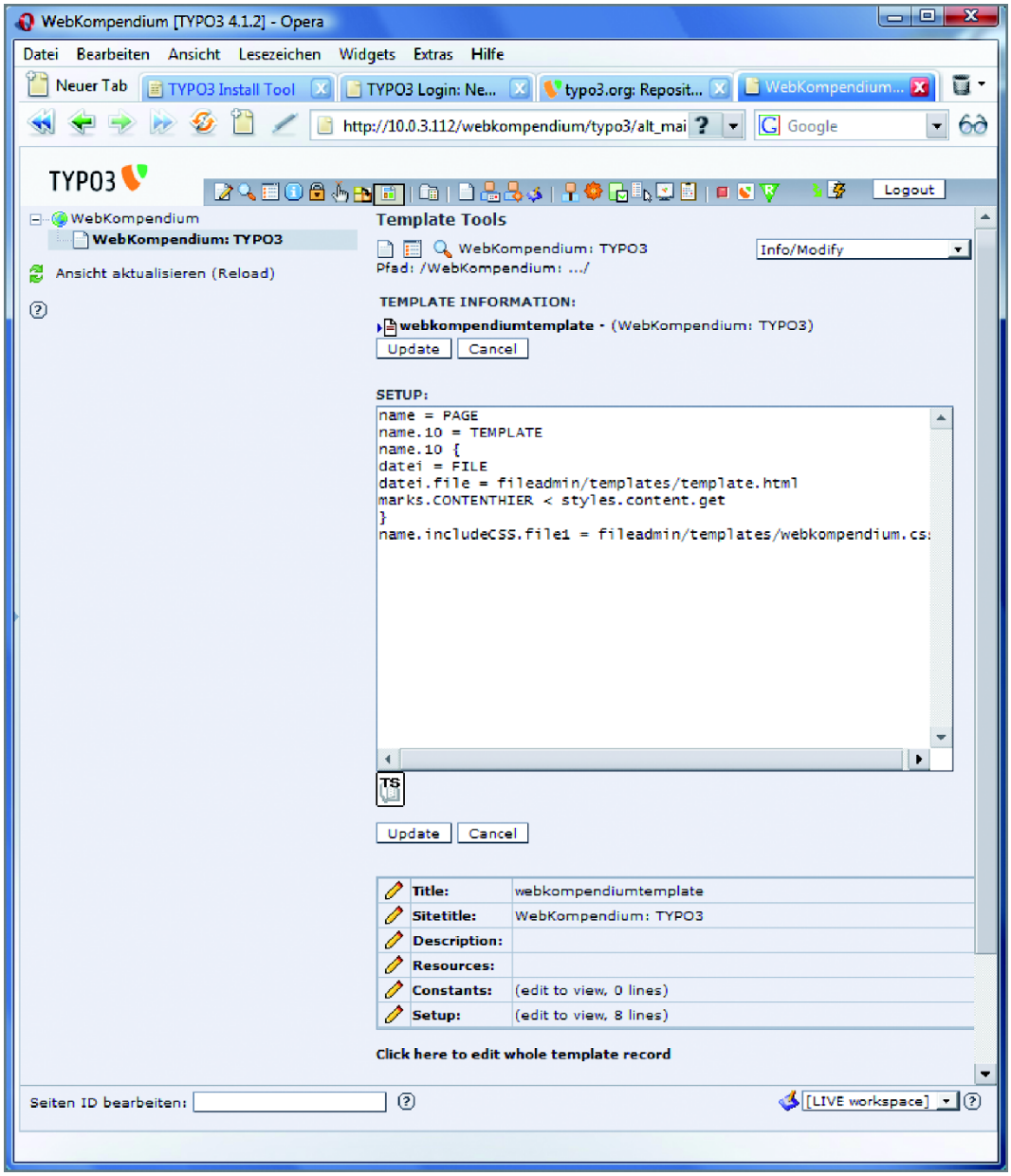



**Abbildung 28.12:**
Root-Template für die Site

Es werden zwei weitere Dateien für die Gestaltung der Site benötigt:

- die verwendete CSS-Datei;
- ein (einfaches) HTML-Template für die Verbindung des TYPO3-Templates nach Abbildung 28.12 mit der zu rendernden HTML-Ausgabe; ein solches einfaches Template ist in Abbildung 28.13 zu sehen.

Beide Dateien und die anzuzeigende Grafikdatei müssen in das CMS integriert werden, hierzu dient die TYPO3-interne Dateiverwaltung im Filemanager.

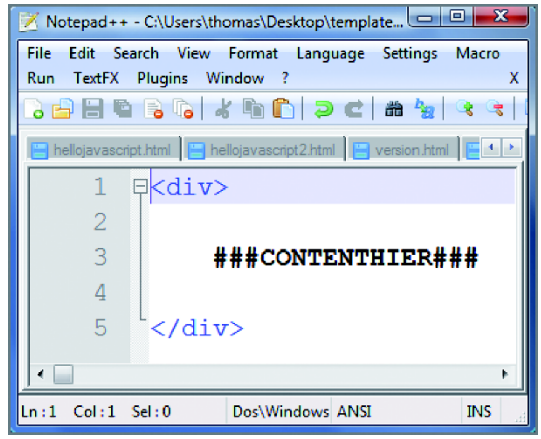



**Abbildung 28.13:** Einfaches HTML-Template

Die Verbindung des TYPO3-Templates mit der CSS-Datei und dem HTML-Template geschieht innerhalb des TYPO3-Templates unter „setup“ mittels TypoScript, der TYPO3-eigenen Konfigurationssprache. Der Kern dieser Verbindung ist:

```
page = PAGE
page.typeNum = 0
page.10 = TEMPLATE
page.10 {
  template = FILE
  template.file = fileadmin/templates/markers.htm
  marks.MARKER0 < temp.tmenu
  marks.MARKER1 < styles.content.get
}
page.includeCSS.file1 = fileadmin/templates/webkompendium.css
```

**Listing 28.1:** TypoScript für die Verknüpfung mit CSS und dem HTML-Template „markers.htm“

Das vollständige TypoScript ist in Listing 28.2 zu sehen.

Für die Einbindung von News wird eine spezielle Extension verwendet: tt_news. Hierbei wird für jede News ein eigenes Element erzeugt (wieder mit spezifischen Eigenschaften wie Titel, Zusammenfassung, Text usw.).

Diese Extension ist ausgesprochen mächtig und implementiert für ein neues Inhaltselement Funktionalitäten wie Achivierung, Beziehungen zwischen diesen Elementen und News-Kategorien. Wichtig für tt_news ist der Modus: Im Modus `LATEST` werden nur die neuesten Elemente angezeigt, die dann verknüpft sind zu einer Seite, die die ausgewählte News im `SINGLE`-Modus, also vollständig, anzeigt. Wichtige weitere Konfigurationsschritte sind die Angabe der News-Quelle („SysFolder“), die Anzahl der in `LATEST` angezeigten Elemente sowie deren Anzeigenlänge. Die Darstellung der News wird über ein eigenes Template gesteuert.

Das Endergebnis im Frontend ist eine TYPO3-Seite nach Abbildung 28.14.

**Abbildung 28.14:**
Die erste Seite im Frontend

### 28.2.7 TypoScript

*TypoScript* ist die TYPO3-eigene Sprachkonstruktion zur Steuerung des Verhaltens der TYPO3-Elemente. Im Web [web] ist auch eine vollständige deutsche Referenz für TypoScript zu finden.

Da TYPO3 den Zustand der Site intern über (mehrdimensionale) PHP-Arrays steuert, dient TypoScript letztlich der einfachen Manipulation dieser Arrays; TypoScript ist keine vollständige Scriptsprache, wie sie hier vorgestellt wurden, sondern eher wie XML eine Beschreibungssprache für Informationsstrukturen. Dabei ist es das Ziel von TypoScript, dass programmierunerfahrene Anwender damit einfach ihre Site und alle Elemente konfigurieren können.

Für das konkrete Beispiel hier das vollständige TypoScript für das setup-Element der Startseite:

```
// CONFIG
config.removeDefaultJS = 1
config.inlineStyle2TempFile = 1
config.admPanel = 1

// TMENU
temp.tmenu = HMENU
temp.tmenu{
entryLevel = 0
  1 = TMENU
  1.wrap = <ul>|</ul>
  1{

    NO{
wrapItemAndSub = <li>|</li>
    }
    ACT=1
    ACT{
      wrapItemAndSub = <li class="menu-level1-active">|</li>
    }
    CUR=1
    CUR{
      wrapItemAndSub = <li class="menu-level1-current-active">|</li>
    }
    IFSUB=1
    IFSUB{
      wrapItemAndSub = <li class="menu-level1-with-subpage">|</li>
    }
  }
  2 = TMENU
  2.wrap = <ul class="menu-level2">|</ul>
  2{
    NO{
      wrapItemAndSub = <li>|</li>
    }
    ACT=1
    ACT{
      wrapItemAndSub = <li class="menu-level2-active">|</li>
    }
    CUR=1
    CUR{
      wrapItemAndSub = <li class="menu-level2-current-active">|</li>
    }
    IFSUB=1
    IFSUB{
      wrapItemAndSub = <li class="menu-level2-with-subpage">|</li>
    }
  }
}
```

```
page = PAGE
page.typeNum = 0
page.10 = TEMPLATE
page.10 {
template = FILE
template.file = fileadmin/templates/markers.htm
        marks.MARKER0 < temp.tmenu
marks.MARKER1 < styles.content.get

}
page.includeCSS.file1 = fileadmin/templates/webkompendium.css
```

**Listing 28.2:** TypoScript des setup-Elements im TYPO3-Template der Startseite: Verknüpfung mit allgemeinem Template `markers.htm`

### 28.2.8 Frontend-Editing mit TYPO3

Der Umgang mit dem Backend ist für Administratoren, insbesondere aber für Redakteure, nicht leicht zu erlernen. Für die Gruppe der Redakteure bietet das *Frontend-Editing* eine attraktive Alternative: Voraussetzung ist ein erfolgreiches Anmelden als Backend-User, welcher im Gegensatz zu einem Frontend-Benutzer im Wurzelelement verankert ist.

Die Extension simulateBE ermöglicht zusätzlich die Verknüpfung von Frontend- und Backend-Benutzern, so dass ein angemeldeter FE-Nutzer mit den Rechten des zugeordneten BE-Benutzers über das Frontend direkt Änderungen vornehmen kann (Abbildung 28.15.

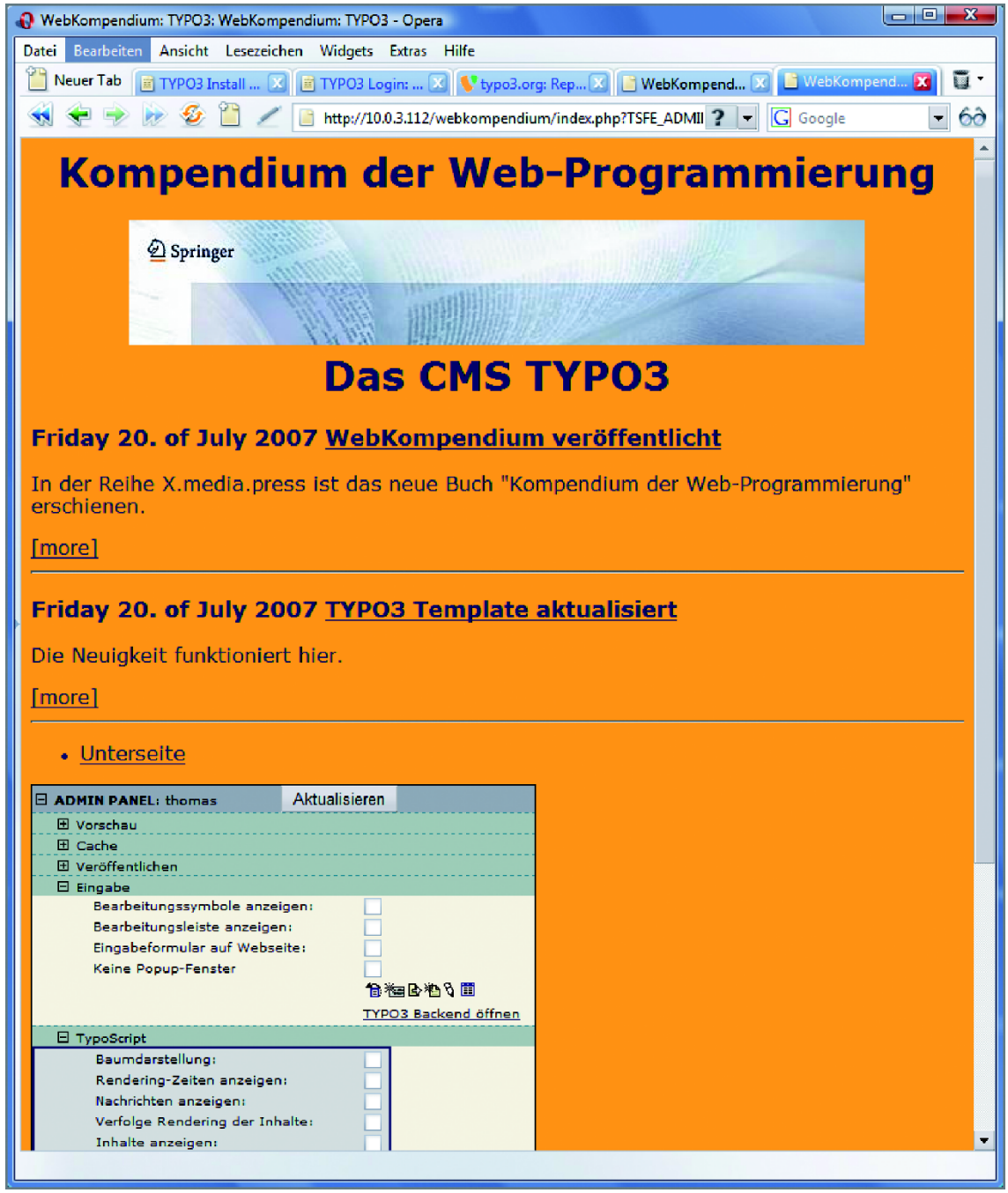



**Abbildung 28.15:**
Frontend-Editing mit TYPO3

# 29

# Performance und Testverfahren für Web-Applikationen

## 29.1 Bedeutung der Testverfahren

Die professionelle Entwicklung von Web-Applikationen setzt heute sinnvollerweise Testverfahren ein. Dabei geht es zum einen um die Überprüfung der Performance des Verfahrens, zum anderen auch um die automatisierte Kontrolle der Korrektheit der Anwendung: Liefert sie bei identischer Eingabe das korrekte Ergebnis?

