Beispiele und Lösungen für PHP-Programmierer

3. Auflage
Aktuell zu PHP 5.3

PHP 5 Kochbuch

David Sklar & Adam Trachtenberg

Überarbeitung und Aktualisierung von Carsten Lucke, Matthias Brusdeylins, Ulrich Speidel & Stephan Schmidt

O'REILLY®

3. AUFLAGE

PHP 5 Kochbuch

David Sklar & Adam Trachtenberg

Überarbeitung und Aktualisierung von
Carsten Lucke, Matthias Brusdeylins,
Ulrich Speidel & Stephan Schmidt

O'REILLY®
Beijing · Cambridge · Farnham · Köln · Sebastopol · Taipei · Tokyo

Kommentare und Fragen können Sie gerne an uns richten:
O'Reilly Verlag
Balthasarstr. 81
50670 Köln
Tel.: 0221/9731600
Fax: 0221/9731608
E-Mail: kommentar@oreilly.de

1. Auflage 2003
2. Auflage 2005
3. Auflage 2009

Die 2. Auflage der Originalausgabe erschien 2006 unter dem Titel
PHP Cookbook, 2nd Edition im Verlag O'Reilly Media, Inc.

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte bibliografische Daten
sind im Internet über *http://dnb.ddb.de* abrufbar.

Deutsche Übersetzung: Lars Schulten, Köln
Lektorat: Alexandra Follenius, Köln
Korrektorat: Sibylle Feldmann, Düsseldorf
Fachgutachten: Frank Kleine, Karlsruhe & Gerd Schaufelberger, Dresden
Satz: III-satz, Husby, www.drei-satz.de
Umschlaggestaltung: Michael Oreal, Köln
Produktion: Karin Driesen & Andrea Miß, Köln
Belichtung, Druck und buchbinderische Verarbeitung:
Druckerei Kösel, Krugzell; www.koeselbuch.de

ISBN 978-3-89721-904-5

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Inhalt

Einleitung

PHP ist der Motor hinter Millionen von dynamischen Webanwendungen. Aufgrund seiner vielfältigen Möglichkeiten, seiner gut verständlichen Syntax und seiner Unterstützung für verschiedene Betriebssysteme und Webserver ist PHP zur idealen Sprache für schnelle Webentwicklungen wie auch für den methodischen Aufbau komplexer Systeme geworden.

Einer der Hauptgründe für den Erfolg von PHP als Web-Skriptsprache ist dessen Ursprung als Hilfsmittel zur Verarbeitung von HTML-Formularen und zur Erzeugung von Webseiten. PHP ist also sehr webfreundlich. Außerdem ist PHP polyglott – es kann mit einer Vielzahl von Datenbanken kommunizieren und kennt diverse Internetprotokolle. PHP erleichtert die Analyse von Browser-Daten und die Generierung von HTTP-Anfragen. Dieser webbezogene Fokus wirkt sich natürlich auch auf die Rezepte und Beispiele dieses *PHP 5 Kochbuchs* aus.

Das Buch ist eine Sammlung von Lösungen für gängige Aufgaben bei der Arbeit mit PHP. Wir haben uns bemüht, für alle Programmierer – vom Einsteiger bis zum Spezialisten – interessantes Material zusammenzutragen. Wenn wir damit Erfolg hatten, werden Sie einiges (oder vielleicht eine ganz Menge) aus dem *PHP 5 Kochbuch* lernen können. Hier gibt es Tipps für Programmierer, die jeden Tag mit PHP zu tun haben, und auch für Entwickler, die von einer anderen Programmiersprache zu PHP gewechselt sind.

PHP ist als Quellcode und in Binärform kostenlos unter *http://www.php.net/* als Download erhältlich. Die PHP-Website enthält außerdem Installationsanleitungen, umfassende Dokumentationen sowie Verweise auf Online-Ressourcen, Benutzergruppen, Mailinglisten und weitere PHP-Ressourcen.

An wen sich dieses Buch richtet

Dieses Buch ist für Programmierer geschrieben, die mit PHP Probleme lösen müssen. Wenn Sie PHP noch gar nicht kennen, sollte es eher Ihr zweites PHP-Buch werden. Wenn Sie bereits mit PHP vertraut sind, können Sie mithilfe dieses Buchs spezielle Probleme überwinden und Ihre Fähigkeiten weiterentwickeln (zumindest was Ihre Programmieraktivitäten betrifft). Außerdem kann das *PHP 5 Kochbuch* Ihnen zeigen, wie Sie bestimmte Aufgaben mithilfe von PHP lösen, mit denen Sie in einer anderen Sprache bereits vertraut sind, zum Beispiel das Versenden von E-Mails oder die Entwicklung eines SOAP-Servers. Auch Programmierer, die Anwendungen von anderen Sprachen nach PHP konvertieren, werden in diesem Buch einen treuen Begleiter finden.

Deutsche Überarbeitung für PHP 5.3

Die deutsche Ausgabe des *PHP Kochbuchs* wurde 2005 von Ulrich Speidel und Stephan Schmidt komplett überarbeitet und aktualisiert. Diese 2. Auflage behandelte PHP 5.0.x und befasste sich auch bereits mit einigen Änderungen in der Version 5.1.

Die vorliegende 3. Auflage des *PHP 5 Kochbuchs* deckt alle wichtigen Neuerungen ab, die die Veröffentlichung von PHP 5.3 mit sich gebracht hat. Die PHP-Experten Matthias Brusdeylins und Carsten Lucke haben dafür alle Rezepte und Codebeispiele geprüft und wenn nötig umgeschrieben, und sie haben das Kochbuch um eine Reihe von neuen Rezepten ergänzt.

Diese Auflage befasst sich nun unter anderem mit den maßgeblichen Änderungen im Objektmodell von PHP 5.3, die sich im Zusammenhang mit der lang erwarteten Einführung von *Namensräumen* ergeben haben. Weitere Themen wie das *Late Static Binding* und die Möglichkeit zur Erstellung von *Lambda-Funktionen* und *Closures* werden ebenso behandelt wie kleinere Erweiterungen. Dazu gehören zum Beispiel die *NOWDOC-Notation* und auch neue magische Methoden wie *__callStatic()*.

Mit PHP 5.3 wurden diverse neue PHP-Erweiterungen eingeführt. Diese Neuauflage bespricht viele davon, unter anderem den neuen *MySQL Native Driver (mysqlnd)* als performante Alternative zur *MySQL Client Library (libmysql)*. Außerdem wird die neue *PHAR-Erweiterung* behandelt.

Eine Vielzahl von Rezepten ist überarbeitet worden, um maßgebliche Neuerungen in Erweiterungen wie der *Standard PHP Library (SPL)* und der Datumserweiterung *ext/date* zu reflektieren.

Die Neuerungen von PHP 5.3 bieten einen immensen Zugewinn an Funktionalität gegenüber PHP 5.2.x, und die Autoren können Ihnen das Upgraden auf die neue Version nur wärmstens empfehlen. Endlich müssen Sie die fehlende Namensraumunterstützung nicht mehr durch komplexe Klassennamen und Verzeichnisstrukturen kompensieren. Hilfreiche Hinweise für den Umstieg finden Sie unter *http://php.net/migration53*.

Aufbau dieses Buchs

Wir gehen nicht davon aus, dass Sie sich hinsetzen und dieses Buch von Anfang bis Ende durchlesen (obwohl wir uns freuen würden, wenn Sie es täten!). PHP-Programmierer sind ständig mit vielfältigen Herausforderungen in einem weiten Themenbereich konfrontiert. Greifen Sie zum *PHP 5 Kochbuch*, wenn Sie auf ein konkretes Problem stoßen. Jedes Rezept enthält eine in sich geschlossene Problemlösung, die Ihnen einen Anstoß zur Bewältigung Ihrer Aufgabe gibt. Wenn ein Rezept weitergehende Fragen aufwirft, enthält es Verweise auf diesbezügliche Rezepte sowie auf andere Online- und Offlineressourcen.

Manchmal kann es aber auch sinnvoll sein, ein ganzes Kapitel auf einmal zu lesen. Die Rezepte steigern sich in der Regel vom Einfachen zum Schwierigen, und Beispielprogramme am Ende vieler Kapitel stellen die Themen im Zusammenhang dar. Die Einführungen zu den Kapiteln bieten einen Überblick über ihren jeweiligen Inhalt, enthalten Hintergrundinformationen und weisen auf besonders interessante Rezepte hin.

Das Buch beginnt mit vier Kapiteln über grundlegende Datentypen:

- Kapitel 1, *Strings*, behandelt Einzelheiten wie die Verarbeitung von Substrings oder Groß- und Kleinschreibung und zeigt, wie man Strings in kleinere Teile zerlegt und kommaseparierte Daten analysiert.
- Kapitel 2, *Zahlen,* erläutert Operationen mit Fließkomma- und Zufallszahlen, die Konvertierung zwischen verschiedenen Zahlensystemem sowie die Formatierung von Zahlen.
- Kapitel 3, *Datum und Zeit*, zeigt Ihnen, wie Sie Datums- und Zeitwerte manipulieren und formatieren, mit Zeitzonen und der Sommerzeit umgehen und die Zeit auf Mikrosekunden genau bestimmen.
- Kapitel 4, *Arrays*, behandelt Array-Operationen wie das Abarbeiten in Schleifen, das Vermischen sowie das Ändern, Entfernen und Extrahieren einzelner Elemente.

Danach folgen Kapitel zu den Elementen, aus denen Programme zusammengesetzt werden, sowie zu einigen Themenbereichen, die in PHP 5 neu hinzugekommen sind:

- Kapitel 5, *Variablen*, behandelt bemerkenswerte Eigenheiten von PHP bezüglich des Umgangs mit Variablen, darunter Vorgabewerte, statische Variablen und die Umwandlung von komplexen Datentypen in eine String-Repräsentation.
- Bei den Rezepten in Kapitel 6 geht es um die Verwendung von Funktionen in PHP: die Verarbeitung von Argumenten, die Über- und Rückgabe von Variablen als Referenzen, die Erzeugung von Funktionen zur Laufzeit, Lambda-Funktionen und Closures sowie die Bestimmung der Geltungsbereiche von Variablen.
- Kapitel 7, *Klassen und Objekte*, behandelt das Objektmodell von PHP 5. Hier gibt es Rezepte zu Namensräumen, zur Anwendung des Überladens und des Polymorphismus, zur Definition von Konstruktoren und zum Kopieren von Objekten.

- Kapitel 8, *Effizienter Umgang mit Daten*, beschreibt die Anwendung von vier neuen nützlichen Konzepten in PHP 5: Iteratoren, mit denen Sie Objekte einfach in einer `foreach`-Schleife bearbeiten können; Streams, mit denen Sie auf beliebige Datenquellen so zugreifen können wie auf eine Datei; Wrapper, mit denen Sie Datenquellen für die Stream-Funktionen vorbereiten können, und Filter, mit denen Sie Daten, die Sie in Streams schreiben oder aus ihnen auslesen, transparent modifizieren können.
- Kapitel 9, *Fehlerbehandlung mit Exceptions*, zeigt Ihnen, wie Sie Ihre Anwendungen durch den Einsatz von Ausnahmen (Exceptions) robuster gestalten und Code zur Fehlerbehandlung von anderem Code trennen.

Die folgenden sechs Kapitel widmen sich Themen rund um die klassische Webprogrammierung:

- Kapitel 10, *Web-Grundlagen*, behandelt Cookies, Header, Authentifizierung, Konfigurationsvariablen und weitere Grundlagen von Webanwendungen.
- Kapitel 11, *Formulare*, zeigt, wie man Eingaben validiert, mit mehrseitigen Formularen arbeitet, Formulare mit Fehlermeldungen anzeigt und mit Sonderzeichen in den Benutzerdaten umgeht.
- Kapitel 12, *Datenbankzugriff*, erläutert die Unterschiede zwischen Textdateien, DBM- und SQL-Datenbanken und geht auf den neuen MySQL Native Driver (*mysqlnd*) ein. Außerdem zeigt es, wie man mithilfe der Datenbank-Abstraktionsschicht PHP Data Objects (PDO) eindeutige Kennzeichen zuweist, Zeilen wiederfindet, Daten verändert, Anführungszeichen setzt und Debugging-Informationen protokolliert.
- Kapitel 13, *Web-Automatisierung*, hat das Auslesen von URLs und die Verarbeitung von HTML-Code zum Schwerpunkt, behandelt aber auch den Gebrauch von Templates und zeigt, wie man Server-Zugriffsprotokolle analysiert.
- Kapitel 14 behandelt XML und die verschiedenen APIs, mit denen XML verarbeitet werden kann, wie DOM, SAX, XSLT sowie die PECL-Extensions xmlWriter und xmlReader oder auch die nur in PHP 5 verfügbare Erweiterung simpleXML. Weiterhin erfahren Sie, wie Sie wiederkehrende Arbeiten mithilfe von PEAR-Paketen vereinfachen.
- Kapitel 15, *Webservices*, befasst sich mit den Webservice-Standards XML-RPC, SOAP und RSS-Feeds. Außerdem zeigt es, wie Sie mit REST-Webservices umgehen.

Im darauffolgenden Buchabschnitt finden Sie eine Reihe von Kapiteln über weitere PHP-Features sowie über Erweiterungen, die viele nützliche Funktionalitäten bieten. Diese Rezepte helfen Ihnen beim Aufbau von Anwendungen, die unübertroffen robust, sicher, benutzerfreundlich und effizient sind.

- Kapitel 16 behandelt reguläre Ausdrücke, mit deren Hilfe beispielsweise die Gültigkeit von E-Mail-Adressen geprüft und der Inhalt von HTML-Tags gelesen werden kann, sowie von »gierigem« und »nicht-gierigem« Matching.

- In Kapitel 17, *Verschlüsselung und Sicherheit*, geht es um Verschlüsselung, unter anderem um das Generieren und Speichern von Passwörtern, den Austausch von verschlüsselten Daten mit anderen, die Speicherung von verschlüsselten Daten in einer Datei oder Datenbank und den Gebrauch von SSL.
- Kapitel 18, *Grafik*, zeigt Ihnen, wie Sie Grafiken erzeugen, indem Sie Text, Linien, geometrische Figuren und Kurven zeichnen.
- Kapitel 19, *Internationalisierung und Lokalisierung*, hilft Ihnen dabei, Ihre Anwendungen weltweit benutzerfreundlich zu gestalten, und enthält Rezepte zur Verwendung von Locales sowie lokalisierten Texten, Datums- und Zeitwerten, Währungsbeträgen und Bildern.
- Kapitel 20, *Internetdienste*, erörtert netzwerkbezogene Aufgaben wie das Lesen und Versenden von E-Mails und Newsgroup-Beiträgen, den Gebrauch von FTP und LDAP und die Ausführung von DNS- und Whois-Abfragen.

Kapitel 21 und Kapitel 22 behandeln das Dateisystem:

- Kapitel 21 konzentriert sich dabei auf den Umgang mit Dateien: wie man sie öffnet und schließt, temporäre Dateien benutzt, Dateien sperrt, komprimierte Dateien versendet und den Inhalt von Dateien verarbeitet.
- Kapitel 22 behandelt Verzeichnisse und Datei-Metadaten; dieses Kapitel enthält Rezepte zur Veränderung der Zugriffsrechte und der Eigentümerschaft von Dateien, zum Verschieben und Löschen von Dateien und der Verarbeitung aller Dateien eines Verzeichnisses.

Den Abschluss bilden zwei Kapitel über Themen, mit denen die Möglichkeiten von PHP erweitert werden:

- Kapitel 23, *PHP auf der Kommandozeile*, behandelt den Einsatz von PHP außerhalb der Webprogrammierung. Die Rezepte befassen sich mit Kommandozeilen-bezogenen Themen wie dem Parsen von Programmparametern und dem Einlesen von Passwörtern sowie mit der zeilenweisen Verarbeitung von Dateien, dem Forken von Prozessen oder der Programmierung eines Servers.
- Kapitel 24 behandelt PEAR, das *PHP Extension and Application Repository* und die PHP Archiv-(PHAR-)Erweiterung. PEAR ist eine Sammlung von PHP-Programmen, die diverse Funktionen und PHP-Erweiterungen zur Verfügung stellt. PEAR-Module verwenden wir an verschiedenen Stellen im Buch, aber Kapitel 24 zeigt, wie Sie diese installieren und aktualisieren. Weitere Rezepte machen Sie mit Funktionen von PEAR vertraut, mit denen Sie unter anderem die PEAR-Infrastruktur für eigene Anwendungen nutzen können.

Weiterführende Quellen

Websites

Es gibt eine gewaltige Menge an online verfügbarem PHP-Referenzmaterial. Dort finden Sie alles, vom kommentierten PHP-Handbuch bis zu Sites mit regelmäßig erscheinenden Artikeln und Tutorials. Hier sind einige besonders wichtige Sites:

Das kommentierte PHP-Handbuch: http://www.php.net/manual/de
: In 17 Sprachen verfügbar, enthält sowohl offizielle Dokumentationen der Funktionen und der Besonderheiten der Sprache als auch Kommentare von Anwendern.

PHP-Mailinglisten: http://www.php.net/mailing-lists.php
: Es gibt viele PHP-Mailinglisten zur Installation, Programmierung und Erweiterung von PHP sowie zu verschiedenen anderen Themen. Eine Web-Oberfläche zum Lesen der Mailinglisten finden Sie unter *http://news.php.net/*.

PHP-Präsentationsarchiv: http://conf.php.net/
: Eine Sammlung von Präsentationen zu PHP, die bei verschiedenen Konferenzen vorgeführt wurden.

PEAR: http://pear.php.net/
: PEAR bezeichnet sich selbst als »ein Framework und Verteilungssystem für wiederverwendbare PHP-Komponenten«. Sie finden dort viele nützliche PHP-Klassen und Beispielprogramme.

PHP.net: A Tourist's Guide: http://www.php.net/sites.php
: Unter dem Dach von *php.net* ist dies ein »Reiseführer« zu verschiedenen Websites.

PHP Knowledge Base: http://php.faqts.com/
: Viele Antworten und Fragen aus der PHP-Community und Links zu anderen Quellen.

PHP DevCenter: http://www.onlamp.com/php/
: Eine Sammlung von PHP-Artikeln und Tutorials mit einer guten Mischung aus Einführung und fortgeschritteneren Themen.

Bücher

In diesem Abschnitt weisen wir Sie auf Bücher hin, die hilfreiche Referenzen und Tutorials für die Erstellung von PHP-Anwendungen bieten. Die meisten von ihnen behandeln speziell die Web-Programmierung; achten Sie auch auf Bücher über MySQL, HTML, XML und HTTP.

- *Programmieren mit PHP* von Rasmus Lerdorf, Kevin Tatroe und Peter MacIntyre (O'Reilly Verlag)
- *PHP Design Patterns* von Stephan Schmidt (O'Reilly Verlag)
- *HTML & XHTML – Das umfassende Handbuch* von Chuck Musciano und Bill Kennedy (O'Reilly Verlag)

- *Dynamic HTML: The Definitive Reference* von Danny Goodman (O'Reilly)
- *High Performance MySQL – Optimierung, Backups, Replikation und Lastverteilung* von Baron Schwartz, Peter Zaitsev, Vadim Tkachenko, Jeremy Zawodny, Arjen Lentz, & Derek Balling (O'Reilly Verlag)
- *JavaScript – Das umfassende Referenzwerk* von David Flanagan (O'Reilly Verlag)
- *CSS – Das umfassende Handbuch* von Eric A. Meyer (O'Reilly Verlag)
- *Reguläre Ausdrücke* von Jeffrey E. F. Friedl (O'Reilly Verlag)
- *XML in a Nutshell* von Elliotte Rusty Harold und W. Scott Means (O'Reilly Verlag)
- *Vom Mythos des Mann-Monats. Essays zum Software-Engineering* von Frederick P. Brooks (Mitp)

In diesem Buch verwendete Konventionen

Programmierkonventionen

Wir haben in der Regel die Anfangs- und Schlusskennzeichen `<?php ?>` bei den Beispielen in diesem Buch weggelassen, außer in den Beispielen, in denen die Kennzeichen im Programmrumpf enthalten sind. Um Namenskonflikte zu vermeiden, beginnen die Funktions- und Klassennamen im *PHP 5 Kochbuch* mit `pc_`.

Die Beispiele sind so geschrieben worden, dass sie unter der PHP-Version 5.3 laufen. Die Beispielprogramme sollten sowohl unter Unix als auch unter Windows funktionieren, sofern es nicht im Text anders vermerkt ist.

Schriftkonventionen

Folgende typographische Konventionen wurden in diesem Buch benutzt:

Kursiv
: Für Datei- und Verzeichnisnamen, E-Mail-Adressen und URLs. Ebenso für neue Begriffe bei ihrer Definition.

`Nichtproportionalschrift`
: Für Code-Listings und Schlüsselwörter, Variablen, Funktionen, Befehlsoptionen, Parameter, Klassennamen und HTML-Tags, wenn sie im Text auftauchen.

`Nichtproportionalschrift fett`
: Für die Markierung von Zeilen in der Ausgabe der Code-Listings und Befehlszeilen, die der Benutzer schreibt.

`Nichtproportionalschrift kursiv`
: Als genereller Platzhalter, um Elemente anzuzeigen, die durch aktuelle Werte in Ihrem eigenen Programm ersetzt werden müssen.

Die Codebeispiele zu diesem Buch

Es gibt eine Website von O'Reilly, auf der Sie Beispiele, Errata und weitere Informationen zu diesem Buch finden können:

http://www.oreilly.de/catalog/phpckbk3ger/

Viele der in diesem Buch abgedruckten Codebeispiele stehen dort zum Herunterladen zur Verfügung. Bitte klicken Sie auf den Beispiel-Link auf der Katalogseite zum Buch.

Die Codebeispiele verwenden

Dieses Buch ist dazu da, Ihnen bei Ihrer Arbeit zu helfen. In der Regel können Sie den Code dieses Buchs in Ihren Programmen und Dokumentationen verwenden. Sie brauchen uns nicht um Erlaubnis zu fragen, solange Sie nicht einen beachtlichen Teil des Codes wiedergeben. Beispielsweise benötigen Sie keine Erlaubnis, um ein Programm zu schreiben, das einige Codeteile aus diesem Buch verwendet. Für den Verkauf oder die Verbreitung einer CD-ROM mit Beispielen von O'Reilly-Büchern brauchen Sie auf jeden Fall unsere Erlaubnis. Die Beantwortung einer Frage durch das Zitieren dieses Buchs und seiner Codebeispiele benötigt wiederum keine Erlaubnis. Wenn Sie einen erheblichen Teil der Codebeispiele dieses Buchs in die Dokumentation Ihres Produkts einfügen, brauchen Sie eine Erlaubnis.

Wir freuen uns über einen Herkunftsnachweis, bestehen aber nicht darauf. Eine Referenz enthält i.d.R. Titel, Autor, Verlag und ISBN, zum Beispiel: »*PHP 5 Kochbuch, 3. Auflage* von David Sklar, Adam Trachtenberg, Carsten Lucke, Matthias Brusdeylins, Ulrich Speidel & Stephan Schmidt, Copyright 2009, O'Reilly Verlag, ISBN 978-3-89721-904-5.«

Wenn Sie meinen, Ihre Verwendung unserer Codebeispiele könnte den angemessenen Gebrauch oder die hier gegebene Erlaubnis überschreiten, nehmen Sie einfach mit uns über *kommentar@oreilly.de* Kontakt auf.

KAPITEL 1

Strings

1.0 Einführung

Strings in PHP sind Zeichenfolgen wie etwa »Wir halten diese Wahrheiten für selbstverständlich« oder »Es war einmal« oder auch »111211211«. Wenn Sie Daten aus einer Datei lesen oder auf einem Webbrowser ausgeben, werden Ihre Daten durch Strings repräsentiert.

Auf individuelle Zeichen in Strings können Sie wie in C mit einer Schreibweise im Stil von Array-Indizes zugreifen. Das erste Zeichen in dem String hat den Index 0. Ein Beispiel:

```
$nachbar = 'Hilda';
print $nachbar[3];
d
```

PHP-Strings unterscheiden sich jedoch von C-Strings dadurch, dass sie binär-sicher sind (d.h., sie können Null-Bytes enthalten) und auf Anweisung wachsen oder schrumpfen können. Ihre Größe ist nur durch den verfügbaren Speicherplatz beschränkt.

Sie können Strings auf vier Arten initialisieren, die in Form und Verhalten Perl und Unix-Shells ähneln: mit einfachen Anführungszeichen, mit doppelten Anführungszeichen, mit dem »Here-Document-Format« (Heredoc) und mit dem »Now-Document-Format« (Nowdoc). Bei Strings, die in einfachen Anführungszeichen stehen, müssen Sie nur ein Sonderzeichen in eine Escape-Sequenz umwandeln, und zwar den Backslash, sowie auch das einfache Anführungszeichen selbst:

```
print 'Ich bin zum Kurfürstendamm gegangen.';
print 'Ich bin zum Ku\'damm gegangen.';
print 'Würden Sie $1.75 für 1/4 Liter Leitungswasser zahlen?';
print 'In Strings mit doppelten Anführungszeichen werden Zeilenwechel durch \n dargestellt';
Ich bin zum Kurfürstendamm gegangen.
Ich bin zum Ku'damm gegangen.
Würden Sie $1.75 für 1/4 Liter Leitungswasser zahlen?
In Strings mit doppelten Anführungszeichen werden Zeilenwechsel durch \n dargestellt.
```

Da PHP in Strings mit einfachen Anführungszeichen nicht nach zu interpolierenden Variablen und kaum nach Escape-Sequenzen sucht, kann man auf diese Weise Strings schnell und klar definieren.

In Strings mit doppelten Anführungszeichen werden keine einfachen Anführungszeichen in Form von Escape-Sequenzen erkannt, aber es werden interpolierte Variablen und die in Tabelle 1-1 aufgeführten Escape-Sequenzen erkannt.

Tabelle 1-1: Escape-Sequenzen in Strings mit doppelten Anführungszeichen

Escape-Sequenzen	Zeichen
`\n`	Zeilenwechsel (ASCII 10)
`\r`	Wagenrücklauf (ASCII 13)
`\t`	Tabulator (ASCII 9)
`\\`	Backslash
`\$`	Dollar-Zeichen
`\"`	Doppeltes Anführungszeichen
`\{`	Linke geschweifte Klammer
`\}`	Rechte geschweifte Klammer
`\[`	Linke eckige Klammer
`\]`	Rechte eckige Klammer
`\0` bis `\777`	Oktalzahl
`\x0` bis `\xFF`	Hexadezimalzahl

Zum Beispiel:

```
print "Ich bin zum Ku'damm gegangen.";
print "Die Soße kostet \$10.25.";
$preis = '$10.25';
print "Die Soße kostet $preis.";
print "Die Soße kostet \$\061\060.\x32\x35.";
Ich bin zum Ku'damm gegangen.
Die Soße kostet $10.25.
Die Soße kostet $10.25.
Die Soße kostet $10.25.
```

Die letzte Zeile des Codes gibt den Soßenpreis korrekt aus, denn das Zeichen `1` hat den ASCII-Code 49 (dezimal) oder 061 (oktal). Das Zeichen `0` ist ASCII 48 (dezimal) oder 060 (oktal), `2` ist ASCII 50 (dezimal) oder 32 (hexadezimal), und `5` ist ASCII 53 (dezimal) oder 35 (hexadezimal).

In den als Heredoc formatierten Strings werden die gleichen Interpolationen und Escape-Sequenzen wie in gewöhnlichen Strings mit doppelten Anführungszeichen erkannt. Lediglich die doppelten Anführungszeichen selbst müssen für die Ausgabe nicht mit einer Escape-Sequenz geschützt werden. Heredocs beginnen mit `<<<` und einem Token. Dieses Token (ohne umgebende Leerzeichen), gegebenenfalls gefolgt von einem Semikolon zum Abschließen der Anweisung, beendet das Heredoc. Ein Beispiel:

```
print <<< END
Es sieht seltsam aus, wenn auf Schildern Folgendes steht:
   Das Original "Root" Beer
   "Kostenloses" Geschenk
   Schuhe putzen, während "Sie" warten
oder wenn sie andere falsch gesetzte Anführungszeichen enthalten.
END;
Es sieht seltsam aus, wenn auf Schildern Folgendes steht:
   Das Original "Root" Beer
   "Kostenloses" Geschenk
   Schuhe putzen, während "Sie" warten
oder wenn sie andere falsch gesetzte Anführungszeichen enthalten.
```

Bei Heredocs bleiben Zeilenumbrüche, Leerzeichen und Anführungszeichen erhalten. Das Kennzeichen für das String-Ende wird normalerweise der Konvention entsprechend in Großbuchstaben geschrieben, wobei auf Groß- und Kleinschreibung geachtet werden muss. Deshalb ist Folgendes in Ordnung:

```
print <<< PETERSILIE
Leicht frisch anbauen können Sie:
- Petersilie
- Schnittlauch
auf Ihrer Fensterbank
PETERSILIE;
```

Ebenso dies:

```
print <<< HUNDE
Wenn Sie Haustiere mögen, schreien Sie:
HUNDE UND KATZEN SIND SUPER!
HUNDE;
```

Heredocs sind hilfreich, wenn man HTML mit eingefügten Variablen ausgeben möchte:

```
if ($uebrige_karten > 0) {
    $url = '/deal.php';
    $text = 'Mehr Karten austeilen';
} else {
    $url = '/new-game.php';
    $text = 'Neues Spiel beginnen';
}
print <<< HTML
Es sind <b>$uebrige_karten</b> &uuml;brig.
<p>
<a href="$url">$text</a>
HTML;
```

In diesem Fall muss das Semikolon hinter der Begrenzung für das String-Ende stehen, damit PHP weiß, dass die Anweisung beendet ist. Nehmen wir an, zwei Karten seien übrig, dann sieht die Ausgabe wie folgt aus:

```
Es sind <b>2</b> &uuml;brig.
<p>
<a href="/deal.php">Mehr Karten austeilen</a>
```

Manchmal dürfen Sie jedoch kein Semikolon benutzen:

```
$a = <<< END
Es war einmal ein
END
. ' Junge!';
print $a;
Es war einmal ein Junge!
```

In diesem Fall wird der Ausdruck in der folgenden Zeile fortgesetzt, deshalb benutzen Sie kein Semikolon. Beachten Sie auch, dass der Operator für die String-Verkettung in einer anderen Zeile stehen muss als das Kennzeichen für das String-Ende, damit PHP das Ende des Strings erkennen kann.

Seit PHP 5.3 existiert außerdem die Nowdoc-Syntax. Alle Regeln, die für Heredocs gelten, treffen auch auf Nowdocs zu. Der Unterschied ist, dass PHP in Nowdocs nicht nach zu interpolierenden Variablen und auch nach nur wenigen Sonderzeichen sucht. Als Faustregel kann man sagen, dass Nowdocs analog zu Strings mit einfachen Anführungszeichen funktionieren und Heredocs analog zu Strings mit doppelten Anführungszeichen. Nowdocs werden ebenfalls durch eine <<<-Sequenz eingeleitet, der nachfolgende Identifier wird aber in einfache Anführungszeichen eingeschlossen:

```
$uebrige_karten = 2;
if ($uebrige_karten > 0) {
    $url = '/deal.php';
    $text = 'Mehr Karten austeilen';
} else {
    $url = '/new-game.php';
    $text = 'Neues Spiel beginnen';
}
print <<< 'HTML'
Es sind <b>$uebrige_karten</b> &uuml;brig.
<p>
<a href="$url">$text</a>
HTML;
Es sind <b>$uebrige_karten</b> &uuml;brig.
<p>
<a href="$url">$text</a>
```

Wie Sie sehen, werden im Vergleich zum Heredoc-Beispiel die Variablen nicht durch ihre Werte ersetzt.

1.1 Auf Teil-Strings zugreifen

Sie möchten einen Teil eines Strings extrahieren und dabei an einer bestimmten Stelle im String beginnen. Beispielsweise benötigen Sie die ersten acht Buchstaben eines Benutzernamens, der in ein Formular eingeben worden ist.

Lösung

Benutzen Sie `substr()`, um Ihre Teil-Strings auszuschneiden:

```
$substring = substr($string,$start,$length);
$username = substr($_REQUEST['username'],0,8);
```

Diskussion

Wenn `$start` und `$length` positiv sind, gibt `substr()` die Anzahl von `$length` Zeichen aus dem String, beginnend mit `$start`, zurück. Das erste Zeichen im String ist an der Position 0:

```
print substr('Achte auf diesen Baum',6,5);
auf d
```

Wenn Sie `$length` weglassen, gibt `substr()` den String von `$start` bis zum Ende des ursprünglichen Strings zurück:

```
print substr('Achte auf diesen Baum',15);
n Baum
```

Wenn `$start` plus `$length` hinter dem String-Ende liegt, gibt `substr()` den gesamten String von `$start` an zurück:

```
print substr('Achte auf diesen Baum',28,5);
aum
```

Wenn `$start` negativ ist, zählt `substr()` rückwärts vom String-Ende, um festzustellen, wo der Teil-String beginnen soll:

```
print substr('Achte auf diesen Baum',-6);
print substr('Achte auf diesen Baum',-17,5);
n Baum
e auf
```

Wenn `$length` negativ ist, zählt `substr()` rückwärts vom String-Ende, um festzustellen, wo der Teil-String endet:

```
print substr('Achte auf diesen Baum',15,-2);
print substr('Achte auf diesen Baum',-4,-1);
n Ba
Bau
```

Siehe auch

Die Dokumentation zu `substr()` unter *http://www.php.net/substr*.

1.2 Teile von Strings ersetzen

Problem

Sie möchten einen Teil eines Strings durch einen anderen String ersetzen. Sie möchten beispielsweise alles außer den letzten vier Ziffern einer Kreditkartennummer vor der Ausgabe verbergen.

Lösung

Benutzen Sie substr_replace():

```
// Alles von der Position $start bis zum Ende von $alter_string
// wird durch $neuer_substring ersetzt.
$neuer_string = substr_replace($alter_String,$neuer_substring,$start);

// $laenge Zeichen, beginnend bei der Position $Start, werden durch
// $neuer_substring ersetzt.
$neuer_string = substr_replace($alter_string,$neuer_substring,$start,$laenge);
```

Diskussion

Ohne das Argument $length ersetzt substr_replace() alles von $start bis zum String-Ende. Wenn $length angegeben ist, wird nur die genannte Anzahl von Zeichen ersetzt:

```
print substr_replace('Mein Haustier ist ein blauer Hund.','Fisch.',22);
print substr_replace('Mein Haustier ist ein blauer Hund.','gelb',22,4);
$kreditkarte = '4111 1111 1111 1111';
print substr_replace($credit_card,'xxxx ',0,strlen($kreditkarte)-4);

Mein Haustier ist ein Fisch.
Mein Haustier ist ein gelber Hund.
xxxx 1111
```

Wenn $start negativ ist, wird der neue Teil-String bei $start-Zeichen, gezählt vom Ende von $alter_string und nicht vom Anfang, eingefügt:

```
print substr_replace('Mein Haustier ist ein blauer Hund.','Fisch.',-12);
print substr_replace('Mein Haustier ist ein blauer Hund.','gelb',-12,4);

Mein Haustier ist ein Fisch.
Mein Haustier ist ein gelber Hund.
```

Wenn $start und $laenge 0 sind, wird der neue Teil-String am Anfang von $alter_string eingefügt:

```
print substr_replace('Mein Haustier ist ein blauer Hund','Titel: ',0,0);

Titel: Mein Haustier ist ein blauer Hund.
```

Die Funktion substr_replace() ist nützlich, wenn Sie einen Text erhalten haben, der so umfangreich ist, dass man ihn nicht in einem Stück anzeigen kann, und Sie möchten einen Teil des Texts mit einem Verweis auf den übrigen Text anzeigen. Das folgende Beispiel zeigt die ersten 25 Zeichen einer Nachricht mit einer Ellipse an, auf die ein Link auf den weiteren Text folgt:

```
$r = mysql_query("SELECT id,message FROM messages WHERE id = $id") or die();
$ob = mysql_fetch_object($r);
printf('<a href="more-text.php?id=%d">%s</a>',
       $ob->id, substr_replace($ob->message,' ...',25));
```

Die Seite *more-text.php* kann mithilfe der im Query-String übergebenen Nachrichten-ID die vollständige Nachricht einlesen und anzeigen.