Die Leistungsfähigkeit, die Performance einer Web-Applikation, ist von besonderer Bedeutung, denn der Besucher einer Site ist es nicht gewohnt, auf die Serverantwort länger zu warten. Aus Usability-Gründen werden maximale Antwortzeiten von deutlich unter zehn Sekunden angegeben. Bei zu langer Reaktionszeit sendet der Benutzer meistens über „Reload“ den Request erneut, was das Serversystem noch mehr belastet und zu falschen Zuständen führen kann – oder er bricht den kompletten Vorgang ab.

Ein ausgiebiges Testen, verbunden mit einer Analyse der „Bottlenecks“ ist deshalb für Web-Applikationen sehr wichtig. Ein moderner Ansatz hierfür soll nun vorgestellt werden.

## 29.2 Performance mit JMeter

Ein umfangreiches Werkzeug zur Performance-Analyse kommt ebenfalls von der Apache Group: *JMeter* [web].

JMeter ist eine reine Java-Desktop-Applikation, welche ausgesprochen bequem vordefinierte HTTP- und FTP-Requests an einen Webserver und SQL-Abfragen an einen Datenbankserver schicken kann. Aufgrund der Implementierung in Java ist JMeter plattformunabhängig einsetzbar. Nach der vorgegebenen Installation wird JMeter über eine GUI (realisiert mit Swing) bedient (Abbildung 29.2).

### 29.2.1 Der Testplan

Im ersten Schritt ist ein Testplan zu erstellen, der festlegt, welche Requests an die Web-Applikation zu senden sind. Der über Bearbeiten|Hinzufügen|Non-Tests Elements erreichbare Proxy-Server kann dabei verwendet werden, um während der Navigation durch die Web-Applikation die Schritte aufzuzeichnen und daraus ein Test-Szenario zu generieren.

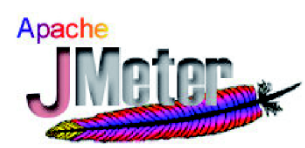



**Abbildung 29.1:** JMeter

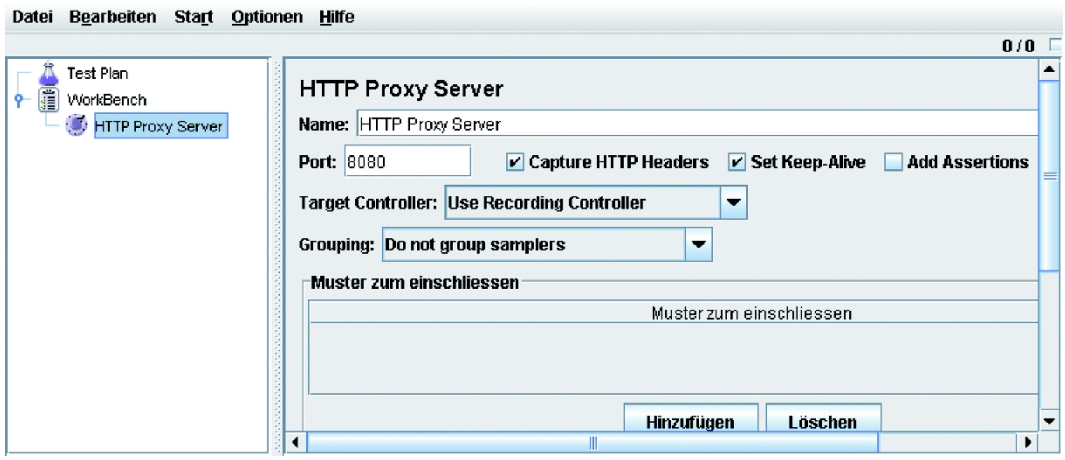



**Abbildung 29.2:** JMeter-GUI mit Konfiguration des Proxy-Servers

Im Menü dieses Proxy-Servers ist insbesondere ein „Recording Controller“ über Hinzufügen|Logik Kontroller|Recording Controller hinzuzufügen, der den eigentlichen Testplan dann aufzeichnet.

Hiermit sind die wesentlichen Konfigurationen des JMeter schon abgeschlossen und der Testplan kann erstellt werden: Mit einem beliebigen Browser werden nun – allerdings über den Proxy-Server des JMeters – Anfragen an die Web-Applikation gesendet und dabei vom Recording Controller, aktiviert über Start, aufgezeichnet. So generiert der Recording Controller aus der realen Benutzung der Applikation einen Testplan, welcher später einfach durchlaufen werden kann. Der Recording Controller kann die so erzeugte Aufzeichung der Abfrage zur Übersichtlicheit in Gruppen einteilen (Abbildung 29.3, linkes Fenster).

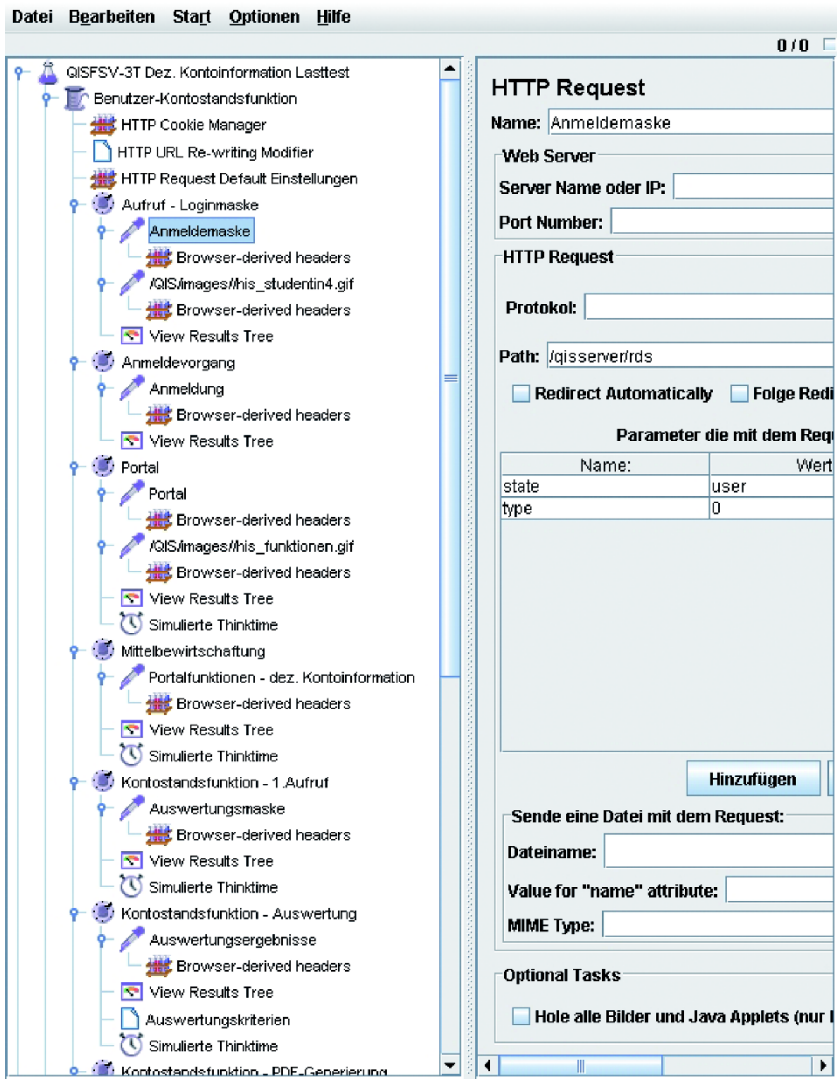



**Abbildung 29.3:**
Testplan im JMeter (Fenster links) und HTTP Request-Element (Fenster rechts)

### 29.2.2 Durchführung des Tests

Nachdem ein Test mit diesen Hilfen des JMeters definiert wurde, kann er automatisch durchgeführt werden. Hierbei ist auch festzulegen, welche Antworten akzeptiert werden (Test auf Korrektheit) und wie viele Anfragen an die Web-Applikation gesendet werden (Performance). Bei der Anzahl der Requests ist auch deren zeitliche Abfolge entscheidend: Werden alle Requests parallel (Default-Fall) oder zeitlich versetzt gesendet? Es können zwischen einzelnen Schritten auch einzelne Verzögerungen eingebaut werden, um ein reales Benutzerverhalten besser zu simulieren.

Viele dieser Konfigurationen geschehen über das HTTP Request-Konfigurationselement (Abbildung 29.3, Fenster rechts).

Über Start|Start wird der Test dann durchgeführt.

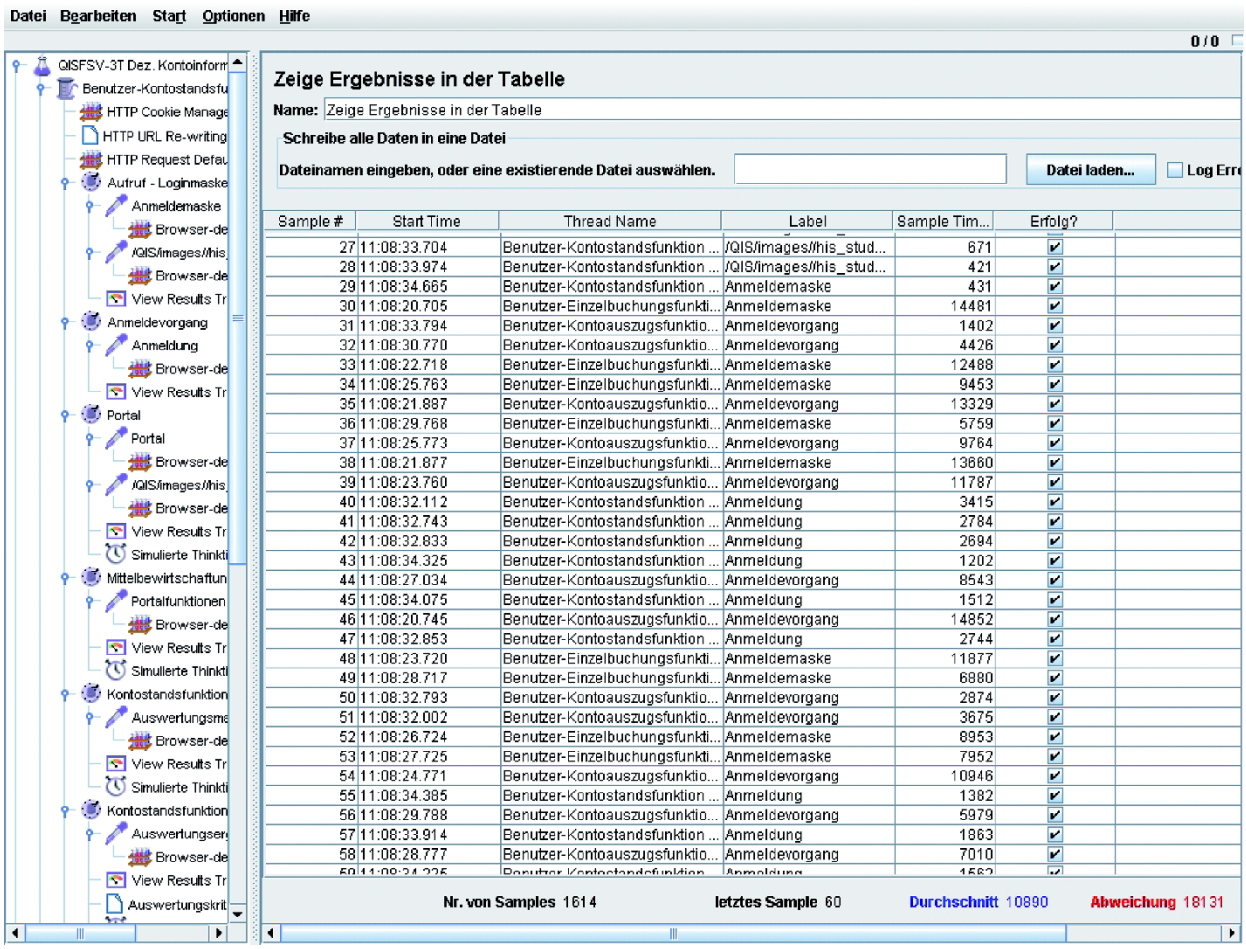



**Abbildung 29.4:** Durchführung eines Tests (hier 80 HTTP Requests) mit JMeter

Für komplexere Web-Applikationen kann JMeter auch verschiedene Sessions verwalten, um real den Zugriff von verschiedenen Clients – und insbesondere die Anmeldung mit verschiedenen Accounts – zu simulieren. JMeter bietet hierfür den wichtigen HTTP Cookie Manager.[1]

### 29.2.3 Auswertung des Tests

Nach Durchlaufen des Tests ist dieser auszuwerten; auch hierfür stellt JMeter nützliche Funktionalitäten bereit.

Über Hinzufügen|Lauscher|... sind Auswertungswerkzeuge hinzuzufügen und zu konfigurieren, wobei wieder die zwei beschriebenen Zielsetzungen von Testverfahren zu bedenken sind.

#### 29.2.3.1 Funktionale Korrektheit der Web-Applikation

Bereits die direkte Durchführung des JMeter-Tests nach Abbilundg 29.4 zeigt in der letzten Spalte, ob der Test erfolgreich war, also das erwartete Ergebnis geliefert hat. Über View Result Tree kann ein JMeter-Element verwendet werden, welches die Analyse dieser Ergebnisse, also die Analyse der Response der

[1] der Zusammenhang von Sessions und Cookies ist in Kapitel 26 beschrieben.

Web-Applikation, detaillierter ermöglicht; die Antworten können dabei in reiner Textform oder (eher einfach) als HTML mit einer integrierten Browsersicht angezeigt werden.

#### 29.2.3.2 Performance der Web-Applikation

Der Performance-Test mit JMeter verwendet eine Vielzahl gleichzeitiger Requests an die Web-Applikation, wobei über ein Timer-Element des JMeter die Abfolge der einzelnen Requests genau erklärt werden kann; dies dient wie schon erwähnt der Simulation realer Bedingungen (Benutzerverhalten).

Eine einfache Auswertung liefert der Menüpunkt Zeige Ergebnis in der Tabelle, JMeter kann aber die Performance-Ergebnisse auch grafisch auswerten. Eine derartige Auswertung, eingerichtet über GrafikHinzufügen|Lauscher|Graph Results ist in Abbildung 29.5 zu sehen, wobei neben der grafischen Information auch die in der letzten Zeile enthaltene statistische Auswertung wichtig ist.

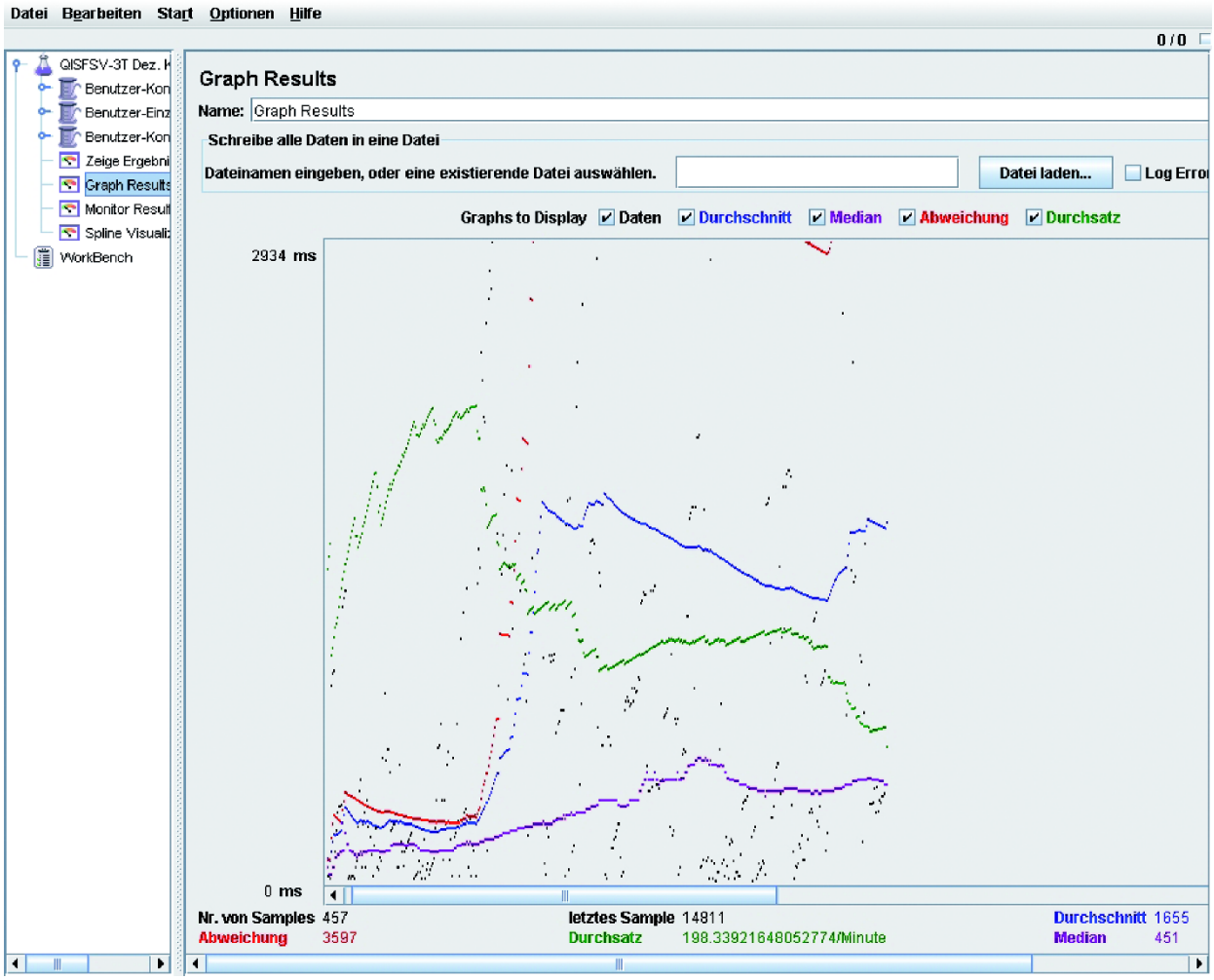



**Abbildung 29.5:**
Grafische Testanalyse mit JMeter

### 29.2.4 Die Bedeutung von JMeter

Mit JMeter liegt ein freies, und sehr effizientes Werkzeug für automatisierte Testverfahren bereit:

**JMeter ist ein gutes und praktikables Werkzeug zum Testen von Web-Applikationen.**

Ein Nachteil, zumindest zum momentanen Zeitpunkt, sei aber noch erwähnt: Die Dokumentation von JMeter ist nicht sehr umfassend, so dass der Anwender hier mitunter nur durch aktives Ausprobieren die vielen Möglichkeiten des JMeter erkunden wird – dieses ist aber eine sehr lohnende Aufgabe.

### 29.2.5 Performance mit Perl-LWP

Eine weitere Möglichkeit für einen einfachen Test einer Web-Applikation liefert Perl: In Abschnitt 10.8 wurde das LWP-Modul für Perl vorgestellt. Mit diesem lassen sich HTTP-Requests absenden, und damit eine Test-Applikation entwerfen; die Antworten können dann durch Perl einfach ausgewertet werden.

## 29.3 Typisches Ergebnis und Performance-Optimierung

Die Performance-Analyse einer Web-Applikation bringt in fast allen Fällen das gleiche, charakteristische Ergebnis: Der wesentliche Teil der Antwortzeit wird nicht durch das Netzwerk und nicht durch die Web-Applikation selbst, also die Ausführung der Programmierung, verursacht, sondern durch das Datenbanksystem.

Von daher ist es ein häufig sehr zielführender Ansatz, möglichst viele Informationen für die Web-Applikation im Speicher zu halten:

- Beim Starten der Web-Applikation erfolgt einmalig eine komplexere Abfrage der Datenbank. Hieraus baut die Anwendung speicherintern die notwendigen Daten auf und hält sie dort.

  Bei der Beantwortung von Anfragen kann auf einige, sich häufig wiederholende Datenbankzugriffe dadurch verzichtet werden.
- Ein Problem bei dieser Architektur ist die Tatsache, dass die Web-Applikation eine Änderung am Datenbestand nicht bemerkt und so zu falschen Ergebnissen führt. Als Ausweg ist in solchen Fällen die Web-Applikation neu zu starten.

Zur Verdeutlichung kann ein webbasierten Haushaltsinformationssystem dienen: Die Vielzahl der Einzelbuchungen bleibt natürlich in der Datenbank, aber alle Anfragen orientieren sich an einer vorhandenen Kontenstruktur. Diese wird beim Starten der Web-Applikation als Baumstruktur in den Speicher geladen und dort, also ohne Datenbankzugriff, ausgewertet. Dann muss nur noch sichergestellt werden, dass eine Änderung der Kontenstruktur in der Datenbank durch die Anwendung berücksichtigt wird.

Insgesamt sollte abschließend festgehalten werden:

**Testverfahren sind für die professionelle Entwicklung von Web-Applikationen ein sehr wichtiges Werkzeug und sollten intensiv genutzt werden.**

# Sicherheit im Web

Web-Anwendungen laufen naturgemäß im offenen Netz ab, damit sind die Rechnersysteme prinzpiell angreifbar. War in den noch nicht zu lange vergangenen Anfangszeiten des Internets die Gemeinde ein vertrauter und sich gegenseitig vertrauender Kreis, werden heute auf prominente Server die Angriffe in der Maßeinheit „tausend Angriffe pro Sekunde“ gemessen.

Das unerlaubte Verändern einer fremden Web-Site wird als Web-Site-Defacement bezeichnet; aus der Tatsache eines derartigen Eingriffes folgt aber meistens nicht notwendig der befürchtete Fall, dass der Angreifer Systemadministrator-Berechtigung (root-Rechte) erlangt hat. Die Site www.zone-h.org [web] gibt eine Übersicht über die aktuellen Fälle von Web-Site-Defacement; mehrere tausend Fälle pro Tag sind leider üblich.

**Für den Anbieter von webbasierten Diensten im offenen Netz ergibt sich damit zwangsweise die Notwendigkeit, dem Aspekt der Sicherheit zentrale Bedeutung beizumessen.**

Vertiefende Literatur zu diesem Thema ist vielfältig vorhanden, etwa [KS05] und [MR01].

IHS IRAN HACKERS SABOTAGE WAS HERE
Atomic energy is our right
even with threating us NO one can rule us not to use atomic power , it is our right and we ( all iranian people ) are united in this matter
we are being industrialized and being industrialized means need for more energy and this energy should come from somewhere
we want from iran government than quit NPT as soon as possible and close the UK embassy in iran where all of these problems come from

**Abbildung 30.1:** Die Site www.opensuse.org am 2.10.2005

## 30.1 Die Netzwerkstruktur

Eine typische Web-Anwendung, nehmen wir wieder unser Beispiel aus Kapitel 7, basiert auf dem sicheren Zusammenspiel mehrerer Rechner:

- dem Client, der den HTTP-Request an den Webserver sendet;
- dem Webserver, der den HTTP-Request entgegennimmt;
- dem Applikationsserver, der die Web-Programmierung ausführt;
- einem Datenbankserver.

In vielen Fällen werden Web- und Applikationsserver auf der gleichen Hardware betrieben, der Client ist immer entfernt, der Datenbankserver kann auf einer separaten Maschine, aber auch auf der gleichen, betrieben werden.

Ein grundlegender Ansatz zur Sicherheit besteht darin, nach außen nur diejenigen Server sichtbar zu machen, die notwendig sind, und das ist hier nur der Webserver.

Eine fortgeschrittene Netzstruktur besteht aus mehreren Teilnetzen, die über Switches verbunden sind. Auf jedem Switch werden zusätzlich Zugriffsregeln (Access-rules) implementiert, um zu steuern, welche Zugriffe gültig bzw. verboten sind. Alle von außen erreichbaren Server sind in einem vorgelagerten Netz, meistens als „Demilitarisierte Zone“ (DMZ) bezeichnet. Bereits vor dieser Zone findet eine Filterung nach den Netzwerkports statt; eine typische Regelung wäre etwa:

```
erlaube von allen Adressen nur Zugriffe auf Ports
   80   (http)
   443  (https)
   22   (ssh)
```

Die Ports 80 und 443 sind für den HTTP(s)-Zugang notwendig; Port 22 wird häufig für die Administration über SSH zugelassen, wie in 1.6.7.3 aber aufgeführt unbedingt ohne Kennwort, stattdessen sollte ein passphrase-geschütztes

Zertifikat verwendet werden.

In dieser DMZ muss dann notwendig der Webserver stehen. Dieser nimmt die Requests aller Clients entgegen und leitet die Parameter an einen Applikationsserver weiter, etwa an ein CGI-Programm oder an das PHP-Modul des Apache oder an eine Tomcat-Instanz. Häufig befindet sich dieser Applikationsserver – in vielen Fällen ist dies, wie wir wissen, sogar zwingend – auf der gleichen Maschine wie der Webserver, er kann aber auch auf einer anderen Maschine betrieben werden. In diesem Fall kann bereits der Applikationsserver in ein sichereres Netz verlagert werden. Die Rechner in der DMZ haben üblicherweise „echte“, also im Internet öffentlich erreichbare ip-Adressen, da sie über eine Namensauflösung mittels DNS (vgl. 1.6.6) erreichbar sein müssen.

Das eigentliche interne Netz beinhaltet häufig den Datenbankserver. Hier kommen sinnvollerweise interne, nichtroutbare ip-Adressen nach Tabelle 1.1 zum Einsatz, die beim Übergang in externe Netze über NAT (vgl. 1.6.3.3) übersetzt werden.

### 30.1.1 Der SSH-Zugang

Zahlreiche Angriffe erfolgen permanent über den *SSH-Zugang*. Sie können durch Ändern des Standard-Ports (22) für SSH reduziert werden. Nützlicher noch ist die Einschränkung des SSH-Zugangs durch eine lokale Firewall für nur wenige ausgewählte ip-Adressen, so dass der Zugriffsversuch von anderen Adressen direkt blockiert wird.

### 30.1.2 Der Datenbankserver

Eine besondere Position hat also der Datenbankserver. Da in vielen konkreten Anwendungen nur Informationssysteme implementiert werden, die eine bestehende Datenbank über das Web lesbar zur Verfügung stellen, ist auch die Alternative zu bedenken, mit einer Schattendatenbank, einer Kopie der Originaldatenbank, zu arbeiten. Dann kann diese Kopie auch in der DMZ liegen. Ein zusätzlicher Vorteil dieses Vorgehens ist, dass die Web-Zugriffe nicht das Produktionssystem im internen Netz belasten.

## 30.2 Die notwendige Konfiguration

Zahlreiche erfolgreiche Angriffe geschehen nicht aufgrund von Fehlern in der Software – etwa des Webservers –, sondern aufgrund von falscher Konfiguration der Anwendungen. Inzwischen sind zwar die standardmäßig ausgelieferten Konfigurationseinstellungen in vielen Fällen sinnvoll und auch auf Sicherheitsaspekte bedacht, dennoch gilt:

**Default-Konfigurationen sollten nie ungesehen und unüberlegt übernommen werden.**

Wir haben schon einige der zentralen Konfigurationsdateien für Web-Anwendungen kennengelernt. Die wichtigsten sind:

- die Apache-Konfiguration, üblicherweise in der Datei `httpd.conf`.

  In der unter 6.3.5 beschriebenen Konfigurationsdatei des Apache sind zahlreiche sicherheitsrelevante Einstellungen zu beachten, darunter

  - die Zugriffsrechte auf die einzelnen Verzeichnisse (vgl. Abbildung 30.2), insbesondere die Frage, wo ausführbare CGI-Anwendungen zugelassen werden. Über die Direktive `script-alias` wird hier meistens nur der Ordner `cgi-bin` samt seiner Unterordner zugelassen, der außerhalb des `document-root` des Apache liegen sollte. Außerdem sind für dieses Verzeichnis keine Leserechte notwendig.
  - Zulassen oder Verbieten von Optionen wie des Directory-Listings oder des Zugriffes auf user-Verzeichnisse;
  - auch die Ausgabe der `server-signature` sollte überdacht werden, da durch die Bekanntgabe des Webservers leicht nach spezifischen Schwachstellen gesucht werden kann; wie wir wissen, ist die Server-Version aber auch im HTTP-Header (vgl. 1.6.5) enthalten.
- die PHP-Konfiguration in der `php.ini`.
  (Vorsicht, es kann mehrere geben; über das gewohnte `phpinfo()` können Sie einfach erfahren, welche verwendet wird).
- die Tomcat-Konfiguration in der `server.xml`.