Siehe auch

Die Dokumentation zu `substr_replace()` unter *http://www.php.net/substr-replace*.

1.3 Einen String zeichenweise verarbeiten

Problem

Sie müssen alle Zeichen in einem String einzeln verarbeiten.

Lösung

Durchlaufen Sie jedes Zeichen des Strings mit `for` in einer Schleife. Dieses Beispiel zählt die Vokale in einem String:

```
$string = "An diesem Wochenende möchte ich ein zahmes Huhn kaufen.";
$vokale = 0;
for ($i = 0, $j = strlen($string); $i < $j; $i++) {
    if (strstr('aeiouäöüAEIOUÄÖÜ',$string[$i])) {
        $vokale++;
    }
}
```

Diskussion

Die zeichenweise Verarbeitung von Strings ermöglicht es, auf einfache Art und Weise Gleichniszahlen-Reihen zu berechnen:

```
Function lookandsay($s) {
    // Den Rückgabewert mit einem leeren String initialisieren.
    $r = '';
    /* $m enthält das Zeichen, das wir gerade zählen.
     * Mit dem ersten Zeichen im String initialisieren.*/
    $m = $s[0];
    // $n ist die Zahl der bereits gesehenen Zeichen in $m,
    // mit 1 initialisieren
    $n = 1;
    for ($i = 1, $j = strlen($s); $i < $j; $i++) {
        // Wenn dieses Zeichen dasselbe ist wie das letzte,
        if ($s[$i] == $m) {
            // den Zähler für dieses Zeichen erhöhen,
            $n++;
        } else {
            // andernfalls Zähler und Zeichen an Rückgabewert hängen.
            $r .= $n.$m;
```

```
            // Gesuchtes Zeichen auf aktuelles Zeichen setzen
            $m = $s[$i];
            // und den Zähler zurück auf 1 setzen.
            $n = 1;
        }
    }
    // Zusammengestellten String zusammen mit letztem Zähler
    // und letztem String zurückgeben.
    return $r.$n.$m;
}

for ($i = 0, $s = 1; $i < 10; $i++) {
    $s = lookandsay($s);
    print "$s\n";
}
```

```
1
11
21
1211
111221
312211
13112221
1113213211
31131211131221
13211311123113112211
```

Gleichniszahlen-Reihen bezeichnet man auf Englisch auch als »Look and Say«-Sequenzen, weil jedes Element das darstellt, was man erhält, wenn man das vorhergehende Element betrachtet und sagt, was darin enthalten ist. Wenn Sie sich beispielsweise das erste Element mit der 1 anschauen, sagen Sie »eine Eins«. Also ist das zweite Element »11«. Das sind »zwei Einsen«, deshalb ist das dritte Element »21«. Auf die gleiche Art und Weise finden Sie hier »eine Zwei und eine Eins«, folglich ist das vierte Element »1211« und so weiter.

Siehe auch

Die Dokumentation zu `for` unter *http://www.php.net/for*; mehr über Gleichniszahlen-Reihen unter *http://mathworld.wolfram.com/LookandSaySequence.html*.

1.4 Einen String wort- oder zeichenweise umkehren

Problem

Sie möchten die Reihenfolge der Wörter oder Zeichen in einem String umkehren.

Lösung

Benutzen Sie `strrev()`, um in einem String die Zeichen umzukehren:

```
print strrev('Dies ist kein Palindrom.');
```

```
.mordnilaP niek tsi seiD
```

Um die Reihenfolge der Wörter umzukehren, müssen Sie den String an den Wortgrenzen in ein Array umwandeln, dann zunächst die Wörter umkehren und sie anschließend neu zusammenfügen:

```
$s = "Es war einmal eine Schildkröte.";
// Den String in seine Wörter aufbrechen.
$words = explode(' ',$s);
// Das Array der Wörter umkehren.
$words = array_reverse($words);
// Den String neu aufbauen.
$s = join(' ',$words);
print $s;
```

```
Schildkröte. eine einmal war Es
```

Diskussion

Wenn Sie die Wörter eines Strings umkehren möchten, können Sie auch alles in eine Zeile schreiben:

```
$reversed_s = join(' ',array_reverse(explode(' ',$s)));
```

Siehe auch

Rezept 21.7 behandelt die Folgen der Verwendung anderer Zeichen, die keine Leerzeichen sind, als Wortgrenzen; die Dokumentationen zu `strrev()` unter *http://www.php.net/strrev* und `array_reverse()` unter *http://www.php.net/array-reverse*.

1.5 Tabulatoren expandieren und komprimieren

Problem

Sie möchten Leerzeichen innerhalb eines Strings in Tabulator-Zeichen umwandeln (oder umgekehrt). Dabei soll der Text an den Tabstopps ausgerichtet bleiben. So möchten Sie beispielsweise Benutzern formatierten Text in standardisierter Form anzeigen.

Lösung

Verwenden Sie `str_replace()`, um Leerzeichen in Tabs oder Tabs in Leerzeichen umzuwandeln:

```
$r = mysql_query("SELECT message FROM messages WHERE id = 1") or die();
$ob = mysql_fetch_object($r);
$tabbed = str_replace(' ',"\t",$ob->message);
```

```
$spaced = str_replace("\t",' ',$ob->message);

print "mit Tabs: <pre>$tabbed</pre>";
print "mit Leerzeichen: <pre>$spaced</pre>";
```

Die Funktion `str_replace()` berücksichtig allerdings bei der Umwandlung keine Tabstopps. Wenn Sie nach jedem achten Zeichen einen Tabstopp haben möchten, muss in einer Zeile, die mit einem Wort mit fünf Zeichen und einem Tab beginnt, das Tab durch drei Leerzeichen ersetzt werden und nicht nur durch eins. Mithilfe der in Beispiel 1-1 dargestellten Funktion `pc_tab_expand()` können Sie Tabulator-Zeichen so in Leerzeichen umwandeln, dass Tabstopps berücksichtigt werden.

Beispiel 1-1: pc_tab_expand()

```
function pc_tab_expand($a) {
  $tab_stop = 8;
  while (strstr($a,"\t")) {
    $a = preg_replace('/^([^\t]*)(\t+)/e',
                      "'\\1'.str_repeat(' ',strlen('\\2') *
                       $tab_stop - strlen('\\1') % $tab_stop)",$a);
  }
  return $a;
}

$spaced = pc_tab_expand($ob->message);
```

Mit der Funktion `pc_tab_unexpand()` können Sie Leerzeichen in Tabs zurückverwandeln, wie Beispiel 1-2 zeigt.

Beispiel 1-2: pc_tab_unexpand()

```
function pc_tab_unexpand($x) {
  $tab_stop = 8;

  $lines = explode("\n",$x);
  for ($i = 0, $j = count($lines); $i < $j; $i++) {
    $lines[$i] = pc_tab_expand($lines[$i]);
    $e = preg_split("/(.\{$tab_stop})/",$lines[$i],-1,PREG_SPLIT_DELIM_CAPTURE);
    $lastbit = array_pop($e);
    if (!isset($lastbit)) { $lastbit = ''; }
    if ($lastbit == str_repeat(' ',$tab_stop)) { $lastbit = "\t"; }
    for ($m = 0, $n = count($e); $m < $n; $m++) {
      $e[$m] = preg_replace('/  +$',"\t",$e[$m]);
    }
    $lines[$i] = join('',$e).$lastbit;
  }
  $x = join("\n", $lines);
  return $x;
}

$tabbed = pc_tab_unexpand($ob->message);
```

Beide Funktionen nehmen einen String als Argument an und geben ihn entsprechend verändert zurück.

Diskussion

Beide Funktionen gehen davon aus, dass sich an jeder achten Stelle ein Tabstopp befindet; dies kann jedoch geändert werden, wenn man die Einstellung der Variablen `$tab_stop` anpasst.

Der reguläre Ausdruck in `pc_tab_expand()` sucht nach einer Gruppe von Tabs und zugleich auch den gesamten Text vor dieser Tab-Gruppe. Der Text vor der Tab-Gruppe muss mit einbezogen werden, weil die Länge dieses Texts bestimmt, durch wie viele Leerzeichen die Tabs ersetzt werden müssen, damit der anschließende Text mit dem nächsten Tabstopp übereinstimmt. Die Funktion ersetzt nicht einfach jeden Tab durch acht Leerzeichen, sondern richtet den Text hinter den Tab-Zeichen nach den Tabstopps aus.

Auf ähnliche Weise sucht auch `pc_tab_unexpand()` nicht einfach nach acht aufeinander folgenden Leerzeichen, um sie durch ein Tab-Zeichen zu ersetzen. Sie teilt jede Zeile in Einheiten von acht Zeichen und ersetzt dann Leerzeichen am Ende dieser Einheiten (wenn es mindestens zwei sind) durch Tabs. Dieses Vorgehen erhält nicht nur die Ausrichtung des Texts an den Tabstopps aufrecht, sondern spart auch Platz im String.

Siehe auch

Die Dokumentation zu `str_replace()` unter *http://www.php.net/str-replace*.

1.6 Die Groß- und Kleinschreibung in Texten ändern

Problem

Sie müssen Text in Großbuchstaben oder Kleinbuchstaben umwandeln oder die Schreibung von Buchstaben innerhalb eines Strings auf andere Weise verändern. Sie möchten beispielsweise die Anfangsbuchstaben von Namen groß-, den Rest aber kleinschreiben.

Lösung

Verwenden Sie `ucfirst()` oder `ucwords()`, um die Anfangsbuchstaben eines oder mehrerer Wörter großzuschreiben:

```
print ucfirst("wie geht es dir heute?");
print ucwords("the prince of wales");
Wie geht es dir heute?
The Prince Of Wales
```

Verwenden Sie strtolower() oder strtoupper(), um die Groß-/Kleinschreibung eines gesamten Strings zu verändern:

```
print strtoupper("ich schreie nicht!");
// Tags müssen kleingeschrieben werden, um XHTML-kompatibel zu sein.
print strtolower('<A HREF="eins.php">eins</A>');
ICH SCHREIE NICHT!
<a href="eins.php">eins</a>
```

Diskussion

Mithilfe von ucfirst() wandeln Sie das erste Zeichen eines Strings in einen Großbuchstaben um:

```
print ucfirst('affengesicht');
print ucfirst('1 affengesicht');
Affengesicht
1 affengesicht
```

Beachten Sie, dass die zweite Zeile nicht als »1 Affengesicht« ausgegeben wird. Seit PHP 5.3 existiert die Funktion lcfirst(), die das Gegenstück zu ucfirst() ist und somit den ersten Buchstaben einer Zeichenkette in einen Kleinbuchstaben umwandelt:

```
print lcfirst("SCHREI MICH NICHT AN!");
sCHREI MICH NICHT AN!
```

Mit ucwords() schalten Sie das erste Zeichen jedes Worts innerhalb eines Strings in Großbuchstaben um:

```
print ucwords('1 affengesicht');
print ucwords("war's frau schulze-schaumburg?");
1 Affengesicht
War's Frau Schulze-schaumburg?
```

Wie erwartet, schreibt ucwords() das »s« in »War's« nicht groß. Aber die Funktion schreibt auch das »s« in »Schulze-schaumburg« nicht groß. Für ucwords() ist jede Abfolge von Zeichen ohne Leerzeichen ein Wort, wenn es auf ein oder mehrere Leerzeichen folgt. Da sowohl ' als auch - keine Leerzeichen sind, sieht ucwords() das »s« in beiden Fällen nicht als Anfangsbuchstaben eines Worts an.

Sowohl ucfirst() als auch ucwords() lassen die Groß-/Kleinschreibung aller nicht am Anfang stehenden Buchstaben unverändert:

```
print ucfirst('macWorld sagt, ich sollte ein iBook kaufen');
print ucwords('eTunaFish.com kauft itunaFish.Com!');
MacWorld sagt, ich sollte ein iBook kaufen
ETunaFish.com Kauft ItunaFish.Com!
```

Die Funktionen strtolower() und strtoupper() dagegen arbeiten mit den gesamten Strings und nicht mit nur einzelnen Zeichen. Alle Buchstaben des Alphabets werden durch strtolower() in Kleinbuchstaben umgewandelt, während strtoupper() aus allen Buchstaben Großbuchstaben macht:

```
print strtolower("Ich programmierte den WOPR und den TRS-80.");
print strtoupper('"since feeling is first" ist ein gedicht von e. e. cummings.');
ich programmierte den wopr und den trs-80.
"SINCE FEELING IS FIRST" IST EIN GEDICHT VON E. E. CUMMINGS.
```

Bei der Entscheidung darüber, was Groß- und was Kleinbuchstaben sind, beachten diese Funktionen Ihre Locale-Einstellungen.

Siehe auch

Weitere Informationen über Locale-Einstellungen finden Sie in Kapitel 19; die Dokumentationen zu `ucfirst()` unter *http://www.php.net/ucfirst*, `ucwords()` unter *http://www.php.net/ucwords*, `strtolower()` unter *http://www.php.net/strtolower* und `strtoupper()` unter *http://www.php.net/strtoupper*.

1.7 Funktionen und Ausdrücke in Strings interpolieren

Problem

Sie möchten das Ergebnis einer Funktion oder eines Ausdrucks in einen String einfügen.

Lösung

Verwenden Sie den Operator für die String-Verkettung (.), wenn der einzufügende Wert nicht direkt im String stehen kann:

```
print 'Sie haben '.($_REQUEST['Jungen'] + $_REQUEST['Mädchen']).' Kinder.';
print "Das Wort '$wort' ist ".strlen($wort).' Zeichen lang.';
print 'Sie schulden '.$betraege['Zahlung'].' sofort';
print "Der Kreisdurchmesser beträgt ".$circle->getDiameter().' Zentimeter.';
```

Diskussion

Variablen, Objekt-Eigenschaften und Array-Elemente können Sie in Strings mit doppelten Anführungszeichen direkt einfügen (sofern der Index nicht in Anführungszeichen steht):

```
print "Ich habe $kinder Kinder.";
print "Sie schulden $betraege[Zahlung] sofort.";
print "Der Kreisdurchmesser beträgt $circle->diameter Zentimeter.";
```

Die direkte Interpolation und die String-Verkettung lassen sich auch auf Heredocs anwenden. Die Interpolation durch verkettete Strings kann bei Heredocs etwas merkwürdig aussehen, da der Heredoc-Begrenzer und der Operator für die String-Verkettung in verschiedenen Zeilen stehen müssen:

```
print <<< END
Die Uhrzeit in diesem Moment ist
```

```
END
. strftime('%c') . <<< END
 und morgen ist es
END
. strftime('%c',time() + 86400);
```

Außerdem müssen Sie bei der Arbeit mit Heredocs darauf achten, dass Sie in den String an den richtigen Stellen Leerzeichen einfügen, damit der String insgesamt korrekt erscheint. Im vorhergehenden Beispiel muss sich nach »Die Uhrzeit in diesem Moment ist« am Zeilenende ein Leerzeichen befinden, und bei »und morgen ist es« muss ein Leerzeichen am Anfang und am Ende der Zeile stehen.

Siehe auch

Informationen zur Syntax für die Interpolation von Variablen (wie etwa `${"amount_$i"}`) finden Sie in Rezept 5.4; siehe außerdem die Dokumentationen zum Operator für die String-Verkettung unter *http://www.php.net/language.operators.string*.

1.8 Leerzeichen aus einem String entfernen

Problem

Sie möchten Leerzeichen am Anfang oder am Ende eines Strings entfernen. Sie möchten beispielsweise Benutzereingaben vor dem Validieren bereinigen.

Lösung

Verwenden Sie `ltrim()`, `rtrim()` oder `trim()`. `ltrim()` entfernt Leerraum vom String-Anfang, `rtrim()` vom String-Ende und `trim()` sowohl vom Anfang als auch vom Ende eines Strings:

```
$zipcode = trim($_REQUEST['plz']);
$no_linefeed = rtrim($_REQUEST['text']);
$name = ltrim($_REQUEST['name']);
```

Diskussion

Bei diesen Funktionen sind die folgenden Zeichen als Leerraum definiert: Zeilenvorschub, Wagenrücklauf, Leerzeichen, horizontaler und vertikaler Tab und Null.

Wenn Sie Leerraum abschneiden, sparen Sie Speicherplatz; außerdem kann beispielsweise die Anzeige formatierter Daten oder Texte innerhalb von `<pre>`-Tags präzisier erscheinen. Auch wenn Sie Benutzereingaben vergleichen, sollten Sie zuerst die Daten beschneiden, damit nicht jemand, der versehentlich »98052« als Zip-Code eingibt, einen Fehler korrigieren muss, der in Wirklichkeit keiner ist. Und vor einem exakten Textver-

gleich stellen Sie so auch sicher, dass beispielsweise »Salami\n« und »Salami« als gleich angesehen werden. Sinnvollerweise sollten Sie String-Daten vor dem Speichern in der Datenbank normalisieren, indem Sie die Leerzeichen entfernen.

Die `trim()`-Funktionen können auch benutzerspezifische Zeichen aus einem String entfernen. Sie übergeben die zu entfernenden Zeichen als zweites Argument. Einen Bereich von Zeichen können Sie dabei mit zwei Punkten zwischen dem ersten und dem letzten Zeichen der Gruppe angeben.

```
// Ziffern und Leerzeichen vom Zeilenanfang entfernen.
print ltrim('10 PRINT A$',' 0..9');
// Semikola von Zeilenende entfernen.
print rtrim('SELECT * FROM turtles;',';');
PRINT A$
SELECT * FROM turtles
```

PHP stellt auch `chop()` als Alias für `rtrim()` zur Verfügung. Sie sollten jedoch besser `rtrim()` verwenden, weil das `chop()` in PHP sich anders verhält als das `chop()` in Perl (das ohnehin zugunsten von `chomp()` als veraltet erklärt wurde) und die Verwendung dieser Funktion andere verwirren kann, die Ihre Programme lesen.

Siehe auch

Die Dokumentationen zu `trim()` unter *http://www.php.net/trim*, `ltrim()` unter *http://www.php.net/ltrim* und `rtrim()` unter *http://www.php.net/rtrim*.

1.9 Kommaseparierte Daten zerlegen

Problem

Ihnen liegen Daten im CSV-Format (*Comma Separated Values*) vor, beispielsweise eine aus Excel oder aus einer Datenbank exportierte Datei, und nun möchten Sie die Datensätze und Felder extrahieren und in ein Format bringen, das Sie mit PHP bearbeiten können.

Lösung

Wenn sich die CSV-Daten in einer Datei befinden (oder mittels einer URL verfügbar sind), können Sie die Datei mit `fopen()` öffnen und die Daten mit `fgetcsv()` einlesen. Folgendermaßen geben Sie die Daten in einer HTML-Tabelle aus:

```
$fp = fopen('sample2.csv','r') or die("Kann die Datei nicht öffnen");
print "<table>\n";
while($csv_line = fgetcsv($fp,1024)) {
    print '<tr>';
    for ($i = 0, $j = count($csv_line); $i < $j; $i++) {
        print '<td>'.$csv_line[$i].'</td>';
    }
```

```
    print "</tr>\n";
}
print '</table>\n';
fclose($fp) or die("Kann die Datei nicht schließen");
```

Sollten sich die CSV-Daten nicht in einer Datei befinden, sondern als Zeichenkette in einer Variablen vorliegen, können Sie die Funktion `str_getcsv()` verwenden. Diese funktioniert analog zur hier vorgestellten Funktion `fgetcsv()`.

Diskussion

Das zweite Argument von `fgetcsv()` muss länger sein als die Maximallänge einer Zeile in Ihrer CSV-Datei. (Vergessen Sie nicht, den Leerraum am Ende der Zeile mitzuzählen.) Wenn Sie CSV-Zeilen lesen, die länger als 1 KByte sind, sollten Sie die in diesem Rezept verwendete 1024 so verändern, dass es für Ihre Zeilenlänge ausreicht.

Als drittes Argument können Sie `fgetcsv()` optional ein Begrenzungszeichen übergeben, das an die Stelle des Kommas (,) treten soll. Die Verwendung eines anderen Begrenzers steht jedoch ein wenig im Widerspruch zum Sinn der CSV als einfache Möglichkeit für den Austausch tabellarischer Daten.

Lassen Sie sich nicht dazu verleiten, anstelle von `fgetcsv()` einfach Zeilen einzulesen, die Sie dann mit `explode()` an den Kommata zerteilen. CSV ist viel komplizierter, denn es müssen auch eingebettete Kommata und doppelte Anführungszeichen berücksichtigt werden. Der Verwendung von `fgetcsv()` schützt Sie und Ihren Code vor subtilen Fehlern.

PHP bietet mit der Funktion `fputcsv()` auch das entsprechende Gegenstück zu `fgetcsv()`, um Daten aus einem Array in eine CSV-Datei zu schreiben:

```
$autos = array(
    array('Auto', 'Spitzname',),
    array('Dodge Charger', 'General Lee',),
    array('Ford Mustang Shelby GT', 'Eleanor')
);
$fp = fopen('csvout.csv', 'w');
foreach ($autos as $zeile) {
    fputcsv($fp, $zeile);
}
fclose($fp);
```

Die Funktion `fputcsv()` erwartet als ersten Parameter ein File-Handle und als zweiten Parameter die zu schreibenden Daten als Array. Weitere Parameter können genutzt werden, um ein Trennzeichen sowie ein Begrenzungszeichen für die Daten festzulegen.

Siehe auch

Die Dokumentationen zu `fgetcsv()` unter *http://www.php.net/fgetcsv* und zu `fputcsv()` unter *http://www.php.net/fputcsv*.

1.10 Begrenzte Daten mit fester Länge zerlegen

Problem

Sie müssen Datensätze mit festgelegten Längen in Strings aufteilen.

Lösung

Verwenden Sie `substr()`:

```
$fp = fopen('feste-laenge-saetze.txt','r') or die ("Kann Datei nicht öffnen");
while ($s = fgets($fp,1024)) {
    $fields[1] = substr($s,0,10);  // erstes Feld:  die ersten zehn Zeichen der Zeile
    $fields[2] = substr($s,10,5);  // zweites Feld: die nächsten fünf Zeichen der Zeile
    $fields[3] = substr($s,15,12); // drittes Feld: die nächsten 12 Zeichen der Zeile
    // Diese Funktion macht irgendetwas mit den Feldern.
    process_fields($fields);
}
fclose($fp) or die("Kann Datei nicht schließen");
```

oder `unpack()`:

```
$fp = fopen('feste-laenge-saetze.txt','r') or die ("Kann Datei nicht öffnen");
while ($s = fgets($fp,1024)) {
    // Assoziatives Array mit den Schlüsseln "titel", "autor" und "jahr"
    $fields = unpack('A25titel/A14autor/A4jahr',$s);
    // Diese Funktion macht irgendetwas mit den Feldern.
    process_fields($fields);
}
fclose($fp) or die("Kann Datei nicht schließen");
```

Diskussion

Daten, bei denen jedem Feld eine feste Anzahl von Zeichen pro Zeile zugeteilt ist, können so wie diese Liste mit Büchern, Titeln und Erscheinungsjahren aussehen:

```
$booklist=<<<END
Elmer Gantry             Sinclair Lewis1927
Das Scarlatti-Erbe       Robert Ludlum 1971
Das Parsifal-Mosaik      Robert Ludlum 1982
Sophies Wahl             William Styron1979
END;
```

In jeder Zeile nimmt der Titel die ersten 25 Zeichen ein, der Autorenname die nächsten 14 Zeichen und das Erscheinungsjahr die letzten 4 Zeichen. Wenn Sie die Breite dieser Felder kennen, ist es nahe liegend, dass Sie mit `substr()` die Felder in ein Array zerlegen:

```
$books = explode("\n",$booklist);

for($i = 0, $j = count($books); $i < $j; $i++) {
  $book_array[$i]['titel'] = substr($books[$i],0,25);
  $book_array[$i]['autor'] = substr($books[$i],25,14);
```

```
  $book_array[$i]['jahr'] = substr($books[$i],39,4);
}
```

Wenn Sie aus `$booklist` mit `explode()` ein Zeilen-Array machen, ist der in einer Schleife laufende Code der gleiche, und zwar unabhängig davon, ob er mit einem einzelnen String oder mit einer Reihe von aus einer Datei gelesenen Zeilen operiert.

Sie können die Schleife flexibler gestalten, wenn Sie die Feldnamen und -breiten in einem separaten Array angeben. Dieses Array können Sie dann einer Funktion wie `pc_fixed_width_substr()`, wie in Beispiel 1-3 gezeigt, übergeben, die die Daten zerlegt.

Beispiel 1-3: pc_fixed_width_substr()

```
function pc_fixed_width_substr($fields,$data) {
  $r = array();
  for ($i = 0, $j = count($data); $i < $j; $i++) {
    $line_pos = 0;
    foreach($fields as $field_name => $field_length) {
      $r[$i][$field_name] = rtrim(substr($data[$i],$line_pos,$field_length));
      $line_pos += $field_length;
    }
  }
  return $r;
}

$book_fields = array('titel' => 25,
                     'Autor' => 14,
                     'jahr' => 4);

$book_array = pc_fixed_width_substr($book_fields,$books);
```

Die Variable `$line_pos` enthält jeweils den Beginn eines Felds und wird beim Durchlaufen einer Zeile um die vorhergehende Feldlänge weitergerückt. Sie sollten `rtrim()` benutzen, um Leerzeichen am Ende jedes Felds zu entfernen.

Sie können auch `unpack()` anstelle von `substr()` benutzen, um Felder zu extrahieren. Statt Feldnamen und -breite als assoziatives Array anzugeben, müssen Sie für `unpack()` einen Format-String aufbauen. Ein Extraktor für Felder mit fester Breite, der `unpack()` verwendet, kann wie die Funktion `pc_fixed_width_unpack()` aus Beispiel 1-4 aussehen.

Beispiel 1-4: pc_fixed_width_unpack()

```
function pc_fixed_width_unpack($format_string,$data) {
  $r = array();
  for ($i = 0, $j = count($data); $i < $j; $i++) {
    $r[$i] = unpack($format_string,$data[$i]);
  }
  return $r;
}

$book_array = pc_fixed_width_unpack('A25titel/A14autor/A4jahr',
                                    $books);
```

Da das Format A für unpack() »mit Leerzeichen aufgefüllter String« bedeutet, brauchen Sie mit rtrim() die Leerzeichen am Ende des Strings nicht zu entfernen.

Sobald die Felder durch beide Funktionen in ein $book_array zerlegt wurden, können die Daten als HTML-Tabelle ausgegeben werden, zum Beispiel folgendermaßen:

```
$book_array = pc_fixed_width_unpack('A25titel/A14autor/A4jahr', $books);
print "<table>\n";
// Kopfzeile ausgeben.
print '<tr><td>';
print join('</td><td>',array_keys($book_array[0]));
print "</td></tr>\n";
// Datenzeilen einzeln ausgeben.
foreach ($book_array as $row) {
    print '<tr><td>';
    print join('</td><td>',array_values($row));
    print "</td></tr>\n";
}
print '</table>\n';
```

Wenn Sie Daten mit </td><td> zusammenfügen, erhalten Sie eine Tabellenzeile, die am Anfang kein <td> und am Ende kein </td> enthält. Eine vollständige Tabellenzeile erhalten Sie, wenn Sie vor den zusammengefügten Daten <tr><td> und danach </td></tr> ausgeben.

Die beiden Funktionen substr() und unpack() bieten die gleichen Möglichkeiten, wenn es sich bei den Daten mit fester Länge um Strings handelt. Hierbei ist unpack() ist die bessere Lösung, sofern die Feld-Elemente nicht nur einfache Strings sind.

Siehe auch

Weitere Informationen zu unpack() finden Sie im Rezept 1.13 und unter *http://www.php.net/unpack*; Rezept 4.8 behandelt join().

1.11 Strings aufteilen

Problem

Sie müssen Strings in Teile zertrennen. Sie möchten beispielsweise auf jede Zeile einzeln zugreifen, die ein Benutzer in einem <textarea>-Formularfeld einträgt.

Lösung

Verwenden Sie explode(), wenn ein konstanter String die Teile trennt:

```
$words = explode(' ','Mein Satz ist nicht sehr kompliziert');
```

Verwenden Sie split() oder preg_split(), wenn Sie einen regulären Ausdruck nach POSIX oder Perl benötigen, um den Separator zu beschreiben:

```
$words = split(' +','Dieser Satz  enthält  einige zusätzliche  Leerzeichen.');
$words = preg_split('/\d\. /','Mein Tag: 1. aufstehen 2. mich anziehen 3. Toast essen');
$lines = preg_split('/[\n\r]+/',$_REQUEST['textarea']);
```

Verwenden Sie spliti() oder das /i-Flag für preg_split(), um nach Trennzeichen unabhängig von der Groß- und Kleinschreibung zu suchen:

```
$words = spliti(' x ','31 cm x 22 cm X 9 cm');
$words = preg_split('/ x /i','31 cm x 22 cm X 9 cm');
```

Diskussion

Die einfachste Lösung hiervon ist explode(). Übergeben Sie der Funktion Ihren Trennzeichen-String, den String, der getrennt werden soll, und eine optionale Obergrenze für die Anzahl der zurückzugebenden Elemente:

```
$zwerge = 'Seppl,Schlafmütz,Happy,Brummbär,Hatschi,Pimpel,Chef';
$zwerg_array = explode(',',$zwerge);
```

Jetzt ist $zwerg_array ein Array mit sieben Elementen:

```
print_r($zwerg_array);
Array
(
    [0] => Seppl
    [1] => Schlafmütz
    [2] => Happy
    [3] => Brummbär
    [4] => Hatschi
    [5] => Pimpel
    [6] => Chef
)
```

Wenn die festgelegte Grenze kleiner als die Anzahl der möglichen Teil-Strings ist, enthält der letzte Teil den ganzen Rest:

```
$dwarf_array = explode(',',$zwerge,5);
print_r($zwerg_array);
Array
(
    [0] => Seppl
    [1] => Schlafmütz
    [2] => Happy
    [3] => Brummbär
    [4] => Hatschi,Pimpel,Chef
)
```

Der Separator wird von explode() exakt verglichen. Wenn Sie ein Komma und ein Leerzeichen als Trennzeichen angeben, wird der String nur bei einem von einem Leerzeichen gefolgten Komma getrennt – nicht wenn nur ein Komma oder nur ein Leerzeichen auftritt.

Bei split() haben Sie mehr Flexibilität. Diese Funktion verwendet einen regulären Ausdruck nach POSIX anstelle eines literalen String als Separator:

```
$mehr_zwerge = 'Cheeky,Fatso, Wonder Boy, Chunky,Growly, Groggy, Winky';
$mehr_zwerge_array = split(', ?',$mehr_zwerge);
```

Dieser reguläre Ausdruck trennt bei jedem Komma, dem optional ein Leerzeichen folgen kann; auf diese Weise werden alle neuen Teile korrekt behandelt. Diejenigen, die ein Leerzeichen in ihrem Namen haben, werden nicht zerschnitten, aber alle werden korrekt geteilt, ganz gleich, ob sie lediglich durch ein Komma oder durch ein Komma mit Leerzeichen getrennt sind:

```
print_r($mehr_zwerge_array);
Array
(
    [0] => Cheeky
    [1] => Fatso
    [2] => Wonder Boy
    [3] => Chunky
    [4] => Growly
    [5] => Groggy
    [6] => Winky
)
```

Der Funktion split() ähnlich ist preg_split(), die statt eines POSIX-Prozessors für reguläre Ausdrücke einen Perl-kompatiblen Prozessor verwendet. Bei preg_split() können Sie verschiedene Perl-ähnliche Erweiterungen für reguläre Ausdrücke nutzen. Außerdem gibt es noch einige Tricks; beispielsweise können Sie den Separator-Text in das zurückgegebene String-Array einfügen:

```
$math = "3 + 2 / 7 - 9";
$stack = preg_split('/ *([+\-\/*]) */',$math,-1,PREG_SPLIT_DELIM_CAPTURE);
print_r($stack);
Array
(
    [0] => 3
    [1] => +
    [2] => 2
    [3] => /
    [4] => 7
    [5] => -
    [6] => 9
)
```

Der reguläre Separator-Ausdruck sucht die vier mathematischen Operatoren (+, -, /, *), die optional von Leerzeichen umgeben sein können. Das Flag PREG_SPLIT_DELIM_CAPTURE veranlasst preg_split() dazu, die Übereinstimmungen innerhalb der Klammern als Teil des regulären Separator-Ausdrucks in dem zurückgegebenen String-Array einzufügen. Nur die Klasse der mathematischen Operator-Zeichen steht in Klammern, daher enthält das übergegebene Array keine Leerzeichen.

Siehe auch

Reguläre Ausdrücke werden eingehender in Kapitel 16 erörtert; die Dokumentation zu `explode()` finden Sie unter *http://www.php.net/explode*, zu `split()` unter *http://www.php.net/split* und zu `preg_split()` unter *http://www.php.net/preg-split*.

1.12 Text an bestimmten Zeilenlängen umbrechen

Problem

Sie müssen einen String in Zeilen umbrechen. Beispielsweise möchten Sie einen Text innerhalb der Tags `<pre>` und `</pre>` anzeigen, er muss aber in einem normal großen Browser-Fenster Platz finden.

Lösung

Benutzen Sie `wordwrap()`:

```
$s = "Vor viermal zwanzig und sieben Jahren haben unsere Väter auf diesem Kontinent eine
neue Nation ins Leben gerufen, in Freiheit gezeugt und dem Grundsatz verpflichtet, dass
alle Menschen gleich geschaffen sind.";
print "<pre>\n".wordwrap($s)."\n</pre>";
<pre>
Vor viermal zwanzig und sieben Jahren haben unsere Väter auf diesem
Kontinent eine neue Nation ins Leben gerufen, in Freiheit gezeugt und dem
Grundsatz verpflichtet, dass alle Menschen gleich geschaffen sind.
</pre>
```

Diskussion

Standardmäßig bricht `wordwrap()` den Text nach 75 Zeichen pro Zeile um. Mit einem optionalen zweiten Argument kann eine andere Zeilenlänge angegeben werden:

```
print wordwrap($s,50);
Vor viermal zwanzig und sieben Jahren haben unsere
Väter auf diesem Kontinent eine neue Nation ins
Leben gerufen, in Freiheit gezeugt und dem
Grundsatz verpflichtet, dass alle Menschen gleich
geschaffen sind.
```

Auch andere Zeichen als »\n« können für den Zeilenumbruch verwendet werden. Für einen doppelten Zeilenabstand nehmen Sie »\n\n«:

```
print wordwrap($s,50,"\n\n");
Vor viermal zwanzig und sieben Jahren haben unsere

Väter auf diesem Kontinent eine neue Nation ins

Leben gerufen, in Freiheit gezeugt und dem
```

```
Grundsatz verpflichtet, dass alle Menschen gleich

geschaffen sind.
```

Es gibt noch ein optionales viertes Argument für `wordwrap()`, das die Behandlung von Wörtern festlegt, die länger als die angegebene Zeilenlänge sind. Wenn dieses Argument 1 ist, werden solche Wörter umbrochen. Andernfalls überschreiten sie die festgelegte Zeilenlänge:

```
print wordwrap('Donaudampfschifffahrt',10);
print wordwrap('Donaudampfschifffahrt',10,"\n",1);
donaudampfschifffahrt

donaudampf
schifffahrt
```

Siehe auch

Die Dokumentation zu `wordwrap()` unter *http://www.php.net/wordwrap*.