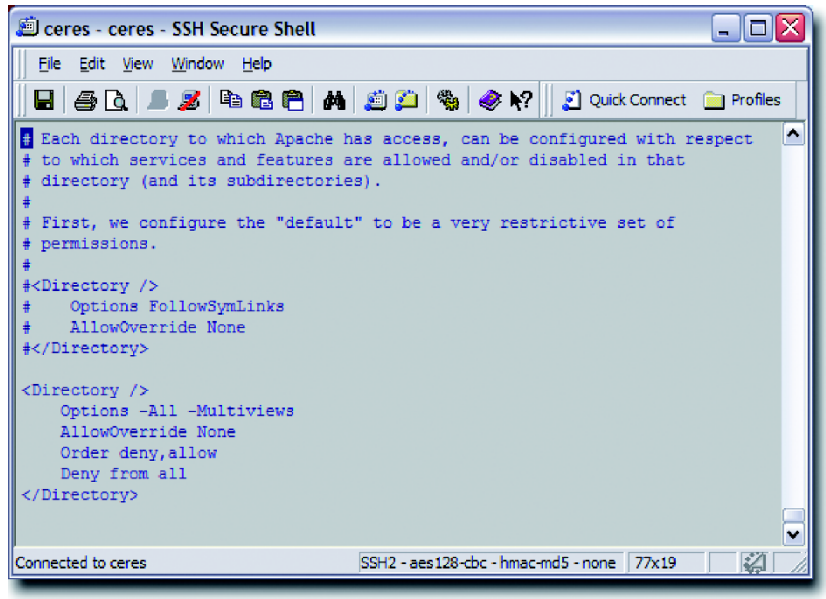



**Abbildung 30.2:** Prinzip der Rechteregelung bei Apache auf der Ebene von Verzeichnissen, Locations oder auch von einzelnen Dateien

Zur Offenheit der Konfiguration gehört die im Zusammenhang mit PHP diskutierte Einstellung der einfachen globalen Variablen, also des Systemschalters

```
register_globals
```

der `php.ini`. Erst seit der Version 4.2 wird dieser default-mäßig auf `Off` gesetzt, um ein direktes Überschreiben von Variablen durch einen HTTP-Request mit Datenteil zu verhindern.

### 30.2.1 Nur die notwendigen Dienste

Zur Konfiguration eines Rechnersystems gehört neben seiner netzwerkseitigen Integration wesentlich die Frage, welche Dienste auf diesem System laufen sollen. Auch hier gilt die zentrale Regel: weniger ist mehr.

**Nur die notwendigen Dienste sollten auch tatsächlich betrieben werden.**

Typische Beispiele für häufig unnötige Dienste auf einem Webserver sind etwa `sendmail` und das Grafiksystem `X`: beide sind beliebte Einfallslöcher und werden für Web-Anwendungen meistens nicht benötigt.[1] Dabei ist es wiederum sicherer, den jeweiligen Dienst gar nicht erst zu installieren; wird nur auf das Starten verzichtet, ist der Dienst weiter potenziell startbar, ein Restrisiko mit dem Dienst bleibt.

#### 30.2.1.1 SSL und TLS

Die meisten SSL-Implementierungen für den Apache-Webserver basieren auf OpenSSL mit dem von Ralf Engelschall bereitgestellten Apache-Modul mod_ssl (vgl. 6.3.7.3). In dieser Konstellation ist Apache in der Lage, https zu unterstützen.

Leider hat sich in der Vergangenheit mehrfach das hierbei grundlegende OpenSSL als Schwachstelle erwiesen (buffer overflow-Angriffe), weshalb man den Einsatz von SSL nur dann anbieten sollte, wenn dies auch notwendig ist. Wird SSL verwendet, ist auch hier auf ein kontinuierliches Updaten auf die aktuellste Version für das Apache-Modul, aber insbesondere für OpenSSL zu achten.

## 30.3 Die Apache-Kennung

In 6.3.2 haben wir das Prozessmodell von Apache 1.3 und 2 kennengelernt; während Apache 1.3 auf Unix für neue Requests Kindprozesse zu deren Bearbeitung startet, verwenden Apache 2 und Apache 1.3 auf Windows hierfür Threads.

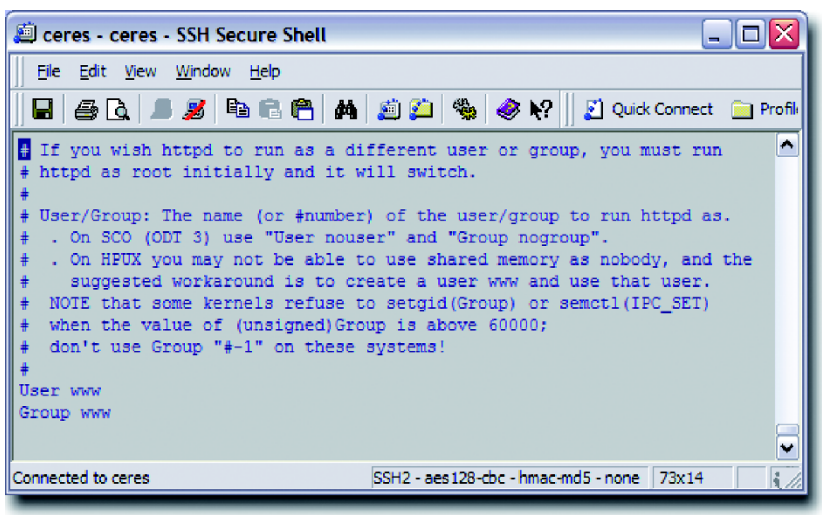



**Abbildung 30.3:** Festlegung der Benutzer- und Gruppenkennung für Apache-Prozesse auf Unix

Im ersten Block der Apache-Konfiguration kann konfiguriert werden, mit welcher *Benutzerkennung* die Kindprozesse bzw. die Threads ausgeführt werden (vgl. Abbildung 30.3). Der entsprechende Abschnitt lautet beispielsweise:

[1] es gibt aber Ausnahmen, etwa benötigen manche Java-Anwendungen zur Grafikgenerierung über awt auf Unix-Systemen doch `X`.

```
User   www
Group  www
```

Es ist dringend angeraten, diesen Mechanismus zu nutzen, um den Webserver mit einer nicht-priviligierten Kennung, die keinen Zugriff auf Systemdateien hat, auszuführen (wie hier im Beispiel www oder etwa http).

Unter Windows kann man eine derartige Absicherung erreichen, indem man unter dienste die entsprechenden Einstellungen für eine derartige Benutzerkennung trifft (Abbildungen 30.4 und 30.5).

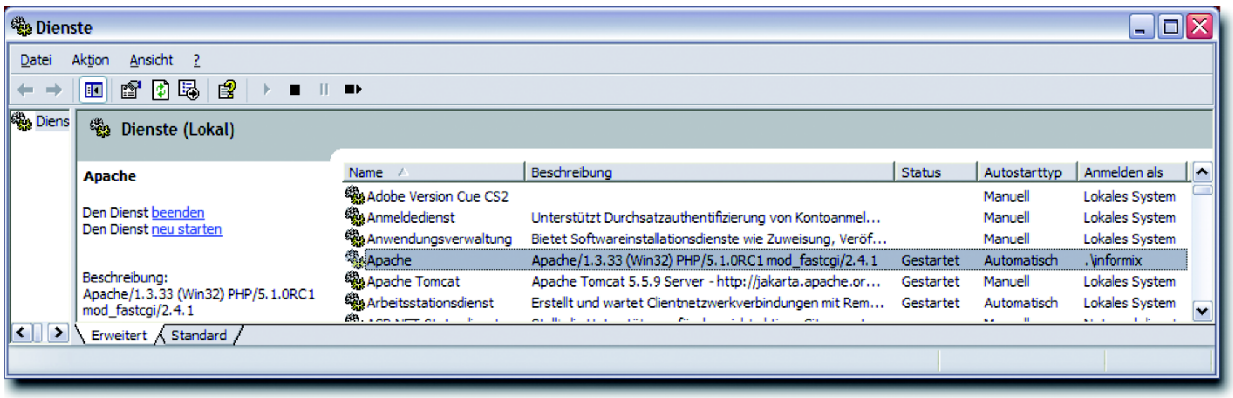



**Abbildung 30.4:** Der Apache-Dienst unter Windows, ausgeführt in einer Kennung „informix“

## 30.4 Der Highend-Angriff: DOS und DDOS

Hinter *Denial of Service* (DOS) wird das „Ausschalten“ eines Web-Dienstes durch Überschütten mit Anfragen verstanden. Dies kann mit einem einfachen Perl-Script mit LWP (vgl. 10.8) einfach realisiert werden. Verteilt sich der Angriff auf mehrere ausgehende Rechner, wird von *Distributed Denial of Service* (DDOS) gesprochen.

DDOS wird etwa von einigen Viren verwendet, die zu einem bestimmten Zeitpunkt eine Web-Site angreifen – der Lieblingsgegner ist eindeutig www.microsoft.com. Diese Angriffe sind nicht so leicht abzuwehren; am einfachsten ist eine Änderung der Adresse des Webservers kurz vor dem Angriff.

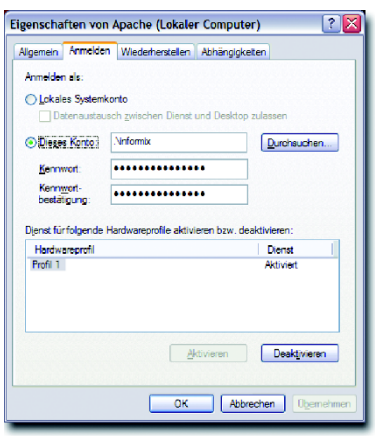



**Abbildung 30.5:** Festlegen der Apache-Benutzerkennung für win32-Dienst

## 30.5 Nicht zu viel verraten

Ein weiteres wichtiges Prinzip, um die Sicherheit zu erhöhen, besteht darin, nicht unnötig viel über den Webserver zu verraten. Dazu sind einige wenige Grundregeln wichtig.

- Besonders ungeschickt ist eine öffentlich zugängliche `phpinfo()`, da hierdurch sehr umfassend Auskunft über das Gesamtsystem gegeben wird.
- Die Apache-Module server_status und server_info geben sehr hilfreiche Übersicht über Zustand und Konfiguration des Apache (vgl. 6.3.7.1); auch diese sollten keinesfalls allgemein zugänglich sein.
- Die Qualität der Suchmaschinen, insbesondere natürlich Google, darf nicht unterschätzt werden. Um gezielt Bereiche des Webservers von der Indizierung durch Suchmaschinen auszunehmen, dient eine einfache Textdatei

`robots.txt`, welche im Startverzeichnis des Webservers („htdocs“) abgelegt wird. Hier kann konfiguriert werden, welche Suchmaschine (agent) welche Bereiche indiziert. Ein typisches Beispiel zeigt Abbildung 30.6.

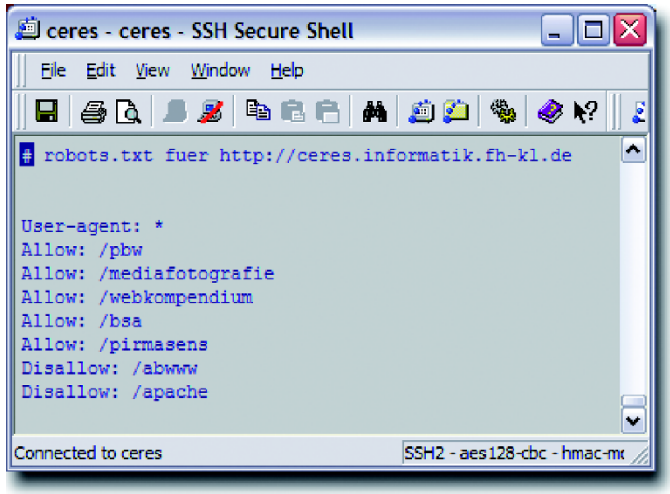



**Abbildung 30.6:**
Beispiel für `robots.txt`-Datei

## 30.6 Selbstanalyse

Wichtige Informationen über den Zustand des eigenen Servers bekommt man direkt aus der Anwendung einschlägiger Analysetools. Das älteste und bekannteste ist das Security Administrator Tool for Analyzing Networks (SATAN) [web].

Der Einsatz derartiger Hilfen ist zweischneidig, da potenzielle Angreifer natürlich ebenso damit arbeiten und Lücken eines Servers einfach erspähen können.

**Vor dem Einsatz eines derartigen Programms sollte – vergleichbar dem Einsatz eines Sniffers (vgl. 1.6.11.1) – der lokale Netzwerkadministrator informiert werden. Auch ist hier die rechtliche Situation zu beachten, da es in Deutschland Bestrebungen gibt, derartige Tools generell zu verbieten.**

### 30.6.1 Werkzeuge zur Analyse

Das bereits erwähnte Werkzeug „SATAN“ hat inzwischen zahlreiche, sehr leistungsfähige Nachfolger gefunden. Zwei davon sollen hier erwähnt werden:

- `nmap` [web] ist ein auf Unix-Systemen verbreitetes Tool für die Analyse offener Ports auf einem entfernten Rechner.
- `nessus` [web] ist ein komfortableres Analysetool, welches auch gleich Informationen und Hinweise zur Verbesserung des Systems gibt; es ist unter Unix und Mac OS-X verfügbar.

## 30.7 Sicherheit auf dem Client

Natürlich ist nicht nur der Webserver, sondern auch der Webclient bei Zugriff auf das offene Netz permanent gefährdet. Seitens des Clients können einige Regeln die Gefährdung deutlich reduzieren.

### 30.7.1 Java Applets: die Sandbox

Java Applets, die wir in Kapitel 18 kennengelernt haben, laufen standardmäßig „in ihrem eigenen Sandkasten“, der *Sandbox,* ab und können auf keine weiteren Systenmressourcen zugreifen, sind also sicher. In den Anfangstagen der Programmiersprache Java hatte dieses Konzept teilweise Lücken aufgezeigt, die aber schon lange geschlossen sind. Aufgrund der heute sehr geringen Verbreitung von Applets ist hier insgesamt das Gefahrenpotenzial klein.

Höheren Zugriff auf Systemressourcen haben signierte Applets. Hier sollten unbedingt nur wirklich vertrauenswürdige Applets akzeptiert werden. Aufgrund des (immer noch) hohen Aufwands für signierte Applets sind diese aber ausgesprochen selten.