1.13 Binärdaten in einem String speichern

Problem

Sie möchten einen String zerlegen, der als binäre Struktur kodierte Werte enthält, oder Werte in einen solchen String kodieren. Beispielsweise möchten Sie Zahlen in ihrer binären Darstellung speichern und nicht als ASCII-Zeichenfolgen.

Lösung

Verwenden Sie `pack()`, um binäre Daten in einen String zu speichern:

```
$packed = pack('S4',1974,106,28225,32725);
```

Verwenden Sie `unpack()`, um binäre Daten aus einem String zu extrahieren:

```
$nums = unpack('S4',$packed);
```

Diskussion

Das erste Argument für `pack()` ist ein Format-String, der beschreibt, wie die in den übrigen Argumenten übergebenen Daten kodiert werden sollen. Der Format-String `S4` veranlasst `pack()` dazu, aus den Eingabedateien vier unsignierte kurze 16-Bit-Zahlen in rechnerspezifischer Byte-Ordnung zu erzeugen. Angenommen, die Eingabe ist 1974, 106, 2822 und 32725, dann ergibt dieses acht Bytes: 182, 7, 106, 0, 65, 110, 213 und 127. Jedes Zwei-Byte-Paar entspricht einer der übergebenen Zahlen: 7 * 256 + 182 ergibt 1974; 0 * 256 + 106 ist gleich 106; 110 * 256 + 65 = 28225; 127 * 256 + 213 = 32725.

Das erste Argument für unpack() ist ebenfalls ein Format-String, und das zweite Argument sind die zu dekodierenden Daten. Wenn man den Format-String S4 übergibt, erhält man aus der 8-Byte-Sequenz, die pack() erzeugt hat, ein Array mit den vier ursprünglichen Zahlen als Elemente zurück:

```
print_r($nums);
Array
(
    [1] => 1974
    [2] => 106
    [3] => 28225
    [4] => 32725
)
```

Bei unpack() kann dem Zeichenformat und dem Zähler ein String folgen, der als Array-Schlüssel verwendet werden soll. Ein Beispiel:

```
$nums = unpack('S4num',$packed);
print_r($nums);
Array
(
    [num1] => 1974
    [num2] => 106
    [num3] => 28225
    [num4] => 32725
)
```

Mehrere Zeichenformate müssen für unpack() durch / getrennt werden:

```
$nums = unpack('S1a/S1b/S1c/S1d',$packed);
print_r($nums);
Array
(
    [a] => 1974
    [b] => 106
    [c] => 28225
    [d] => 32725
)
```

Die Formatzeichen, die man mit pack() und unpack() verwenden kann, sind in Tabelle 1-2 aufgeführt:

Tabelle 1-2: Formatzeichen für pack() und unpack()

Formatzeichen	Datentypen
a	mit NUL-Zeichen aufgefüllter String
A	mit Leerzeichen aufgefüllter String
h	Hexadezimal-String, niederwertiges Nibble zuerst
H	Hexadezimal-String, höherwertiges Nibble zuerst
c	signed char
C	unsigned char

Tabelle 1-2: Formatzeichen für pack() und unpack() (Fortsetzung)

Formatzeichen	Datentypen
s	signed short (16 Bit, rechnerspezifische Byte-Anordnung)
S	unsigned short (16 Bit, rechnerspezifische Byte-Anordnung)
n	unsigned short (16 Bit, Big-Endian-Byte-Anordnung)
v	unsigned short (16 Bit, Little-Endian-Byte-Anordnung)
i	signed int (rechnerspezifische Länge und Byte-Anordnung)
I	unsigned int (rechnerspezifische Länge und Byte-Anordnung)
l	signed long (32 Bit, rechnerspezifische Byte-Anordnung)
L	unsigned long (32 Bit, rechnerspezifische Byte-Anordnung)
N	unsigned long (32 Bit, Big-Endian-Byte-Anordnung)
V	unsigned long (32 Bit, Little-Endian-Byte-Anordnung)
f	float (rechnerspezifische Länge und Repräsentation)
d	double (rechnerspezifische Länge und Repräsentation)
x	NUL-Byte
X	ein Byte zurück
@	bis zur absoluten Position mit NUL-Zeichen auffüllen

Bei a, A, h und H zeigt eine Zahl nach dem Formatzeichen die String-Länge an. Beispielsweise bezeichnet A25 einen 25 Zeichen langen, mit Leerzeichen aufgefüllten String. Bei anderen Formatzeichen gibt die folgende Zahl an, wie viele Daten dieser Art nacheinander in einem String erscheinen. Mit einem * können Sie alle übrigen verfügbaren Daten erfassen.

Mit unpack() können Sie Konvertierungen zwischen Datentypen vornehmen. In diesem Beispiel wird das Array $ascii mit den ASCII-Werten aller Zeichen in $s gefüllt:

```
$s = 'Schnabeltier';
$ascii = unpack('c*',$s);
print_r($ascii);
Array
(
    [1] => 83
    [2] => 99
    [3] => 104
    [4] => 110
    [5] => 97
    [6] => 98
    [7] => 101
    [8] => 108
    [9] => 116
    [10] => 105
    [11] => 101
    [12] => 114
)
```

Siehe auch

Die Dokumentation zu `pack()` unter *http://www.php.net/pack* und zu `unpack()` unter *http://www.php.net/unpack*.

KAPITEL 2

Zahlen

2.0 Einführung

Im täglichen Leben kann man Zahlen leicht identifizieren. Es gibt Zeitangaben wie 15:00 oder Preise wie EUR 1,29 für einen Liter Milch. Zahlen können wie π (Verhältnis zwischen Kreisumfang und -durchmesser) aussehen. Zahlen können auch ziemlich groß sein, wie die Avogadro-Zahl, die ungefähr 6×10^{23} beträgt. In PHP sind Zahlen für alle diese Dinge geeignet.

Jedoch sind für PHP nicht alle diese Zahlen einfach nur »Zahlen«, sondern sie werden in zwei Gruppen eingeteilt: Integer-Zahlen (ganze Zahlen) und Fließkommazahlen. Integer-Zahlen sind Zahlen wie –4, 0, 5 und 1,975. Fließkommazahlen sind Dezimalzahlen wie –1,23, 0,0, 3,14159 und 9,9999999999.

PHP sorgt aber meistens dafür, dass Sie sich um die Unterschiede zwischen den beiden Zahlentypen keine Gedanken machen müssen, denn Integer-Zahlen werden automatisch in Fließkommazahlen konvertiert und Fließkommazahlen in Integer-Zahlen. So können Sie einfach über die dahinter verborgenen Details hinwegsehen. Es bedeutet auch: `3/2` ist `1.5` und nicht `1`, wie dies in manchen Programmiersprachen der Fall ist. PHP konvertiert Strings ebenfalls automatisch in Zahlen und umgekehrt. So ist beispielsweise `1+"1"` gleich `2`.

Diese für uns angenehme Ignoranz kann jedoch manchmal Probleme verursachen. Erstens können Zahlen nicht unendlich groß oder klein sein; es gibt eine Mindestgröße von 2,2e−308 und ein Maximalgröße von ungefähr 1,8e308.[1] Wenn Sie größere (oder kleinere) Zahlen benötigen, müssen Sie die Bibliotheken BCMath oder GMP benutzen, die in Rezept 2.13 behandelt werden.

1 Tatsächlich sind diese Zahlen plattformspezifisch, aber die Werte sind üblich, weil sie dem 64-Bit-EEE-Standard 754 entsprechen.

Des Weiteren ist nicht garantiert, dass Fließkommazahlen absolut korrekt sind, denn sie sind nur bis auf eine kleine positive oder negative Abweichung genau. Nun sind diese kleinen Abweichungen für die meisten Gelegenheiten klein genug, aber in manchen Situationen können sie Probleme verursachen. Menschen betrachten beispielsweise eine Sechs mit einer nach dem Dezimalkomma folgenden endlosen Kette von Neunen automatisch als eine Sieben. PHP jedoch denkt, dass es eine Sechs mit sehr vielen Neunen ist. Deshalb gibt PHP, wenn Sie nach dem Integer-Wert der Fließkommazahl fragen, 6 zurück und nicht 7. Aus ähnlichen Gründen sind Fließkommazahlen nicht nützlich, wenn eine an der 200. Dezimalstelle stehende Ziffer signifikant sein soll. Auch hier bieten die Bibliotheken BCMath und GMP eine Lösung. Aber in den meisten Fällen verhält sich PHP beim Umgang mit Zahlen benutzerfreundlich, und Sie können Zahlen genau so behandeln, wie Sie es im wirklichen Leben tun.

2.1 Prüfen, ob ein String eine gültige Zahl enthält

Problem

Sie möchten sicherstellen, dass ein String eine Zahl enthält. Beispielsweise möchten Sie eine Altersangabe validieren, die ein Benutzer in das Eingabefeld eines Formulars geschrieben hat.

Lösung

Verwenden Sie `is_numeric()`:

```
if (is_numeric('fünf')) { /* falsch */ }

if (is_numeric(5))      { /* wahr  */ }
if (is_numeric('5'))    { /* wahr  */ }

if (is_numeric(-5))     { /* wahr  */ }
if (is_numeric('-5'))   { /* wahr  */ }
```

Diskussion

Die Funktion `is_numeric()` kann nicht nur mit Zahlen arbeiten, sondern auch auf numerische Strings angewendet werden. Der Unterschied besteht hier darin, dass die ganze Zahl 5 und der String 5 in PHP technisch gesehen nicht dasselbe sind.[2]

2 Ein besonders krasses Beispiel für diesen Unterschied gab es während des Übergangs von PHP 3 zu PHP 4. In PHP 3 gab `empty('0')` den Wert `false` zurück, aber seit PHP 4 liefert dieser Aufruf `true`. Andererseits gab `empty(0)` immer `true` zurück und tut dies immer noch. (Tatsächlich müssen Sie `empty()` mit *Variablen* aufrufen, die `'0'` und `0` enthalten.) Einzelheiten hierzu finden Sie in der Einführung zu Kapitel 5.

Es ist hilfreich, dass `is_numeric()` auch Dezimalzahlen wie `5.1` richtig erkennt; Zahlen mit Tausender-Trennzeichen wie beispielsweise in `5,100` bewirken jedoch, dass `is_numeric()` den Wert `false` zurückgibt.[3]

Um Ihre Zahl von den Tausender-Separatoren zu befreien, bevor Sie `is_numeric()` aufrufen, verwenden Sie `str_replace()`:

```
is_numeric(str_replace($number, ',', ''));
```

Man kann den Typ einer Zahl mit einer Vielzahl von Funktionen überprüfen, die selbsterklärende Namen haben: `is_bool()`, `is_float()` (oder `is_double()`, `is_real()`; sie sind alle gleich) und `is_int()` (oder `is_integer()`, `is_long()`).

Siehe auch

Die Dokumentationen zu `is_numeric()` unter *http://www.php.net/is-numeric* und zu `str_replace()` unter *http://www.php.net/str-replace*.

2.2 Fließkommazahlen vergleichen

Problem

Sie möchten prüfen, ob zwei Fließkommazahlen gleich sind.

Lösung

Benutzen Sie einen kleinen Delta-Wert und überprüfen Sie, ob die Zahlen innerhalb dieses Deltas gleich sind:

```
$delta = 0.00001;

$a = 1.00000001;
$b = 1.00000000;

if (abs($a - $b) < $delta) { /* $a und $b sind gleich */ }
```

Diskussion

Fließkommazahlen werden in binärer Form gespeichert, wobei es für Mantissen und Exponenten nur jeweils eine endliche Anzahl von Bits gibt. Sie bekommen einen Overflow-Fehler, wenn diese Bits überschritten werden. Als Folge davon glaubt PHP manchmal nicht (wie andere Programmiersprachen auch), dass zwei gleiche Zahlen wirklich gleich sind, da sie sich vielleicht ganz am Ende unterscheiden.

3 PHP verwendet Punkte als Dezimalzeichen, wie dies im englischen Sprachraum und bei den meisten Programmiersprachen der Fall ist. Das Komma dient im Englischen als Trennzeichen für Tausender. (Anm. d. Übers.)

Um dieses Problem zu vermeiden, sollten Sie, statt auf `$a == $b` zu prüfen, feststellen, ob der Abstand zwischen der ersten und der zweiten Zahl kleiner ist als ein sehr kleiner Betrag (`$delta`). Die Größe Ihres Delta-Werts sollte die kleinste Abweichung zwischen zwei Zahlen sein, die für Sie noch wesentlich ist. Mit `abs()` erhalten Sie den absoluten Wert des Abstands.

Siehe auch

Rezept 2.3 mit Informationen über das Runden von Fließkommazahlen; die Dokumentation zu Fließkommazahlen in PHP unter *http://www.php.net/language.types.float*.

2.3 Fließkommazahlen runden

Problem

Sie möchten Fließkommazahlen runden, entweder um einen Integer-Wert zu erhalten oder um die Anzahl der Dezimalstellen zu begrenzen.

Lösung

Verwenden Sie `round()`, um eine Zahl auf die nächste Ganzzahl zu runden:

```
$number = round(2.4);   // $number = 2
```

Verwenden Sie `ceil()` zum Aufrunden:

```
$number = ceil(2.4);    // $number = 3
```

Verwenden Sie `floor()` zum Abrunden:

```
$number = floor(2.4);   // $number = 2
```

Diskussion

Wenn eine Zahl genau zwischen zwei Ganzzahlen liegt, rundet PHP von der 0 weg:

```
$number = round(2.5);    // $number ist 3

$number = round(-2.5);   // $number ist -3
```

Wie in Rezept 2.2 erwähnt, ergeben Fließkommazahlen aufgrund der Art und Weise, wie sie intern durch den Computer gespeichert werden, nicht immer exakte Werte. Dies kann zu Situationen führen, in denen eine offensichtlich erscheinende Antwort nicht zutrifft. Ein Wert, von dem Sie erwarten, dass er einen Dezimalteil von »0,5« hat, könnte stattdessen »0,499999...9« (mit vielen Neunen) oder »0,500000...1« (mit vielen Nullen und einer nachfolgenden Eins) sein. PHP bezieht daher automatisch einen kleinen Delta-Wert in Rundungsoperationen mit ein, sodass Sie sich darüber keine Sorgen machen müssen.

Eine festgelegte Anzahl von Ziffern nach dem Dezimaltrenner erhalten Sie, wenn Sie `round()` ein optionales Argument für die Genauigkeit übergeben. So können Sie zum Beispiel den Gesamtpreis aller Waren im Einkaufswagen eines Benutzers ausrechnen:

```
$cart = 54.23;
$tax = $cart * .05;
$total = $cart + $tax;       // $total = 56.9415

$final = round($total, 2);   // $final = 56.94
```

Siehe auch

Rezept 2.2 mit Informationen über den Vergleich von Fließkommazahlen; die Dokumentation zu `round()` unter *http://www.php.net/round*.

2.4 Mit Bereichen von Integer-Zahlen arbeiten

Problem

Sie möchten ein Programmstück auf einen Bereich von Integer-Werten anwenden.

Lösung

Benutzen Sie die Funktion `range()`, die ein Array mit Ganzzahlen zurückgibt:

```
foreach(range($start,$end) as $i) {
    plot_point($i);
}
```

Anstelle von `range()` kann die Verwendung einer `for`-Schleife effizienter sein. Dabei können Sie auch mit anderen Werten als 1 inkrementieren, zum Beispiel:

```
for ($i = $start; $i <= $end; $i += $increment) {
    plot_point($i);
}
```

Diskussion

Schleifen wie diese werden häufig verwendet. Damit können Sie beispielsweise eine Funktion grafisch darstellen und die Werte für verschiedene Punkte auf Ihrem Graph ausrechnen. Wenn Sie die NASA sind, können Sie einen Count-down bis zum Abschuss eines Spaceshuttles durchführen.

Im ersten Beispiel gibt `range()` ein Array mit Werten von `$start` bis `$end` zurück. Dann liest `foreach` jedes einzelne Element aus und weist es innerhalb der Schleife der Variablen `$i` zu. Der Vorteil der Verwendung von `range()` liegt in der Kürze, aber diese Technik hat einige kleine Nachteile. Erstens kann ein großes Array überflüssigen Speicherplatz bean-

spruchen. Außerdem können Sie die Folge immer nur um eins erhöhen, und daher ist es nicht möglich, eine Reihe von geraden Integer-Zahlen in einer Schleife zu durchlaufen.

Seit PHP 4.1 kann `$start` größer als `$end` sein. In diesem Fall sind die von `range()` zurückgegebenen Zahlen in absteigender Reihenfolge. Auch können Sie auf diese Weise Zeichenfolgen durchlaufen:

```
print_r(range('l', 'p'));

Array
(
    [0] => l
    [1] => m
    [2] => n
    [3] => o
    [4] => p
)
```

Die Schleifen-Methode `for` benutzt einfach eine einzelne Ganzzahl und vermeidet das Array völlig. Sie ist zwar länger, aber Sie haben eine größere Kontrolle über die Schleife, da Sie `$i` freier inkrementieren oder dekrementieren können. Außerdem können Sie `$i` aus dem Inneren der Schleife heraus modifizieren; dies ist bei `range()` nicht möglich, weil PHP zu Beginn der Schleife das gesamte Array einliest und Veränderungen des Arrays keine Auswirkungen auf die Abfolge der Elemente haben.

Siehe auch

Rezept 4.3 mit Einzelheiten zur Initialisierung eines Arrays mit einem Bereich von Ganzzahlen; die Dokumentation zu `range()` unter *http://www.php.net/range*.

2.5 Zufallszahlen innerhalb eines Bereichs generieren

Problem

Sie möchten eine Zufallszahl generieren, die in einem Zahlenbereich liegt.

Lösung

Benutzen Sie `mt_rand()`:

```
// Zufallszahl zwischen $upper und $lower einschließlich
$random_number = mt_rand($lower, $upper);
```

Diskussion

Zufallszahlen können sehr hilfreich sein, wenn Sie auf einer Seite ein zufällig ausgewähltes Bild anzeigen, die Startposition in einem Spiel willkürlich bestimmen, per Zufall einen

Datensatz aus einer Datenbank auswählen oder ein eindeutiges Session-Kennzeichen generieren möchten.

Um eine Zufallszahl zwischen zwei Endpunkten zu generieren, übergeben Sie `mt_rand()` zwei Argumente:

```
$random_number = mt_rand(1, 100);
```

Wenn Sie `mt_rand()` ohne Argumente aufrufen, gibt die Funktion eine Zahl zwischen `0` und der größtmöglichen Zufallszahl zurück, die Sie mit `mt_getrandmax()` feststellen können.

Echte Zufallszahlen zu generieren ist für einen Computer eine schwierige Aufgabe. Computer können hervorragend Anweisungen gewissenhaft abarbeiten, Spontaneität ist dagegen nicht ihre Stärke. Wenn Sie einem Computer beibringen wollen, Zufallszahlen zu generieren, müssen Sie ihm eine Anzahl festgelegter, wiederholbarer Befehle geben; die bloße Tatsache, dass sie wiederholbar sind, steht der gewünschten Zufälligkeit entgegen.

PHP hat zwei verschiedene Generatoren für Zufallszahlen: eine klassische Funktion namens `rand()` und eine bessere Funktion mit dem Namen `mt_rand()`. MT steht hier für den Mersenne Twister, der nach dem französischen Mönch und Mathematiker Marin Mersenne und der mit ihm in Verbindung gebrachten Art von Primzahlen benannt ist. Der Algorithmus beruht auf diesen Primzahlen. Da `mt_rand()` zufälliger und schneller arbeitet als `rand()`, ist diese Funktion gegenüber `rand()` zu bevorzugen.

Wenn Sie eine ältere Version von PHP als 4.2 verwenden, müssen Sie den Generator durch einen Aufruf von `mt_srand()` (beziehungsweise `srand()`) mit einem Seed-Wert initialisieren, bevor Sie in einem Skript zum ersten Mal `mt_rand()` (oder `rand()`) benutzen. Der *Seed* ist eine Zahl, die der Zufallsfunktion als Basis für die Generierung der von ihr gelieferten Zufallszahlen dient; auf diese Weise wird das oben erwähnte Dilemma zwischen Wiederholbarkeit und Zufall gelöst. Mithilfe von `microtime()`, einer hochpräzisen Zeitfunktion, erhalten Sie einen Seed-Wert, der sehr schnell wechselt und sich aller Wahrscheinlichkeit nach nicht wiederholt – dies sind für einen guten Seed wünschenswerte Eigenschaften. Nach dem anfänglichen Seed müssen Sie den Zufallszahlengenerator von PHP 4.2 nicht erneut initialisieren. Die Versionen PHP 4.2 und höher nehmen diese Initialisierung automatisch für Sie vor, aber wenn Sie vor dem ersten Aufruf von `mt_rand()` manuell einen Seed bereitstellen, ersetzt PHP diesen nicht durch einen Wert eigener Wahl.

Wenn Sie einen zufälligen Datensatz aus einer Datenbank auswählen wollen – am besten ermitteln Sie zuvor die Gesamtzahl der Zeilen in einer Tabelle –, dann wählen Sie eine Zufallszahl in diesem Bereich aus und fragen anschließend diese Zeile von der Datenbank ab:

```
$sth = $dbh->query('SELECT COUNT(*) AS count FROM quotes');
if ($row = $sth->fetchRow()) {
    $count = $row[0];
} else {
    die ($row->getMessage());
}

$random = mt_rand(0, $count - 1);
```

```
$sth = $dbh->query("SELECT quote FROM quotes LIMIT $random,1");
while ($row = $sth->fetchRow()) {
    print $row[0] . "\n";
}
```

Dieser Ausschnitt liest die Gesamtzahl der Tabellenzeilen aus, errechnet eine Zufallszahl innerhalb dieses Bereichs und verwendet schließlich `LIMIT $random,1`, um eine Zeile der Tabelle, beginnend an der Position `$random`, mit `SELECT` zu lesen.

Alternativ dazu können Sie Folgendes tun, sofern Sie MySQL 3.23 oder höher verwenden:

```
$sth = $dbh->query('SELECT quote FROM quotes ORDER BY RAND() LIMIT 1');
while ($row = $sth->fetchRow()) {
    print $row[0] . "\n";
}
```

In diesem Fall ordnet MySQL die Zeilen per Zufall an und gibt anschließend die erste Zeile zurück.

Siehe auch

Rezept 2.6 dazu, wie man verzerrte Zufallszahlen generieren kann; die Dokumentationen zu `mt_rand()` unter *http://www.php.net/mt-rand* und `rand()` unter *http://www.php.net/rand*; das MySQL-Handbuch über `RAND()` unter *http://dev.mysql.com/doc/mysql/en/Mathematical_functions.html*.

2.6 Verzerrte Zufallszahlen generieren

Problem

Sie möchten Zufallszahlen generieren, wobei diese Zahlen aber ungleichmäßig verteilt sein sollen, sodass Zahlen in bestimmten Bereichen häufiger auftauchen als andere. Beispielsweise möchten Sie eine Reihe von Werbebanner-Einblendungen verteilen, die proportional zur Anzahl der für jede Werbekampagne noch übrigen Einblendungen ist.

Lösung

Verwenden Sie die Funktion `pc_rand_weighted()`, die Beispiel 2-1 zeigt.

Beispiel 2-1: pc_rand_weighted()

```
// Gibt den gewichtet-zufällig ausgewählten Schlüssel zurück
function pc_rand_weighted($numbers) {
    $total = 0;
    foreach ($numbers as $number => $weight) {
        $total += $weight;
        $distribution[$number] = $total;
    }
```

Beispiel 2-1: pc_rand_weighted() (Fortsetzung)

```
        $rand = mt_rand(0, $total - 1);
        foreach ($distribution as $number => $weights) {
            if ($rand < $weights) { return $number; }
        }
    }
```

Diskussion

Stellen Sie sich vor, Sie verwenden statt eines Arrays, in dem die Werte aus der Anzahl der verbleibenden Einblendungen bestehen, ein anderes Array, in dem jede Anzeige genauso oft vorkommt wie die Anzahl der noch verbliebenen Einblendungen. Dann können Sie einfach eine ungewichtete zufällige Stelle innerhalb dieses Arrays herauspicken, und das ist dann die Anzeige, die erscheint.

Diese Technik kann viel Speicherplatz verbrauchen, wenn Sie noch Millionen von Einblendungen haben. Stattdessen können Sie auch ausrechnen, wie groß das Array wäre (indem Sie die verbleibenden Einblendungen zusammenzählen), eine Zufallszahl innerhalb der Größe dieses imaginären Arrays herausgreifen und dann dieses Array durchgehen, wobei Sie ausrechnen, welche Anzeige der von Ihnen herausgegriffenen Zahl entspricht. Zum Beispiel:

```
$ads = array('ford' => 12234, // Inserent, verbleibende Einblendungen
             'att'  => 33424,
             'ibm'  => 16823);

$ad = pc_rand_weighted($ads);
```

Siehe auch

Rezept 2.5 darüber, wie man Zufallszahlen innerhalb eines Bereichs generieren kann.

2.7 Logarithmen berechnen

Problem

Sie möchten den Logarithmus einer Zahl berechnen.

Lösung

Für Logarithmen auf der Basis *e* (natürlicher Logarithmus) verwenden Sie `log()`:

```
$log = log(10);          // 2.30258092994
```

Für Logarithmen auf der Basis 10 verwenden Sie `log10()`:

```
$log10 = log10(10);      // 1
```

Für Logarithmen auf anderen Basen benutzen Sie pc_logn():

```
function pc_logn($number, $base) {
    return log($number) / log($base);
}

$log2  = pc_logn(10, 2); // 3.3219280948874
```

Diskussion

Sowohl log() als auch log10() sind nur für Zahlen definiert, die größer als null sind. Die Funktion pc_logn() wendet eine Formel an, derzufolge der Logarithmus einer Zahl zur Basis n gleich dem natürlichen Logarithmus der Zahl geteilt durch den natürlichen Logarithmus von n ist.

Siehe auch

Die Dokumentationen zu log() unter *http://www.php.net/log* und log10() unter *http://www.php.net/log10*.

2.8 Potenzen berechnen

Problem

Sie möchten einen Exponentialausdruck berechnen.

Lösung

Um eine Potenz von *e* zu berechnen, benutzen Sie exp():

```
$exp = exp(2);        // 7.3890560989307
```

Um eine beliebige Potenz zu berechnen, benutzen Sie pow():

```
$exp = pow( 2, M_E);  // 6.5808859910179

$pow = pow( 2, 10);   // 1024
$pow = pow( 2, -2);   // 0.25
$pow = pow( 2, 2.5);  // 5.6568542494924

$pow = pow(-2, 10);   // 1024
$pow = pow( 2, -2);   // 0.25
$pow = pow(-2, -2.5); // NAN (Fehler: keine Zahl)
```

Diskussion

Die eingebaute Konstante M_E ist ein Näherungswert für die Eulersche Zahl *e*. Sie ist gleich 2.7182818284590452354. Somit sind exp($n) und pow(M_E, $n) identisch.

Mit exp() und pow() kann man leicht sehr große Zahlen erzeugen. Wenn Sie über die maximale Zahlengröße von PHP (annähernd 1.8e308) hinausgehen möchten, lesen Sie in Rezept 2.13 nach, wie man Funktionen mit beliebiger Genauigkeit verwendet. Bei diesen Funktionen gibt PHP INF (unendlich/infinity) zurück, wenn das Ergebnis zu groß ist, und NAN (keine Zahl/not-a-number) im Fall eines Fehlers.

Siehe auch

Die Dokumentationen zu pow() unter *http://www.php.net/pow* und exp() unter *http://www.php.net/exp* sowie Informationen über vordefinierte mathematische Konstanten unter *http://www.php.net/math*.

2.9 Zahlen formatieren

Problem

Sie haben eine Zahl, die Sie mit Tausender- und Dezimalseparatoren ausgeben möchten. Beispielsweise möchten Sie die Preise der Waren in einem Einkaufswagen anzeigen.

Lösung

Benutzen Sie die Funktion number_format(), um als Ganzzahl zu formatieren:

```
$number = 1234.56;
print number_format($number);    // 1235 wegen Aufrundung
```

Geben Sie die Anzahl der Dezimalstellen an, wenn die Zahl als Dezimalzahl formatiert werden soll:

```
print number_format($number, 2); // 1234.56
```

Diskussion

Die number_format()-Funktion formatiert eine Zahl, indem sie die (im englischen Sprachraum üblichen) Dezimal- und Tausender-Separatoren einfügt. Wenn Sie die Trennzeichen manuell angeben möchten, übergeben Sie sie als dritten und vierten Parameter:

```
$number = 1234.56;
print number_format($number, 2, '@', '#'); // 1#234@56
```

Das dritte Argument wird als Dezimalzeichen benutzt, und das letzte trennt die Tausender. Wenn Sie diese Optionen verwenden, müssen beide Argumente angegeben sein.

Standardmäßig rundet number_format() die Zahl auf die nächste Ganzzahl. Wenn Sie die gesamte Zahl erhalten möchten, aber vorher nicht wissen, wie viele Ziffern dem Dezimaltrenner in Ihrer Zahl folgen, können Sie Folgendes benutzen:

```
$number = 1234.56; // Ihre Zahl
list($int, $dec) = explode('.', $number);
print number_format($number, strlen($dec));
```

Siehe auch

Die Dokumentation zu `number_format()` unter *http://www.php.net/number-format*.

2.10 Den Plural korrekt ausgeben

Problem

Sie möchten Wörter, die auf den Werten von Variablen basieren, korrekt in den Plural setzen. Beispielsweise geben Sie einen Text zurück, der von der Trefferzahl einer Suche abhängt.

Lösung

Benutzen Sie einen Konditionalausdruck:

```
$number = 4;
print "Ihre Suche hat $number " . ($number == 1 ? 'Ergebnis' : 'Ergebnisse') . '.';

Ihre Suche hat 4 Ergebnisse.
```

Diskussion

Folgendermaßen können Sie die Zeile geringfügig kürzer schreiben:

```
print "Ihre Suche hat $number Ergebnis" . ($number == 1 ? '' : 'se') . '.';
```

Bei unregelmäßigen Pluralbildungen (zum Beispiel »Haus« und »Häuser«) finden wir es klarer, wenn man das gesamte Wort austauscht und nicht nur einzelne Buchstaben.

Eine weitere Möglichkeit besteht in der Verwendung einer Funktion für sämtliche Pluralbildungen, wie die Funktion `pc_may_pluralize()` in Beispiel 2-2 zeigt:[4]

Beispiel 2-2: pc_may_pluralize()

```
function pc_may_pluralize($singular_word, $amount_of) {

    // Array mit besonderen Pluralbildungen
    $plurals = array(
        'fish' => 'fish',
        'person' => 'people',
```

4 Anm. d. Übers.: Wegen der vielen Unregelmäßigkeiten bietet sich eine solche allgemeine Funktion für die deutsche Sprache weniger an. Das Beispiel bezieht sich daher auf englische Texte.

Beispiel 2-2: pc_may_pluralize() (Fortsetzung)

```
    );

    // Nur eins
    if (1 == $amount_of) {
        return $singular_word;
    }

    // Mehr als eins, spezieller Plural
    if (isset($plurals[$singular_word])) {
        return $plurals[$singular_word];
    }

    // Mehr als eins, normaler Plural: 's' an das Wort hängen
    return $singular_word . 's';
}
```

Dies sind einige Beispiele:

```
$number_of_fish = 1;
print "I ate $number_of_fish " . pc_may_pluralize('fish', $number_of_fish) . '.';

$number_of_people = 4;
print 'Soylent Green is ' . pc_may_pluralize('person', $number_of_people) . '!';

I ate 1 fish.
Soylent Green is people!
```

Wenn solche Pluralformen an verschiedenen Stellen Ihres Codes vorkommen, kann der Einsatz einer Funktion wie `pc_may_pluralize()` die Lesbarkeit erhöhen. Um die Funktion zu verwenden, übergeben Sie `pc_may_pluralize()` die Singularform des Worts als erstes Argument und die Anzahl als zweites. Innerhalb der Funktion gibt es ein großes Array `$plurals`, das alle Spezialfälle enthält. Wenn die Anzahl `$amount` den Wert `1` hat, gibt sie das ursprüngliche Wort zurück. Wenn der Wert größer ist und eine spezielle Pluralform existiert, gibt die Funktion diese zurück. In allen anderen Fällen hängt sie einfach ein »s« an das Ende des Worts.

2.11 Trigonometrische Funktionen berechnen

Problem

Sie möchten trigonometrische Funktionen wie Sinus, Kosinus und Tangens verwenden.

Lösung

PHP unterstützt viele trigonometrische Funktionen von Hause aus: `sin()`, `cos()` und `tan()`:

```
$cos = cos(2.1232);
```

Sie können auch deren Umkehrungen verwenden: `asin()`, `acos()`und `atan()`:

```
$atan = atan(1.2);
```

Diskussion

Diese Funktionen gehen davon aus, dass ihre Argumente als Bogenmaß und nicht in Graden angegeben sind (siehe Rezept 2.12, falls dies ein Problem ist).

Die Funktion `atan2()` übernimmt die beiden Variablen `$x` und `$y` und errechnet `atan($x/$y)`. Sie gibt jedoch immer das richtige Vorzeichen zurück, weil sie beide Parameter berücksichtigt, wenn sie die Quadranten des Ergebnisses bestimmt.

Für den Sekans, den Kosekans und den Kotangens berechnen Sie manuell die reziproken Werte von `sin()`, `cos()`und `tan()`:

```
$n = .707;

$secans    = 1 / sin($n);
$cosecans  = 1 / cos($n);
$cotangens = 1 / tan($n);
```

Seit PHP 4.1 können Sie auch die folgenden hyperbolischen Funktionen verwenden: `sinh()`, `cosh()` `tanh()` sowie natürlich `asin()`, `cosh()` und `atanh()`. Die Umkehrfunktionen werden jedoch unter Windows nicht unterstützt.

Siehe auch

Rezept 2.12 zur Ausführung trigonometrischer Operationen in Graden anstelle von Bogenmaßen; die Dokumentationen zu `sin()` unter *http://www.php.net/sin*, `cos()` unter *http://www.php.net/cos*, `tan()` unter *http://www.php.net/tan*, `asin()` unter *http://www.php.net/asin*, `acos()` unter *http://www.php.net/acos*, `atan()` unter *http://www.php.net/atan* und `atan2()` unter *http://www.php.net/atan2*.

2.12 Trigonometrische Funktionen mit Graden anstelle von Bogenmaßen berechnen

Problem

Sie haben Zahlen als Gradangaben und möchten trotzdem die trigonometrischen Funktionen verwenden.

Lösung

Benutzen Sie `deg2rad()` und `rad2deg()` für Ihre Ein- und Ausgaben:

```
$cosinus = rad2deg(cos(deg2rad($grad)));
```

Diskussion

Per Definition sind 360 Grad gleich dem Bogenmaß 2π, man kann also leicht manuell zwischen diesen beiden Formaten konvertieren. Diese Funktionen verwenden jedoch den PHP-internen Wert für π, daher können Sie eine Antwort mit hoher Präzision erwarten. Um auf diese Zahl für andere Berechnungen zuzugreifen, verwenden Sie die Konstante `M_PI`, die 3.1415926535897932384 beträgt.

Es gibt keine eingebaute Unterstützung für Neugrad. Dies wird als Feature angesehen, nicht als Bug.

Siehe auch

Rezept 2.11 zu trigonometrische Grundlagen; die Dokumentationen zu `deg2rad()` unter *http://www.php.net/deg2rad* und `rad2deg()` unter *http://www.php.net/rad2deg*.

2.13 Mit sehr großen oder kleinen Zahlen arbeiten

Problem

Sie müssen Zahlen verwenden, die zu groß (oder zu klein) für die eingebauten Fließkommazahlen von PHP sind.

Lösung

Verwenden Sie eine der Bibliotheken BCMath oder GMP.

Wenn Sie BCMath benutzen:

```
$sum = bcadd('1234567812345678', '8765432187654321');

// $sum ist jetzt der String '9999999999999999'
print $sum;
```

Und bei GMP:

```
$sum = gmp_add('1234567812345678', '8765432187654321');

// $sum ist nun eine GMP-Ressource und kein String;
// konvertieren Sie sie mit gmp_strval()
print gmp_strval($sum);
```

Diskussion

Die BCMath-Bibliothek zu benutzen ist nicht schwierig. Sie geben Ihre Zahlen als Strings ein, und die jeweilige Funktion gibt die Summe (oder die Differenz, das Produkt usw.) als

String zurück. Die Reichweite der Aktionen, die Sie auf Zahlen anwenden können, beschränkt sich bei BCMath auf die Grundrechenarten.

Die GMP-Bibliothek ist seit PHP 4.0.4 verfügbar. Zwar akzeptieren die meisten GMP-Funktionen auch Integer-Zahlen und Strings als Argumente, aber sie übergeben Zahlen eher in der Form von Ressourcen, die im Prinzip Zeiger auf die eigentlichen Zahlen darstellen. Anders als die BCMath-Funktionen, die Strings zurückgeben, geben die GMP-Funktionen also nur Ressourcen zurück. Sie übergeben dann die Ressource einer beliebigen GMP-Funktion, wo sie die Rolle einer Zahl übernimmt.

Die einzige Schattenseite zeigt sich, wenn Sie die Ressource mit einer Nicht-GMP-Funktion betrachten oder verwenden wollen. Dann müssen Sie sie mithilfe von `gmp_strval()` oder `gmp_intval()` explizit konvertieren.

GMP-Funktionen sind großzügig in Bezug auf das, was sie akzeptieren, zum Beispiel:

```
$four = gmp_add(2, 2);            // Sie können Integer-Werte übergeben
$eight = gmp_add('4', '4');       // oder Strings
$twelve = gmp_add($four, $eight); // oder GMP-Ressourcen.
print gmp_strval($twelve);        // Gibt 12 aus.
```

Mit GMP-Zahlen können Sie viel mehr machen als nur addieren. Beispielsweise können Sie auch Potenzen berechnen, große Fakultäten schnell ausrechnen, den größten gemeinsamen Teiler finden und andere exotische Berechnungen durchführen:

```
// Die Potenz einer Zahl berechnen.
$pow = gmp_pow(2, 10);             // 1024

// Große Fakultäten sehr schnell berechnen.
$factorial = gmp_fact(20);         // 2432902008176640000

// Den größten gemeinsamen Teiler (GCD, Greatest Common Denominator) finden.
$gcd = gmp_gcd (123, 456);         // 3

// Weitere exotische Berechnungen.
$legdendre = gmp_legendre(1, 7);   // 1
```

Die Bibliotheken BCMath und GMP sind nicht notwendigerweise bei allen PHP-Konfigurationen aktiviert. Von PHP 4.0.4 an aufwärts ist BCMath mit PHP gebündelt und damit wahrscheinlich verfügbar. GMP wird jedoch nicht mit PHP geliefert und muss daher heruntergeladen und installiert werden, außerdem müssen Sie bei der Konfiguration PHP instruieren, dass es GMP verwendet. Anhand der Werte von `function_defined('bcadd')` und `function_defined('gmp_init')` stellen Sie fest, ob Sie BCMath und GMP verwenden können.

Siehe auch

Die Dokumentationen zu BCMath unter *http://www.php.net/bc* und GMP unter *http://www.php.net/gmp*.

2.14 Zwischen Zahlensystemen konvertieren

Problem

Sie müssen eine Zahl von einer Basis zu einer anderen Basis konvertieren.

Lösung

Benutzen Sie die Funktion `base_convert()`:

```
$hex = 'a1';                          // Hexadezimalzahl (Basis 16)

// Von der Basis 16 zur Basis 10 konvertieren.
$decimal = base_convert($hex, 16, 10); // $decimal ist jezt 161.
```

Diskussion

Die Funktion `base_convert()` verwandelt einen numerischen String auf einer Basis zu dem entsprechenden String auf einer anderen Basis. Sie funktioniert für alle Basen von 2 bis 36 einschließlich. Dabei benutzt sie die Buchstaben a bis z als zusätzliche Symbole für Basen über 10. Das erste Argument ist die zu konvertierende Zahl, darauf folgt die Basis, in der sie sich befindet, und die Basis, zu der konvertiert werden soll.

Es gibt auch einige wenige spezialisierte Funktionen für die Konvertierung in und von Basis 10 und die häufig verwendeten anderen Basen 2, 8 und 16. Sie heißen `bindec()` und `decbin()`, `octdec()` und `decoct()` sowie `hexdec()` und `dechex()`:

```
// Zur Basis 10 konvertieren.
print bindec(11011); // 27
print octdec(33);    // 27
print hexdec('1b');  // 27

// Von der Basis 10 konvertieren.
print decbin(27);    // 11011
print decoct(27);    // 33
print dechex(27);    // 1b
```

Eine andere Möglichkeit ist die Verwendung der Funktion `sprintf()`, mit deren Hilfe Sie Dezimalzahlen in Binär-, Oktal- und Hexdezimalzahlen konvertieren können. Außerdem enthält sie eine große Bandbreite an Formatierungsmöglichkeiten, darunter führende Nullen und die Wahl zwischen Groß- und Kleinschreibung bei Hexadezimalzahlen.

Angenommen, Sie möchten HTML-Farbwerte ausgeben:

```
printf('#%02X%02X%02X', 0, 102, 204); // #0066CC
```

Siehe auch

Die Dokumentation zu `base_convert()` unter *http://www.php.net/base-convert* und die `sprintf()`-Formatierungsoptionen unter *http://www.php.net/sprintf*.

2.15 Mit anderen Zahlen als Dezimalzahlen rechnen

Problem

Sie möchten mathematische Operationen mit Zahlen ausführen, die nicht als Dezimalzahlen, sondern als Oktal- oder Hexadezimalzahlen formatiert sind. Beispielsweise möchten Sie websichere Farben in Hexadezimal-Darstellung ausrechnen.

Lösung

Stellen Sie der Zahl ein führendes Symbol voran, an der PHP erkennt, dass es sich nicht um Basis 10 handelt. Die folgenden Werte sind alle gleich:

```
0144  // Basis 8
 100  // Basis 10
0x64  // Basis 16
```

Hier sehen Sie, wie man von 1 bis 15 (dezimal) zählen kann, dabei aber eine hexadezimale Darstellung verwendet:

```
for ($i = 0x1; $i < 0x10; $i++) { print "$i\n"; }
```

Diskussion

Auch wenn Sie hexadezimal formatierte Zahlen in einer `for`-Schleife benutzen, werden normalerweise alle Zahlen als Dezimalzahlen ausgegeben. Mit anderen Worten, der Code in der Lösung gibt nicht »..., 8, 9, a, b, ...« aus. Um Hexadezimalzahlen auszugeben, sollten Sie eine der in Rezept 2.14 aufgeführten Methoden verwenden. Hier ist ein Beispiel:

```
for ($i = 0x1; $i < 0x10; $i++) { print dechex($i) . "\n"; }
```

Bei den meisten Berechnungen sind Dezimalzahlen einfacher zu benutzen. Manchmal jedoch ist es logischer, auf eine andere Basis überzuwechseln, zum Beispiel wenn Sie die 216 websicheren Farben verwenden. Jeder Web-Farbcode hat die Form *RRGGBB*, wobei *RR* die rote Farbe, *GG* die grüne Farbe und *BB* die blaue Farbe ist. Jede Farbe ist in Wirklichkeit eine zweistellige hexadezimale Zahl zwischen 0 und FF.

Das Besondere an websicheren Farben ist, dass *RR*, *GG* und *BB* jeweils eine der folgenden sechs Zahlen sein muss: 00, 33, 66, 99, CC und FF (in dezimaler Schreibung: 0, 51, 102, 153, 204, 255). So ist 003366 websicher, aber 112233 nicht. Web-sichere Farben lassen sich ohne Farbvermischung auf einem 256-Farben-Display anzeigen.

Wenn Sie eine Liste dieser Zahlen erzeugen, verwenden Sie in der folgenden Schleife eine hexadezimale Notation, um die hexadezimale Grundlage dieser Liste deutlich zu machen:

```
for ($rr = 0; $rr <= 0xFF; $rr += 0x33)
    for ($gg = 0; $gg <= 0xFF; $gg += 0x33)
        for ($bb = 0; $bb <= 0xFF; $bb += 0x33)
            printf("%02X%02X%02X\n", $rr, $gg, $bb);
```

In den Schleifen werden alle möglichen websicheren Farben berechnet. Statt sie jedoch mit Dezimalzahlen zu durchlaufen, benutzen Sie die hexadezimale Notation und verdeutlichen damit den hexadezimalen Zusammenhang zwischen den Zahlen. Geben Sie die Werte mithilfe von `printf()` aus, um sie als groß geschriebene hexadezimale Zahlen zu formatieren, die mindestens zwei Ziffern lang sind. Zahlen mit einer Ziffer werden dabei mit einer vorangestellten Null ausgegeben.

Siehe auch

Rezept 2.14 für Details zur Konvertierung zwischen Zahlensystemen; Kapitel 3, »Webdesign-Prinzipien für Designer aus der Druckbranche«, in *Webdesign in a Nutshell* (O'Reilly Verlag).

KAPITEL 3

Datum und Zeit

3.0 Einführung

Auf den ersten Blick scheint die Darstellung und Manipulation von Datums- und Zeitwerten einfach zu sein, aber je verschiedenartiger und komplizierter Ihre Leserschaft zusammengesetzt ist, desto schwieriger wird es. Verteilen sich Ihre Benutzer beispielsweise über mehr als eine Zeitzone? Vielleicht werden Ihre Leser auch durch Zeitstempel wie »2009-07-18 14:56:34 CET« verschreckt und sollten lieber mit vertrauteren Darstellungsweisen wie »Samstag, den 18. Juli 2009 (14:56 Uhr)« beruhigt werden. Die Zahl der Stunden zwischen heute 10 Uhr und heute 19 Uhr zu berechnen ist recht einfach. Aber wie sieht es mit 3 Uhr und dem Mittag des ersten Tages des nächsten Monats aus? Die Rezepte 3.5 and 3.6 behandeln die Berechnung von Differenzen zwischen Datumswerten.

Solche Berechnungen und Manipulationen werden durch die Sommerzeit (Daylight Saving Time, DST) noch unübersichtlicher. Dank der Sommerzeit gibt es Zeitwerte, die gar nicht existieren (in den europäischen Ländern zwischen 2 und 3 Uhr am letzten Sonntag des Monats März), und Zeitwerte, die es zweimal gibt (in Europa zwischen 2 Uhr und 3 Uhr des letzten Sonntags im Oktober). Einige Ihrer Benutzer können an Orten wohnen, an denen die Sommerzeit gilt, während dies bei anderen nicht der Fall ist. Die Rezepte 3.12, 3.13 und 3.14 zeigen Möglichkeiten für die Arbeit mit Zeitzonen und der Sommerzeit.

Der Umgang mit der Zeit wird in Programmen viel einfacher, wenn man zwei Konventionen folgt: Erstens sollten Sie die Zeit stets als Coordinated Universal Time (abgekürzt UTC, auch bekannt als GMT, Greenwich Mean Time) behandeln, der Mutter aller Zeitzonen, für die es keine Sommerzeit gibt. Es ist die Zeitzone am Längengrad 0, und alle anderen Zeitzonen werden durch deren (positiven oder negativen) Abstand zu dieser Zeitzone definiert. Behandeln Sie zweitens die Zeit nicht als ein Array mit einzelnen Werten für Tag, Monat, Jahr, Minute, Sekunde usw., sondern als Sekunden, die seit dem Unix-Epochenbeginn vergangen sind: seit dem 1. Januar 1970 Mitternacht (natürlich

UTC). Dadurch wird die Berechnung von Intervallen viel einfacher, und PHP verfügt über eine Fülle von Funktionen, die Sie bei Umwandlungen zwischen Epochen-Zeitstempeln und von Menschen lesbaren Zeitdarstellungen unterstützen.

Die Funktion `mktime()` generiert einen Epochen-Zeitstempel aus einer Reihe angegebener Zeitbestandteile, während `date()` einen formatieren Zeit-String liefert, wenn man ihr einen Epochen-Zeitstempel übergibt. Mithilfe dieser Funktionen können Sie zum Beispiel herausfinden, auf welchen Wochentag der Neujahrstag 1986 fiel:

```
$stamp = mktime(0,0,0,1,1,1986);
print date('l',$stamp);
Wednesday
```

Die Funktion `mktime()` gibt hier den Zeitstempel für den 1. Januar 1986 um Mitternacht zurück. Das Formatzeichen `l` veranlasst die Funktion `date()`, den vollständigen Namen des Wochentags (in englischer Sprache) zurückzugeben, der dem angegebenen Epochen-Zeitstempel entspricht. Rezept 3.4 geht auf die Einzelheiten der für `date()` verfügbaren Formatzeichen ein.

In diesem Buch bezieht sich der Ausdruck *Epochen-Zeitstempel* auf einen Zähler für die Sekunden seit dem Beginn der Unix-Epoche. *Zeitbestandteile* (oder *Datumsbestandteile* beziehungsweise *Zeit- und Datumsbestandteile*) bezeichnet ein Array oder eine Gruppe von Zeit- und Datumskomponenten wie Tag, Monat, Jahr, Stunde, Minute und Sekunde. Mit *formatierter Zeit-String* (oder *formatierter Datums-String* usw.) ist ein String gemeint, der eine bestimmte Anordnung von Zeit- und Datumsbestandteilen enthält, zum Beispiel »2009-03-12«, »Mittwoch, 11:23 A.M.« oder »25. Februar«.

Wenn Sie den Epochen-Zeitstempel zur internen Repräsentation der Zeit verwendeten, konnten Sie damit auch alle Y2K-Probleme vermeiden, denn die Differenz zwischen 946702799 (1999-12-31 23:59:59 UTC) und 946702800 (2000-01-01 00:00:00 UTC) wird genau wie jede andere Differenz zwischen zwei Zeitstempel behandelt. Allerdings könnten Sie ein Y2038-Problem bekommen. Am 19. Januar 2038 3:14:07 (UTC) werden seit dem 1. Januar 1970 genau 2147483647 Sekunden vergangen sein. Was das Besondere an 2147483647 ist? Diese Zahl ist gleich $2^{31} - 1$, und das ist die größte Ganzzahl, die durch eine 32 Bit lange Integer-Zahl mit Vorzeichen ausgedrückt werden kann. (Das 32. Bit wird für das Vorzeichen benötigt.)

Wie sieht die Lösung aus? Irgendwann vor dem 19. Januar 2038 sollten Sie zu einer Hardware übergehen, die zum Beispiel 64-Bit-Speicher für die Zeit verwendet. Das hilft Ihnen dann über die nächsten 292 Milliarden Jahre hinweg. Bereits 39 Bits würden bis etwa zum Jahr 10680 reichen, über den Zeitpunkt hinaus, an dem der Y10K-Bug die durch Kaltfusion angetriebenen Fabriken der Erde ebenso außer Betrieb gesetzt haben wird wie die überlichtschnellen Weltraumstationen. Das Jahr 2038 mag Ihnen noch weit entfernt erscheinen, aber so dachten die COBOL-Programmierer der 1950er und 1960er auch schon über das Jahr 2000. Wiederholen Sie nicht deren Fehler!

Mit PHP 5.3 hat sich diese Situation bereits gebessert. Das objektorientierte Interface der Datumserweiterung arbeitet ab dieser PHP-Version mit 64 Bit. Wenn Sie also ausschließlich diese Klassen verwenden und nicht auf die bisher bekannten Funktionen bauen, umgehen Sie das beschriebene Problem der Epochen-Zeitstempel:

```
$j   = 2057;
$m   = 9;
$d   = 8;
$st  = 14;
$min = 0;
$sek = 0;

$zeit1 = mktime($st, $min, $sek, $m, $d, $j);
var_dump($zeit1);

$zeit2 = new DateTime("$j-$m-$d $st:$min:$sek");
print $zeit2->format(DateTime::ISO8601);
var_dump($zeit2->getTimestamp());
bool(false)
2057-09-08T14:00:00+0100
bool(false)
```

Wie Sie an der Ausgabe des Codebeispiels erkennen, ist die Klasse `DateTime` in der Lage, mit einem Datum außerhalb der 32-Bit-Grenze der Epochen-Zeitstempel zu arbeiten. Versuchen wir aber, über die Funktion `mktime()` einen Zeitstempel zu erzeugen, liefert diese den Rückgabewert `false`. Wenn es Ihnen möglich ist, sollten Sie also auf das objektorientierte Interface der Datumserweiterung von PHP setzen.

3.1 Das aktuelle Datum und die aktuelle Zeit feststellen

Problem

Sie möchten wissen, welches Datum oder welche Zeit Sie haben.

Lösung

Mit `strftime()` oder `date()` erhalten Sie einen formatierten Zeit-String:

```
print strftime('%c');
print date('r');
Mon 10 Aug 2009 18:23:45 CEST
Mon, 10 Aug 2009 18:23:45 +0100
```

Seit PHP 5.2 können Sie auch mit der Klasse `DateTime` arbeiten. Wenn Sie den Konstruktor aufrufen, ohne irgendwelche Parameter anzugeben, repräsentiert das erzeugte Objekt das aktuelle Datum und die aktuelle Uhrzeit. Mit der Methode `format()` können Sie

einen beliebig formatierten Zeit-String erzeugen. Die Methode erwartet einen Parameter, der das Format festlegt. Dieser folgt den gleichen Regeln wie der erste Parameter der oben gezeigten Funktion `date()`.

```
$now = new DateTime;
print $now->format('r');
Mon, 10 Aug 2009 18:23:45 +0100
```

Verwenden Sie `getdate()` oder `localtime()`, wenn Sie Zeitbestandteile benötigen:

```
$now_1 = getdate();
$now_2 = localtime();
print "$now_1[hours]:$now_1[minutes]:$now_1[seconds]";
print "$now_2[2]:$now_2[1]:$now_2[0]";
18:23:45
18:23:45
```

Diskussion

Die Funktionen `strftime()`, `date()` und `DateTime::format()` können diverse formatierte Zeit- und Datums-Strings generieren; sie werden in Rezept 3.4 detaillierter behandelt. Die beiden Funktionen `localtime()` und `getdate()` liefern dagegen Arrays, deren Elemente verschiedene Bestandteile der angegebenen Zeit enthalten.[1]

Das von `getdate()` zurückgegebene assoziative Array enthält die in Tabelle 3-1 aufgeführten Schlüssel/Wert-Paare.

Tabelle 3-1: Das von getdate() zurückgegebene Array

Schlüssel	Wert
`seconds`	Sekunden
`minutes`	Minuten
`hours`	Stunden
`mday`	Tag des Monats
`wday`	Wochentag, numerisch (Sonntag ist 0, Samstag ist 6)
`mon`	Monat, numerisch
`year`	Jahr, numerisch
`yday`	Tag des Jahres, numerisch (z.B. 299)
`weekday`	Wochentag, textuell, vollständig (z.B. »Friday«)
`month`	Monat, textuell, vollständig (z.B. »January«)

1 Die Darstellung der Zeit mit `strftime()` hängt vom Betriebssystem und dem eingestellten Locale ab. In den Beispielen dieses Kapitels wird davon ausgegangen, dass das Locale für die deutsche Sprache verwendet wird – z.B. mit der Funktion `setlocale(LC_ALL,'German')`. Vgl. auch Kapitel 19 zur Lokalisierung.

Hier sehen Sie zum Beispiel, wie man mit getdate() den Monat, den Tag und das Jahr ausgibt:

```
$a = getdate();
printf('%s %d, %d',$a['month'],$a['mday'],$a['year']);
August 7, 2009
```

Übergeben Sie getdate() einen Epochen-Zeitstempel als Argument, enthält das zurückgegebene Array den diesem Zeitstempel entsprechenden Wert für die lokale Zeit. Beispielsweise ergeben Monat, Tag und Jahr für den Epochen-Zeitstempel 163727100:

```
$a = getdate(163727100);
printf('%s %d, %d',$a['month'],$a['mday'],$a['year']);
March 11, 1975
```

Die Funktion localtime() liefert ein Array mit Zeit- und Datumsbestandteilen. Auch sie kann einen Epochen-Zeitstempel als optionales erstes Argument übernehmen sowie einen Booleschen Wert als zweites Argument. Wenn dieses zweite Argument true ist, gibt localtime() ein assoziatives Array anstelle eines numerisch indizierten Arrays zurück. Die Schlüssel dieses Arrays sind dieselben wie die Member der Struktur tm_struct, die von der C-Funktion localtime() geliefert werden, und sind in Tabelle 3-2 dargestellt.

Tabelle 3-2: Von localtime( ) zurückgegebenes Array

Numerische Position	Schlüssel	Wert
0	tm_sec	Sekunden
1	tm_min	Minuten
2	tm_hour	Stunde
3	tm_mday	Tag des Monats
4	tm_mon	Monat des Jahrs (Januar ist 0)
5	tm_year	Jahre seit 1900
6	tm_wday	Wochentag
7	tm_yday	Tag des Jahres
8	tm_isdst	Gilt die Sommerzeit?

Sehen Sie hier, wie Sie beispielsweise mit localtime() das heutige Datum im Format Monat.Tag.Jahr ausgeben können:

```
$a = localtime();
$a[4] += 1;
$a[5] += 1900;
print "$a[3].$a[4].$a[5]";
7.8.2009
```

Der Monat wird vor der Ausgabe um eins erhöht, da localtime() beim Zählen der Monate mit 0 für Januar beginnt, wir aber 1 ausgeben möchten, wenn der aktuelle

Monat der Januar ist. Dementsprechend wird das Jahr um 1900 erhöht, weil `localtime()` die Jahre mit 0 für 1900 zu zählen beginnt.

Wie `getdate()` akzeptiert auch `localtime()` einen Epochen-Zeitstempel als erstes Argument und liefert dann die Zeitbestandteile für diesen Zeitstempel:

```
$a = localtime(163727100);
$a[4] += 1;
$a[5] += 1900;
print "$a[4]/$a[3]/$a[5]";
11.3.1975
```

Siehe auch

Die Dokumentationen zu `strftime()` unter *http://www.php.net/strftime*, `date()` unter *http://www.php.net/date*, `getdate()` unter *http://www.php.net/getdate* und `localtime()` unter *http://www.php.net/localtime*. Informationen zur Klasse `DateTime` finden Sie unter *http://php.net/manual/class.datetime.php*.

3.2 Datums- und Zeitbestandteile in einen Epochen-Zeitstempel konvertieren

Problem

Sie möchten wissen, welcher Epochen-Zeitstempel einem Satz von Zeit- und Datumsbestandteilen entspricht.

Lösung

Verwenden Sie `mktime()` oder die Klasse `DateTime`, wenn Ihre Zeit- und Datumsbestandteile in der lokalen Zeitzone liegen:

```
// 10. März 1975 19:45:03 Uhr lokale Zeit
$dann = mktime(19,45,3,3,10,1975);
$dann = new DateTime('10 March 1975 19:45:03');
$dann->getTimestamp();
```

Die Funktion `mktime()` liefert direkt einen Zeitstempel vom Typ `Integer`. Wollen Sie ein Objekt der Klasse `DateTime` erzeugen, rufen Sie den Konstruktor auf und übergeben eine Zeichenkette, die das gewünschte Datum repräsentiert. Das Format dieser Zeichenkette entspricht dem der Funktion `strtotime()`, die in Rezept 3.10 genauer beschrieben wird. Den Zeitstempel-Wert erhalten Sie durch Aufruf der ab PHP 5.3 verfügbaren Methode `getTimestamp()`.

Verwenden Sie `gmmktime()`, wenn Ihre Zeit- und Datumsbestandteile in der GMT liegen:

```
// 10. März 1975 19:45:03 Uhr GMT
$dann = gmmktime(19,45,3,3,10,1975);
```

Wenn Sie keine Argumente übergeben, erhalten Sie das Datum und die Zeit in der lokalen oder der UTC-Zeitzone:

```
$jetzt = mktime();
$jetzt_utc = gmmktime();
```

Diskussion

Jede der Funktionen `mktime()` und `gmmktime()` übernimmt die Bestandteile eines Datums und einer Zeit (Stunde, Minute, Sekunde, Monat, Tag, Jahr, Sommerzeit-Flag) und liefert den zugehörigen Unix-Epochen-Zeitstempel. Die Komponenten werden von `mktime()` als lokale Zeitangabe betrachtet, während `gmmktime()` sie als UTC-Datum und -Zeit behandelt. Bei beiden Funktionen gibt es als siebtes, optionales Argument das DST-Flag für die Sommerzeit (1, wenn die Sommerzeit gilt, 0, wenn nicht). Diese Funktionen liefern nur für Zeiten innerhalb der Unix-Epoche sinnvolle Ergebnisse. Die meisten Systeme speichern Zeitstempel in einer 32-Bit-Integer-Zahl mit Vorzeichen, daher bedeutet »in der Epoche« normalerweise zwischen dem 13. Dezember 1901, 8:45:01 Uhr und dem 19. Januar 2038, 3:14:07 Uhr.

Die Klasse `DateTime` verwendet genau wie `mktime()` die lokale Zeit. Über die Funktion `date_default_timezone_set` oder den *php.ini*-Parameter `date.timezone` können Sie die verwendete Zeitzone festlegen. Alternativ können Sie ein Objekt der Klasse `DateTimeZone` als zweites Konstruktor-Argument übergeben.

In dem folgenden Beispiel enthält `$stamp_jetzt` den Zeitstempel für den Moment, in dem `mktime()` aufgerufen wird, und `$stamp_zukunft` den Epochen-Zeitstempel für den 4. Juni 2012, 15:25 Uhr:

```
$stamp_jetzt = mktime();
$stamp_zukunft = mktime(15,25,0,6,4,2012);

print $stamp_jetzt;
print $stamp_zukunft;
1028782421
1338837900
```

Beide Zeitstempel können wiederum `strftime()` übergeben werden, um formatierte Zeit-Strings zu erhalten:

```
print strftime('%c',$stamp_jetzt);
print strftime('%c',$stamp_zukunft);
Don 08 Aug 2002 00:53:41 CEST
Mon 04 Jun 2012 15:25:00 CEST
```

Die obigen Aufrufe von `mktime()` wurden auf einem auf EDT (Sommerzeit im Osten der USA) eingestellten Computer durchgeführt, der gegenüber GMT um vier Stunden nachgeht. Daher liefert der Aufruf von `gmmktime()` einen Epochen-Zeitstempel, der um 14400 Sekunden (vier Stunden) kleiner ist als der von `mktime()` gelieferte:

```
$stamp_jetzt = gmmktime();
$stamp_zukunft = gmmktime(15,25,0,6,4,2012);
```

```
print $stamp_jetzt;
print $stamp_zukunft;
1028768021
1338823500
```

Wenn Sie diese von `gmmktime()` generierten Epochen-Zeitstempel wiederum an `strftime()` weitergeben, erscheinen formatierte Zeit-Strings, die vier Stunden früher liegen:

```
print strftime('%c',$stamp_jetzt);
print strftime('%c',$stamp_zukunft);
Mit 07 Aug 2002 20:53:41 CEST
Mon 04 Jun 2012 11:25:00 CEST
```

Siehe auch

Rezept 3.3 für Informationen darüber, wie man einen Epochen-Zeitstempel zurück in Datums- und Zeitbestandteile konvertiert; die Dokumentationen zu `mktime()` unter *http://www.php.net/mktime* und `gmmktime()` unter *http://www.php.net/gmmktime*. Informationen zu `date_default_timezone_set` finden Sie unter *http://php.net/manual/function.date-default-timezone-set.php*. und zur `DateTime`-Klasse unter *http://php.net/manual/class.datetime.php*.

3.3 Einen Epochen-Zeitstempel in Zeit- und Datumsbestandteile konvertieren

Problem

Sie benötigen einen Satz von Zeit- und Datumsbestandteilen, der einem Epochen-Zeitstempel entspricht.

Lösung

Übergeben Sie der Funktion `getdate()` einen Epochen-Zeitstempel:

```
$time_parts = getdate(163727100);
```

Diskussion

Die von `getdate()` gelieferten Zeitbestandteile werden in Tabelle 3-1 weiter unten ausführlich dargestellt; sie beziehen sich auf die lokale Zeit. Wenn Sie die Zeitbestandteile zu einem bestimmten Epochen-Zeitstempel in einer anderen Zeitzone benötigen, sehen Sie in Rezept 3.12 nach.

Ab PHP 5.3 können Sie auch die `DateTime`-Klasse verwenden. Diese bietet die Methode `setTimestamp()` an, der Sie einen Epochen-Zeitstempel übergeben können. Nach dem

Aufruf dieser Methode repräsentiert das DateTime-Objekt das Datum und die Uhrzeit des Zeitstempels:

```
$time = new DateTime;
$time->setTimestamp(163727100);
```

Siehe auch

Rezept 3.2 dazu, wie man Zeit- und Datumsbestandteile zurück in einen Epochen-Zeitstempel konvertiert; Rezept 3.12 zum Umgang mit Zeitzonen; die Dokumentation zu getdate() unter *http://www.php.net/getdate*. Informationen zur DateTime-Klasse finden Sie unter *http://php.net/manual/class.datetime.php*.

3.4 Datum oder Zeit in einem bestimmten Format ausgeben

Problem

Sie müssen das Datum oder die Zeit in einer speziellen Formatierung ausgeben.

Lösung

Verwenden Sie date() oder strftime():

```
print strftime('%c');
print date('d.m.Y');
Die 28 Jul 2009 11:31:08 CEST
28.07.2009
```

Ab PHP 5.2 können Sie zudem die Klasse DateTime verwenden. Ein Aufruf der Methode format() liefert einen beliebig formatierbaren Zeit-String. Diese Methode erwartet einen Parameter und funktioniert genau so wie die date()-Funktion. Die Methode wird daher nicht gesondert beschrieben bzw. in den nachfolgenden Tabellen aufgeführt.

```
$date = new DateTime;
print $date->format('d.m.Y');
28.07.2009
```

Diskussion

Sowohl date() als auch strftime() sind flexible Funktionen, die formatierte Zeit-Strings mit einer Vielzahl von Komponenten generieren können. Die Formatzeichen für diese Funktionen sind in Tabelle 3-3 aufgeführt. Die Windows-Spalte gibt an, ob das jeweilige Formatzeichen auch von unter Windows laufenden Systemen unterstützt wird.

Tabelle 3-3: Formatzeichen für strftime() und date()

Typ	strftime()	date()	Beschreibung	Bereich	Windows
Stunde	%H	H	Stunde, numerisch, 24-Stunden-Uhr	00–23	Ja
Stunde	%I	h	Stunde, numerisch, 12-Stunden-Uhr	01–12	Ja
Stunde	%k		Stunde, numerisch, 24-Stunden-Uhr, führende Null als Leerzeichen	0–23	Nein
Stunde	%l		Stunde, numerisch, 12-Stunden-Uhr, führende Null als Leerzeichen	1–12	Nein
Stunde	%p	A	Angabe AM oder PM für aktuelles Locale		Ja
Stunde	%P	a	Angabe am/pm für aktuelles Locale		Nein
Stunde		G	Stunde, numerisch, 24-Stunden-Uhr, ohne führende Null	0–23	Nein
Stunde		g	Stunde, numerisch, 12-Stunden-Uhr, ohne führende Null	0–1	Nein
Minute	%M	i	Minute, numerisch	00–59	Ja
Sekunde	%S	s	Sekunde, numerisch	00–61[a]	Ja
Tag	%d	d	Tag des Monats, numerisch	01–31	Ja
Tag	%e		Tag des Monats, numerisch, führende Null als Leerzeichen	1–31	Nein
Tag	%j	z	Tag des Jahres, numerisch	001–366 bei `strftime()`; 0–365 bei `date()`	Ja
Tag	%u		Wochentag, numerisch (Montag ist 1)	1–7	Nein
Tag	%w	w	Wochentag, numerisch (Sonntag ist 0)	0–6	Ja
Tag		j	Tag des Monats, numerisch, ohne führende Null	1–31	Nein
Tag		S	Englischer Ordinalzahl-Zusatz für den Tag des Monats, textuell	»st,« »th,« »nd,« »rd«	Nein
Woche	%a	D	Abgekürzter Name des Wochentags, Text für aktuelles Locale		Ja
Woche	%A	l	Vollständiger Name des Wochentags, Text für aktuelles Locale		Ja
Woche	%U		Wochennummer im Jahr, numerisch; erster Sonntag ist erster Tag der ersten Woche	00–53	Ja
Woche	%V	W	Wochennummer im Jahr nach ISO 8601:1988, numerisch; Woche 1 ist die erste Woche, die mindestens 4 Tage im aktuellen Jahr hat; Montag ist der erste Tag der Woche	01–53	Nein
Woche	%W		Wochennummer im Jahr, numerisch; erster Montag ist erster Tag der ersten Woche	00–53	Ja
Monat	%B	F	Vollständiger Name des Monats, Text für aktuelles Locale		Ja
Monat	%b	M	Abgekürzter Name des Monats, Text für aktuelles Locale		Ja
Monat	%h		Dasselbe wie %b		Nein

Tabelle 3-3: Formatzeichen für strftime() und date() (Fortsetzung)

Typ	strftime()	date()	Beschreibung	Bereich	Windows
Monat	%m	m	Monat, numerisch	01–12	Ja
Monat		n	Monat, numerisch, ohne führende Null	1–12	Nein
Monat		t	Länge des Monats in Tagen, numerisch	28, 29, 30, 31	Nein
Jahr	%C		Jahrhundert, numerisch	00–99	Nein
Jahr	%g		Wie %G, aber ohne das Jahrhundert	00–99	Nein
Jahr	%G		Jahr nach ISO 8601 mit Jahrhundert, numerisch; das vierstellige Jahr entspricht der ISO-Wochennummer; wie %y, jedoch wird, wenn die ISO-Wochennummer zum vorigen oder nächsten Jahr gehört, stattdessen dieses Jahr verwendet		Nein
Jahr	%y	y	Jahr ohne Jahrhundert, numerisch	00–99	Ja
Jahr	%Y	Y	Jahr, numerisch, mit Jahrhundert		Ja
Jahr		L	Flag für Schaltjahr (ja ist 1)	0, 1	Nein
Zeitzone	%z	O	Abstand von GMT in Stunden, +/-HHMM (z.B. –0400, +0230)	–1200–+1200	Ja, ist aber wie %Z
Zeitzone	%Z	T	Zeitzone, Name oder Abkürzung, textuell		Ja
Zeitzone		I	Flag für Sommerzeit (ja ist 1)	0, 1	Nein
Zeitzone		Z	Abstand von GMT in Stunden; westlich von GMT negativ, östlich von GMT positiv	–43200–43200	Nein
Zusammengesetzt	%c		Standardformat für Datum und Zeit im aktuellen Locale		Ja
Zusammengesetzt	%D		Entspricht %m/%d/%y		Nein
Zusammengesetzt	%F		Entspricht %Y-%m-%d		Nein
Zusammengesetzt	%r		Zeitangabe in AM-/PM-Darstellung für aktuelles Locale		Nein
Zusammengesetzt	%R		Zeit in 24-Stunden-Darstellung für aktuelles Locale		Nein
Zusammengesetzt	%T		Zeit in 24-Stunden-Darstellung (wie %H:%M:%S)		Nein
Zusammengesetzt	%x		Standardmäßiges Datumsformat für aktuelles Locale (ohne Zeit)		Ja
Zusammengesetzt	%X		Standardmäßiges Zeitformat für aktuelles Locale (ohne Datum)		Ja
Zusammengesetzt		r	Nach RFC 822 formatiertes Datum (z.B. »Thu, 20 Aug 2009 16:01:07 +0200«)		Nein
Andere	%s	U	Sekunden seit der Epoche		Nein
Andere		B	Swatch-Internet-Zeit		Nein
Formatierung	%%		Literales Zeichen %		Ja
Formatierung	%n		Zeichen für Zeilenumbruch		Nein
Formatierung	%t		Zeichen für Tabulator		Nein

[a] Der Bereich für Sekunden reicht bis 61, um Schaltsekunden zu berücksichtigen.