### 30.7.2 JavaScript

Ebenso wie bei Java Applets traten bei JavaScript immer wieder Sicherheitslücken auf (vgl. Kapitel 15). Hier liegt das Problem aber in einer fehlerhaften Implementierung der Scriptsprache im ausführenden Prozess – also im Webbrowser. Der unsichere Prozess hat somit direkt die vollen Benutzerrechte des Anwenders selbst, eine Kapselung wie beim Applet findet nicht statt. Zusätzlich traten bei JavaScript deutlich mehr Lücken auf als bei den Applets, weshalb insgesamt JavaScript als unsicher(er) gilt.

Auf der anderen Seite ist heute die breite Nutzung des Webs ohne JavaScript kaum möglich. Als Ausweg ist zu empfehlen, keine veralteten Browser-Versionen zu verwenden (und damit eine möglicherweise angreifbare JavaScript-Implementierung), sondern stets auf die jeweils neueste Version zu aktualisieren.

### 30.7.3 Cookies

In Kapitel 25 wurde die Bedeutung und Arbeitsweise von Cookies vorgestellt. Aufgrund eines etwas irreführenden Artikels sind heutzutage Cookies ausgesprochen unbeliebt, was aber nicht valide mit Sicherheitsproblemen zu begründen ist.

### 30.7.4 Die Masse macht’s: der InternetExplorer

In 5.2 wurde das Problem der ungleichen Verteilung der Browser angesprochen; aufgrund der seit Jahren starken Dominanz des InternetExplorers (IE) konzentrieren sich die meisten clientseitigen Angriffe eindeutig auf diesen Browser.

Als Web-Entwickler in jedem Bereich wird man heute nicht auf den IE verzichten können, schon alleine deshalb, weil man die eigenen Seiten im IE testen muss. Abgesehen davon ist es aber sicherlich denkbar, die Verwendung des IE auf ein Minimum zu reduzieren.

## 30.8 Lokale Firewalls

Zu den vorgestellten Konzepten zur Erhöhung der Rechnersicherheit kommt als eine beliebte Variante hinzu, die involvierten Rechner mit einer *lokalen Firewall* auszustatten. Hierfür ist im Bereich Linux etwa `iptables` beliebt,

während für Windows verschiedene Produkte von Microsoft und Fremdherstellern wie Symantec, McAffee oder Kerio existieren. Zu Windows XP gehört eine sehr rudimentäre Firewall, die praktisch keine Konfigurationsmöglichkeiten beinhaltet, während Vista hier weiter ist. Insgesamt haben aber lokale Firewalls auf Windows einen schlechten Ruf.

Beim Einsatz von Firewalls gibt es das großzügige Vorgehen zu verbieten, was gefährlich ist, und das strenge Vorgehen nur zu erlauben, was sicher ist. Das erste Vorgehen wird als Black List bezeichnet, das zweite als White List. Für Webserver ist nur das Vorgehen über eine White List empfehlenswert.

### 30.8.1 Access-Lists

Firewall-Techniken verwenden, wenn sie sinnvoll konfigurierbar sind, Zugriffs-Listen, *access-Lists*. Diese regeln, welcher Zugriff von welchen Rechnern (ip-Bereichen) auf welchen Ports zulässig ist. Access-Lists können sowohl für den Zugriff auf den Rechner als auch umgekehrt für die Zugriffsmöglichkeiten des Rechners selbst nach außen verwendet werden.

Eine denkbare Liste für einen Webserver kann folgendermaßen aussehen:

```
allow from all port 80
allow from all port 443
deny all
```

Diese Listen werden stets von oben nach unten gelesen; der erste Treffer, also die erste zutreffende Regel, beendet die weitere Überprüfung. In diesem Beispiel handelt es sich somit um eine White-List, da zwei Zugriffe explizit erlaubt werden – die weltweiten Zugriffe auf den http- und den https-Port – alle anderen Zugriffe auf diesen Rechner sind verboten. Dieses Prinzip der Access-Listen ist nicht nur für die Konfiguration von Firewalls zentral, sondern wird weiter bei der Programmierung einer Vielzahl von Netzwerkkomponenten wie Switches und Router verwendet; Ciscos Betriebssystem IOS verwendet es vielfach.

Es sind, je nach eingesetzter Software, weitere Konfigurationen von Firewalls möglich, etwa zeitliche Beschränkungen. Abbildung 30.7 zeigt eine entsprechende lokale Firewall auf Windows.

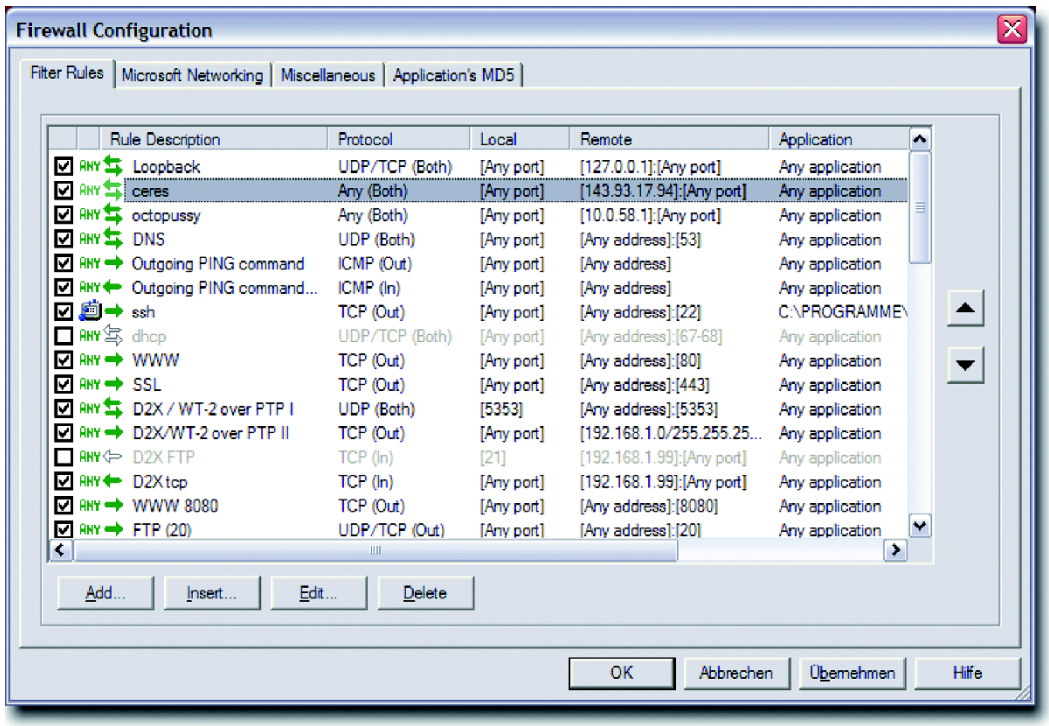



**Abbildung 30.7:** Zugriffsregeln in einer lokalen Firewall Kerio 2.1.5

## 30.9 Zusammenfassung Sicherheit

Eines ist heute sicher: Das Thema „Sicherheit und Web“ wird noch lange Zeit ein zentrales Thema der Informatik sein. Jede Nachlässigkeit hier wird schnell und hart bestraft.

### 30.9.1 Immer „up to date“

Eine Kernregel gilt im Zusammenhang mit der Sicherheit immer:

**Rechnersysteme mit Verbindung zum offenen Netz sollten unbedingt auf dem aktuellen Softwarestand gehalten werden. Das betrifft gleichermaßen das Betriebssystem, den Webserver und alle Applikationssoftware wie Java oder Perl und den Webbrowser.**

### 30.9.2 Information

Nicht nur die Software auf den Rechnersystemen, auch der Wissensstand der beteiligten Personen, insbesondere des Systemadministrators, muss stets auf dem neuesten Stand gehalten werden. Dazu gehört die kontinuierliche Partizipation an Security-Foren, um schnell auf neue Lücken reagieren zu können.

### 30.9.3 Aufwand beachten

Das Problem Sicherheit für den Webserver muss, wie wir gesehen haben, auf verschiedenen Ebenen berücksichtigt werden. Wichtig für den Betreiber eines Webservers ist dabei stets zu beachten:

**Sicherheit für einen Webserver gibt es nicht umsonst, der Betrieb eines Servers ist stets mit einem nicht unerheblichen Aufwand verbunden.**

# 31

# Quo vadis? Web 2.0 und die weitere Entwicklung

## 31.1 Die Bedeutung der einzelnen Techniken

Die Gardner Group hat die Relevanz neuer Techniken in der Informatik analysiert, und kam dabei unabhängig von der jeweiligen Technik zu einem ähnlichen Ergebnis, das in Abbildung 31.1 zu sehen ist.

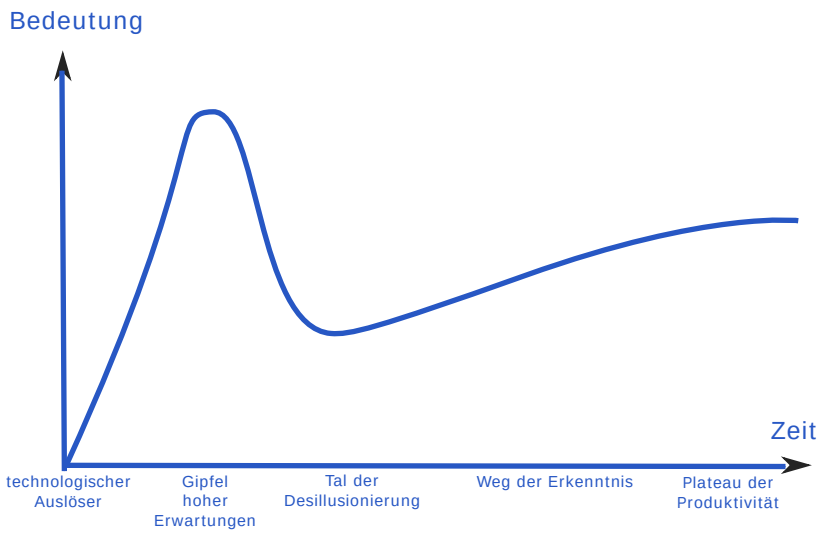



**Abbildung 31.1:** Typische Marktrelevanz neuer Techniken

Die in diesem Buch vorgestellten Techniken befinden sich aktuell natürlich auf ganz verschiedenen Punkten dieser Kurve: Ajax und Ruby on Rails sind gerade in der ersten Euphoriephase, das Applet ist im „Tal der Ernüchterung“ verschwunden und PHP, Perl/CGI sowie serverseitiges Java haben den Übergang zum kritischen Produktionszustand geschafft.

Neue Techniken bereiten sich laufend vor, den Euphorie-Gipfel zu erstürmen. Hierzu zählen der Versuch des neuen Java6, Scriptsprachen zu integrieren, und die neue Scriptsprache Groovy.

Momentan ist kein Bereich der Informatik derart dynamisch und in Bewegung wie die Programmierung für das Internet; einige der hier vorgestellten Techniken sind gerade seit drei Jahren verfügbar, andere sind schon lange Klassiker der Softwaretechnik. Dabei werden noch viele eine Entwicklung nach Abbildung 31.1 erleben – und einige werden auch das „Tal der Ernüchterung“ dabei nicht mehr verlassen.

Sicher, ganz sicher scheint heute zu sein, dass die Bedeutung „des Web“ noch wesentlich weiter steigen wird, und dass neben einzelnen Techniken die Sicherheit eine immer wichtigere Rolle spielt. Für künftige Web-Applikationen wird dabei etwa die Authentifizierung sehr wichtig werden, die digitale Identität. Viele weitere Themen werden die künftige Entwicklung der Web-Programmierung noch prägen – es wird weiter ein hochaktuelles und spannendes Thema bleiben.

## 31.2 Web 2.0

Ein Begriff soll hier noch separat behandelt werden, der momentan eine große Popularität, ganz auf dem ersten Gipfel nach Abbildung 31.1, erlebt: *Web 2.0.*

Wie schnell die Entwicklung von Technologietrends heute ist, erkennt man auch daran dass Web 2.0 erst im Oktober/November 2005 vorgestellt wurde, bereits heute aber schon die Diskussion intensiv prägt. Der Begriff geht auf Dale Dougherty und Craig Cline zurück und steht nicht für eine einzelne Technik, sondern für einen Paradigmenwechsel im Umgang mit dem Web.

Dougherty und Cline sehen das klassische „Web 1.0“ als Medium des Veröffentlichens, der Kommunikation in eine Richtung. Web 2.0 hingegen steht für „teilnehmen“, der Besucher gestaltet das Medium mit. Beispiele für Web 2.0 sind also interaktive Web-Sites, etwa die Wikipedia oder das Bildportal Flickr. Web 2.0-Angebote können mit vielen der hier vorgestellten Techniken realisiert werden, eine zentrale Bedeutung wird aber Ajax zukommen, da hier die Möglichkeit der verteilten Client-Server-Applikation am sinnvollsten zu realisieren ist.

Abbildung 31.2 zeigt einige der wichtigsten Begriffe um das Web 2.0 in einer ebenfalls zum Web 2.0 zählenden Darstellungsweise, der „Wolke“: die Position und Größe der Begriffe verdeutlicht ihre Beziehung zueinander und ihre Relevanz.

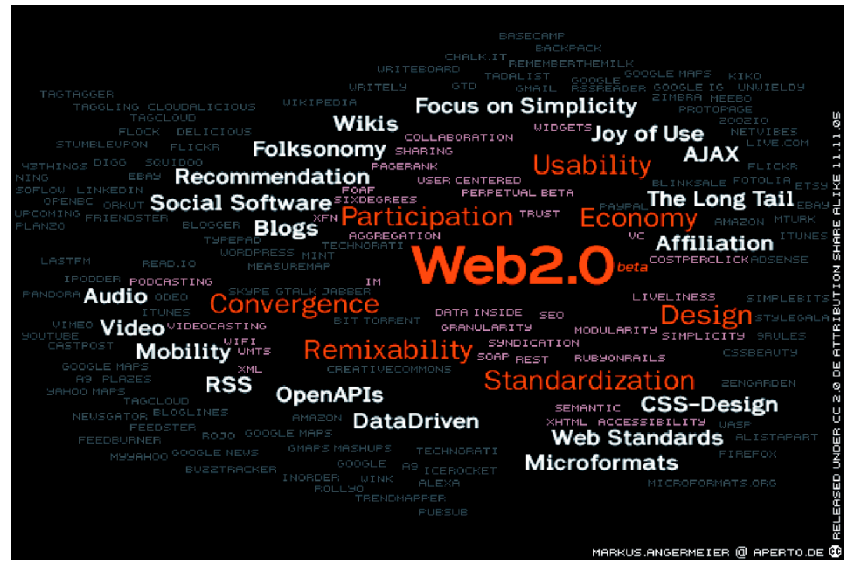



**Abbildung 31.2:**
Typische Begriffe von Web 2.0 (nach Markus Angermeier, November 2005)

Allerdings setzt schon rasch auch hier gewisse Skepsis ein; tatsächlich ist Web 2.0 – wie viele hier vorgestellten Themen – nicht grundsätzlich neu, sondern nur eine neue Betrachtungsweise bestehender Ansätze.