Das erste Argument ist bei beiden Funktionen ein Format-String, und das zweite Argument ist ein Epochen-Zeitstempel. Wenn Sie das zweite Argument weglassen, verwenden beide Funktionen das aktuelle Datum und die aktuelle Zeit. Während `date()` und `strftime()` mit der lokalen Zeit arbeiten, haben sie jeweils ein UTC-orientiertes Gegenstück (`gmdate()` und `gmstrftime()`).

Die Formatzeichen für `date()` sind PHP-spezifisch, während `strftime()` die Funktion `strftime()` aus der C-Bibliothek verwendet. Daher ist `strftime()` leichter verständlich für Programmierer, die von einer anderen Sprache zu PHP kommen, allerdings ist das Verhalten dadurch auch auf verschiedenen Plattformen etwas unterschiedlich. Windows unterstützt nicht so viele `strftime()`-Formatbefehle wie die meisten Unix-basierten Systeme. Außerdem erwartet `strftime()`, dass jedem Formatzeichen ein `%` vorangestellt ist (denken Sie an `printf()`), daher kann man mit dieser Funktion leichter Strings mit vielen interpolierten Zeit- und Datumswerten erzeugen.

Zum Beispiel kann man am 15. Juli 2002 für folgende Anzeige:

```
Es ist nach 12 Uhr am 15. Juli 2002
```

den folgenden Code mit `strftime()` verwenden:

```
print strftime("Es ist nach %H Uhr am %d. %B %Y");
```

Mit `date()` sieht das so aus:[2]

```
print "Es ist nach ".date('H').' Uhr am '.date('d. F Y');
```

Zeichen, die nichts mit dem Datum zu tun haben, sind für `strftime()` akzeptabel, da diese Funktion nach dem `%`-Zeichen sucht, um zu entscheiden, wo die entsprechende Zeitinformation interpoliert werden soll. Bei `date()` gibt es dagegen kein solches Kennzeichen, daher können Sie hier in den Format-String fast nur Leer- und Interpunktionszeichen einfügen. Wenn Sie den für `strftime()` vorgesehenen Format-String der Funktion `date()` übergeben:

```
print date("Es ist nach %H Uhr am %d. %B %Y");
```

kommt dabei höchstwahrscheinlich nicht heraus, was Sie sich vorstellen:

```
E44 494431 7pmc12 %12 102673018412Mon, 15 Jul 2002 12:49:44 +0200 pm07 %15. %492 %2002
```

Um mit `date()` leicht zu interpolierende Zeitbestandteile zu generieren, gruppieren Sie alle zum Datum gehörenden Teile von `date()` in einen String. Dabei trennen Sie die verschiedenen Komponenten durch ein Zeichen, das von `date()` nicht in etwas anderes übersetzt wird und das nicht selbst in Ihrem Teil-String vorkommt. Dann wenden Sie `explode()` mit diesem Trennzeichen an und speichern dadurch alle Teile des von `date()` gelieferten Ergebnisses in ein Array; dieses können Sie einfach in den Ausgabe-String interpolieren:

```
$ar = explode(':',date("H:d. F Y"));
print "Es ist nach $ar[0] Uhr am $ar[1]";
```

2 Mit `date()` wird der Monatsname auf Englisch dargestellt.

Bei Verwendung der date()-Funktion oder der DateTime-Klasse können Sie vordefinierte Konstanten zur formatierten Ausgabe von Daten nutzen:

```
print date(DATE_ISO8601);
$date = new DateTime;
print $date->format(DateTime::ISO8601);
2009-06-13T13:37:39+0200
2009-06-13T13:37:39+0200
```

Auf der Seite *http://php.net/manual/class.datetime.php#datetime.constants.types* finden Sie eine Übersicht über die verfügbaren Konstanten.

Siehe auch

Die Dokumentationen zu date() unter *http://www.php.net/date* und strftime() unter *http://www.php.net/strftime*; zu Unix-basierten Systemen siehe *man strftime* bezüglich Ihrer systemspezifischen Optionen für strftime(); unter Windows siehe *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vclib/html/_crt_strftime.2c_.wcsftime.asp* zu Einzelheiten über strftime(). Informationen zur DateTime-Klasse finden Sie unter *http://php.net/manual/class.datetime.php*.

3.5 Die Differenz zwischen zwei Datumswerten berechnen

Problem

Sie möchten die Zeitdifferenz zwischen zwei Datumswerten berechnen. Beispielsweise möchten Sie einem Benutzer mitteilen, wie viel Zeit seit der letzten Anmeldung bei Ihrer Site vergangen ist.

Lösung

Konvertieren Sie beide Datumswerte in Epochen-Zeitstempel und subtrahieren Sie den einen vom anderen. Verwenden Sie den folgenden Code, um die Differenz in Wochen, Tage, Stunden, Minuten und Sekunden aufzuteilen:

```
// 10. Mai 1965 19:32:56 Uhr
$epoch_1 = mktime(19,32,56,5,10,1965);
// 20. November 1962 04:29:11 Uhr
$epoch_2 = mktime(4,29,11,11,20,1962);

$diff_seconds  = $epoch_1 - $epoch_2;
$diff_weeks    = floor($diff_seconds/604800);
$diff_seconds -= $diff_weeks   * 604800;
$diff_days     = floor($diff_seconds/86400);
$diff_seconds -= $diff_days    * 86400;
$diff_hours    = floor($diff_seconds/3600);
$diff_seconds -= $diff_hours   * 3600;
```

```
$diff_minutes  = floor($diff_seconds/60);
$diff_seconds -= $diff_minutes * 60;

print "Zeitdifferenz: $diff_weeks Wochen, $diff_days Tage, ";
print "$diff_hours Stunden, $diff_minutes Minuten und $diff_seconds ";
print "Sekunden sind zwischen beiden Zeitpunkten vergangen.";
Zeitdifferenz: 128 Wochen, 6 Tage, 14 Stunden, 3 Minuten
und 45 Sekunden sind zwischen beiden Zeitpunkten vergangen.
```

Beachten Sie, dass die Differenz nicht in größere Einheiten als Wochen (z.B. Monate oder Jahre) aufgeteilt werden. Diese Einheiten haben variable Längen und ergeben daher keine genaue Angabe der Zeitdifferenz.

Diskussion

Hier geschehen einige merkwürdige Dinge, derer Sie sich bewusst sein sollten. Zunächst einmal liegen 1962 und 1965 vor dem Beginn der Epoche. Glücklicherweise versagt `mktime()` auf freundliche Weise und erzeugt in beiden Fällen einen negativen Zeitstempel. Die ist in Ordnung, da der absolute Zeitwert bei den fraglichen Zeitstempeln nicht benötigt wird, sondern nur die Differenz zwischen beiden. Solange die Epochen-Zeitstempel für die Datumswerte in den Bereich eines Integer-Werts mit Vorzeichen fallen, wird deren Differenz korrekt berechnet.

Des Weiteren würde eine Wanduhr (oder ein Kalender) einen etwas anderen Betrag für die Zeitveränderung zwischen den beiden Datumswerten anzeigen, da sie sich auf verschiedenen Seiten der Sommerzeit-Umstellung befinden. Das Ergebnis der Subtraktion zwischen den Epochen-Zeitstempeln ergibt den korrekten Betrag für die *abgelaufene* Zeit, aber der vom Menschen wahrgenommene Zeitunterschied weicht um eine Stunde ab. Was ist beispielsweise am letzten Sonntagmorgen im März, wenn die Sommerzeit aktiviert wird, die Differenz zwischen 01:30 und 04:30 Uhr? Es scheinen drei Stunden zu sein, aber die Epochen-Zeitstempel für die beiden Zeiten liegen nur 7200 Sekunden auseinander – zwei Stunden. Wenn eine örtliche Uhr um eine Stunde vorgestellt (oder im Oktober um eine Stunde zurückgedreht) wird, hat dies keine Auswirkung auf den stetigen Lauf des Epochen-Zeitstempels. In Wirklichkeit sind auch nur zwei Stunden vergangen, aber aufgrund der Uhrzeit-Manipulation scheinen es drei Stunden zu sein.

Wenn Sie die tatsächlich abgelaufene Zeit messen möchten (was normalerweise der Fall sein wird), ist dieses Vorgehen in Ordnung. Wenn Sie aber an der Differenz dessen interessiert sind, was die Uhr an zwei Zeitpunkten anzeigt, verwenden Sie zur Berechnung des Intervalls julianische Tage, wie es in Rezept 3.6 gezeigt wird.

Um einem Benutzer die seit der letzten Anmeldung vergangene Zeit anzuzeigen, müssen Sie die Differenz zwischen der jetzigen und der letzten Anmeldezeit herausfinden:

```
$epoch_1 = time();
$r = mysql_query("SELECT UNIX_TIMESTAMP(last_login) AS login
                  FROM user WHERE id = $id") or die();
$ob = mysql_fetch_object($r);
```

```
$epoch_2 = $ob->login;

$diff_seconds  = $epoch_1 - $epoch_2;
$diff_weeks    = floor($diff_seconds/604800);
$diff_seconds -= $diff_weeks   * 604800;
$diff_days     = floor($diff_seconds/86400);
$diff_seconds -= $diff_days    * 86400;
$diff_hours    = floor($diff_seconds/3600);
$diff_seconds -= $diff_hours   * 3600;
$diff_minutes  = floor($diff_seconds/60);
$diff_seconds -= $diff_minutes * 60;

print "Sie haben sich vor $diff_weeks Wochen, $diff_days Tagen, ";
print "$diff_hours Stunden, $diff_minutes Minuten und "
print "$diff_seconds Sekunden zum letzten Mal angemeldet.";
```

Siehe auch

Rezept 3.6 zur Berechnung der Differenz zwischen zwei Datumswerten mit julianischen Tagen; Rezept 3.11 zur Addition und Subtraktion mit einem Datum; die Dokumentation zur MySQL-Funktion `UNIX_TIMESTAMP()` unter *http://dev.mysql.com/doc/mysql/en/Date_and_time_functions.html*.

3.6 Den Abstand zwischen zwei Datumswerten über julianische Tage ermitteln

Problem

Sie möchten die Differenz zwischen zwei Datumswerten ermitteln, wobei aber die Anzeige der Uhr maßgeblich sein soll und nicht die tatsächlich abgelaufene Zeit.

Lösung

Seit PHP 5.3 können Sie mit den Klassen `DateTime` und `DateInterval` arbeiten, die es sehr einfach machen, den Abstand zwischen zwei Daten zu berechnen:

```
$datum1 = new DateTime('10 May 1965 19:32:56');
$datum2 = new DateTime('20 November 1962 04:29:11');
$intervall = $datum1->diff($datum2);
```

Sollten Sie eine PHP-Version vor 5.3 einsetzen, erhalten Sie mithilfe von `gregoriantojd()` das julianische Datum für eine Gruppe von Datumsbestandteilen. Subtrahieren Sie den einen julianischen Tag von dem anderen und finden Sie so die Datumsdifferenz. Dann konvertieren Sie die Zeitbestandteile in Sekunden und subtrahieren die eine Zeit von der anderen, um die Zeitdifferenz zu ermitteln. Wenn die Zeitdifferenz kleiner als 0 ist, ver-

mindern Sie die Datumsdifferenz um eins und korrigieren die Zeitdifferenz, sodass sie auf den vorangehenden Tag zutrifft. Dies ist der zugehörige Code:

```
$diff_date = gregoriantojd($date_1_mo, $date_1_dy, $date_1_yr) -
             gregoriantojd($date_2_mo, $date_2_dy, $date_2_yr);
$diff_time = $date_1_hr * 3600 + $date_1_mn * 60 + $date_1_sc -
             $date_2_hr * 3600 - $date_2_mn * 60 - $date_2_sc;
if ($diff_time < 0) {
   $diff_date--;
   $diff_time = 86400 - $diff_time;
}
```

Diskussion

Seit PHP 5.3 ist es sehr leicht, die zeitliche Differenz zu berechnen, die zwischen zwei Objekten der Klasse `DateTime` liegt. Die Methode `DateTime::diff()` liefert als Ergebnis ein Objekt der Klasse `DateInterval`, das wie folgt aussieht:

```
$datum1 = new DateTime('10 May 1965 19:32:56');
$datum2 = new DateTime('20 November 1962 04:29:11');
$intervall = $datum1->diff($datum2);
var_dump($intervall);
object(DateInterval)#3 (8) {
  ["y"]=>
  int(2)
  ["m"]=>
  int(5)
  ["d"]=>
  int(20)
  ["h"]=>
  int(15)
  ["i"]=>
  int(3)
  ["s"]=>
  int(45)
  ["invert"]=>
  int(1)
  ["days"]=>
  int(902)
}
```

Um die Informationen des `DateInterval`-Objekts zu verwenden, können Sie direkt auf die entsprechende Eigenschaft zugreifen oder die `format()`-Methode verwenden, die entsprechende Formatzeichen (z.B. %y für das Jahr) bereitstellt:

```
print $intervall->format(
    '%y Jahre, %m Monate, %d Tage, %h Stunden, %i Minuten, %s Sekunden');
print $intervall->days . " Tage";
print $intervall->h . " Stunden";
print $intervall->i . " Minuten";
print $intervall->s . " Sekunden";
2 Jahre, 5 Monate, 20 Tage, 15 Stunden, 3 Minuten, 45 Sekunden
```

```
902 Tage
15 Stunden
3 Minuten
45 Sekunden
```

In PHP-Versionen vor 5.3 nutzen Sie die Funktion `gregoriantojd()`. Dadurch dass Sie die Differenzen mit julianischen Tagen berechnen, operieren Sie außerhalb des Bereichs der Epochen-Sekunden und beziehen die Sommerzeitabweichungen mit ein. Angenommen, Sie haben zwei Arrays mit den Komponenten der beiden Tage:

```
// 10. Mai 1965 19:32:56 Uhr
list($date_1_yr, $date_1_mo, $date_1_dy, $date_1_hr, $date_1_mn, $date_1_sc)=
    array(1965, 5, 10, 19, 32, 56);
// 20. November 1962 04:29:11 Uhr
list($date_2_yr, $date_2_mo, $date_2_dy, $date_2_hr, $date_2_mn, $date_2_sc)=
    array(1962, 11, 20, 4, 29, 11);

$diff_date = gregoriantojd($date_1_mo, $date_1_dy, $date_1_yr) -
             gregoriantojd($date_2_mo, $date_2_dy, $date_2_yr);
$diff_time = $date_1_hr * 3600 + $date_1_mn * 60 + $date_1_sc -
             $date_2_hr * 3600 - $date_2_mn * 60 - $date_2_sc;
if ($diff_time < 0) {
   $diff_date--;
   $diff_time = 86400 - $diff_time;
}
$diff_weeks = floor($diff_date/7); $diff_date -= $diff_weeks * 7;
$diff_hours = floor($diff_time/3600); $diff_time -= $diff_hours * 3600;
$diff_minutes = floor($diff_time/60); $diff_time -= $diff_minutes * 60;

print "Zwischen den beiden Datumswerten liegen $diff_weeks Wochen, ";
print "$diff_date Tage, $diff_hours Stunden, $diff_minutes Minuten ";
print "und $diff_time Sekunden.";
Zwischen den beiden Datumswerten liegen 128 Wochen, 6 Tage,
15 Stunden, 3 Minuten und 45 Sekunden.
```

Bei diesem Vorgehen wird die Zeit auf Basis der Uhrzeit berechnet, und daher ist das Ergebnis um eine Stunde größer als das aus Rezept 3.5. Der 10. Mai liegt in der Sommerzeit und der 11. November in der normalen Zeit.

Die Funktion `gregoriantojd()` gehört zur PHP-Kalendererweiterung, ist also nur verfügbar, wenn diese Erweiterung geladen ist.

Siehe auch

Rezept 3.5 zur Berechnung der Differenz von zwei Datumswerten als abgelaufene Zeit; Rezept 3.11 zum Addieren und Subtrahieren von einem Datum; die Dokumentation `DateTime` unter *http://php.net/manual/class.datetime.php*, zu `DateInterval` unter *http://php.net/manual/class.datetime.php*, zu `gregoriantojd()` unter *http://www.php.net/gregoriantojd*; einen Überblick zum julianischen Datumssystem finden Sie unter *http://tycho.usno.navy.mil/mjd.html*.

3.7 Den Tag der Woche, des Monats, des Jahres oder die Kalenderwoche des Jahres ermitteln

Problem

Sie möchten wissen, welchen Tag oder welche Woche innerhalb des Jahres, welchen Tag in der Woche oder welchen Tag im Monat Sie haben. Beispielsweise möchten Sie an jedem Montag oder jedem Ersten des Monats eine spezielle Mitteilung ausgeben.

Lösung

Verwenden Sie die entsprechenden Argumente für `date()` oder `strftime()`:

```
print strftime("Heute ist der %d. Tag des Monats und der %j. Tag des Jahres.");
print 'Heute ist der '.date('d').'. Tag des Monats und der '.date('z').'.
        Tag des Jahres.';
```

Ab PHP 5.2 können Sie alternativ auch mit der Klasse `DateTime` arbeiten. Wenn Sie ein Objekt dieser Klasse erzeugt haben, können Sie die Methode `format()` aufrufen – diese erwartet einen Parameter zur Definition des Formats und funktioniert wie die zuvor gezeigte `date()`-Funktion:

```
$date = new DateTime;
print 'Heute ist der '.$date->format('d').'. Tag des Monats und der '. $date->
format('z').'. Tag des Jahres.';
```

Diskussion

Die Funktionen `date()` und `strftime()` verhalten sich nicht genau gleich. Die Tage des Jahres beginnen bei `date()` mit 0, bei `strftime()` aber mit 1. Tabelle 3-4 enthält alle Formatzeichen für Tages- und Wochennummern, die von `date()` und `strftime()` verstanden werden.

Tabelle 3-4: Formatzeichen für Tages- und Wochennummern

Typ	strftime()	date()	Beschreibung	Bereich	Windows
Tag	%d	d	Tag des Monats, numerisch	01–31	Ja
Tag	%e		Tag des Monats, numerisch, einstellige Werte mit vorangestelltem Leerzeichen	1-31	Nein
Tag	%j	z	Tag des Jahres, numerisch	001–366 bei `strftime()`; 0–365 bei `date()`	Ja
Tag	%u	N	Wochentag, numerisch (Montag ist 1)	1–7	Nein
Tag	%w	w	Wochentag, numerisch (Sonntag ist 0)	0–6	Ja
Tag		j	Tag des Monats, numerisch, ohne führende Nullen	1-31	Nein

Tabelle 3-4: Formatzeichen für Tages- und Wochennummern (Fortsetzung)

Typ	strftime()	date()	Beschreibung	Bereich	Windows
Tag		S	Anhang der englischen Aufzählung für einen Monatstag, zwei Zeichen	"st", "nd", "rd" oder "th"	Nein
Woche	%a	D	Abgekürzter Names des Wochentags entsprechend dem aktuell verwendeten Locale		Ja
Woche	%A	l	Ausgeschriebener Tag der Woche entsprechend dem aktuell verwendeten Locale		Ja
Woche	%U		Kalenderwoche im Jahr, numerisch; erster Sonntag ist erster Tag der ersten Woche	00–53	Ja
Woche	%V	W	Kalenderwoche im Jahr nach ISO 8601:1988, numerisch; Woche 1 ist die erste Woche, die mindestens 4 Tage im aktuellen Jahr hat; Montag ist der erste Tag der Woche	01–53	Nein
Woche	%W		Kalenderwoche, numerisch, beginnend mit dem ersten Montag als erstem Tag der ersten Woche	00-53	Ja

Um nur an Montagen etwas anzuzeigen, verwenden Sie `date()` mit dem Formatzeichen `w` oder `strftime()` mit dem Formatzeichen `%w`:

```
if (1 == date('w')) {
    print "Willkommen zum Beginn Ihrer Arbeitswoche.";
}

if (1 == strftime('%w')) {
    print "Nur noch 4 Tage bis zum Wochenende!";
}
```

Es gibt verschiedene Möglichkeiten zur Berechnung der Kalenderwoche und des Wochentags; achten Sie also darauf, dass Sie die richtige auswählen. Nach dem ISO-Standard (ISO 8601) beginnt die Woche am Montag, und die Wochentage werden von 1 (Montag) bis 7 (Sonntag) nummeriert. Die Woche Nummer 1 ist die erste Woche, deren Donnerstag in dem Jahr liegt. Die erste Woche des Jahres ist demnach die erste Woche, von der sich die meisten Tage in diesem Jahr (und nicht im alten) befinden. Die Kalenderwoche reichen von 01 bis 53.

Andere Standards für die Nummerierung der Wochen haben den Bereich 00 bis 53, wobei sich wahrscheinlich einige Tage der Woche 53 des einen Jahres mit der Woche 00 des Folgejahres überschneiden.

Solange Sie innerhalb Ihrer Programme konsistent sind, sollte es keine Probleme geben, aber seien Sie vorsichtig, wenn Sie Daten mit anderen PHP-Programmen oder der Datenbank austauschen. Beispielsweise behandelt MySQL in der Funktion `DAYOFWEEK()` dem ODBC-Standard entsprechend den Sonntag als ersten Tag der Woche und nummeriert die Tage von 1 bis 7. Die MySQL-Funktion `WEEKDAY()` dagegen sieht den Montag als ersten Tag der Woche an und nummeriert die Tage von 0 bis 6. Bei der `WEEK()`-Funktion

schließlich können Sie wählen, ob die Woche am Sonntag oder Montag beginnen solle, aber sie ist nicht mit dem ISO-Standard kompatibel.

Siehe auch

Die Dokumentationen zu `date()` unter *http://www.php.net/date* und `strftime()` unter *http://www.php.net/strftime*; die MySQL-Funktionen `DAYOFWEEK()`, `WEEKDAY()` und `WEEK()` sind unter *http://dev.mysql.com/doc/mysql/en/Date_and_time_functions.html* dokumentiert.

3.8 Start- und Enddatum einer Woche errechnen

Problem

Sie möchten das Datum des ersten und letzten Tages einer Woche errechnen.

Lösung

Setzen Sie eine geschickte Kombination aus den Funktionen `date()` und `strtotime()` ein:

```
$jahr = 2009;
$kalenderwoche = 18;

// Montag (=>1)
print date(DateTime::ISO8601, strtotime("{$jahr}-W{$kalenderwoche}"));

// Sonntag (=>7)
print date(DateTime::ISO8601, strtotime("{$jahr}-W{$kalenderwoche}-7"));
2009-04-27T00:00:00+0200
2009-05-03T00:00:00+0200
```

Diskussion

Die Funktion `strtotime()` wandelt eine Zeichenkette, die ein Datum repräsentiert, in einen Epochen-Zeitstempel um. Wir nutzen das aus und übergeben ihr einen String im Format *JJJJ-Wxx* bzw. *JJJJ-Wxx-T*. Die ersten vier Stellen repräsentieren das Jahr. Dem Bindestrich folgt ein *W*, das anzeigt, dass die beiden folgenden Stellen *xx* die Kalenderwoche darstellen. Optional können noch ein Bindestrich und eine Ziffer von 1 bis 7 folgen, die den Tag der Woche darstellt. Dabei steht die 1 für den Montag und die 7 für den Sonntag. Folgt nach der Wochennummer keine weitere Angabe, berechnet `strtotime()` das Datum für den Montag der angegebenen Woche.

Im Beispiel sind die übergebenen Zeichenketten also *2009-W18* (Montag der 18. Kalenderwoche im Jahr 2009) und *2009-W18-7* (Sonntag der 18. Kalenderwoche im Jahr 2009). Es wäre auch möglich, die Angaben ohne Bindestriche zu machen – das Ergebnis bliebe dasselbe.

Siehe auch

Das Rückgabeformat der Funktion `date()` kann sehr detailliert gesteuert werden. Im Beispiel wird eine Konstante der `DateTime`-Klasse von PHP genutzt (`DateTime::ISO8601`), und damit wird festgelegt, dass der Rückgabewert als Zeichenkette im ISO-8601-Datumsformat erfolgen soll. Die `DateTime`-Klasse stellt weitere Konstanten zur Verfügung, die auf der Seite *http://php.net/manual/class.datetime.php#datetime.constants.types* beschrieben werden. Natürlich sind auch sämtliche anderen von `date()` unterstützten Formatierungen möglich: *http://php.net/manual/function.date.php*. Informationen zum ISO-8601-Standard finden Sie z.B. bei Wikipedia unter *http://de.wikipedia.org/wiki/ISO_8601*.

3.9 Ein Datum validieren

Problem

Sie möchten die Gültigkeit eines Datums überprüfen. Zum Beispiel möchten Sie sicherstellen, dass ein Benutzer kein Geburtsdatum wie den 30. Februar 1962 angeben kann.

Lösung

Verwenden Sie `checkdate()`:

```
$valid = checkdate($monat,$tag,$jahr);
```

Diskussion

Die Funktion `checkdate()` gibt `true` zurück, wenn `$monat` zwischen 1 und 12 liegt, `$jahr` zwischen 1 und 32767 sowie `$tag` zwischen 1 und der korrekten Maximalzahl der Tage für `$monat` und `$jahr`. Schaltjahre werden von `checkdate()` korrekt behandelt und dem gregorianischen Kalender entsprechend interpretiert.

Da `checkdate()` einen derart weiten Bereich für gültige Jahre hat, sollten Sie die Benutzereingaben zusätzlich überprüfen, wenn Sie beispielsweise ein gültiges Geburtsdatum erwarten. Das *Guinness Buch der Rekorde* behauptet, das höchste jemals von einer Person erreichte Alter sei 122 Jahre. Um zu prüfen, ob das Alter eines Benutzers entsprechend dem angegebenen Geburtstag zwischen 18 und 122 Jahren liegt, verwenden Sie die in Beispiel 3-1 dargestellte Funktion `pc_checkbirthdate()`.

Beispiel 3-1: pc_checkbirthdate()

```
function pc_checkbirthdate($month,$day,$year) {
    $min_age = 18;
    $max_age = 122;

    if (! checkdate($month,$day,$year)) {
        return false;
```

Beispiel 3-1: pc_checkbirthdate() (Fortsetzung)

```
    }

    list($this_year,$this_month,$this_day) = explode(',',date('Y,m,d'));

    $min_year = $this_year - $max_age;
    $max_year = $this_year - $min_age;

    print "$min_year,$max_year,$month,$day,$year\n";

    if (($year > $min_year) && ($year < $max_year)) {
        return true;
    } elseif (($year == $max_year) &&
              (($month < $this_month) ||
               (($month == $this_month) && ($day <= $this_day)))) {
        return true;
    } elseif (($year == $min_year) &&
              (($month > $this_month) ||
               (($month == $this_month && ($day > $this_day))))) {
        return true;
    } else {
        return false;
    }
}
```

Hier ein paar Beispiele für die Verwendung der Funktion:

```
// Den 3. Dezember 1974 prüfen
if (pc_checkbirthdate(12,3,1974)) {
    print "Sie d&uuml;rfen auf diese Website zugreifen.";
} else {
    print "Sie sind zu jung f&uuml;r diese Website.";
    exit();
}
```

Die Funktion stellt zunächst mithilfe von `checkdate()` sicher, dass die Datumsangaben `$month`, `$day` und `$year` ein gültiges Datum repräsentieren. Danach prüfen diverse Vergleichsoperationen, ob sich das angegebene Datum im Bereich zwischen `$min_age` und `$max_age` befindet.

Wenn sich `$year` innerhalb des nicht-inklusiven Bereichs `$min_year` bis `$max_year` befindet, ist das Datum definitiv im zulässigen Bereich, und die Funktion gibt `true` zurück. Andernfalls sind zusätzliche Prüfungen erforderlich. Wenn `$year` gleich `$max_year` ist (d.h., im Jahr 2002 ist `$year` gleich 1984), muss `$month` vor dem aktuellen Monat liegen. Wenn `$month` gleich dem aktuellen Monat ist, muss `$day` gleich dem aktuellen Tag sein oder davor liegen. Wenn `$year` gleich `$min_year` ist (d.h., im Jahr 2002 ist `$year` gleich 1880), muss `$month` nach dem aktuellen Monat liegen. Wenn `$month` gleich dem aktuellen Monat ist, muss `$day` nach dem aktuellen Tag liegen. Wenn keine dieser Bedingungen erfüllt ist, liegt das angegebene Datum außerhalb des zulässigen Bereichs, und die Funktion gibt `false` zurück.

Die Funktion liefert `true`, wenn das angegebene Datum genau `$min_age` Jahre vor dem aktuellen Tag liegt, aber `false`, wenn es genau `$max_age` nach dem aktuellen Datum liegt. Dies bedeutet, dass die Funktion Ihnen zwar an Ihrem 18. Geburtstag den Zugang gestattet, aber nicht am 123.

Siehe auch

Die Dokumentation zu `checkdate()` unter *http://www.php.net/checkdate*; Informationen über die nach dem *Guinness-Buch* älteste Person sind unter *http://www.guinnessworldrecords.com* zu finden (navigieren Sie zu »The Human Body«, »Age and Youth« und dann zu »Oldest Woman Ever«).

3.10 Datums- und Zeitwerte aus Strings lesen

Problem

Sie müssen ein Datum oder eine Zeit aus einem String in ein Format bringen, das Sie in Berechnungen verwenden können. Beispielsweise möchten Sie Datumsausdrücke wie »last Thursday« in einen Epochen-Zeitstempel konvertieren.

Lösung

Am einfachsten können Sie einen Datums- oder Zeit-String mit der Funktion `strtotime()` zerlegen, die eine Vielzahl von Datums- und Zeit-Strings, die für Menschen lesbar sind, in Epochen-Zeitstempel verwandeln kann:[3]

```
$a = strtotime('march 10'); // Vorgabe ist das aktuelle Jahr.
```

Diskussion

Die Grammatik von `strtotime()` ist ebenso kompliziert wie umfassend, daher machen Sie sich am besten mit dieser Funktion vertraut, indem Sie eine Menge unterschiedlicher Zeitangaben ausprobieren. Wenn Sie an den Grundlagen der Funktion interessiert sind, sehen Sie sich *ext/standard/parsedate.y* in der PHP-Source-Distribution an.

Sollten Sie PHP 5.2 oder höher einsetzen, können Sie alternativ auch mit der `DateTime`-Klasse arbeiten. Bei der Erzeugung eines Objekts per Konstruktoraufruf können Sie als ersten Parameter den gleichen String übergeben wie bei `strtotime()`. Der einzige Unterschied besteht darin, dass die Funktion `strtotime()` einen Epochen-Zeitstempel im `Integer`-Format und der Konstruktoraufruf eine Instanz der Klasse `DateTime` liefert. Haben Sie die Instanz erzeugt, können Sie sämtliche Möglichkeiten nutzen, die Ihnen die Klasse bietet.

3 Die Funktion `strtotime()` kann mit textuellen Daten nur in englischer Sprache umgehen.

Die Funktion `strtotime()` versteht Wörter, die sich auf die aktuelle Zeit beziehen:

```
$a = strtotime('now');
print strftime('%c',$a);
$a = strtotime('today');
print strftime('%c',$a);

Mon 10 Aug 2009 20:35:10 CEST
Mon 10 Aug 2009 20:35:10 CEST
```

Sie versteht verschiedene Möglichkeiten, die Zeit und das Datum zu identifizieren:

```
$a = strtotime('5/12/1994');
print strftime('%c',$a);
$a = strtotime('12 may 1994');
print strftime('%c',$a);

Don 12 May 1994 00:00:00 CEST
Don 12 May 1994 00:00:00 CEST
```

Sie versteht relative Datums- und Zeitangaben:

```
$a = strtotime('last thursday');   // am 12. August 2002
print strftime('%c',$a);
$a = strtotime('2001-07-12 2pm + 1 month');
print strftime('%c',$a);
Don 08 Aug 2002 00:00:00 CEST
Mon 12 Aug 2002 14:00:00 CEST
```

Sie versteht Zeitzonen. Wenn die folgenden Zeilen in einem Computer in der Zeitzone EDT laufen, wird die gleiche Zeit angezeigt:

```
$a = strtotime('2002-07-12 2pm edt + 1 month');
print strftime('%c',$a);

Mon 12 Aug 2002 14:00:00 EDT
```

Wenn aber Folgendes auf einem Rechner in der EDT läuft, gibt er die gleiche Zeit in EDT aus, wenn es 2 Uhr nachmittages in MDT ist (zwei Stunden vor EDT):

```
$a = strtotime('2002-07-12 2pm mdt + 1 month');
print strftime('%c',$a);

Mon 12 Aug 2002 16:00:00 EDT
```

Sind das Datum und die Zeit, die Sie aus einem String herauslesen möchten, in einem vorher bekannten Format angegeben, können Sie anstelle des `strtotime()`-Aufrufs einen regulären Ausdruck bilden, der die verschiedenen von Ihnen benötigten Datums- und Zeitbestandteile herausgreift. Folgendermaßen können Sie beispielsweise ein Datum im Format »YYYY-MM-DD HH:MM:SS« zerlegen, wie es in einem `DATETIME`-Feld unter MySQL erscheint:

```
$date = '1974-12-03 05:12:56';
preg_match('/(\d{4})-(\d{2})-(\d{2}) (\d{2}):(\d{2}):(\d{2})/',$date,$date_parts);
```

Dabei werden Jahr, Monat, Tag, Stunde, Minute und Sekunde in `$date_parts[1]` bis `$date_parts[6]` abgelegt. (`preg_match()` schreibt den gesamten übereinstimmenden Ausdruck in `$date_parts[0]`.)

Mithilfe eines regulären Ausdrucks können Sie das Datum und die Zeit auch aus längeren Strings extrahieren, die noch weitere Informationen enthalten (aus den Benutzereingaben oder einer Datei, die Sie einlesen). Wenn Sie aber genau wissen, an welcher Position im String sich das Datum befindet, geht dies mit `substr()` noch schneller:

```
$date_parts[0] = substr($date,0,4);
$date_parts[1] = substr($date,5,2);
$date_parts[2] = substr($date,8,2);
$date_parts[3] = substr($date,11,2);
$date_parts[4] = substr($date,14,2);
$date_parts[5] = substr($date,17,2);
```

Sie können auch `split()` verwenden:

```
$ar = split('[- :]',$date);
print_r($ar);
Array
(
    [0] => 1974
    [1] => 12
    [2] => 03
    [3] => 05
    [4] => 12
    [5] => 56
)
```

Aber Vorsicht: PHP konvertiert zwar ohne weiteres Zahlen und Strings, aber Zahlen, die mit einer 0 beginnen, werden als Oktalzahlen (mit der Basis 8) angesehen. Daher erhält man bei 03 und 05 zwar 3 und 5, aber 08 und 09 ergeben *nicht* 8 und 9.

`preg_match()` und `strtotime()` sind beim Zerlegen eines Datums, das wie »YYYY-MM-DD HH:MM:SS« formatiert ist, gleichermaßen effizient, aber `ereg()` ist etwa viermal langsamer. Wenn Sie die einzelnen Teile eines Datums-Strings benötigen, geht dies mit `preg_match()` bequemer, dagegen ist `strtotime()` offensichtlich flexibler.

Siehe auch

Die Dokumentation zu `strtotime()` unter *http://www.php.net/strtotime* und zur DateTime-Klasse unter *http://php.net/manual/class.datetime.php*.

3.11 Addition und Subtraktion mit einem Datum

Problem

Sie möchten ein Intervall zu einem Datum addieren oder von einem Datum subtrahieren.

Lösung

Abhängig von der Repräsentation des Datums und des Intervalls verwenden Sie entweder `strtotime()` oder einige arithmetische Operationen.

Wenn Datum und Intervall in einem geeigneten Format vorliegen, geht es mit `strtotime()` am einfachsten:

```
$geburtstag = 'March 10, 1975';
$hurra_geschafft = strtotime("$geburtstag - 9 months ago");
```

Wenn sich das Datum in einem Epochen-Zeitstempel befindet und das Intervall in Sekunden ausgedrückt werden kann, subtrahieren Sie einfach das Intervall vom Zeitstempel:

```
$geburtstag = 163727100;
$schwangerschaft = 36 * 7 * 86400; // 36 Wochen
$hurra_geschafft = $geburtstag - $schwangerschaft;
```

Wenn Sie PHP >= 5.3.0 einsetzen, können Sie sehr komfortabel mit den Klassen `DateTime` und `DateInterval` der ext/date-Erweiterung arbeiten. Diese berücksichtigt zudem Gegebenheiten wie Schaltjahre, Zeitzonen und auch die Sommerzeit.

```
$intervall = new DateInterval('P9M'); // 9 Monate
$geburtstag = new DateTime('March 10, 1975');
print $geburtstag->format(DateTime::ISO8601);
$hurra_geschafft = $geburtstag->add($intervall);
print $hurra_geschafft->format(DateTime::ISO8601);
1975-03-10T00:00:00+0100
1975-12-10T00:00:00+0100
```

Im gezeigten Beispiel beträgt das Intervall 9 Monate (`P9M`). Die Festlegung des Intervalls erfolgt per Konstruktoraufruf. Als Parameter muss ein ISO-8601-Intervall-Format-String in der Form `P1Y2M3DT04H05M06S` angegeben werden. Dieser beispielhaft gezeigte String definiert folgendes Intervall: 1 Jahr, 2 Monate, 3 Tage, 4 Stunden, 5 Minuten, 6 Sekunden. Hat man ein Intervall- und ein Datumsobjekt, kann man bequem das Intervall zum Datum hinzuaddieren oder davon abziehen. Dazu stellt die Klasse `DateTime` die Methoden `add()` und `sub()` zur Verfügung, denen man das gewünschte `DateInterval`-Objekt übergibt.

Diskussion

Ab PHP 5.3 empfiehlt sich der Einsatz der Klassen `DateTime` und `DateInterval`. Die Funktion `strtotime()` eignet sich besonders für Intervalle mit variablen Längen, zum Beispiel Monate. Wenn Sie `strtotime()` nicht verwenden können, konvertieren Sie Ihr Datum in einen Epochen-Zeitstempel und addieren oder subtrahieren das entsprechende Intervall in Sekunden. Dies ist am sinnvollsten bei Intervallen von fester Länge, wie Tage oder Wochen:

```
$jetzt = time();
$naechste_woche = $jetzt + 7 * 86400;
```

Bei dieser Methode können Sie allerdings Probleme bekommen, wenn die Endpunkte Ihres Intervalls auf verschiedenen Seiten einer Sommerzeit-Umstellung liegen. In diesem Fall ist einer Ihrer Tage mit fester Länge nicht 86 400 Sekunden lang; vielmehr sind es in Abhängigkeit von der Jahreszeit 82 800 oder 90 000 Sekunden. Wenn Sie in Ihrer Anwendung ausschließlich UTC verwenden, müssen Sie sich darüber keine Gedanken machen. Aber wenn Sie die lokale Zeit verwenden müssen, können Sie mithilfe des julianischen Datums die Tage ohne Probleme mit diesem Schluckauf zählen. Mit `unixtojd()` und `jdtounix()` können Sie Konvertierungen zwischen Epochen-Zeitstempeln und julianischen Tagen durchführen:

```
$jetzt = time();
$heute = unixtojd($jetzt);
$naechste_woche = jdtounix($heute + 7);
// Vergessen Sie nicht, die Stunden, Minuten und Sekunden
// wieder hinzuzufügen.
$naechste_woche += 3600 * date('H',$jetzt) + 60 * date('i',$jetzt) + date('s',$jetzt);
```

Die Funktionen `unixtojd()` und `jdtounix()` gehören zur PHP-Kalendererweiterung, daher sind sie nur verfügbar, wenn diese Erweiterung geladen ist.

Siehe auch

Rezept 3.5 zur Ermittlung von Differenzen zwischen Datumswerten als abgelaufene Zeit; Rezept 3.6 zur Ermittlung der Differenz zwischen zwei Datumswerten als julianische Tage; die Dokumentationen zu `strtotime()` unter *http://www.php.net/strtotime*, `unixtojd()` unter *http://www.php.net/unixtojd* und `jdtounix()` unter *http://www.php.net/jdtounix*.

3.12 Die Zeit mit Zeitzonen berechnen

Problem

Sie müssen mit Zeiten in unterschiedlichen Zeitzonen rechnen. Beispielsweise möchten Sie Ihren Benutzern Informationen geben, die sich auf deren Ortszeit und nicht auf die Zeit Ihres Servers beziehen.

Lösung

Für einfache Berechnungen können Sie einfach den Abstand zwischen den zwei Zeitzonen addieren oder subtrahieren:

```
// Wenn die lokale Zeit EST ist.
$time_parts = localtime();
// Kalifornien (PST) ist drei Stunden früher.
$california_time_parts = localtime(time() - 3 * 3600);
```

Nutzen Sie PHP 5.1 oder neuer, können Sie mit der Funktion `date_default_timezone_set()` die von PHP verwendete Zeitzone anpassen. Das folgende Beispiel gibt die aktuelle Zeit für zwei unterschiedliche Zeitzonen aus – einmal für New York und einmal für Paris:

```
$now = time();
date_default_timezone_set('America/New_York');
print date('c', $now);
date_default_timezone_set('Europe/Paris');
print date('c', $now);
2009-05-18T05:13:55-04:00
2009-05-18T11:13:55+02:00
```

Noch leichter ist es, wenn Sie mit PHP 5.2 oder neuer arbeiten. Dann stehen Ihnen die Klassen `DateTime` und `DateTimeZone` zur Verfügung, mit denen die Datumsberechnung wiederum einfacher wird:

```
$datum = new DateTime('now', new DateTimeZone('America/New_York'));
print $datum->format(DateTime::ISO8601);
$datum->setTimezone(new DateTimeZone('Europe/Paris'));
print $datum->format(DateTime::ISO8601);
2009-05-18T05:13:55-0400
2009-05-18T11:13:55+0200
```

Auf Unix-basierten Systemen setzen Sie, wenn Sie die Abstände zwischen den Zeitzonen nicht kennen, einfach die Umgebungsvariable `TZ` auf die gewünschte Zeitzone:

```
putenv('TZ=PST8PDT');
$california_time_parts = localtime();
```

Diskussion

Bevor wir tiefer in die Einzelheiten der Zeitzonen einsteigen, möchten wir eine Erklärung weitergeben, die das U.S. Naval Observatory unter *http://tycho.usno.navy.mil/tzones.html* anbietet. Im Einzelnen heißt es dort, dass die offiziellen, weltweiten Zeitzonen etwas fragil sind, »da Staaten souveräne Mächte sind, die ihre Zeiterfassungssysteme nach Belieben ändern können und dies auch tun«. Wir denken daran, dass wir hier den Unberechenbarkeiten der internationalen Beziehungen unterliegen, wenn wir hier einige Möglichkeiten für den Umgang mit vielen Zeitzonen der Erde vorstellen.

Die Datums- und Zeitfunktionen von PHP sind in PHP 5.1 überarbeitet worden, und das Handling von Zeitzonen wurde deutlich verbessert. Die in PHP 5.1 hinzugefügten Funktionen `date_timezone_default_set()` und `date_timezone_default_get()` erleichtern einen Wechsel zwischen unterschiedlichen Zeitzonen immens. Zudem steht die neue Konfigurationsdirektive `date.timezone` zur Voreinstellung der Standard-Zeitzone in der Datei *php.ini* zur Verfügung, die anstelle von `date_timezone_default_set()` verwendet werden kann. Über diese beiden Funktionen müssen Sie nur die gewünschte Zeitzone einstellen und können dann wie gewohnt mit `date()` oder `strftime()` arbeiten. Wenn Sie eine Applikation entwickeln, die Datumsinformationen für Leute in unterschiedlichen Zeitzonen ausgeben soll, bestünde eine gute Technik darin, die Standard-Zeitzone in der *php.*

ini auf `GMT` zu setzen und dann in Ihrem Programmcode vor einer Datumsberechnung die gewünschte Zeitzone (z.B. entsprechend einer Benutzereinstellung) über `date_timezone_default_set()` einzustellen.

Eine weitere Vereinfachung der zeitzonenbezogenen Berechnung von Datumswerten bringen die Klassen `DateTime` und `DateTimeZone`, die ab PHP 5.2 verfügbar sind. Wenn Sie mit Datumswerten arbeiten (`DateTime`-Objekte), können Sie dediziert die Zeitzone, auf die sich das Datum beziehen soll, angeben. Das kann direkt im Konstruktor erfolgen oder über die Methode `DateTime::setTimezone()`.

Zeitzonen, die PHP kennt und unterstützt, finden Sie im PHP-Manual (*http://php.net/manual/timezones.php*). Die Namen der Zeitzonen (z.B. `America/New_York` oder `Africa/Dar_es_Salaam`) entsprechen denen der populären `zoneinfo`-Datenbank.

In früheren PHP-Versionen (vor PHP 5.1) können Sie die Abstände zwischen den Zeitzonen in Ihrem Programm mit einem Array berücksichtigen, das die verschiedenen Zeitverschiebungen der Zeitzonen gegenüber der UTC enthält. Sobald Sie festgestellt haben, in welcher Zeitzone sich ein Benutzer befindet, addieren Sie diese Differenz zu der jeweiligen UTC-Zeit, und diejenigen Funktionen, die UTC-Zeit ausgeben (z.B. `gmdate()`, `gmstrftime()`), können die korrekt angepasste Zeit anzeigen.

```
// Aktuelle Zeit ermitteln.
$jetzt = time();

// Kalifornien ist gegenüber der UTC 8 Stunden zurück.
$jetzt += $pc_timezones['PST'];

// Mit gmdate() oder gmstrftime() die für Kalifornien passende Zeit ausgeben.
print gmstrftime('%c',$jetzt);
```

Der obige Code benutzt das folgende Array `$pc_timezones`, das die Abstände gegenüber der UTC enthält:

```
// Aus Time::Timezone von Perl
$pc_timezones = array(
  'GMT'  =>   0,           // Greenwich Mean
  'UTC'  =>   0,           // Universal (Coordinated)
  'WET'  =>   0,           // Westeuropa
  'WAT'  =>  -1*3600,      // Westafrika
  'AT'   =>  -2*3600,      // Azoren
  'NFT'  =>  -3*3600-1800, // Neufundland
  'AST'  =>  -4*3600,      // USA-Atlantik-Standard
  'EST'  =>  -5*3600,      // USA-Osten-Standard
  'CST'  =>  -6*3600,      // USA-Zentral-Standard
  'MST'  =>  -7*3600,      // USA-Mountain-Standard
  'PST'  =>  -8*3600,      // USA-Pazifik-Standard
  'YST'  =>  -9*3600,      // Yukon-Standard
  'HST'  => -10*3600,      // Hawaii-Standard
  'CAT'  => -10*3600,      // Zentralalaska
  'AHST' => -10*3600,      // Alaska-Hawaii-Standard
  'NT'   => -11*3600,      // Nome
  'IDLW' => -12*3600,      // International Date Line West
```

```
    'CET'  =>  +1*3600,      // Mitteleuropa
    'MET'  =>  +1*3600,      // Mitteleuropa
    'MEWT' =>  +1*3600,      // Mitteleuropa-Winter
    'SWT'  =>  +1*3600,      // Schweden-Winter
    'FWT'  =>  +1*3600,      // Frankreich-Winter
    'EET'  =>  +2*3600,      // Osteuropa, Russland Zone 1
    'BT'   =>  +3*3600,      // Bagdad, Russland Zone 2
    'IT'   =>  +3*3600+1800, // Iran
    'ZP4'  =>  +4*3600,      // Russland Zone 3
    'ZP5'  =>  +5*3600,      // Russland Zone 4
    'IST'  =>  +5*3600+1800, // Indien-Standard
    'ZP6'  =>  +6*3600,      // Russland Zone 5
    'SST'  =>  +7*3600,      // Südsumatra, Russland Zone 6
    'WAST' =>  +7*3600,      // Westaustralien-Standard
    'JT'   =>  +7*3600+1800, // Java
    'CCT'  =>  +8*3600,      // China-Küste, Russland Zone 7
    'JST'  =>  +9*3600,      // Japan-Standard, USSR Zone 8
    'CAST' =>  +9*3600+1800, // Zentralaustralien-Standard
    'EAST' => +10*3600,      // Ostaustralien-Standard
    'GST'  => +10*3600,      // Guam-Standard, Russland Zone 9
    'NZT'  => +12*3600,      // Neuseeland
    'NZST' => +12*3600,      // Neuseeland-Standard
    'IDLE' => +12*3600       // International Date Line East
  );
```

Auf Unix-Systemen können Sie die *zoneinfo*-Bibliothek zum Konvertieren verwenden. Dadurch wird Ihr Code kompakter und die Sommerzeit transparenter, wie in Rezept 3.14 zu sehen.

Um *zoneinfo* aus PHP heraus zu nutzen, führen Sie alle Ihre internen Datumsberechnungen mit Epochen-Zeitstempeln aus. Generieren Sie diese aus den Zeitbestandteilen mit der Funktion `pc_mktime()`, die in Beispiel 3-2 dargestellt wird.

Beispiel 3-2: pc_mktime()

```
function pc_mktime($tz,$hr,$min,$sec,$mon,$day,$yr) {
    putenv("TZ=$tz");
    $a = mktime($hr,$min,$sec,$mon,$day,$yr);
    putenv('TZ=EST5EDT'); //EST5EDT entsprechend der Zeitzone Ihres Servers ändern!
    return $a;
}
```

Durch den Aufruf von `putenv()` vor `mktime()` gaukeln Sie den von `mktime()` benutzten Systemfunktionen eine andere Zeitzone vor. Nach dem `mktime()`-Aufruf muss die korrekte Zeitzone wiederhergestellt werden. An der Ostküste der Vereinigten Staaten ist dies `EST5EDT`. Ändern Sie das auf den zum Standort Ihres Servers passenden Wert (vgl. Tabelle 3-5).

Die Funktion `pc_mktime()` verwandelt Zeitbestandteile in Epochen-Zeitstempel. Das Gegenstück zum Umwandeln von Epochen-Zeitstempeln in formatierte Zeit-Strings und Zeitbestandteile ist `pc_strftime()`, dargestellt in Beispiel 3-3.

Beispiel 3-3: pc_strftime()

```
function pc_strftime($tz,$format,$timestamp) {
    putenv("TZ=$tz");
    $a = strftime($format,$timestamp);
    putenv('TZ=EST5EDT'); // EST5EDT entsprechend der Zeitzone Ihres Servers ändern!
    return $a;
}
```

Dieses Beispiel verwendet die gleiche Manipulation der Systemfunktionen wie `pc_mktime()`, um von `strftime()` die richtigen Ergebnisse zu erhalten.

Das Schöne an diesen Funktionen ist, dass Sie sich keine Gedanken darüber machen müssen, wie groß die Abstände zwischen der UTC und den verschiedenen Zeitzonen sind, ob gerade die Sommerzeit gilt oder ob es andere Unregelmäßigkeiten bei den Zeitzonen-Abständen gibt. Sie setzen einfach die richtige Zeitzone und überlassen alles andere den Systembibliotheken.

Beachten Sie, dass der Wert der Variablen `$tz` bei beiden obigen Funktionen nicht der Name einer Zeitzone, sondern eine *zoneinfo*-Zone ist. *zoneinfo*-Zonen sind spezifischer als Zeitzonen, da sie sich auf bestimmte Orte beziehen. Tabelle 3-5 enthält die Zuordnung der jeweiligen *zoneinfo*-Zonen zu einigen UTC-Abständen. Die letzte Spalte (DST) gibt an, ob es in der Zone eine Sommerzeit gibt.

Tabelle 3-5: zoneinfo-Zonen

UTC-Abstand (Std.)	UTC-Abstand (Sek.)	zoneinfo-Zone	DST?
–12	–43200	Etc/GMT+12	Nein
–11	–39600	Pacific/Midway	Nein
–10	–36000	US/Aleutian	Ja
–10	–36000	Pacific/Honolulu	Nein
–9	–32400	America/Anchorage	Ja
–9	–32400	Etc/GMT+9	Nein
–8	–28800	PST8PDT	Ja
–8	–28800	America/Dawson_Creek	Nein
–7	–25200	MST7MDT	Ja
–7	–25200	MST	Nein
–6	–21600	CST6CDT	Ja
–6	–21600	Canada/Saskatchewan	Nein
–5	–18000	EST5EDT	Ja
–5	–18000	EST	Nein
–4	–14400	America/Halifax	Ja
–4	–14400	America/Puerto_Rico	Nein
–3.5	–12600	America/St_Johns	Ja

Tabelle 3-5: zoneinfo-Zonen (Fortsetzung)

UTC-Abstand (Std.)	UTC-Abstand (Sek.)	zoneinfo-Zone	DST?
–3	–10800	America/Buenos_Aires	Nein
0	0	Europe/London	Ja
0	0	GMT	Nein
1	3600	CET	Ja
1	3600	GMT-1	Nein
2	7200	EET	Nein
2	7200	GMT-2	Nein
3	10800	Asia/Baghdad	Ja
3	10800	GMT-3	Nein
3.5	12600	Asia/Tehran	Ja
4	14400	Asia/Dubai	Nein
4	14400	Asia/Baku	Ja
4.5	16200	Asia/Kabul	Nein
5	18000	Asia/Tashkent	Nein
5.5	19800	Asia/Calcutta	Nein
5.75	20700	Asia/Katmandu	Nein
6	21600	Asia/Novosibirsk	Ja
6	21600	Etc/GMT-6	Nein
6.5	23400	Asia/Rangoon	Nein
7	25200	Asia/Jakarta	Nein
8	28800	Hongkong	Nein
9	32400	Japan	Nein
9.5	34200	Australia/Darwin	Nein
10	36000	Australia/Sydney	Ja
10	36000	Pacific/Guam	Nein
12	43200	Etc/GMT-13	Nein
12	43200	Pacific/Auckland	Ja

Die Sommerzeit beginnt nicht in allen Ländern der Erde am selben Tag oder zur selben Zeit. Um die Zeit für einen internationalen Standort, an dem es die Sommerzeit gibt, korrekt zu berechnen, wählen Sie eine *zoneinfo*-Zone, die mit dem gesuchten Standort so genau wie möglich übereinstimmt.

Siehe auch

Rezept 3.14 zum Umgang mit der Sommerzeit; die Dokumentationen zu `date_default_timezone_set()` unter *http://php.net/manual/function.date-default-timezone-set.php*, zu

date_default_timezone_get() unter *http://php.net/manual/function.date-default-timezone-get.php*, zu DateTime unter *http://php.net/manual/class.datetime.php*, zu DateTimeZone unter *http://php.net/manual/class.datetimezone.php*, zu putenv() unter *http://www.php.net/putenv*, zu localtime() unter *http://www.php.net/localtime*, zu gmdate() unter *http://www.php.net/gmdate* und zu gmstrftime() unter *http://www.php.net/gmstrftime*; PHP bekannte Zeitzonen finden Sie unter *http://php.net/manual/timezones.php*; die *zoneinfo*-Zonennamem sowie Längen- und Breitengrade für Hunderte von Orten rund um die Welt sind unter *ftp://elsie.nci.nih.gov/pub/* verfügbar (suchen Sie nach den neuesten Dateien, deren Namen mit *tzdata* beginnen); zahlreiche Links zu historischen und technischen Informationen über Zeitzonen sind unter *http://www.twinsun.com/tz/tz-link.htm* zu finden.

3.13 Geografische Lageinformationen zu einer Zeitzone bestimmen

Problem

Sie möchten geografische Lageinformationen zu einer Zeitzone ermitteln.

Lösung

Die ext/date-Erweiterung stellt ab PHP 5.3.0 für die Klasse DateTimeZone eine Methode namens getLocation() zur Verfügung. Damit können Sie den Längen- und Breitengrad der gewünschten Zeitzone ermitteln:

```
$tz = new DateTimeZone('Europe/Berlin');
print_r($tz->getLocation());
Array
(
    [country_code] => DE
    [latitude] => 52.5
    [longitude] => 13.36666
    [comments] =>
)
```

Der Rückgabewert der Funktion ist ein Array, das unter anderem den Ländercode sowie den Längen- und Breitengrad der Zeitzone enthält.

Diskussion

Die Informationen, die Sie mit dieser Methode erhalten, können Sie in Ihrer Applikation z.B. für die Lokalisierung verwenden. Sie könnten beispielsweise einen erhöhten Komfort bei der Auswahl einer Zeitzone bieten, indem Sie anstelle eines Drop-down-Felds zusätzlich die Zeitzonen auf einer Landkarte (Google-Maps o.Ä.) visualisieren.

Die Funktion verwendet intern die *timezonedb*, die in der PHP Extension Community Library (PECL) zu finden ist. Diese Erweiterung wird standardmäßig mit PHP ausgeliefert. Ein Download könnte für Sie trotzdem interessant sein, denn die mitgelieferte Datei *timezonedb.h* enthält u.a. die Informationen darüber, welche Zeitzonen in welcher Version der Erweiterung unterstützt werden. Informationen zur *timezonedb*-Extension finden Sie unter *http://pecl.php.net/package/timezonedb*.

Siehe auch

Weitere Information zur `getLocation()`-Methode finden Sie im PHP-Manual unter *http://php.net/manual/datetimezone.getlocation.php*. Allgemeine Informationen zu Zeitzonen liefert z.B. Wikipedia: *http://de.wikipedia.org/wiki/Zeitzone*; PHP bekannte Zeitzonen finden Sie unter *http://php.net/manual/timezones.php*.

3.14 Die Sommerzeit berücksichtigen

Problem

Sie müssen sicherstellen, dass bei Ihren Zeitberechnungen die Sommerzeit korrekt berücksichtigt wird.

Lösung

In PHP-Versionen ab PHP 5.1 können Sie mit der Funktion `date_timezone_default_set()` die gewünschte Zeitzone einstellen. Diese Zeitzonen berücksichtigen die korrekte Einbeziehung der Sommerzeit:

```
// Denver/Colorado mit Sommerzeit
date_default_timezone_set('America/Denver');
// 4. Juli 2008 ist im Sommer
$sommer = mktime(12,0,0,7,4,2008);
print date('c', $sommer);
// Phoenix/Arizona hat keine Sommerzeit
date_default_timezone_set('America/Phoenix');
print date('c', $sommer);
2008-07-04T12:00:00-06:00
2008-07-04T11:00:00-07:00
```

Verfügen Sie über PHP 5.2 oder neuer, können Sie auch sehr komfortabel mit den Klassen `DateTime` und `DateTimeZone` arbeiten:

```
$sommer = new DateTime(
        '2008-07-04T12:00:00', new DateTimeZone('America/Denver'));
print $sommer->format(DateTime::ISO8601);
$sommer->setTimezone(new DateTimeZone('America/Phoenix'));
print $sommer->format(DateTime::ISO8601);
2008-07-04T12:00:00-0600
2008-07-04T11:00:00-0700
```

Sollten Sie PHP < 5.1 einsetzen, müssen Sie eine andere Methode wählen. Die *zoneinfo*-Bibliothek berechnet die Effekte der Sommerzeit korrekt. Wenn Sie ein Unix-basiertes System verwenden, können Sie die Vorteile der *zoneinfo* mit `putenv()` nutzen:

```
putenv('TZ=MST7MDT');
print strftime('%c');
```

Wenn Sie *zoneinfo* nicht verwenden können, verändern Sie fest einprogrammierte Zeitzonen-Abstände in Abhängigkeit davon, ob die örtliche Zeitzone der Sommerzeit unterliegt. Mit `localtime()` können Sie feststellen, ob gerade die Sommerzeit gilt:

```
// Die aktuelle UTC-Zeit ermitteln.
$jetzt = time();

// Kalifornien ist 8 Stunden gegenüber UTC zurück.
$jetzt -= 8 * 3600;

// Ist es die Sommerzeit?
$ar = localtime($jetzt,true);
if ($ar['tm_isdst']) { $jetzt += 3600; }

// Mit gmdate() oder gmstrftime() geben Sie die für Kalifornien zutreffende Zeit aus.
print gmstrftime('%c',$jetzt);
```

Diskussion

Dieses Vorgehen, bei dem Sie den Epochen-Zeitstempel um den Abstand der Zeitzone von der UTC verändern und dann mit `gmdate()` oder `gmstrftime()` der Zeitzone entsprechende Zeitinformationen ausgeben, ist flexibel – es funktioniert von jeder Zeitzone aus –, jedoch sind die Sommerzeit-Berechnungen etwas ungenau. In den kurzen Zeiträumen, in denen der Sommerzeit-Status des Servers von dem der gewünschten Zeitzone abweicht, ist das Ergebnis nicht korrekt. Beispielsweise hat am ersten Sonntag im April um 03:30 Uhr EDT (nach der Umstellung auf die Sommerzeit) die Umschaltung in der Pazifik-Zeitzone (wo es dann 23:30 Uhr ist) noch nicht stattgefunden. Ein Server in der EDT-Zeitzone, der diese Methode anwendet, berechnet die kalifornische Zeit mit sieben Stunden nach der UTC, während es in Wirklichkeit acht Stunden sind. Ab 06:00 Uhr EDT (das entspricht 03:00 Uhr PDT) gilt in beiden Zeitzonen die Sommerzeit, und die Berechnung ist wieder korrekt (wobei Kalifornien sieben Stunden gegenüber der UTC zurückgesetzt wird).

Siehe auch

Rezept 3.12 zum Umgang mit Zeitzonen; die Dokumentationen zu `date_default_timezone_set()` unter *http://php.net/manual/function.date-default-timezone-set.php*, zu `DateTime` unter *http://php.net/manual/class.datetime.php*, zu `DateTimeZone` unter *http://php.net/manual/class.datetimezone.php*, zu `putenv()` unter *http://www.php.net/putenv*, zu `localtime()` unter *http://www.php.net/localtime*, zu `gmdate()` unter *http://www.php.net/*

gmdate und zu `gmstrftime()` unter *http://www.php.net/gmstrftime*; detaillierte Informationen über die Sommerzeit finden sich unter *http://www.zeitumstellung.de/*.

3.15 Zeitangaben mit hoher Genauigkeit generieren

Problem

Sie müssen die Zeit in einer höheren als sekundengenauen Auflösung messen, zum Beispiel um eindeutige Kennzeichnungen zu generieren.

Lösung

Verwenden Sie `microtime()`:

```
list($microseconds,$seconds) = explode(' ', microtime());
```

Diskussion

Die Funktion `microtime()` liefert einen String, der den Millisekundenteil der seit dem Epochenbeginn vergangenen Zeit, ein Leerzeichen und dann die Sekunden seit dem Epochenbeginn enthält. Der Rückgabewert `0.41644100 1026683258` bedeutet zum Beispiel, dass seit dem Epochenbeginn 1026683258,41644100 Sekunden vergangen sind. Die Funktion gibt einen String anstelle eines Double-Werts zurück, da die Kapazität eines Double nicht ausreicht, um den gesamten Wert in Mikrosekunden-Genauigkeit aufzunehmen. In PHP-Versionen ab 5.0.0 akzeptiert `microtime()` einen optionalen Parameter vom Datentyp `bool`. Setzen Sie diesen auf `true`, liefert die Funktion den Rückgabewert als `float`-Variable, die eine Wertangabe in Sekunden darstellt.

Die Zeitangabe einschließlich der Mikrosekunden ist hilfreich beim Generieren eindeutiger Kennzeichen. In Kombination mit der aktuellen Prozesskennung ergeben sich garantiert eindeutige IDs, solange ein Prozess nicht mehr als eine ID pro Mikrosekunde generieren kann:

```
list($mikrosekunden,$sekunden) = explode(' ',microtime());
$id = $sekunden.$mikrosekunden.getmypid();
```

Auf Multithreading-Systemen ist diese Methode allerdings nicht völlig sicher, denn dort können mit einer gewissen (wenn auch sehr geringen) Wahrscheinlichkeit zwei Threads gleichzeitig im selben Prozess `microtime()` aufrufen.

Siehe auch

Die Dokumentation zu `microtime()` unter *http://www.php.net/microtime*. Zur Generierung eindeutiger IDs können Sie die Funktion `uniqid()` verwenden: *http://php.net/manual/function.uniqid.php*.

3.16 Periodisch wiederkehrende Ereignisse berechnen

Problem

Sie möchten periodisch wiederkehrende Termine berechnen. Dabei gehen Sie von einem Startdatum aus und suchen nach Terminen, die in einem bestimmten Zeitintervall bis zu einem Endtermin liegen. Alternativ wollen Sie die Anzahl der Wiederholungstermine anstelle eines Endtermins angeben.

Lösung

Seit PHP 5.3 können Sie zur Berechnung von periodisch wiederkehrenden Terminen die Klasse `DatePeriod` nutzen. Einmal entsprechend Ihren Bedürfnissen erzeugt, erlaubt die Klasse das Iterieren über die anfallenden Termine z. B. in einer `foreach`-Schleife:

```
$start = new DateTime('2009-05-02');
$ende = new DateTime('2009-08-01');
$intervall = new DateInterval('P7D');

// Das Intervall beträgt genau 3 Jahre, 6 Monate,
// 4 Tage, 12 Stunden, 30 Minuten und 5 Sekunden.
//$interval = 'P3Y6M4DT12H30M5S';

$p1 = new DatePeriod($start, $intervall, $ende);
foreach($p1 as $termin) {
    print $termin->format(DateTime::ISO8601);
}
2009-05-02T00:00:00+0200
2009-05-09T00:00:00+0200
2009-05-16T00:00:00+0200
2009-05-23T00:00:00+0200
2009-05-30T00:00:00+0200
2009-06-06T00:00:00+0200
2009-06-13T00:00:00+0200
2009-06-20T00:00:00+0200
2009-06-27T00:00:00+0200
2009-07-04T00:00:00+0200
2009-07-11T00:00:00+0200
2009-07-18T00:00:00+0200
2009-07-25T00:00:00+0200

$p2 = new DatePeriod(
    $start, $intervall, $wdhlg = 5, DatePeriod::EXCLUDE_START_DATE);
foreach($p2 as $termin) {
    print $termin->format(DateTime::ISO8601);
}
2009-05-09T00:00:00+0200
2009-05-16T00:00:00+0200
2009-05-23T00:00:00+0200
2009-05-30T00:00:00+0200
2009-06-06T00:00:00+0200
```

Arbeiten Sie mit PHP-Versionen vor 5.3, identifizieren Sie die Anfangszeit mit `time()` und `strftime()`. Wenn Ihr Intervall eine feste Länge hat, können Sie die Tage in einer Schleife durchlaufen. Wenn nicht, müssen Sie jeden folgenden Tag darauf überprüfen, ob er in den gewünschten Zeitraum fällt.

Beispielsweise hat eine Woche immer sieben Tage, daher können Sie alle Tage der aktuellen Woche in einer fest programmierten Schleife durchlaufen:

```
// Einen Zeitbereich für die aktuelle Woche bilden.
$jetzt = time();

// Wenn es vor 3 Uhr ist, $jetzt erhöhen, damit wir nicht
// beim Ermitteln des Wochenanfangs durch die Sommerzeit
// gestört werden.
if (3 < strftime('%H', $jetzt)) { $jetzt += 7200; }

// Welcher Wochentag ist heute?
$heute = strftime('%w', $jetzt);

// Vor wie vielen Tagen war der Wochenbeginn?
$start_tag = $jetzt - (86400 * $heute);

// Alle Tage der Woche ausgeben.
for ($i = 0; $i < 7; $i++) {
  print strftime('%c',$start_tag + 86400 * $i);
}
```

Diskussion

Ab PHP 5.3 verfügt PHP über die Klassen `DatePeriod` und `DateInterval`. Damit ist das Problem der Berechnung periodisch wiederkehrender Termine sehr einfach zu lösen. Das Intervall wird verwendet, um die Häufigkeit des Auftretens eines Termins zu definieren. Im gezeigten Beispiel beträgt das Intervall 7 Tage (`P7D`). Die Festlegung des Intervalls kann per Konstruktoraufruf erfolgen. Hier muss ein ISO-8601-Intervall-Format-String in der Form `P1Y2M3DT04H05M06S` angegeben werden. Der beispielhaft gezeigte String definiert folgendes Intervall: 1 Jahr, 2 Monate, 3 Tage, 4 Stunden, 5 Minuten, 6 Sekunden.

Die Klasse `DatePeriod` verfügt über mehrere Konstruktoren. Zwei davon lösen genau die in der Problemstellung geschilderte Aufgabenstellung:

```
DatePeriod::__construct(
    DateTime $start, DateInterval $interval, int $recurrences
    [, int $options])
DatePeriod::__construct (
    DateTime $start, DateInterval $interval, DateTime $end
    [, int $options])
```

Nutzen Sie den ersten Konstruktor, um ein Objekt zu erzeugen, das ausgehend von einem Startdatum und einem Intervall eine gewünschte Anzahl von wiederkehrenden Terminen berechnet. Der zweite Konstruktor ermöglicht es dagegen, ausgehend von

einem Startdatum und einem Intervall alle entsprechend wiederkehrenden Termine bis zu einem definierten Enddatum zu errechnen. Über den Parameter `$options` können Sie steuern, ob das Startdatum mit in die Menge der berechneten Termine fallen soll. Übergeben Sie hier die Konstante `DatePeriod::EXCLUDE_START_DATE`, um diesen Termin auszuschließen.