**Abbildung 31.3:**
Spiegel Online: „Blase 2.0“

Einen Schritt weiter zu größerer Komplexität, insbesondere zur maschinellen semantischen Verarbeitung der Informationen im Web geht der von Tim Berners-Lee eingeführte Begriff Semantic Web, siehe etwa [PB06]. In diesem Zusammenhang werden teilweise bereits Begriffe wie „Web 3.0“ verwendet. Die weitere Entwicklung wird zeigen, ob auf diesen Ansätzen tatsächlich ein neuer Internet-Boom basieren wird, oder ob es nur der marketingmäßig aufbereitete normale technische Fortschritt ist (vgl. Abbildung 31.3).

# Persönliche Worte

*Und wandelt mit bedächt'ger Schnelle*
*Vom Himmel durch die Welt zur Hölle.*

*Goethe, Faust I*

*Warum schreibt man so ein Buch?*

Lieber Leser,

diese Frage habe ich mir in den letzten Wochen und Monaten öfters, ja sogar oft, gestellt. Einfache Antworten wie „materielle Gründe“ scheiden aus, denn auch bei bestem Erfolg liegt der effektive Stundenlohn weit unterhalb aller gesetzlichen Phantasien

Bleiben ideelle Werte.

Zum einen die Hoffnung und der Wunsch, Anwendern und Studierenden etwas nützliches zu vermitteln; das ist im Kern mein Beruf, und das liegt mir sehr am Herzen – und ist im Einklang mit Web 2.0 und mit „Social Web“ die Zukunft unserer Gesellschaft.

Zum anderen war die Arbeit an diesem Buch nicht nur eine Belastung, sondern auch eine Freude. Ausgesprochen gut war von Anfang an die Zusammenarbeit mit dem Springer-Verlag. Mit Hermann Engesser und Dorothea Glaunsinger in Heidelberg verbindet mich inzwischen keine geschäftliche, sondern eine freundschaftliche Beziehung.
Auf der TEX-nischen Seite war auch dieses Buch eine kleine Herausforderung, und hier war Frank Holzwarth bei Springer ein an Kompetenz nicht zu überbietender Partner, der mich sehr unterstützt hat.

Auch an der Hochschule haben viele ihren Anteil am Zustandekommen dieses Werks. An erster Stelle möchte ich Torsten Kockler danken, dem neben vielen grafischen Elementen dieses Buches auch inhaltlich wichtige Verbesserungen zu verdanken sind. Ebenso haben viele Studierende und Mitarbeiter der Hochschule dieses Werk wesentlich verbessert, und Eure Unterstützung war einer der schönsten Aspekte. Alle kann ich hier nicht aufzählen, aber Marcel Remmy und Christian Schommer sind unbedingt für ihren besonderen Einsatz zu erwähnen.

Das schönste beim Erstellen dieses Buches war aber die unfassbare Unterstützung durch den Menschen an meiner Seite; bedeutet der Entschluss, eine derartige Aufgabe anzugehen, zunächst einen Verzicht und eine Belastung der Partnerschaft, war Deine unvergleichlich kompetente, wichtige und tiefgründige Unterstützung für unser Projekt das schönste, was dabei eintreten konnte.

*Deine Liebe ist die größte Kraft meines Lebens.*

# A

# Internetlinks

Für die Web-Programmierung stehen in großer Vielzahl sehr nützliche und aktuelle Ressourcen im Internet bereit. Einige sehr wichtige sind hier aufgeführt.

Unter

- `http://www.webkompendium.de`

ist eine direkt verlinkte, umfassende und aktuelle Liste der Links zum Thema dieses Buches zu finden.

## A.1 Zu Kapitel 1

Allgemeines

- `http://www.dfn.de` DFN-Verein
- `http://www.netcraft.com` Netcraft
- `http://de.selfhtml.org` Selfhtml

tcp/ip

- `http://www.ieee.org`

## A.2 Zu Kapitel 2

HTML

- `http://www.w3.org` W3C-Site
- `http://de.selfhtml.org` das auf Stefan Münz zurückgehende Selfhtml
- `http://tidy.sourceforge.net` HTML Tidy
- `http://validator.w3c.org` HTML Validator

CSS

- `http://www.w3.org/Style/CSS` CSS Definition
- `http://de.selfhtml.org/css` CSS Erläuterungen
- `http://www.css4you.de` CSS Erläuterungen und Beispiele

Barrierefreiheit

- `http://http://www.w3.org/WAI` WAI-Site
- `http://www.bitvtest.de` BITV-Test

## A.3 Zu Kapitel 3

Betriebssysteme:

- http://www.netcraft.com
- http://www.sun.de
- http://www.debian.de Debian-Site
- http://www.gentoo.de Gentoo-Site
- http://www.ubuntuusers.de

Datenbankmanagementsysteme:

- http://www.mysql.com
- http://www.postgresql.org
- http://www.sqlite.org
- http://hsqldb.org
- http://db.apache.org/derby

XAMPP

- http://www.apachefriends.org/de/xampp.html

## A.4 Zu Kapitel 4

- http://www.stack.nl/~dimitri/doxygen/index.html Doxygen

## A.5 Zu Kapitel 5

- http://www.cstr.ed.ac.uk/projects/festival Festival Screen Reader
- http://www.webstandards.org/action/acid2 Browsertest ACID2

## A.6 Zu Kapitel 6

Apache

- http://www.apache.org

Auswertung der Logfiles

- http://domainunion.de/mirrors/webalizer.com
- http://awstats.sourceforge.net

## A.7 Zu Kapitel 8

Eclipse

- http://www.eclipse.org Site des Eclipse-Projektes
- http://www.sqlexplorer.org

Firebug

- http://www.getfirebug.com

Datenbank-Tools

- http://www.minq.se/products/dbvis DbVisualizer
- http://www.phpmyadmin.net phpMyAdmin

## A.8 Zu Kapitel 9

- http://hoohoo.ncsa.uiuc.edu/cgi

## A.9 Zu Kapitel 10

Perl

- http://www.perl.org offizielle Seite von Larry Wall
- http://www.perl.com Perl-Site des O'Reilly-Verlags
- http://www.cpan.org Comprehensive Perl Archive Network
- http://www.activestate.com Perl-Distributionen für verschiedene Betriebssysteme
- http://e-p-i-c.sourceforge.net das EPIC-Plugin für Perl (10.3.4)
- http://www.informatik.uni-freiburg.de/~diesch/perl.html

CGI-Modul

- http://stein.cshl.org/WWW/software/CGI Informationen von Lincoln Stein zum CGI-Modul
- http://search.cpan.org/dist/CGI.pm CGI-Modul bei CPAN

## A.10 Zu Kapitel 11

- http://www.php.net offizielle PHP-Site
- http://www.zend.com Site von Zend
- http://php-faq.de praktische FAQ zum aktuellen PHP
- http://www.phpdoc.de Informationen zu phpdoc
- http://www.phpeclipse.de PHP-Plugin für Eclipse
- http://de2.php.net/manual/de/migration5.php Migration PHP Version 4 zu Version 5
- http://de.php.net/pdo PHP Data Objects (PDO)

## A.11 Zu Kapitel 12

- http://www.python.org Python-Hauptseite u.a. mit Downloads und Online-Dokumentation
- http://pydev.sf.net und http://pydev.sourceforge.net Entwicklungsumgebung: empfehlenbswertes Eclipse Plugin
- http://www.python.org/doc/current/modindex.html Übersicht über die Python-Standardmodule
- http://sourceforge.net/projects/mysql-python MySQL-Modul für Python
- http://www.jython.org/Project/index.html das Jython-Projekt
- http://zope.org
- http://plone.org
- http://www.djangoproject.com
- http://www.modpython.org

## A.12 Zu Kapitel 13

- http://www.ruby-lang.org Ruby-Hauptseite
- http://www.ruby-doc.org Ruby-Dokumentationssammlung
- http://ww.rubywiki.de ein Wiki rund um Ruby
- http://www.rubycenteal.com zahlreiche Informationen und Ressourcen zu Ruby
- http://rubyeclipse.sourceforge.net Ruby-Plugin „rubyeclipse“ für die IDE Eclipse
- http://modruby.net das mod_ruby-Apache-Modul
- http://www.eruby.org Informationen zu eRuby
- http://rubyforge.org/projects/rubyforapache Ruby-Apache-Pakete insbesondere für Windows mit Installer
- http://www.tmtm.org/en/mysql/ruby Ruby-MySQL-Modul
- http://rubyforge.org/projects/ruby-dbi/ Ruby-DBI
- http://jruby.codehaus.org das JRuby-Projekt

## A.13 Zu Kapitel 15

- http://www.ecma-international.org/publications/standards/Ecma-262.htm ECMA-262
- http://developer.mozilla.org/en/docs/New_in_JavaScript_1.7 JavaScript-Version 1.7
- http://labs.adobe.com/technologies/jseclipse JavaScript-Plugin für Eclipse von Adobe/Macromedia
- http://json.org json
- http://www.w3.org/TR/DOM-Level-3-Core DOM Level 3

## A.14 Zu Kapitel 16

- http://www.adaptivepath.com/publications/essays/archives/000385.php Initialartikel zu Ajax (Abbildung 16.1)
- http://www.eclipse.org/atf Ajax-Plugin für Eclipse
- http://dojotoolkit.org das dojo-Toolkit

## A.15 Zu Kapitel 17

- http://de3.php.net/ming Ming-Bibliothek für PHP
- http://gnash.lulu.com gnash (GNU Flash)

## A.16 Zu Kapitel 19

- http://www.fastcgi.com

## A.17 Zu Kapitel 20

- http://pear.php.net die PEAR-Site
- http://pecl.php.net die PECL-Site
- http://oss.backendmedia.com/index.php?area=MDB2

## A.18 Zu Kapitel 21

- http://smarty.php.net Smarty-Site

## A.19 Zu Kapitel 22

- http://www.djangoproject.com django Site
- http://www.djangobook.com django Online-Dokumentation

## A.20 Zu Kapitel 23

- http://www.rubyonrails.org Rails Site
- http://api.rubyonrails.org Rails-API
- http://www.radrails.org das radrails-Plugin für Eclipse

## A.21 Zu Kapitel 24

- http://java.sun.com/javaee J2EE-Site bei Sun
- http://java.sun.com/products/servlet/index.jsp Java Servlets
- http://tomcat.apache.org Tomcat-Server
- http://www.eclipsetotale.com/tomcatPlugin.html Sysdeo-Plugin
- http://java.sun.com/javase/technologies/database/index.jsp
- http://java.sun.com/products/jndi
- http://java.sun.com/products/jsp/index.jsp

## A.22 Zu Kapitel 27

- http://www.iana.org/assignments/media-types Übersicht über die MIME-Typen

## A.23 Zu Kapitel 28

- http://www.cmsmatrix.org detaillierter Vergleich „aller“ CMS
- http://www.joomla.org das CMS Joomla!
- http://plone.org das CMS Plone
- http://www.infopark.de/products/1cms/index.html das kommerzielle CMS Fiona von Infopark
- http://www.ilias.de/ios das Learning Management System ILIAS
- http://www.typo3.org das CMS TYPO3
- http://www.typo3.com TYPO3 „für Entscheider“
- http://typo3.com/Geschichte.1268.0.html?&L=2 Geschichte von TYPO3
- http://typo3.org/extensions TYPO3-Extensions einschließlich Suchmöglichkeit
- http://www.typo3.net/tsref TypoScript-Referenz

## A.24 Zu Kapitel 29

- http://jakarta.apache.org/jmeter Apache JMeter

## A.25 Zu Kapitel 30

Allgemeine Informationen

- http://staff.washington.edu/dittrich/misc/satan/
- http://www.netcraft.org
- http://www.frsirt.com
- http://www.zone-h.org
- http://www.bsi.de

Analysewerkzeuge

- http://www.insecure.org/nmap
- http://www.nessus.org

# B

# Abkürzungen

Die moderne Informatik, und im besonderen Maß die Internet-Technologie, verwendet eine Vielzahl von Abkürzungen. Die wichtigsten werden hier mit einem Bezug zu den relevanten Passagen im Buch, aufgelöst.

**Ajax**: Asynchronous JavaScript and XML (16)
**ARPANET**: Advanced Research Projects Agency Network (1.1)
**ASP**: Active Server Pages
**AWT**: Abstract Window Toolkit
**BITV**: Barrierefreie Informationstechnik Verordnung (2.6)
**CGI**: Common Gateway Interface (9)
**CMS**: Content Management System
**CPAN**: Comprehensive Perl Archive Network
**CSS**: Cascading Stylesheets (2.5)
**CVS**: Concurrent Versions System
**DBD**: Database Driver (10.7)
**DBI**: Database Independent Interface (10.7)
**DBMS**: Database Management System
**DDOS**: Distributed Denial of Service (30.4)
**DFN**: Verein zur Förderung eines Deutschen Forschungsnetzes e. V. (1.1)
**DHCP**: Dynamic Host Configuration Protocol
**DNS**: Domain Name Service (1.6.6)
**DOM**: Document Object Model (15.5)
**DOS**: Denial of Service (30.4)
**DTD**: Document Type Definition (2.4.4)
**ECMA**: European Computer Manufacturers Association (15)
**EJB**: Enterprise JavaBeans
**ERP**: Enterprise Resource Planning
**favicon**: Favorites Icon
**FOP**: Formatting Objects Processor (27.4.2.1)
**FTP**: File Transfer Protocol (1.6.7.2)
**HTML**: Hypertext Markup Language (2.3)
**HTTP**: Hypertext Transfer Protocol (1.6.5)
**HTTPS**: HyperText Transfer Protocol Secure
**IDE**: Integrated Development Environment (8.1)
**ip**: Internet Protocol (1.6.3)
**ISO**: International Organization for Standardization (1.6.1)
**jar**: Java Archive
**JDBC**: Java Database Connectivity
**JDK**: Java Development Kit

**JNDI**: Java Naming and Directory Interface (24.3.1.1)
**JSF**: JavaServer Faces (24.5.2)
**JSON**: JavaScript Object Notation (15.8)
**JVM**: Java Virtual Machine
**LAMP**: Linux-Apache-MySQL-PHP
**LAN**: Local Area Network (1.6.3)
**LDAP**: Lightweight Directory Access Protocol
**MIME** Multipurpose Internet Mail Extensions (27.1)
**MVC**: Model-View-Controller (11.6.2)
**NAT**: Network Address Translation (1.6.3)
**ODBC**: Open Database Connectivity
**OO**: Objektorientierung (11.6.1)
**OSI**: Open Systems Interconnection Reference Model (1.6.1)
**PAT**: Port Address Translation (1.6.3.3)
**pdf**: Portable Document Format (27.4)
**PDO**: PHP Data Objects (11.8)
**PEAR**: PHP Extension and Application Repository (20)
**PECL**: PHP Extension Code Library 20
**RFC**: Request for Comments
**SGML**: Standard Generalized Markup Language (2.2)
**SMTP**: Simple Mail Transfer Protocol (1.6.8)
**SOA**: serviceorientierte Architekturen
**SSH**: Secure Shell (1.6.7.3)
**SSL**: Secure Sockets Layer
**SVG**: Scalable Vector Graphics
**swf**: Small Web Format
**tcp**: Transfer Control Protocol (1.6.2)
**TLS**: Transport Layer Security
**UAC**: User Account Control (3.2.3)
**udp**: User Datagram Protocol
**UML**: Unified Modeling Language
**URI**: Uniform Resource Identifier (1.6.9)
**URL**: Uniform Resource Locator (1.6.9)
**URN**: Uniform Resource Names (1.6.9)
**utf-8**: 8-bit Unicode Transformation Format (2.3.5)
**WAI**: Web Accessibility Initiative (2.6)
**WAN**: Wide Area Network (1.6.3)
**war**: Web Application Archive
**WebDAV**: Web-based Distributed Authoring and Versioning (1.6.5.2)
**WWW**: World Wide Web (1.1)
**XML**: Extensible Markup Language (2.4)
**XSS**: Cross-Site-Scripting

# Literatur


[AKW98] ACHTNER, WOLFGANG, STEFAN KUNZ und THOMAS WALTER: *Dimensionen der Zeit*. Wissenschaftliche Buchgesellschaft, zugleich Primus-Verlag, Darmstadt, 1998.

[Arn06] ARNDT, HENRIK: *Integrierte Informationsarchitektur*. X.media.press. Springer, Berlin u.a., 2006.

[BBS04] BÖHRINGER, JOACHIM, PETER BÜHLER und PATRICK SCHLAICH: *Projekte zur Mediengestaltung*. X.media.press. Springer, Berlin u.a., 2004.

[BLL06] BRAUNE, KLAUS, JOACHIM LAMMARSCH und MARION LAMMARSCH: *LaTeX– Basissystem, Layout, Formelsatz*. X.systems.press. Springer, Berlin u.a., 2006.

[BRJ05] BOOCH, GRADY, JAMES RUMBAUGH und IVAR JACOBSON: *The Unified Modeling Language – User Guide*. Addison-Wesley, Reading, Mass., 2. Auflage, 2005.

[DB00] DESCARTES, ALLIGATOR und TIM BUNCE: *Programming the Perl DBI – Database Programming with Perl*. O'Reilly, Beijing u.a., 2000.

[Deh01] DEHNHARDT, WOLFGANG: *Scriptsprachen für dynamische Webauftritte*. Hanser, München u.a., 2001.

[Deu05] DEUTZ, ROBERT: *Mambo*. Xpert.press. Springer, Berlin u.a., 2005.

[Fla06] FLANAGAN, DAVID: *JavaScript – The Definitive Guide*. O'Reilly, Beijing u.a., 5. Auflage, 2006.

[Fue05] FUECKS, HARRY: *PHP 5 für Fortgeschrittene*. dpunkt, Heidelberg, 2005.

[GHJ04] GAMMA, ERICH, RICHARD HELM und RALPH E. JOHNSON: *Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley, München u.a., 2004.

[Hal00] HALL, MARTY: *Core Servlets and Java Server Pages*. Prentice Hall, Upper Saddle River, 2000.

[Häß03] HÄSSLER, ULRIKE: *Cascading Stylesheets*. X.media.press. Springer, Berlin u.a., 2003.

[Höp07] HÖPPEL, RALF ROBERT: *Der Ruby-Atlas*. Xpert.press. Springer, Berlin u.a., 2007.

[Kil02] KILLELEA, PATRICK: *Web Performance Tuning*. O'Reilly, Beijing u.a., 2. Auflage, 2002.

[Kra05] KRAUSE, JÖRG: *PHP 5, Grundlagen und Profiwissen*. Hanser, München u.a., 2. Auflage, 2005.

[KS05] KRIHA, WALTER und ROLAND SCHMITZ: *Internet-Security aus Software-Sicht*. Xpert.press. Springer, Berlin u.a., 2005.

[Lut06] LUTZ, MARK: *Programming Python*. O'Reilly, Beijing u.a., 3. Auflage, 2006.

[MR01] MÜLLER, GÜNTHER und MARTIN REICHENBACH (Herausgeber): *Sicherheitskonzepte für das Internet*, Xpert.press, Berlin u.a., 2001. 5. Berliner Kolloquium der Gottlieb Daimler- und Karl Benz-Stiftung.

[MS04] MEINEL, CHRISOPH und HARALD SACK: *WWW – Kommunikation, Internetworking, Web-Technologien*. Xpert.press. Springer, Berlin u.a., 2004.

[PB06] PELLEGRINI, TASSILO und ANDREAS BLUMAUER: *Semantic Web*. X.media.press. Springer, Berlin u.a., 2006.

[PI99] PATWARDHAN, NATE und CLAY IRVING: *Programmieren mit Perl Modulen*. O'Reilly, Beijing u.a., 1999.

[PR07] PLAG, FLORIAN und ROLAND RIEMPP: *Interaktives Video im Internet mit Flash*. X.media.press. Springer, Berlin u.a., 2007.

[Roj98] ROJAS, RAÚL: *Die Rechenmaschinen von Konrad Zuse*. Springer u.a., Berlin, 1998.

[SB06] STÖCKL, ANDREAS und FRANK BONGERS: *Einstieg in TYPO3 4.0*. Galileo Press, Bonn, 2. Auflage, 2006.

[Sto04] STOYAN, ROBERT: *Management von Webprojekten*. Springer, Berlin u.a., 2004.

[Wal03] WALTER, THOMAS: *Grundlagen der Infomatik*. Hanser, München u.a., 2003.

[Wal04] WALDRAFF, THOMAS: *Digitale Bildauflösung*. X.media.press. Springer, Berlin u.a., 2004.

[Wal05] WALTER, THOMAS: *MediaFotografie – analog & digital*. X.media.press. Springer, Berlin u.a., 2005.

[WCS00] WALL, LARRY, TOM CHRISTIANSEN und RANDAL L. SCHWARTZ: *Programming Perl*. O'Reilly, Beijing u.a., 3. Auflage, 2000.

[Wen07] WENZ, CHRISTIAN: *JavaScript und Ajax*. Galileo Press, Bonn, 7. Auflage, 2007.

[Zie02] ZIEGLER, JOACHIM: *Programmieren lernen mit Perl*. Xpert.press. Springer, Berlin u.a., 2002.

# Personenverzeichnis


Andreessen, Marc 5
Angermeier, Markus 588

Bakken, Stig 427
Baran, Paul 3
Beckers, Rob 18
Berners-Lee, Tim 3, 81, 589
Boutell, Thomas 548

Cline, Craig 588
Combs, Gerald 22

Davies, Donald Watts 3
Dougherty, Dale 588
DuBois, Paul 333

Eich, Brendan 348
Engelschall, Ralf 91, 579

Garrett, Jesse J. 387
Gauß, Carl-Friedrich 137
Goethe, Johann Wolfgang von V, 591
Gosling, James 8
Gutmans, Andi 190

Hader, Oliver 554
Hammond, Mark 258
Hansson, David Heinemeier 463
Heesch, Dimitri van 61
Hugonin, John 293

Lemos, Manuel 434
Lerdorf, Rasmus 190

Masahiro, Tomita 331
Matsumoto, Yukihiro 297
McCool, Rob 81
Münz, Stefan 26, 593

Ohrt, Monte 436

Raggett, Dave 31
Reenskaug, Trygve 58
Rossum, Guido van 257

Skårhø, Kasper 554
Stein, Lincoln 165
Suraski, Zeev 190

Tanenbaum, Andrew 57
Torvalds, Linus 46
Trubetskoy, Gregory 295

Wall, Larry 125

Zmievski, Andrei 190, 436
Zuse, Konrad 409

# Sachverzeichnis


127.0.0.1 10

Abkürzungen 599
Access-Lists 583
ACID2 68
ACID2 Browsertest 69
Acrobat 549
ActionScript 401, 406
AdaptivePath 387
Adobe 548, 549
Adobe GoLive 545
Ajax 387
  Ablauf 390
  ATF 389
  beenden 396
  Beispiel 388, 391
  Eclipse 389
  Grundlage 389
  Parameterübergabe 393
  Parameterübergabe POST 395
  Probleme 400
  TYPO3 554
  Web 2.0 588
  XMLHttpRequest 389, 399
Amazon 7, 519
Apache 81
  Architektur 84
  Benutzerkennung 579
  File-System 86
  Installation 84, 191
  Installation Unix 191
  Installation Windows 191
  Konfiguration 87
  Konfiguration für PHP 193
  Logfiles 91
    access.log 92
    error.log 92
  mod_jk 495
  mod_python 295, 447
  mod_ruby 328
  mod_so 91
  mod_user_track 523
  Modulkonzept 90
  Rails 473
  server_info 580
  server_status 580
  UTF-8 32
  Versionen 82
apache2.conf 87
Applet 409, 582
  Architektur 410
  Einbinden 411
  Idee 409
  Java-Klassen 412
  Probleme 414
  Sandbox 410
  Sicherheit 410
  signiert 411, 582
Aptana 468
ARPANET 3
assoziatives Array 134
Audio 401
Auszeichnungssprachen 25
awstat 92

Barrierefreiheit 42
  BITV 42
  WAI 42
Behindertengleichstellungsgesetz 43
Beispielanwendung 93
  Neueintrag 99, 535
  SQL-Syntax 98
BITV 43
Browser 5, 63
  Firefox 65
  InternetExplorer 64
  Lynx 67
  Opera 66
  Safari 66
  Verteilung 63
Browserweiche 371

CERN-Webserver 5
CGI 113
  Beispiele 115
  C 115
  Einfaches mit Perl 154
  Fehler 160
  Java 116
  mit Java und Formular 120
  Prinzip 114
  Python 280
  QUERY_STRING 115
  Rechte 160
  Ruby 322
  Shebang 154
  Umgebungsvariablen 114
  Unterschied zu fastCGI 417
CGI (Perl-Modul) 165
Cisco IOS 90, 583
CMS
  ILIAS 553
  Joomla 553
  Mambo 553
  NPS 553
  Plone 553
  TYPO3 553, 554
  Vorteile 551
cmsmatrix.org 552
Cocoon 447
Compiler-Flags 85
Content Management Systeme 551
Controller 58
Cookie 519, 582
  Arbeitsweise 522
  Browser 519
  Datenstruktur 523
  Expires 523
  Gültigkeit 523
  PHP 524
  Security 523
  setcookie 526
Cookies
  PHP 524
CPAN 162

Cross Media Publishing 551
CSS 38, 58, 154
  Beispiel WebKompendium 41
  Browser 39
  case-sensitive 40
  CSS-Datei 39
  Grundsyntax 40
  Media-Typ 40
  Selektor 40
  Versionen 39
CVS 101

Dateiformate
  für Web 547
Datenbank-Tools 109
Datenbankserver 48, 577
DBI 173
DbVisualizer 109
DDOS 580
Debian 47
Defacement 575
Derby 50
Deutschlandfunk 401
DFN 3
DHCP 20
DHTML 6, 347
Direktive 87
django 447
  Anwendung 452
  Beispiel 450
  Betrieb 447
  Installation 448
  Mapping 456
  Model 453
  Projekt 450
  Server 450
  Template 457
  View 456
DNS 577
DocumentRoot 89
dojo 390
Dokumentation 59
DOM 368
DOS 580
Doxygen 61, 201
Dreamweaver 105, 197, 382
DTD 35
  Syntax 35
Dublin Core 97

Ebay 7
Eclipse 101
  Ajax 389
  Aptana 468
  Ausführen CGI 160
  Callisto 103
  cdt 116
  DB-Plugin 103
  EPIC 129
  JSEclipse 351
  Perspektive 102
  Projekt 102
  radrails-Plugin 468
  rubyeclipse 299
  SQLExplorer 103
  Sysdeo-Plugin 496
  WTP 103
ECMA-262 348
EJB 515
EPIC 129
Etch 47
ethereal 22
eval 140
Event 380
Event-Handle 382
Exceptions 140

Farbe
  hexadezimale Darstellung 547
  indizierte 546
  websichere 546
fastCGI 417
  Ablauf 419
  Apache-Modul 420
  Daemonize-It 417
  Developer's Kit 419
  Java 423
  Lizenz 425
  Perl 422
  Protokoll 419
  Security 419
  Serverprozess 421
  Vorteile gegenüber CGI 418
Favicon 29, 154
Festival 68
File-Handle 144
Filetests 145
Firebug 107, 356
Firefox 65
Firewall 582
  Access-Lists 583
  Black List 583
  White List 583
Flash 401
  Actionscript 406
  Alternativen 407
  Beispiel 404
  Bühne 402
  Editor 402
  HTML-Einbindung 405
  Intro 402
  Prinzip 401
  Probleme 406
  Tween 404
  Zeitleiste 402
Flickr 588
Formular 29
  Eingabefeld 30
FTP 18

Gardner Group 587
GD 548
GD-Bibliothek 548
Gecko 65
gem 466
Gentoo 47
Geronimo 494, 515
gif 547
Gnash 407
GoLive 106, 197, 382
Google 7, 57, 385, 407, 523, 580
  Ajax 388
  Analytics 92
Gopher 3
Grafik 544
Groovy 587

Hardware 45
Hash 134
Here-Documents
  in Perl 155
  in Python 281
HSQLDB 50
HTML 6, 26
  Body 29
  Dokumententyp 28
  Einbinden eines Bildes 544
  Euro-Symbol 32
  Favicon 29
  Formular 29
    Eingabefeld 30
  Frameset 28
  Header 28
  IMG-Tag 544
  JavaScript 368
  Prinzip 27
  Strict 28
  Templates 457
  Transitional 28
  Varianten 28
  Versionen 26
  Zeichensatz 32
HTML Tidy 30
HTML Validator 30
HTTP 12
  Antwortcodes 17
  Cookie 519
  DELETE 17
  GET 16
  POST 16
  PUT 17
httpd.conf 87
HTTPS 91
Hyperlink 3