Ein Monat oder ein Jahr kann unterschiedlich viele Tage haben, daher müssen Sie die Besonderheiten des jeweiligen Monats oder Jahres berücksichtigen, wenn Sie das Ende des Zeitraums berechnen. Um in einer Schleife alle Tage eines Monats zu durchlaufen, ermitteln Sie den Epochen-Zeitstempel des ersten Tages dieses Monats und den ersten Tag des darauffolgenden Monats. Ab PHP 5.3 können Sie, wie schon gezeigt, `DateTime`, `DateInterval` und `DatePeriod` einsetzen:

```
$heute = new DateTime('today');
// 1. Tag des aktuellen Monats.
$anfang = new DateTime($heute->format('01.m.Y'));
// 1. Tag des folgenden Monats.
$ende = clone $anfang;
$ende->add(new DateInterval('P1M'));
// DatePeriod-Objekt erzeugen.
$periode = new DatePeriod($anfang, new DateInterval('P1D'), $ende);
foreach ($periode as $tag) {
    print $tag->format(DateTime::ISO8601);
}
```

In PHP-Versionen vor 5.3 tun Sie Folgendes: Die Schleifenvariable `$tag` wird jeweils um einen Tag (86400 Sekunden) erhöht, bis sie nicht mehr kleiner als der Epochen-Zeitstempel des nächsten Monatsanfangs ist:

```
// Den Zeitraum für diesen Monat generieren.
$jetzt = time();

// Wenn es vor 3 Uhr ist, $jetzt erhöhen, damit wir nicht
// beim Ermitteln des Monatsanfangs durch die Sommerzeit
// gestört werden.
if (3 < strftime('%H', $jetzt)) { $jetzt += 7200; }

// Welchen Monat haben wir?
$dieser_monat = strftime('%m',$jetzt);

// Epochen-Zeitstempel für den ersten Tag des Monats, Mitternacht.
$tag = mktime(0,0,0,$dieser_monat,1);
// Epochen-Zeitstempel für den ersten Tag des Folgemonats, Mitternacht.
$monatsende = mktime(0,0,0,$dieser_monat+1,1);

while ($tag < $monatsende) {
  print strftime('%c',$tag);
  $tag += 86400;
}
```

Siehe auch

Die Dokumentationen zu `DateTime` unter *http://php.net/manual/class.datetime.php*, zu `DateInterval` unter *http://php.net/manual/class.dateinterval.php*, zu `DateTimeZone` unter *http://php.net/manual/class.datetimezone.php*, zu `time()` unter *http://www.php.net/time* und zu `strftime()` unter *http://www.php.net/strftime*.

3.17 Andere Kalender als den gregorianischen verwenden

Problem

Sie möchten nicht den gregorianischen Kalender verwenden, sondern zum Beispiel den julianischen Kalender, den jüdischen Kalender oder den französischen Revolutionskalender.

Lösung

Die Kalendererweiterung zu PHP bietet Konvertierungsfunktionen für die Arbeit mit dem julianischen, dem französischen und dem jüdischen Kalender. Damit diese Funktionen verwendet werden können, muss die Kalendererweiterung geladen sein.

Diese Funktionen verwenden den julianischen Tageszähler (der sich vom julianischen Kalender unterscheidet) als Zwischenformat, um Informationen untereinander auszutauschen.

Die beiden Funktionen `jdtogregorian()` und `gregoriantojd()` konvertieren zwischen julianischen Tagen und dem uns vertrauten gregorianischen Kalender:

```
$jd = gregoriantojd(3,9,1876);      // 9. März 1876; $jd = 2406323

$gregorian = jdtogregorian($jd);    // $gregorian = 3/9/1876
```

Der gregorianische Kalender ist für den Zeitraum zwischen 4714 vor unserer Zeitrechnung und dem Jahr 9999 gültig.

Diskussion

Mithilfe der Funktionen `jdtojulian()` und `juliantojd()` können Sie Konvertierungen zwischen julianischen Tagen und dem julianischen Kalender durchführen:

```
// 29. Februar 1900 (kein Schaltjahr im gregorianischen Kalender)
$jd = juliantojd(2,29,1900);      // $jd = 2415092
$julian = jdtojulian($jd);        // $julian = 29.2.1900
$gregorian = jdtogregorian($jd);  // $gregorian = 13.3.1900
```

Der gültige Bereich für den julianischen Kalender liegt zwischen 4713 vor unserer Zeitrechnung und dem Jahr 9999; da aber der julianische Kalender erst im Jahr 46 vor unserer Zeitrechnung geschaffen wurde, verärgern Sie möglicherweise julianische Kalender-Puristen, wenn Sie die Funktionen mit einem davor liegenden Datum verwenden.

Für Konvertierungen zwischen julianischen Tagen und dem französischen Revolutionskalender verwenden Sie `jdtofrench()` und `frenchtojd()`:

```
$jd = frenchtojd(8,13,11);       // 13. floréal XI; $jd = 2379714
$french = jdtofrench($jd);       // $french = 13.8.11
$gregorian = jdtogregorian($jd); // $gregorian = 3.5.1803; Verkauf Louisianas an die USA
```

Der gültige Bereich für den französischen Revolutionskalender reicht von September 1792 bis zum September 1806; das ist nicht viel, aber da dieser Kalender nur vom Oktober 1793 bis zum Januar 1806 in Gebrauch war, umfassend genug.

Um ein Datum zwischen julianischen Tagen und dem jüdischen Kalender zu konvertieren, verwenden Sie `jdtojewish()` und `jewishtojd()`:

```
$jd = JewishToJD(6,14,5761);      // 14. Adar 5761; $jd = 2451978
$jewish = JDToJewish($jd);        // $jewish = 14.6.5761
$gregorian = JDToGregorian($jd);  // $gregorian = 9.3.2001
```

Der gültige Bereich für den jüdischen Kalender beginnt bei 3761 vor unserer Zeitrechnung, das ist das Jahr 1 des jüdischen Kalenders.

Siehe auch

Die Dokumentation zu den Kalenderfunktionen unter *http://www.php.net/calendar*; die Geschichte des gregorianischen Kalenders wird unter *http://scienceworld.wolfram.com/astronomy/GregorianCalendar.html* erläutert.

3.18 Programm: Kalender

Die in Beispiel 3-4 dargestellte Funktion `pc_calendar()` gibt einen Monatskalender ähnlich dem *cal*-Programm unter Unix aus. Hier sehen Sie, wie Sie diese Funktion verwenden können:

```
// Den Kalender für den aktuellen Monat ausgeben.
list($monat,$jahr) = explode(',',date('m,Y'));
pc_calendar($monat,$jahr);
```

Die Funktion `pc_calendar()` gibt eine Tabelle mit dem Kalender eines Monats aus. Sie bietet Links zum vorhergehenden und zum nachfolgenden Monat und hebt den aktuellen Tag hervor.

Beispiel 3-4: pc_calendar()

```
<?php
function pc_calendar($month,$year,$opts = '') {
    // Vorgabeoptionen setzen //
    if (! is_array($opts)) { $opts = array(); }
    if (! isset($opts['today_color'])) { $opts['today_color'] = '#FFFF00'; }
    if (! isset($opts['month_link'])) {
        $opts['month_link'] =
            '<a href="'.$_SERVER['PHP_SELF'].'?month=%d&year=%d">%s</a>';
    }

    list($this_month,$this_year,$this_day) = split(',',strftime('%m,%Y,%d'));
    $day_highlight = (($this_month == $month) && ($this_year == $year));

    list($prev_month,$prev_year) =
        split(',',strftime('%m,%Y',mktime(0,0,0,$month-1,1,$year)));
    $prev_month_link = sprintf($opts['month_link'],$prev_month,$prev_year,'&lt;');

    list($next_month,$next_year) =
        split(',',strftime('%m,%Y',mktime(0,0,0,$month+1,1,$year)));
    $next_month_link = sprintf($opts['month_link'],$next_month,$next_year,'&gt;');

?>
<table border="0" cellspacing="0" cellpadding="2" align="center">
        <tr>
                <td align="left">
                        <?php print $prev_month_link ?>
                </td>
                <td colspan="5" align="center">
                <?php print strftime('%B %Y',mktime(0,0,0,$month,1,$year)); ?>
                </td>
                <td align="right">
                        <?php print $next_month_link ?>
                </td>
        </tr>
<?php
    $totaldays = date('t',mktime(0,0,0,$month,1,$year));

    // Die Tage einer Woche ausgeben.
    print '<tr>';
    $weekdays = array('So','Mo','Di','Mi','Do','Fr','Sa');
    while (list($k,$v) = each($weekdays)) {
        print '<td align="center">'.$v.'</td>';
    }
    print '</tr><tr>';
    // Den ersten Tag des Monats auf den richtigen Wochentag ausrichten.
    $day_offset = date("w",mktime(0, 0, 0, $month, 1, $year));
    if ($day_offset > 0) {
        for ($i = 0; $i < $day_offset; $i++) { print '<td> </td>'; }
    }
    $yesterday = time() - 86400;
```

Beispiel 3-4: pc_calendar() (Fortsetzung)

```
    // Die Tage ausgeben.
    for ($day = 1; $day <= $totaldays; $day++) {
        $day_secs = mktime(0,0,0,$month,$day,$year);
        if ($day_secs >= $yesterday) {
            if ($day_highlight && ($day == $this_day)) {
                print sprintf('<td align="center" bgcolor="%s">%d</td>',
                              $opts['today_color'],$day);
            } else {
                print sprintf('<td align="center">%d</td>',$day);
            }
        } else {
            print sprintf('<td align="center">%d</td>',$day);
        }
        $day_offset++;

        // Für jede Woche eine neue Zeile beginnen. //
        if ($day_offset == 7) {
            $day_offset = 0;
            if ($day < $totaldays) { print "</tr>\n<tr>"; }
        }
    }
    // Die letzte Woche mit leeren Zellen auffüllen. //
    if ($day_offset > 0) { $day_offset = 7 - $day_offset; }
    if ($day_offset > 0) {
        for ($i = 0; $i < $day_offset; $i++) { print '<td> </td>'; }
    }
    print '</tr></table>';
}
?>
```

Die Funktion `pc_calendar()` beginnt damit, dass sie die in `$opts` übergebenen Optionen überprüft. Die Farbe, mit der der aktuelle Tag hervorgehoben wird, kann in `$opts['today_color']` als RGB-Wert übergeben werden. Der Vorgabewert ist `#FFFF00` (hellgelb). Außerdem können Sie in `$opts['month_link']` einen Format-String im `printf()`-Stil übergeben, mit dem Sie die Darstellungsweise der Links auf den vorherigen und den nächsten Monat verändern können.

Danach setzt die Funktion `$day_highlight` auf `true`, wenn Monat und Jahr des Kalenders mit dem aktuellen Monat und Jahr übereinstimmen. Die Verweise auf den vorherigen und den nächsten Monat werden in `$prev_month_link` und `$next_month_link` gespeichert, wobei der Format-String in `$opts['month_link']` angewendet wird.

Dann gibt `pc_calendar()` den oberen Teil der HTML-Tabelle aus, der den Monat und eine Tabellenzeile mit Wochentag-Abkürzungen enthält. Mithilfe des von der Funktion `strftime('%w')` zurückgegebenen Wochentags werden leere Tabellenzeilen ausgegeben, damit der erste Tag des Monats auf den richtigen Wochentag ausgerichtet ist. Wenn beispielsweise der erste Tag des Monats ein Dienstag ist, müssen zwei leere Zellen ausgege-

ben werden, um die Spalten für den Sonntag und den Montag in der ersten Zeile der Tabelle auszufüllen.

Nachdem diese Vorabinformationen ausgegeben worden sind, durchläuft `pc_calendar()` den Monat in einer Schleife. Die Funktion gibt für die meisten Tage eine einfache Tabellenzelle aus, jedoch beim aktuellen Tag eine Zelle mit abweichendem Hintergrund. Wenn `$day_offset` den Wert 7 erreicht, ist eine Woche komplett, und eine neue Woche beginnt.

Wenn für alle Tage des Monats eine Tabellenzelle ausgegeben ist, werden so lange leere Zellen hinzugefügt, bis die letzte Zeile der Tabelle ausgefüllt ist. Wenn zum Beispiel der letzte Tag des Monats ein Donnerstag ist, werden zwei zusätzliche Zellen für Freitag und Samstag ausgegeben. Schließlich wird die Tabelle geschlossen, und der Kalender ist fertig.

KAPITEL 4

Arrays

4.0 Einführung

Arrays sind Listen: Listen von Personen, Listen von Größen, Listen von Büchern. Um eine Gruppe zusammenhängender Dinge in einer Variablen zu speichern, verwenden Sie ein Array. Wie eine Liste auf einem Stück Papier haben auch die Elemente eines Arrays eine Reihenfolge. Normalerweise folgt jedes neue Element dem letzten Array-Eintrag, aber genau so, wie Sie einen neuen Eintrag zwischen zwei Zeilen klemmen können, die sich bereits in der Papierliste befinden, können Sie dasselbe auch mit PHP-Arrays tun.

Viele Programmiersprachen kennen nur eine Art Array, die als *numerisches Array* (oder einfach als Array) bezeichnet wird. Bei einem numerischen Array müssen Sie, wenn Sie einen Eintrag finden möchten, seine als *index* bezeichnete Position innerhalb des Arrays kennen. Die Positionen werden durch Zahlen gekennzeichnet: Beginnend bei 0, wird fortlaufend aufwärts gezählt.

In manchen Sprachen gibt es daneben noch eine andere Art von Array: das *assoziative Array*, das auch als *Hash* bezeichnet wird. In einem assoziativen Array sind die Indizes keine Integer-Zahlen, sondern Strings. In einem numerischen Array mit US-Präsidenten könnte »Abraham Lincoln« also den Index 16 haben; in einen assoziativen Array könnte der Index auch »Ehrenwert« lauten. Während allerdings numerische Arrays immer eine strikte Sortierung haben, die durch die Schlüssel bestimmt wird, geben assoziative Arrays häufig keine Garantie über die Anordnung der Schlüssel ab. Die Elemente werden in einer bestimmten Reihenfolge hinzugefügt, aber es gibt keine Möglichkeit, die Reihenfolge später zu bestimmen.

In einigen Sprachen gibt es sowohl numerische als auch assoziative Arrays. Aber normalerweise sind das numerische Array `$praesidenten` und das assoziative Array `$praesidenten` zwei verschiedenartige Arrays. Jeder Array-Typ hat ein spezifisches Verhalten, und dementsprechend müssen Sie mit ihnen umgehen. PHP kennt sowohl numerische als auch assoziative Arrays, aber sie verhalten sich nicht unabhängig voneinander.

Bei PHP sind numerische Arrays zugleich auch assoziative Arrays, und assoziative Arrays sind zugleich numerische Arrays. Worum handelt es sich hier also wirklich? Weder um das eine noch um das andere. Die Trennlinie zwischen beiden schwankt hin und her von einem zum anderen. Dies kann anfangs verwirrend sein, insbesondere wenn Sie ein starres Verhalten gewohnt sind, aber nach kurzer Zeit werden Sie diese Flexibilität zu schätzen lernen.

Um einem Array in einem Schritt mehrere Werte zuzuweisen, verwenden Sie `array()`:

```
$obst = array('Apfel', 'Banane', 'Melone', 'Dattel');
```

Der Wert von `$obst[2]` ist jetzt `'Melone'`.

`array()` ist sehr praktisch, wenn Sie eine kurze Liste bekannter Werte haben. Dasselbe Array kann auch folgendermaßen gebildet werden:

```
$obst[0] = 'Apfel';
$obst[1] = 'Banane';
$obst[2] = 'Melone';
$obst[3] = 'Dattel';
```

oder:

```
$obst[ ] = 'Apfel';
$obst[ ] = 'Banane';
$obst[ ] = 'Melone';
$obst[ ] = 'Dattel';
```

Will man einen Wert einem Array mit einem leeren Subskript zuweisen, ist dies eine Abkürzung für das Hinzufügen eines neuen Elements am Ende des Arrays. PHP stellt also die Länge von `$obst` fest und verwendet diese als Position für den zugewiesenen Wert. Dies setzt natürlich voraus, dass `$obst` nicht auf einen skalaren Wert gesetzt ist, zum Beispiel auf 3, und auch kein Objekt ist. PHP beschwert sich, wenn Sie versuchen, ein Nicht-Array als Array zu behandeln; wenn Sie die Variable an dieser Stelle aber zum ersten Mal verwenden, konvertiert PHP sie automatisch in ein Array und beginnt mit dem Index 0.

Eine identische Möglichkeit bietet die Funktion `array_push()`, die einen neuen Wert oben auf dem Array-Stapel ablegt. Die `$foo[ ]`-Notation entspricht allerdings eher dem traditionellen PHP-Stil und ist außerdem schneller. Manchmal allerdings vermittelt die Verwendung von `array_push()` deutlicher, dass Sie das Array wie einen Stapel benutzen, besonders in Kombination mit der Funktion `array_pop()`, die das letzte Element aus einem Array entfernt und zurückgibt.

Bis jetzt haben wir nur Integer-Werte und Strings in einem Array untergebracht. Bei PHP können Sie aber auch jeden beliebigen anderen Datentyp einem Array-Element zuweisen: Boolesche Werte, Integer- und Fließkommazahlen, Strings, Objekte, Ressourcen, `NULL` und sogar andere Arrays. So können Sie auch Arrays oder Objekte direkt aus einer Datenbank lesen und in ein Array platzieren:

```
while ($row = mysql_fetch_row($r)) {
    $obst[ ] = $row;
}
```

```
while ($obj = mysql_fetch_object($s)) {
    $gemuese[] = $obj;
}
```

Die erste while-Anweisung erzeugt ein Array aus Arrays, die zweite legt ein Array aus Objekten an. In Rezept 4.2 finden Sie mehr darüber, wie man mehrere Elemente unter einem Schlüssel speichert.

Um ein Array zu definieren, das keine Integer-Schlüssel, sondern String-Schlüssel hat, können Sie ebenfalls array() verwenden, geben dabei aber die Schlüssel/Wert-Paare mit => an:

```
$obst = array('rot' => 'Apfel', 'gelb' => 'Banane',
              'beige' => 'Melone', 'braun' => 'Dattel');
```

Nun hat $obst['beige'] den Wert 'Melone'. Dies ist eine Abkürzung für:

```
$obst['rot'] = 'Apfel';
$obst['gelb'] = 'Banane';
$obst['beige'] = 'Melone';
$obst['braun'] = 'Dattel';
```

Jedes Array kann für jeden Schlüssel nur einen eindeutigen Wert enthalten. Durch Zufügen von:

```
$obst['rot'] = 'Erdbeere';
```

wird der Wert von 'Apfel' überschrieben. Sie können aber später immer noch einen weiteren Schlüssel hinzufügen:

```
$obst['orange'] = 'Orange';
```

Je länger Sie mit PHP arbeiten, desto häufiger werden Sie assoziative anstelle von numerischen Arrays einsetzen. Anstatt ein numerisches Array mit String-Werten anzulegen, können Sie ein assoziatives Array erzeugen und die Werte als Schlüssel eintragen. Wenn Sie wollen, können Sie dann noch zusätzliche Informationen als Elementwerte speichern. Dabei gibt es keinen Performance-Nachteil, und PHP bewahrt die Reihenfolge. Und zusätzlich ist das Nachschlagen und Verändern eines Werts einfach, da Sie den Schlüssel bereits kennen.

Wenn Sie ein Array durchlaufen und mit allen oder einem Teil der enthaltenen Elemente arbeiten möchten, verwenden Sie am besten foreach:

```
$obst = array('rot' => 'Äpfel', 'gelb' => 'Bananen',
              'beige' => 'Melonen', 'braun' => 'Datteln');

foreach ($obst as $farbe => $frucht) {
    print "$frucht sind $farbe.\n";
}
Äpfel sind rot.
Bananen sind gelb.
Melonen sind beige.
Datteln sind braun.
```

Bei jedem Durchgang der Schleife weist PHP den nächsten Schlüssel der Variablen `$farbe` zu und den Wert des nächsten Schlüssels der Variablen `$frucht`. Wenn es keine weiteren Elemente mehr im Array gibt, wird die Schleife beendet.

Um ein Array in einzelne Variablen aufzubrechen, verwenden Sie `list()`:

```
$obst = array('Äpfel', 'Bananen', 'Melonen', 'Datteln');
list($rot, $gelb, $beige, $braun) = $obst;
```

In diesem Kapitel werden nicht nur einfache numerische und assoziative Arrays besprochen. PHP 5 kann noch mehr. Rezept 4.28 zeigt, wie Sie ein PHP-Objekt anstelle eines Arrays verwenden können.

4.1 Ein Array anlegen, das nicht mit dem Element 0 beginnt

Problem

Sie möchten einem Array mehrere Elemente auf einmal zuweisen, dabei soll aber der erste Index nicht 0 sein.

Lösung

Veranlassen Sie `array()` dazu, einen anderen Index zu benutzen, indem Sie die Syntax mit `=>` verwenden:

```
$presidents = array(1 => 'Washington', 'Adams', 'Jefferson', 'Madison');
```

Diskussion

Wie in den meisten, wenn auch nicht in allen Computersprachen beginnen PHP-Arrays beim ersten Eintrag mit dem Index 0. Manchmal geben aber die im Array gespeicherten Daten mehr Sinn, wenn die Liste bei 1 beginnt. (Dies erwähnen wir hier nicht nur, um Pascal-Programmierer zu gewinnen.)

In der obigen Lösung ist George Washington der erste Präsident und nicht der nullte; wenn Sie also eine Liste der Präsidenten ausgeben möchten, ist es einfacher, dies so zu tun:

```
foreach ($presidents as $number => $president) {
    print "$number: $president\n";
}
```

als folgendermaßen:

```
foreach ($presidents as $number => $president) {
    $number++;
    print "$number: $president\n";
}
```

Diese Möglichkeit ist nicht auf die Zahl 1 beschränkt, sondern funktioniert mit jedem Integer-Wert:

```
$reconstruction_presidents = array(16 => 'Lincoln', 'Johnson', 'Grant');
```

Sie können => auch mehrmals innerhalb eines Aufrufs verwenden:[1]

```
$whig_presidents = array(9 => 'Harrison', 'Tyler', 12 => 'Taylor', 'Fillmore');
```

In PHP ist es auch erlaubt, negative Zahlen im `array()`-Aufruf zu verwenden. (Tatsächlich funktioniert diese Methode sogar bei Zahlen, die keine Integer sind.) Technisch gesehen ist das, was Sie dann erhalten, ein assoziatives Array. Wie gesagt, die Grenzen verschwimmen zwischen numerischen und assoziativen Arrays bei PHP häufig; dies ist nur ein weiteres Beispiel dafür.

```
$us_leaders = array(-1 => 'George II', 'George III', 'Washington');
```

Wenn Washington der erste US-Anführer ist, ist George III der nullte und dessen Großvater George II der minus-erste.

Natürlich können Sie numerische und String-Schlüssel in einer `array()`-Definition nach Bedarf vermischen, aber dies ist verwirrend und wird selten benötigt:

```
$presidents = array(1 => 'Washington', 'Adams', 'Ehrenwert' => 'Lincoln', 'Jefferson');
```

Dies ist gleichbedeutend mit:

```
$presidents[1]           = 'Washington';   // Schlüssel ist 1
$presidents[]            = 'Adams';        // Schlüssel ist 1 + 1 => 2
$presidents['Ehrenwert'] = 'Lincoln';      // Schlüssel ist 'Ehrenwert'
$presidents[]            = 'Jefferson';    // Schlüssel ist 2 + 1 => 3
```

Siehe auch

Die Dokumentation zu `array()` unter *http://www.php.net/array*.

4.2 Mehrere Array-Elemente unter einem Schlüssel speichern

Problem

Sie möchten mehrere Elemente mit einem einzigen Schlüssel verknüpfen.

Lösung

Speichern Sie die Elemente in ein Array:

1 John Tyler wurde als Harrisons Vizepräsident als Mitglied der (liberalen) Whig-Partei gewählt, aber aus der Partei ausgeschlossen, kurz nachdem Harrison gestorben war und er die Präsidentschaft übernommen hatte.

```
$obst = array('rot' => array('Erdbeere','Apfel'),
              'gelb' => array('Banane'));
```

Oder verwenden Sie ein Objekt:

```
while ($obj = mysql_fetch_object($r)) {
    $obst[] = $obj;
}
```

Diskussion

Bei PHP sind die Schlüssel in einem Array eindeutig, daher können Sie nicht mehr als einen Eintrag einem Schlüssel zuordnen, ohne dabei den alten Wert zu überschreiben. Stattdessen können Sie aber die Werte in einem anonymen Array speichern:

```
$obst['rot'][] = 'Erdbeere';
$obst['rot'][] = 'Apfel';
$obst['gelb'][] = 'Banane';
```

Oder, wenn Sie die Gegenstände in einer Schleife durchlaufen:

```
while (list($farbe,$frucht) = mysql_fetch_array($r)) {
    $obst[$farbe][] = $frucht;
}
```

Um die Einträge auszugeben, arbeiten Sie das Array in einer Schleife ab:

```
foreach ($obst as $farbe=>$farbe_frucht) {
    // $farbe_frucht ist ein Array
    foreach ($farbe_frucht as $frucht) {
        print "$frucht hat die Farbe $farbe.<br>";
    }
}
```

Oder verwenden Sie die Funktion `pc_array_to_comma_string()` aus Rezept 4.9.

```
foreach ($obst as $farbe=>$farbe_frucht) {
    print "Obst mit der Farbe $farbe umfaßt " .
        pc_array_to_comma_string($farbe_frucht) . "<br>";
}
```

Siehe auch

Rezept 4.9 dazu, wie man Arrays mit Kommata ausgibt.

4.3 Ein Array mit einer Folge von Integer-Werten initialisieren

Problem

Sie möchten einem Array eine Reihe aufeinander folgender Integer-Zahlen zuweisen.

Lösung

Verwenden Sie `range($anfang, $ende)`:

```
$karten = range(1, 52);
```

Diskussion

Wenn die Inkremente nicht 1 sind, können Sie `range()` einen dritten Parameter mit der Schrittweite übergeben. Benötigen Sie also ungerade Zahlen:

```
$ungerade = range(1, 52, 2);
```

Und für gerade Zahlen:

```
$gerade = range(2, 52, 2);
```

Siehe auch

Rezept 2.4 dazu, wie man mit einer Reihe von Integer-Werten arbeitet; die Dokumentation zu `range()` unter *http://www.php.net/range*.

4.4 Ein Array durchlaufen

Problem

Sie möchten ein Array in einer Schleife durchlaufen und alle oder einen Teil der darin enthaltenen Elemente verarbeiten.

Lösung

Verwenden Sie `foreach`:

```
foreach ($array as $wert) {
    // Mit $wert arbeiten.
}
```

Oder, um die Schlüssel und Werte eines Arrays zu erhalten:

```
foreach ($array as $schluessel => $wert) {
    // Mit $schluessel und $wert arbeiten.
}
```

Eine andere Technik verwendet `for`:

```
for ($schluessel = 0, $laenge = count($array); $schluessel < $laenge; $schluessel++) {
   // Mit $schluessel und $wert arbeiten.
}
```

Und schließlich können Sie auch `each()` in Kombination mit `list()` und `while` verwenden:

```
reset($array) // Internen Zeiger auf den Array-Beginn zurücksetzen.
while (list($schluessel, $wert) = each ($array)) {
   // Mit $schluessel und $wert arbeiten
}
```

Diskussion

Eine foreach-Schleife stellt die kürzeste Möglichkeit dar, ein Array zu durchlaufen:

```
// foreach mit Werten
foreach ($items as $cost) {
    ...
}

// foreach mit Schlüsseln und Werten
foreach($items as $item => $cost) {
    ...
}
```

Mit foreach durchläuft PHP eine Kopie anstelle des eigentlichen Arrays. Wenn Sie dagegen each() oder for benutzen, durchläuft PHP das Original-Array. Wenn Sie also das Array innerhalb der Schleife verändern, wird das Verhalten so sein, wie Sie es erwarten (oder auch nicht).

Wenn Sie ein Array verändern möchten, sollten Sie es direkt referenzieren:

```
reset($items);
while (list($item, $cost) = each($items)) {
    if (! in_stock($item)) {
        unset($items[$item]);   // Adressiert das Array direkt.
    }
}
```

Die von each() gelieferten Werte sind keine Synonyme für die Originalwerte im Array – es handelt sich um Kopien, daher wird es im Array nicht nachvollzogen, wenn Sie diese verändern. Sie müssen also $items[$item] verändern und nicht $produkt.

Wenn Sie each() verwenden, merkt sich PHP, wo Sie sich innerhalb der Schleife befinden. Wollen Sie nach der Beendigung eines ersten Durchgangs noch einmal von vorn beginnen, rufen Sie reset() auf, um den Zeiger zurück vor den Anfang des Arrays zu stellen. Andernfalls gibt each() den Wert false zurück.

Die for-Schleife kann nur bei Arrays mit aufeinander folgenden Integer-Indices verwendet werden. Sofern Sie nicht die Größe Ihres Arrays verändern wollen, ist es ineffizient, mit count() die Anzahl der $items bei jedem Durchgang der Schleife neu zu berechnen, verwenden Sie daher immer eine Variable $size für die Länge des Arrays:

```
for ($item = 0, $size = count($items); $item < $size; $item++) {
    ...
}
```

Wenn Sie lieber mit nur einer Variablen effizient zählen möchten, sollten Sie rückwärts zählen:

```
for ($item = count($items) - 1; $item >= 0; $item--) {
    ...
}
```

Die assoziative Version der `for`-Schleife sieht folgendermaßen aus:

```
for (reset($array); $schluessel = key($array); next($array) ) {
    ...
}
```

Dies geht jedoch schief, wenn eines der Elemente einen String enthält, dessen Auswertung `false` ergibt; ein vollkommen normaler Wert wie 0 bringt also die Schleife zu einem vorzeitigen Ende.

Schließlich können Sie `array_map()` verwenden, um jedes Element des Arrays einer Funktion zur Verarbeitung zu übergeben:

```
// Alle Wörter kleinschreiben.
$lc = array_map('strtolower', $words);
```

Das erste Argument für `array_map()` ist eine Funktion zur Veränderung eines einzelnen Elements, und das zweite ist das Array, das durchlaufen werden soll.

Generell finden wir diese Funktionen weniger flexibel als die vorangehenden Methoden, aber sie sind gut geeignet zum Verarbeiten und Mischen mehrerer Arrays.

Wenn Sie nicht sicher sind, ob die Daten, die Sie verarbeiten, Skalare oder Arrays sind, müssen Sie sich dagegen absichern, dass Sie `foreach` bei einem Nicht-Array aufrufen. Eine Möglichkeit dafür ist die Verwendung von `is_array()`:

```
if (is_array($items)) {
    // foreach-Schleife für ein Array
} else {
    // Code für eine Skalar
}
```

Eine andere Möglichkeit besteht darin, alle Variablen mit `settype()` zwangsweise zu Arrays zu machen:

```
settype($items, 'array');
// Schleifen-Code für Arrays
```

Auf diese Weise verwandeln Sie Skalarwerte in Arrays mit jeweils einem Element und räumen Ihr Programm auf Kosten einer geringen Mehrbelastung auf.

Siehe auch

Die Dokumentationen zu `for` unter *http://www.php.net/for*, `foreach` unter *http://www.php.net/foreach*, `while` unter *http://www.php.net/while*, `each()` unter *http://www.php.net/each*, `reset()` unter *http://www.php.net/reset* und `array_map()` unter *http://www.php.net/array-map*. Kapitel 8 behandelt Iteratoren, die es Ihnen ermöglichen, auch andere Datenstrukturen mit `foreach` zu durchlaufen.

4.5 Elemente aus einem Array löschen

Problem

Sie möchten ein oder mehrere Elemente aus einem Array entfernen.

Lösung

Zum Löschen eines Elements verwenden Sie `unset()`:

```
unset($array[3]);
unset($array['foo']);
```

Zum Löschen mehrerer, nicht zusammenhängender Elemente nehmen Sie ebenfalls `unset()`:

```
unset($array[3], $array[5]);
unset($array['foo'], $array['bar']);
```

Zum Löschen mehrerer zusammenhängender Elemente können Sie außerdem auch noch `array_splice()` verwenden:

```
array_splice($array, $beginn, $laenge);
```

Diskussion

Wenn Sie diese Funktionen verwenden, entfernen Sie alle Referenzen auf die Elemente aus dem PHP-Programm. Möchten Sie einen Schlüssel im Array behalten, der Wert dazu soll aber leer sein, weisen Sie dem Element einen leeren String zu:

```
$array[3] = $array['foo'] = '';
```

Abgesehen von der Syntax ist es auch ein logischer Unterschied, ob Sie `unset()` verwenden oder dem Element `''` zuweisen. Das erste bedeutet: »Dieses Element existiert nicht mehr«, während das zweite besagt: »Dieses Element existiert noch, sein Wert ist aber ein leerer String.«

Wenn Sie es mit Zahlen zu tun haben, könnte die Zuweisung von `0` eine bessere Alternative sein. Eine Firma, die ihre Produktion von Zahnrädern des Modells XL1000 einstellt, modifiziert ihr Inventar also folgendermaßen:

```
unset($produkte['XL1000']);
```

Wenn sie aber nur vorübergehend keine XL1000-Zahnräder mehr auf Lager hat und Ende der Woche eine neue Lieferung von der Fabrik erwartet wird, passt Folgendes besser:

```
$produkte['XL1000'] = 0;
```

Beim Löschen eines Elements mit `unset()` passt PHP das Array so an, dass Schleifen immer noch korrekt funktionieren. Das Array wird nicht komprimiert, um die entstandenen Löcher aufzufüllen. Das meinen wir, wenn wir sagen, dass alle Arrays assoziativ sind, auch wenn sie numerisch zu sein scheinen. Hier ist ein Beispiel:

```
// Ein "numerisches" Array anlegen.
$tiere = array('Ameise', 'Biene', 'Katze', 'Hund', 'Elch', 'Fuchs');
print $tiere[1];  // gibt 'Biene' aus
print $tiere[2];  // gibt 'Katze' aus
count($tiere);    // ergibt 6

// unset()
unset($tiere[1]); // entfernt das Element $tiere[1] = 'Biene'
print $tiere[1];  // gibt '' aus und löst einen E_NOTICE-Fehler aus
print $tiere[2];  // gibt immer noch 'Katze' aus
count($tiere);    // ergibt 5, obwohl $array[5] 'Fuchs' ist

// Neues Element hinzufügen.
$tiere[ ] = 'Gnu'; // neues Element zufügen (kein Unix)
print $tiere[1];  // gibt '' aus, immer noch leer
print $tiere[6];  // gibt 'gnu' aus, hier ist 'Gnu' gelandet
count($tiere);    // ergibt 6

// Leer-String '' zuweisen.
$tiere[2] = '';   // entleert den Wert
print $tiere[2];  // gibt '' aus
count($tiere);    // ergibt 6, die Anzahl ist nicht vermindert
```

Um das Array zu einem dicht gepackten numerischen Array zu komprimieren, verwenden Sie `array_values()`:

```
$tiere = array_values($tiere);
```

Alternativ re-indiziert auch `array_splice()` Arrays; diese Funktion vermeidet es, Löcher zu hinterlassen:

```
// Ein "numerisches" Array erzeugen.
$tiere = array('Ameise', 'Biene', 'Katze', 'Hund', 'Elch', 'Fuchs');
array_splice($tiere, 2, 2);
print_r($tiere);
Array
(
    [0] => Ameise
    [1] => Biene
    [2] => Elch
    [3] => Fuchs
)
```

Dies ist hilfreich, wenn Sie ein Array als Warteschlange verwenden und Elemente aus der Schlangen entfernen möchten, wobei der direkte Zugriff aber erhalten bleiben soll. Um das erste oder letzte Element eines Arrays sicher zu entfernen, verwenden Sie `array_shift()` beziehungsweise `array_pop()`.

Wenn es aber des Öfteren vorkommt, dass Sie Probleme mit Löchern in Arrays bekommen, liegt dies möglicherweise daran, dass Sie nicht »in PHP denken«. Sehen Sie sich die verschiedenen Möglichkeiten in Rezept 4.4 an, mit denen man Arrays durchlaufen kann und bei denen keine `for`-Schleife benötigt wird.

Siehe auch

Rezept 4.4 mit Schleifentechniken; die Dokumentationen zu `unset()` unter *http://www.php.net/unset*, `array_splice()` unter *http://www.php.net/array-splice* und `array_values()` unter *http://www.php.net/array-values*.

4.6 Die Größe eines Arrays ändern

Problem

Sie möchten die Größe eines Arrays verändern, indem Sie es entweder größer oder kleiner machen, als es gerade ist.