IDE 101
ifconfig 20

ILIAS 553
ImageMagick 548
IMAP 19
InternetExplorer 582
Internetlinks 593
Internetprotokolle 9
IOS 90, 583
ip 10
  v4 10
  v6 11
ipconfig 20
iPhone 66
IPSec 11
ISO 9

J2EE 493, 494
Java 8
  Applet 409, 582
  EJB 515
  fastCGI 423
  J2EE 493, 494
  J2SE 494
  Java 6 50
  JDBC 506
  JNDI 506
  JSF 515
  JSP 511
  JVM 414
  Sandbox 582
  Servlets 494
  Struts 515
javadoc 59, 200
JavaScript 347, 348, 582
  Ajax 387
  Array 358
  aus Serverscript 384
  Browser 348
  Browserabhängigkeit 384
  Browserweiche 371
  Date 368
  DOM 368, 369
  ECMA-262 348
  Entwicklung 348
  Event-Handle 382
  Events 380
  Fehlerkonsole 355
  Fehlersuche 355
  Formularüberprüfung 378
  HTML 368
  Importieren 362
  Iteratoren 362
  JSEclipse 351
  Kommentare 352
  Kontrollstrukturen 359
    Block 360
    do-while-Schleife 360
    Entscheidung 360
    for-Schleife 361
    Iteratoren 362
    Schalter 360
    verlassen 362
    while-Schleife 360
  lastModified bei Serverscript 384
  Link 375
  Logik 359
  Math 368
  Methoden 359
  navigator 370
  Objektorieniterung 364
  Operatoren 359
  screen 372
  Sicherheit 384, 385
  Standardklassen 368
  String 368
  Syntax 356
  Tag 353
  Tags 351
  toString 368
  Variablen 356
  Version 354
  window 373
    document 377
    document forms 377
    document forms elements 378
    frames 377
    history 376
    location 375
  XMLHttpRequest 399
  Zeichenketten 356
JBoss 494, 515
JDBC 175, 506
  Arbeitsweise 506
  Connection-Pool 510
JMeter 569
  Bedeutung 573
  Graph Result 573
  HTTP Cookie Manager 572
  Lauscher 572
  Proxy-Server 569
  Recording Controller 570
  Session 572
  Test auswerten 572
  Test durchführen 571
  Test Korrektheit 572
  Test Performance 573
  Testplan 569
  Timer 573
JNDI 506, 510
jpeg 547
JRuby 335
JSF 515
JSON 383
JSP 511
  Arbeistweise 512
  Aufbau 511
  Designpattern 515
  Direktiven 514
JVM 8, 414, 493
Jython 293

Kerio Firewall 584
Kommentare 59
Komodo 129
Kontrollstrukturen 135
Kubuntu 47

LAMP 191
LAN 10
LDAP 510
Links 593
Links2 67
Linux 46
  Debian 47
Liste 132
Literatur XIV
LiveScript 347, 348
Load Balancing 57
localhost 10
Logfile 91, 159
LWP 185
Lynx 67

Macromedia Dreamweaver 545
MDB2 434
Media-Formate 543
MIME 543
  Übersicht 543
mod_ssl 579
mod_perl 91
mod_php 91
mod_python 295, 447
mod_ruby 91
mod_ssl 91
Model 58
Mosaic-Browser 5
Mozilla 65
MVC 57, 251, 435
  CSS 58
  mit PHP 251
  Servlets 497
  Web 59
MyISAM 94
MySQL 49, 94
  MySQLdb 292

Namensraum 150
NAT 11, 577
Neo 1973 47
nessus 581
Netscape 5, 348
Netzwerkmaske 20
nichtroutbare Adressen 11
nmap 581
Nomenklatur XV
Normalform 96
nslookup 17

Objektorientierung 54
  Begrifflichkeiten 54
  Garbage Collection 244
  Klasse 54
  Konstruktor 54
  Objekt 54
  Referenz 54
ODBC 175
OpenSSL 91, 579
Opera 66
OSI 9

PAT 11
pdf
  FOP 549
  Java 549
pdf-Generierung 548
PDO 255
PEAR 427
  DB 255, 430
    Arbeitsweise 430
  Installation 427
  MDB2 434
  Pakete 429
  Paketmanager 428
  Struktur 427
PECL 427
Performance 53, 569, 574
Perl 125
  beenden 141
  Cookie 527
  Dateizugriff 144
  Datenstrukturen 130
  DBI 173
    Anweisungen außer Select 184
    close 178
    connect 177
    connect allgemein 178
    connect für MySQL 177
    fetchrow_arrayref 179
    fetchrow_hashref 179
    Installation 176
    Prinzip 174
    rows 181
    SELECT-Abfrage 178
  Einbinden von Code 162
  einfaches CGI 154
  EPIC 129
  fastCGI 422
  Funktionen 142
  Here-Documents 155
  Installation 126
  Interpreter 127
  Kommentare 128
  Kontrollstrukturen 135
  List-Operatoren 133
  Lizenz 126
  LWP Performance-Test 574
  Modul CGI 165
  Modul DBI 173
  Modul LWP 185
  Module 160
  Namensraum 150
  package 160
  perldoc 127, 128
  Ratespiel im Web 167
  Referenz 150
  Schleifen 137
  Syntax 126
  Tastatureingabe 131
  Überladen 143
  wantarray 143
Perl-CGI
  Formularfelder 166
  Methode param 167
  Methode Vars 167
PHP 189
  Alternative Syntax 211
  alternative Syntax für Schleifen 215
  Arrays 206
  Bezeichner 201
  Bilder 548
  break und continue 216
  Cookies 524
  Dateizugriff 220
  Datenkapselung 244
  Datenstrukturen 201
  DBMS-Treiber 191
  Exception 250
  Formulare 223
  Hash 207
  Importieren 216
  include 216
  Klasse 244
  Konfiguration 193
  Konstruktor 244
  Kontrollstrukturen 209
  Listenoperationen 207
  Listenzeiger 209
  Logik 210
  Ming 407
  MS Access 241
  MVC 251
  mysqli 239
  Objektorientierung 244
  ODBC 241
  Operatoren 209
  pdf-Generierung 549
  PDO 255
  PEAR 427
  php.ini 193
  phpinfo 193
  Referenzen 223
  register globals 578
  require 216
  safe_mode 194
  Schleifen 212
  Session 530
  setcookie 526
  Smarty 435
  Superglobal 209, 226, 228, 526, 530
  Upload 548
  variable Variablen 223
  Vererbung 247
PHP-Nuke 447
phpinfo 580
phpMyAdmin 109
PHPSESSID 530
PLONE 257
png 547
pop 133
POP3 19
Popup-Fenster 374
Postgres 50
  psycopg 292
ppm 163
ppm-shell 163
Projektmanagement 53
Provider 18
Prozedurale Programmierung 54
push 133
pydev 258
Python 257
  Applets 295
  assoziative Listen 266
  Block 267
  Bytecode 262
  CGI 280
  Cookie 527
  Dateizugriff 271
  django 447
  Exceptions 272
  for 268
  Here-Documents 281
  IDLE 260
  Interpreter 262
  IO 271
  Jython 293
  Jython und Datenbanken 295
  Klasse 272
  Konstruktor 274
  Kontrollstrukturen 267
  LIFO 266
  Listen 265
  Literale 263
  Logik 265
  Methoden 269
    Argumente 270
  mod_python 295, 447
  Modul cgi 282
  Modul cgitb 284
  Modul freeze 279
  Modul math 278

Modul os 279
Modul string 277
Modul time 279
Modul Tkinter 295
Modul urllib 287
Module 277
MySQL 288
MySQLdb 292
Namensräume 270
Objektorientierung 272
psycopg 292
pyc-Datei 262
pydev 258
pysqlite 292
self 273
Syntax 260
Vererbung 275
Verzweigung 267
Web 277
while 268
ZOPE 447, 461

QuickTime 407

Rails 463
Apache 473
Controller 475
CRUD 464, 480
Datenbankanbindung 475
Datenbanktabellen 490
DBMS 465
Installation 466
Mapping 490
Model 475
Module 465
Prinzip 464
routes.rb 490
Scaffolding 464, 480
Template 484
View 484
WEBrick 464, 472
Webserver 464
Referenz 150
reguläre Ausdrücke 152
Rendering 435
Resultset 51
Ruby 297
Aliasing 313
alternative Syntax 310
Array 304
attr 319
BEGIN- und END-Block 308
Bezeichner 305
Bignum 302
Block 307
case-Schalter 309
CGI-Klasse 323
CGI-Programmierung 322
Cookie 527
Datenbanken 330
Datenkapselung 319
Entwicklungsumgebung 299
eRuby 329
Exceptions 320
Fixnum 302
Float 303
for-Schleife 311
FreeRIDE 300
gem 466
Hash 305
Here-Documents 303
HTML-Integration von Ruby 328
HTTP-Server 329
if - else 308
Installation 299
Instanzen 314
Interpreter 300
Iterator 310
Klassen 313
Konstruktor 314
Kontrollstrukturen 307
Kontrollstrukturen verlassen 311
Liste 304
LOAD_PATH 317
Logik 307
Mehrfachvererbung 318
Methodendefinition 312
mod_ruby 328
Modul DBI 331
Modul mysql 331
Modul socket 329
Module 315
NET-Modul 330
numerische Typen 302
Object-Klasse 313, 317
Objektorientierung 312
Operatoren 306
Protokolle 330
Rails-Framework 463
Range 304
rubyeclipse 299
Systemvariablen 306
until-Schleife 311
Vererbung 317
Web 321
Webclient 330
while-Schleife 311
Zeichenketten 303
Ruby on Rails 463

Safari 66
Sandbox 385, 410, 582
SATAN 581
Schichtmodell 9, 57
Schleife
break 140
continue 140
do-while 138
for 139
foreach 139
last 140
next 140
redo 140
until 139
while 138
Schleifen 137
Screen Reader 68
Selfhtml
Farbwähler 7
Semantic Web 589
sendmail 19
Sequenz 144
Serv-U 18
Servlet
Aufwand 500
CGI 505
Deployment 496
Formularverarbeitung 500
Lebenszyklus 503
Request-Methoden 503
Senden von HTML 506
Startwerte 505
Tomcat Webserver 500
Servlets 494
Klasse 498
MVC 497
Tomcat 495
Voraussetzungen 495
Session 529
JMeter 572
PHP 530
PHPSESSID 530
Session-ID 529
SGML 26
Shebang-Zeile 154, 160
shift 133
Sicherheit 575
Apache-Kennung 579
Aufwand 585
ausgelieferte Dateien 159
Datenbankserver 577
Dienste 578
DOS 580
Firewall 582
Konfiguration 577
Netzwerkstruktur 576
SSH 577
Werkzeuge 581
Smarty 58, 255, 435, 436
Beispiel 436
Caching 446
include 445
include_path 436
Instanz 442
Konfiguration 442
Kontrollstrukturen 443

Methoden 442
Template Language 439
Variablen 439
SMTP 19
Sniffer 22
Socket 72
Solaris 46
spiegel.de 552
SQL
Fremdschlüssel 94
Primärschlüssel 94
SQLite 50
pysqlite 292
SSH 18, 577
SSI 337
Beispiele 337
Elemente 339
Flusssteuerung 341
geschachtelt 340
Kennzeichnung 338
Syntax 339
Voraussetzungen 337
SSL 579
Stack 133
Standard-Handle 145
Streaming 12, 401
Struts 447, 515
Sun 8
Superglobals 209
SVG 548

Tabelle
`gmtime` in Perl 156
Browser, Verteilung 63
Cookie-Attribute 523
CSS und ältere Browser 39
Default tcp-Ports der wichtigsten Dienste 19
Event-Handles in JavaScript 383
Filetests in Perl 146
HTML-Attribute für <IMG>-Tag 544
HTTP-Codes 17
HTTP-Syntax 14
logische Operationen in Perl 136
MIME-Typen 544
Modi der Perl-Methode `open` 145
Operatoren in Perl 131
Operatoren in PHP 209
private Netzwerkadressen 11
Schleifenabbruch in Perl 140
Sonderzeichenmaskierung 131
SSI-Elemente 339
Tabellentypen in MySQL 95
Vergleiche in Perl 136
tcp 11
Socket 72
tcp/ip 9
telnet 17
Template 435
django 457
Language 435
Rails 484
Smarty 435
Testverfahren 569
Thread 74, 495
Thunderbird 65
TLS 579
Tomcat 495
Webserver 500
Transaktion 49, 239, 494, 515
TYPO3 554
Ajax 554
Architektur 554
Backend 558
Extensions 559
Frontend-Editing 567
Geschichte 554
Installation 555
1-2-3-Tool 556
localconf.php 557
Projekt 561
TemplaVoila 560
TypoScript 565

Ubuntu 47
udp 12
Überladen 143
Unix 46
unshift 133
Unterporgramm
in Perl 146
URI 19
URL 19
UTF-8 32
Webserver 32

Variable
lokal und global 148
Vektorgrafik 547
Velocity 435, 447, 511, 515
VGA
Farbpalette 546
Virtualisierung 46, 48
Vista 583
Apache-Installation 86

W3C 26, 32
WAI 43
Warteschlange 133
Web 2.0 400, 588
Web-Design 435
Web-Programmierung
Bedeutung der Techniken 587
clientseitige 6
Geschichte 3
Grundlagen 3
Scriptsprachen 8
serverseitige 7
Sprachen 8
Web-Site-Defacement 575
Webalizer 92
Webclient 5, 63
WebDAV 17
Webeditoren 104
WEBrick 472
Webserver 5, 71
Werkzeuge für die Web-Entwicklung 101
Wikipedia 588
Wolke 588
WSAD 295
www.webkompendium.de XV, 593

XAMPP 51, 86, 192
XHTML 38
XML 32
Baumstruktur 34
DTD 35
Syntax 35
Editor 33
Formatierung 36
MABxml 97
MARCXML 97
Prolog 34
Syntax 34
valid 36
Validierung 33
well-formed 36
XMLHttpRequest 389, 399

Z3 409
Zeichenketten 131
Zertifikat
SSH 18
ZOPE 257, 295, 461, 553
Zugriffsregel 576
Zuse 409