Lösung

Lassen Sie mit `array_pad()` ein Array wachsen:

```
// Mit drei beginnen.
$array = array('Apfel', 'Banane', 'Kokosnuss');

// Auf fünf vergrößern.
$array = array_pad($array, 5, '');
```

Nun ergibt `count($array)` die Zahl `5`, und die letzten beiden Elemente enthalten einen leeren String.

Ein Array verkleinern können Sie mit `array_splice()`:

```
// Keine Zuweisung an $array.
array_splice($array, 2);
```

Dadurch werden alle mit Ausnahme der ersten beiden Elemente aus dem `$array` entfernt.

Diskussion

Arrays werden in PHP nicht vordimensioniert, daher kann ihre Größe im laufenden Programm verändert werden.

Um ein Array aufzufüllen, verwenden Sie `array_pad()`. Das erste Argument ist das aufzufüllende Array. Das nächste ist die Länge und die Richtung, in der Sie auffüllen möchten. Um nach rechts aufzufüllen, setzen Sie eine positive Zahl ein, zum Aufzufüllen nach links eine negative. Das dritte Argument ist der Wert, der den neu erzeugten Einträgen zugewiesen werden soll. Die Funktion gibt ein modifiziertes Array zurück und verändert nicht das Original.

Hier sind einige Beispiele:

```
// Ein Array mit vier Elementen erzeugen,
// rechts mit 'Dattel' aufgefüllt.
$array = array('Apfel', 'Banane', 'Kokosnuss');
```

```
$array = array_pad($array, 4, 'Dattel');
print_r($array);
Array
(
    [0] => Apfel
    [1] => Banane
    [2] => Kokosnuss
    [3] => Dattel
)

// Ein Array mit sechs Elementen erzeugen,
// links mit 'Zucchini' aufgefüllt.
$array = array_pad($array, -6, 'Zucchini');
print_r($array);
Array
(
    [0] => Zucchini
    [1] => Zucchini
    [2] => Apfel
    [3] => Banane
    [4] => Kokosnuss
    [5] => Dattel
)
```

Seien Sie vorsichtig: `array_pad($array, 4, 'Dattel')` stellt sicher, dass ein `$array` *mindestens* vier Elemente lang ist, die Funktion fügt nicht vier *neue* Elemente hinzu. In diesem Fall würde, wenn `$array` bereits vier Elemente hätte, `array_pad()` ein unverändertes `$array` liefern.

Beachten Sie, dass Sie auch mit `$array[4]` einen Wert für ein viertes Element deklarieren können:

```
$array = array('Apfel', 'Banane', 'Kokosnuss');
$array[4] = 'Dattel';
```

um Ende ein vierelementiges Array mit den Indizes `0`, `1`, `2` und `4` zu erhalten:

```
Array
(
    [0] => Apfel
    [1] => Banane
    [2] => Kokosnuss
    [4] => Dattel
)
```

Eigentlich verwandelt PHP dies in ein assoziatives Array, das lediglich Integer-Zahlen als Schlüssel hat.

Die Funktion `array_splice()` hat anders als `array_pad()` den Seiteneffekt, dass sie das ursprüngliche Array verändert. Sie gibt das herausgetrennte Array zurück. Aus diesem Grund weisen Sie den zurückgegebenen Wert nicht `$array` zu. Sie können aber, ähnlich wie bei `array_pad()`, einen Teil von rechts oder von links herauslösen. Wenn Sie also `array_splice()` mit `-2` aufrufen, schneiden Sie die letzten beiden Elemente am Ende ab:

```
// Ein Array mit vier Elementen erzeugen.
$array = array('Apfel', 'Banane', 'Kokosnuss', 'Dattel');

// Auf drei Elemente schrumpfen.
array_splice($array, 3);

// Das letzte Element entfernen, entspricht array_pop().
array_splice($array, -1);

// Die einzigen verbleibenden Früchte sind Apfel und Banane.
print_r($array);
Array
(
    [0] => Apfel
    [1] => Banane
)
```

Siehe auch

Die Dokumentationen zu `array_pad()` unter *http://www.php.net/array-pad* und `array_splice()` unter *http://www.php.net/array-splice*.

4.7 Ein Array an ein anderes anfügen

Problem

Sie möchten zwei Arrays zu einem Array kombinieren.

Lösung

Verwenden Sie `array_merge()`:

```
$garten = array_merge($obst, $gemuese);
```

Diskussion

Die Funktion `array_merge()` funktioniert sowohl bei vorab definierten Arrays als auch bei solchen, die an Ort und Stelle mit `array()` definiert werden:

```
$p_sprachen = array('Perl', 'PHP');
$p_sprachen = array_merge($p_sprachen, array('Python'));
print_r($p_sprachen);
Array
(
    [0] => PHP
    [1] => Perl
    [2] => Python
)
```

Die zu mischenden Arrays können also wie `$p_sprachen` bereits vorhanden sein, oder sie können anonyme Arrays wie `array('Python')` sein.

Verwenden Sie hier nicht `array_push()`, da PHP das Array nicht automatisch in eine Reihe voneinander unabhängiger Variablen auflösen würde, stattdessen würden Sie ein geschachteltes Array erhalten. Dementsprechend:

```
array_push($p_sprachen, array('Python'));
print_r($p_sprachen);
Array
(
    [0] => PHP
    [1] => Perl
    [2] => Array
        (
            [0] => Python
        )

)
```

Wenn Sie Arrays mischen, die nur numerische Schlüssel haben, werden die Arrays neu nummeriert, damit keine Werte verloren gehen. Beim Mischen von Arrays mit String-Schlüsseln überschreibt das zweite Array die Werte aller doppelten Schlüssel. Arrays mit beiden Schlüsselarten zeigen beide Arten des Verhaltens. Ein Beispiel:

```
$lc = array('a', 'b' => 'b');   // Kleinbuchstaben als Werte
$uc = array('A', 'b' => 'B');   // Großbuchstaben als Werte
$ac = array_merge($lc, $uc);    // Alle Buchstaben?
print_r($ac);
Array
(
    [0] => a
    [b] => B
    [1] => A
)
```

Das große A wurde umnummeriert von Index 0 zu Index 1, um eine Kollision zu vermeiden, und am Ende hinzugefügt. Das große B hat das kleine b überschrieben und am ursprünglichen Platz im Array ersetzt.

Auch der Operator + kann Arrays mischen. Das Array auf der rechten Seite überschreibt identisch benannte Schlüssel, die auf der linken Seite gefunden werden. Es gibt keine Umsortierung zur Vermeidung von Kollisionen. Um bei obigem Beispiel zu bleiben:

```
print_r($a + $b);
print_r($b + $a);
Array
(
    [0] => a
    [b] => b
)
Array
```

```
(
    [0] => A
    [b] => B
)
```

Da sowohl `a` als auch `A` beide den Schlüssel `0` haben und entsprechend `b` und `B` den Schlüssel `b`, erhalten Sie in der Summe nur zwei Elemente aus den gemischten Arrays.

Im ersten Fall wird aus `$a + $b` einfach `$b`, und im anderen Fall wird aus `$b + $a` dann `$a`.

Wenn Sie allerdings zwei Arrays mit durchgehend verschiedenen Schlüsseln haben, kommt es nicht zu diesem Problem, und das neue Array bildet die Vereinigungsmenge der beiden anderen Arrays.

Siehe auch

Die Dokumentation zu `array_merge()` unter *http://www.php.net/array-merge*.

4.8 Ein Array in einen String verwandeln

Problem

Sie haben ein Array, das Sie in einen ordentlich formatierten String verwandeln möchten.

Lösung

Verwenden Sie `join()`:

```
// Eine kommaseparierte Liste erstellen.
$string = join(',', $array);
```

Oder programmieren Sie selbst eine Schleife:

```
$string = '';

foreach ($array as $schluessel => $wert) {
    $string .= ",$wert";
}

$string = substr($string, 1); // entfernt führendes ","
```

Diskussion

Wenn Sie `join()` benutzen können, sollten Sie dies auch tun; es geht schneller als jede PHP-basierte Schleife. Allerdings ist `join()` nicht besonders flexibel. Erstens fügt die Funktion die Trennzeichen nur zwischen den Elementen ein, nicht davor und danach. Um Elemente in HTML-Tags für Fettschrift zu verpacken und sie mithilfe von Kommata zu trennen, schreiben Sie Folgendes:

```
$links  = '<b>';
$rechts = '</b>';

$html = $links . join("$links,$rechts", $html) . $rechts;
```

Zweitens können Sie bei `join()` nicht zwischen den Werten unterscheiden. Wenn Sie nur einen Teil der Einträge einschließen möchten, müssen Sie eine eigene Schleife verwenden:

```
$string = '';

foreach ($felder as $schluessel => $wert) {
    // Passwort auslassen.
    if ('password' != $key) {
        $string .= ",<b>$wert</b>";
    }
}

$string = substr($string, 1); // entfernt führendes ","
```

Beachten Sie, dass zu jedem Wert ein Trennzeichen hinzugefügt wird, das außerhalb der Schleife wieder entfernt werden muss. Es wirkt vielleicht etwas verschwenderisch, wenn Sie erst etwas hinzufügen, um es später wieder wegzunehmen; dies ist aber viel sauberer als (und in den meisten Fällen ebenso effizient wie) der Versuch, diese Logik innerhalb der Schleife unterzubringen. Sehen Sie selbst:

```
$string = '';
foreach ($felder as $schluessel => $wert) {
    // Passwort auslassen.
    if ('password' != $value) {
        if (!empty($string)) { $string .= ','; }
        $string .= "<b>$wert</b>";
    }
}
```

Hier müssen Sie `$string` jedes Mal prüfen, wenn Sie einen Wert anhängen. Das ist aufwendiger als der einfache Aufruf von `substr()`. Außerdem sollten Sie das Trennzeichen (in diesem Fall das Komma) vor dem Wert einfügen und nicht danach, denn man kann einen String am Anfang schneller kürzen als am Ende.

Siehe auch

Rezept 4.9 zur Ausgabe eines Arrays mit Kommata; die Dokumentationen zu `join()` unter *http://www.php.net/join* und `substr()` unter *http://www.php.net/substr*.

4.9 Ein Array mit Kommata ausgeben

Problem

Sie möchten ein Array so ausgeben, dass die Elemente durch Kommata getrennt sind, wenn es mehr als zwei Elemente sind, vor dem letzten Element aber ein »und« steht.

Lösung

Verwenden Sie die in Beispiel 4-1 dargestellte Funktion `pc_array_to_comma_string()`, die einen korrekten String liefert.

Beispiel 4-1: pc_array_to_comma_string()

```
function pc_array_to_comma_string($array) {

    switch (count($array)) {
    case 0:
        return '';

    case 1:
        return reset($array);

    case 2:
        return join(' und ', $array);

    default:
        $last = array_pop($array);
        return join(', ', $array) . " und $last";
    }
}
```

Diskussion

Wenn Sie eine Liste von Dingen ausgeben wollen, sollte dies in einer grammatikalisch korrekten Form geschehen. Es sieht unschön aus, wenn Sie Texte wie den folgenden anzeigen:

```
$thundercats = array('Lion-O', 'Panthro', 'Tygra', 'Cheetara', 'Snarf');
print 'Zu den ThunderCat-Helden gehören ' . join(', ', $thundercats) . '.';
Zu den ThunderCat-Helden gehören Lion-O, Panthro, Tygra, Cheetara, Snarf.
```

Die Implementierung dieser Funktion ist nicht völlig gradlinig, da `pc_array_to_comma_string()` mit allen Arrays funktionieren soll und nicht nur mit numerischen, die bei `0` beginnen. Bei einer Beschränkung auf diese Teilmenge würde man bei einem Array der Größe eins einfach `$array[0]` zurückgeben. Aber wenn das Array nicht bei `0` beginnt, ist `$array[0]` leer. Sie können sich jedoch die Tatsache zunutze machen, dass die Funktion `reset()`, die einen internen Zeiger des Arrays zurücksetzt, auch den Inhalt des ersten Array-Elements liefert.

Aus ähnlichen Gründen rufen Sie `array_pop()` auf, um das letzte Element zu bekommen, und gehen nicht davon aus, dass es sich bei `$array[count($array)-1]` befindet. Dies ermöglicht Ihnen, `join()` auf das `$array` anzuwenden.

Beachten Sie auch, dass der Code für den zweiten Fall eigentlich auch beim ersten Fall funktioniert. Und der Default-Code funktioniert ebenfalls (wenn auch ineffizient) beim zweiten Fall. Aber die Eigenschaft der Transitivität ist hier nicht anwendbar, daher können Sie den Default-Code nicht auf Arrays der Länge 1 anwenden.

Siehe auch

Rezept 4.8 dazu, wie man ein Array in einen String verwandelt; die Dokumentationen zu `join()` unter *http://www.php.net/join*, `array_pop()` unter *http://www.php.net/array-pop* und `reset()` unter *http://www.php.net/reset*.

4.10 Prüfen, ob sich ein Schlüssel in einem Array befindet

Problem

Sie möchten wissen, ob ein Array einen bestimmten Schlüssel enthält.

Lösung

Verwenden Sie `array_key_exists()`, um zu prüfen, ob ein Array einen bestimmten Schlüssel enthält, unabhängig von dem assoziierten Wert:

```
if (array_key_exists('Schlüssel', $array)) {
    /* in $array befindet sich ein Wert für 'Schlüssel' */
}
```

Verwenden Sie dagegen `isset()`, um zu prüfen, ob ein Array einen bestimmten Schlüssel enthält, dessen Wert ungleich `null` ist:

```
if (isset($array['Schlüssel'])) {
    /* in $array befindet sich ein Wert für 'Schlüssel' ungleich null */
}
```

Diskussion

Die Funktion `array_key_exists()` ignoriert die assoziierten Werte eines Schlüssels vollkommen und prüft nur, ob ein Schlüssel in einem Array existiert. Die Funktion `isset()` verhält sich hingegen genau so wie bei jeder anderen Variablen. Ist der assoziierte Wert eines Array-Schlüssels `null`, liefert die Funktion `false` zurück. In jedem anderen Fall wird `true` zurückgegeben. In der Einführung zu Kapitel 5 finden Sie weitere Informationen über die Wahrheitswerte von Variablen.

Siehe auch

Die Dokumentationen zu `isset()` unter *http://www.php.net/isset* und zu `array_key_exists()` unter *http://php.net/array_key_exists*.

4.11 Prüfen, ob sich ein Element in einem Array befindet

Problem

Sie möchten wissen, ob ein Array einen bestimmten Wert enthält.

Lösung

Verwenden Sie `in_array()`:

```
if (in_array($wert, $array)) {
    // Ein Element des Arrays $array hat $wert als Wert.
}
```

Diskussion

Mit `in_array()` können Sie prüfen, ob ein Element eines Arrays einen bestimmten Wert enthält:

```
$buchsammlung = array('Emma', 'Stolz und Vorurteil, 'Die Abtei von Northanger');
$buch = 'Verstand und Gefühl';

if (in_array($buch, $buchsammlung)) {
    echo 'Habe ich.';
} else {
    echo 'Brauche ich.';
}
```

Standardmäßig vergleicht `in_array()` die Dinge mit dem Gleichtheitsoperator `==`. Um die strikte Gleichheitsprüfung mit `===` zu verwenden, übergeben Sie `true` als dritten Parameter an `in_array()`:

```
$array = array(1, '2', 'drei');

in_array(0, $array);        // wahr!
in_array(0, $array, true);  // falsch
in_array(1, $array);        // wahr
in_array(1, $array, true);  // wahr
in_array(2, $array);        // wahr
in_array(2, $array, true);  // falsch
```

Die erste Prüfung, `in_array(0, $array)`, ergibt `true`, weil PHP beim Vergleich der Zahl `0` mit `drei` den String `drei` in einen Integer-Wert abbildet. Da `drei` aber kein numerischer

String ist, wie es bei 2 der Fall ist, wird daraus 0. Daher glaubt in_array(), es gäbe eine Übereinstimmung.

Als Konsequenz ist es am sichersten, wenn man beim Vergleich von Zahlen mit Daten, zu denen auch Strings gehören können, den strikten Vergleich verwendet.

Wenn Sie merken, dass Sie in_array() mehrmals mit demselben Array aufrufen, ist es möglicherweise besser, ein assoziatives Array zu verwenden, bei dem die ursprünglichen Array-Elemente die Schlüssel des neuen assoziativen Arrays sind. Das Suchen von Einträgen mithilfe von in_array() nimmt lineare Zeit in Anspruch; bei einem assoziativen Array ist der Zeitbedarf konstant.

Wenn Sie das assoziative Array nicht direkt erzeugen können, sondern es aus einem traditionellen Array mit Integer-Schlüsseln konvertieren müssen, verwenden Sie die Funktion array_flip(), mit der man die Schlüssel und Werte eines Arrays vertauscht:

```
$buchsammlung = array('Emma',
                      'Stolz und Vorurteil',
                      'Die Abtei von Northanger');

// Numerisches Array in assoziatives Array konvertieren.
$buchsammlung = array_flip($buchsammlung);
$buch = 'Verstand und Gefühl';

if (isset($buchsammlung[$buch])) {
    echo 'Habe ich.';
} else {
    echo 'Brauche ich.';
}
```

Beachten Sie, dass dabei mehrere Schlüssel mit dem gleichen Wert im getauschten Array zu einem Element verdichtet werden.

Siehe auch

Rezept 4.12 dazu, wie man die Position eines Elements in einem Array bestimmt; die Dokumentationen zu in_array() unter *http://www.php.net/in-array* und array_flip() unter *http://www.php.net/array-flip*.

4.12 Die Position eines Elements in einem Array feststellen

Problem

Sie möchten wissen, ob sich ein Element in einem Array befindet, und wenn dies der Fall ist, möchten Sie auch wissen, an welcher Stelle es ist.

Lösung

Verwenden Sie `array_search()`. Diese Funktion gibt den Schlüssel des gefundenen Elements oder `false` zurück:

```
$position = array_search($wert, $array);
if ($position !== false) {
    // Das Element an der Position $position im Array $array hat den Wert $wert.
}
```

Diskussion

Verwenden Sie `in_array()`, um herauszufinden, ob ein Array einen bestimmten Wert enthält; verwenden Sie `array_search()`, um festzustellen, wo sich der Wert befindet. Da allerdings `array_search()` auch mit Suchvorgängen sinnvoll umgeht, bei denen der Wert nicht gefunden wird, verwendet man besser gleich `array_search()` anstelle von `in_array()`. Der Geschwindigkeitsunterschied ist minimal, und die zusätzliche Information ist potenziell nützlich:

```
$lieblingsspeisen = array(1 => 'Artischocken', 'Brot', 'Blumenkohl', 'Gefüllte Eier');
$speise = 'Blumenkohl';
$position = array_search($speise, $lieblingsspeisen);

if ($position !== false) {
    echo "Meine Lieblingsspeise Nr. $position ist $speise";
} else {
    echo "$speise mag ich nicht!";
}
```

Verwenden Sie den Vergleich mit `!==` gegen `false`, denn wenn Ihr String im Array an der Position `0` gefunden wird, ergibt `if` ein logisches `false`, und das ist nicht das Gemeinte oder Gewünschte.

Wenn sich ein Wert mehrmals in dem Array befindet, gibt es nur die Garantie, dass `array_search()` eins der Exemplare findet, aber nicht unbedingt das erste.

Siehe auch

Rezept 4.11 zur Prüfung, ob sich ein Element in einem Array befindet; die Dokumentation zu `array_search()` unter *http://www.php.net/array-search*; mehr über raffinierte Suchvorgänge in Arrays mithilfe von regulären Ausdrücken finden Sie bei `preg_replace()` unter *http://www.php.net/preg-replace* und in Kapitel 16.

4.13 Elemente finden, die einer bestimmten Prüfung standhalten

Problem

Sie möchten Einträge in einem Array lokalisieren, die bestimmten Anforderungen genügen.

Lösung

Verwenden Sie eine foreach-Schleife:

```
$filme = array(...);

foreach ($filme as $film) {
    if ($film['verkaufszahl'] < 5000000) { $flops[] = $film; }
}
```

Oder array_filter():

```
$filme = array(...);

function flops($film) {
    return ($film['verkaufszahl'] < 5000000) ? 1 : 0;
}

$flops = array_filter($filme, 'flops');
```

Diskussion

Die foreach-Schleifen sind einfach; lassen Sie die Daten abrollen und fügen Sie alle Ihren Kriterien entsprechenden Elemente in das Rückgabe-Array ein.

Wenn Sie nur das erste passende Element benötigen, verlassen Sie die Schleife mit break:

```
foreach ($filme as $film) {
    if ($film['verkaufszahl'] > 200000000) { $kassenschlager = $film; break; }
}
```

Aus einer Funktion können Sie auch direkt zurückkehren:

```
function kassenschlager($filme) {
    foreach ($filme as $film) {
        if ($film['verkaufszahl'] > 200000000) { return $film; }
    }
}
```

Für array_filter() erstellen Sie dagegen zunächst eine Callback-Funktion, die bei allen gewünschten Werten true zurückgibt und bei allen anderen Werten false. Dann weisen Sie PHP mit der Funktion array_filter() an, das Array in der gleichen Weise zu verarbeiten, wie Sie es mit foreach tun.

Aus einem array_filter() kann man nicht vorzeitig entlassen werden, deshalb bietet foreach mehr Flexibilität und ist einfacher zu verstehen. Außerdem ist es einer der wenigen Fälle, bei denen eine eingebaute PHP-Funktion keine deutlich bessere Performance aufweist als die Programmierung auf Benutzerebene.

Siehe auch

Die Dokumentation zu array_filter() unter *http://www.php.net/array-filter*.

4.14 Das Array-Element mit dem größten oder kleinsten Wert finden

Problem

Sie haben ein Array mit Elementen, von denen Sie das mit dem größten oder dem kleinsten Wert herausfinden möchten. Beispielsweise möchten Sie den passenden Maßstab bei der Erstellung eines Histogramms ermitteln.

Lösung

Um das größte Element zu finden, verwenden Sie max():

```
$groesstes = max($array);
```

Um das kleinste Element zu finden, verwenden Sie min():

```
$kleinstes = min($array);
```

Diskussion

Normalerweise gibt max() das größere von zwei Elementen zurück, aber wenn Sie der Funktion ein Array übergeben, durchsucht sie stattdessen das gesamte Array. Leider gibt es aber keine Möglichkeit, mit max() den Index des größten Elements zu finden. Hierzu müssen Sie das Array in umgekehrter Reihenfolge sortieren und dadurch das größte Element an die Position 0 bringen:

```
arsort($array);
```

Danach befindet sich das größte Element in $array[0].

Wenn Sie die Reihenfolge des ursprünglichen Arrays nicht durcheinander bringen möchten, erstellen Sie eine Kopie und sortieren diese:

```
$kopie = $array;
arsort($kopie);
```

Das gleiche Konzept lässt sich auch auf min() anwenden, nur benutzen Sie dann asort() anstelle von arsort().

Siehe auch

Rezept 4.16 zum Sortieren von Arrays; die Dokumentationen zu `max()` unter *http://www.php.net/max*, `min()` unter *http://www.php.net/min*, `arsort()` unter *http://www.php.net/arsort* und `asort()` unter *http://www.php.net/asort*.

4.15 Ein Array umkehren

Problem

Sie möchten die Elemente eines Arrays in die umgekehrte Reihenfolge bringen.

Lösung

Verwenden Sie `array_reverse()`:

```
$array = array('Null', 'Eins', 'Zwei');
$umgekehrt = array_reverse($array);
```

Diskussion

Die Funktion `array_reverse()` kehrt die Reihenfolge der Elemente eines Arrays um. Allerdings können Sie häufig auch ohne diese Funktion auskommen. Wenn Sie ein Array umkehren möchten, das Sie gerade sortiert haben, drehen Sie einfach direkt die Sortierung um. Wenn Sie eine Liste umkehren möchten, die Sie in einer Schleife durchlaufen und verarbeiten möchten, kehren Sie einfach die Schleife um. Anstelle von:

```
for ($i = 0, $size = count($array); $i < $size; $i++) {
    ...
}
```

schreiben Sie Folgendes:

```
for ($i = count($array) - 1; $i >=0 ; $i--) {
    ...
}
```

Verwenden Sie aber wie immer `for`-Schleifen nur mit dicht gepackten Arrays.

Unter Umständen können Sie als weitere Alternative auch die Reihenfolge umdrehen, in der die Elemente im Array platziert werden. Wenn Sie beispielsweise ein Array aus einer Reihe von Zeilen füllen, die von einer Datenbank geliefert werden, können Sie möglicherweise die Abfrage zu `ORDER DESC` ändern. Die genaue Syntax für Ihre Datenbank finden Sie in Ihrem Datenbank-Handbuch.

Siehe auch

Die Dokumentation zu `array_reverse()` unter *http://www.php.net/array-reverse*.

4.16 Ein Array sortieren

Problem

Sie möchten ein Array in einer bestimmten Weise sortieren.

Lösung

Um eine Array im traditionellen Sinn zu sortieren, verwenden Sie `sort()`:

```
$staaten = array('Delaware', 'Pennsylvania', 'New Jersey');
sort($staaten);
```

Um numerisch zu sortieren, übergeben Sie `SORT_NUMERIC` als zweiten Parameter an `sort()`.

```
$treffer = array(1, 10, 2, 20);
sort($treffer, SORT_NUMERIC);
```

Dadurch wird auf die Zahlen in aufsteigender Reihenfolge (`1, 2, 10, 20`) anstelle der lexikographischen Reihenfolge (`1, 10, 2, 20`) zurückgegriffen.

Diskussion

Durch die Funktion `sort()` geht die Schlüssel/Wert-Verknüpfung zwischen den Elementen verloren, und die Einträge werden von `0` an aufwärts neu indiziert. (Die einzige Ausnahme von dieser Regel ist ein Array mit nur einem Element; dieses einzelne Element bekommt dann nicht den Index `0`. Dies ist aber seit PHP 4.2.3 behoben.)

Um die Schlüssel/Wert-Beziehungen zu bewahren, verwenden Sie `asort()`. Die Funktion `asort()` wird normalerweise für assoziative Arrays verwendet, sie kann aber auch nützlich sein, wenn die Indices eines numerischen Arrays eine Bedeutung haben:

```
$staaten = array(1 => 'Delaware', 'Pennsylvania', 'New Jersey');
asort($staaten);

while (list($rang, $staat) = each($staaten)) {
    print "$staat war der $rang. Staat, der den USA beigetreten ist\n";

}
```

Mit `natsort()` können Sie das Array unter Verwendung eines natürlichen Sortierungsalgorithmus sortieren. Bei einer natürlichen Sortierung können Sie in den Elementen Strings und Zahlen mischen und bekommen trotzdem die richtige Antwort.

```
$tests = array('test1.php', 'test10.php', 'test11.php', 'test2.php');
natsort($tests);
```

Die Elemente sind nun folgendermaßen angeordnet: `'test1.php'`, `'test2.php'`, `'test10.php'` und `'test11.php'`. Bei der natürlichen Sortierung kommt die Zahl `10` nach der Zahl `2`, bei der traditionellen Sortierung ist das Gegenteil der Fall. Für eine natürliche Sortierung, die unabhängig von der Groß-/Kleinschreibung ist, verwenden Sie `natcasesort()`.

Um das Array in umgekehrter Reihenfolge zu sortieren, können Sie `rsort()` oder `arsort()` verwenden. Die letztere Funktion unterscheidet sich von `rsort()` durch die Bewahrung der Schlüssel. Ein `natrsort()` oder `natcasersort()` gibt es dagegen nicht. Sie können diesen Funktionen auch `SORT_NUMERIC` übergeben.

Siehe auch

Rezept 4.17 zum Sortieren mit einer benutzerdefinierten Vergleichsfunktion und Rezept 4.18 zum Sortieren mehrerer Arrays; die Dokumentationen zu `sort()` unter *http://www.php.net/sort*, `asort()` unter *http://www.php.net/asort*, `natsort()` unter *http://www.php.net/natsort*, `natcasesort()` unter *http://www.php.net/natcasesort*, `rsort()` unter *http://www.php.net/rsort* und `arsort()` unter *http://www.php.net/arsort*.

4.17 Ein Array über ein berechnetes Feld sortieren

Problem

Sie möchten eine eigene Sortierroutine definieren.

Lösung

Verwenden Sie `usort()` in Kombination mit einer benutzerdefinierten Vergleichsfunktion:

```
// In umgekehrte natürliche Anordnung sortieren.
function natrsort($a, $b) {
    return strnatcmp($b, $a);
}

$tests = array('test1.php', 'test10.php', 'test11.php', 'test2.php');
usort($tests, 'natrsort');
```

Diskussion

Die Vergleichsfunktion muss einen Wert größer als 0 liefern, wenn `$a > $b`, `0`, wenn `$a == $b`, und einen Wert kleiner als 0, wenn `$a < $b` ist. Bei einer umgekehrten Sortierung tut sie das Gegenteil. Die Funktion `strnatcmp()` in der Lösung folgt diesen Regeln.

Um die Sortierung umzukehren, multiplizieren Sie die Rückgabewerte von `strnatcmp($a, $b)` nicht mit `-1`, sondern vertauschen die Anordnung der Argumente zu `strnatcmp($b, $a)`.

Die Sortierfunktion muss kein Wrapper für eine existierende Sortierung sein. Die Funktion `pc_date_sort()` in Beispiel 4-2 zeigt beispielsweise, wie man Datumswerte sortiert.

Beispiel 4-2: pc_date_sort()

```
// Erwartet werden Datumswerte im Format "TT.MM.YYYY".
function pc_date_sort($a, $b) {
    list($a_day, $a_month, $a_year) = explode('.', $a);
```

Beispiel 4-2: pc_date_sort() (Fortsetzung)

```
    list($b_day, $b_month, $b_year) = explode('.', $b);

    if ($a_year  > $b_year ) return  1;
    if ($a_year  < $b_year ) return -1;

    if ($a_month > $b_month) return  1;
    if ($a_month < $b_month) return -1;

    if ($a_day   > $b_day  ) return  1;
    if ($a_day   < $b_day  ) return -1;

    return 0;
}

$dates = array('14.12.2000', '10.08.2001', '07.08.1999');
usort($dates, 'pc_date_sort');
```

Beim Sortieren berechnet `usort()` die Rückgabewerte der Sortierfunktion immer wieder neu, wenn zwei Elemente verglichen werden müssen; dies verlangsamt die Sortierung. Um unnötige Arbeit zu vermeiden, können Sie die Vergleichswerte in einem Cache halten, wie es die Funktion `pc_array_sort()` in Beispiel 4-3 zeigt.

Beispiel 4-3: pc_array_sort()

```
function pc_array_sort($array, $map_func, $sort_func = '') {
    $mapped = array_map($map_func, $array); // Cache für die $map_func()-Werte

    if ('' == $sort_func) {
        asort($mapped);                     // asort() ist schneller als usort()
    }  else {
        uasort($mapped, $sort_func);        // muss Schlüssel bewahren
    }

    while (list($key) = each($mapped)) {
        $sorted[] = $array[$key];           // verwendet sortierte Schlüssel
    }

    return $sorted;
}
```

Um überflüssige Arbeit zu vermeiden, verwendet `pc_array_sort()` das temporäre Array `$mapped` als Cache für die Rückgabewerte. Dann sortiert sie `$mapped`, wobei sie entweder die standardmäßige Sortierreihenfolge oder eine vom Benutzer angegebene Sortierroutine verwendet. Wichtig ist, dass sie eine Sortierung verwendet, bei der die Schlüssel/Wert-Beziehung erhalten bleibt. Standardmäßig benutzt sie `asort()`, denn `asort()` ist schneller als `uasort()`. (Das langsame Verhalten von `uasort()` ist der eigentliche Grund für `pc_array_sort()`.) Schließlich erzeugt sie das sortierte Array `$sorted`, wobei sie die sortierten Schlüssel in `$mapped` als Index für die Werte im ursprünglichen Array verwendet.

Bei kleinen Arrays und einfachen Sortierfunktionen ist usort() schneller, aber wenn die Anzahl der Berechnungen wächst, überholt pc_array_sort() die Funktion usort(). Das folgende Beispiel sortiert die Elemente nach ihrer String-Länge – eine relativ einfache, benutzerdefinierte Sortierung:

```
function pc_u_length($a, $b) {
    $a = strlen($a);
    $b = strlen($b);

    if ($a == $b) return  0;
    if ($a  > $b) return  1;
                  return -1;
}

function pc_map_length($a) {
    return strlen($a);
}

$tests = array('ein', 'zwei', 'drei', 'vier', 'fünf',
               'sechs', 'sieben', 'acht', 'neun', 'zehn');

// schneller für < 5 Elemente mit pc_u_length()
usort($tests, 'pc_u_length');

// schneller für >= 5 Elemente mit pc_map_length()
$tests = pc_array_sort($tests, 'pc_map_length');
```

Hier ist pc_array_sort() schneller als usort(), sobald die Array-Größe fünf Elemente erreicht.

Siehe auch

Rezept 4.16 zur grundlegenden Sortierung und Rezept 4.18 zum Sortieren mehrerer Arrays; die Dokumentationen zu usort() unter *http://www.php.net/usort*, asort() unter *http://www.php.net/asort* und array_map() unter *http://www.php.net/array-map*.

4.18 Mehrere Arrays sortieren

Problem

Sie möchten mehrere Arrays oder ein Array mit mehreren Dimensionen sortieren.

Lösung

Verwenden Sie array_multisort():

Um mehrere Arrays simultan zu sortieren, übergeben Sie mehrere Arrays an array_multisort():

```
$farben = array('Rot', 'Weiß', 'Blau');
$staedte = array('Boston', 'New York', 'Chicago');

array_multisort($farben, $staedte);
print_r($farben);
print_r($staedte);
Array
(
    [0] => Blau
    [1] => Rot
    [2] => Weiß
)
Array
(
    [0] => Chicago
    [1] => Boston
    [2] => New York
)
```

Um mehrere Dimensionen innerhalb eines einzelnen Arrays zu sortieren, übergeben Sie die jeweiligen Array-Elemente:

```
$kram = array('farben' => array('Rot', 'Weiß', 'Blau'),
              'staedte' => array('Boston', 'New York', 'Chicago'));

array_multisort($kram['farben'], $kram['staedte']);
print_r($kram);
Array
(
    [farben] => Array
        (
            [0] => Blau
            [1] => Rot
            [2] => Weiß
        )

    [staedte] => Array
        (
            [0] => Chicago
            [1] => Boston
            [2] => New York
        )

)
```

Um die Art der Sortierung zu ändern, übergeben Sie wie bei `sort()` nach dem Array `SORT_REGULAR`, `SORT_NUMERIC` oder `SORT_STRING`. Um die Sortierreihenfolge zu ändern, übergeben Sie – anders als bei `sort()` – nach dem Array `SORT_ASC` oder `SORT_DESC`. Darüber hinaus können Sie nach dem Array sowohl eine Sortierart als auch eine Sortierreihenfolge angeben.

Diskussion

Die Funktion `array_multisort()` kann mehrere Arrays auf einmal oder ein multidimensionales Array nach einer oder mehreren Dimensionen sortieren. Die Arrays werden wie

Spalten einer Tabelle behandelt, die nach den Zeilen sortiert wird. Das erste Array ist das Haupt-Array, nach dem sortiert wird; alle Elemente in den anderen Arrays werden auf der Basis der sortierten Reihenfolge des ersten Arrays umgeordnet. Wenn der Vergleich zweier Elemente im ersten Array Gleichheit ergibt, wird die Sortierreihenfolge durch das zweiten Array bestimmt und so weiter.

Die Standardeinstellungen für die Sortierung sind `SORT_REGULAR` und `SORT_ASC`, und diese werden nach jedem Array zurückgesetzt; daher gibt es keinen Grund, einen dieser beiden Werte zu übergeben, es sei denn der Klarheit wegen.

```
$zahlen = array(0, 1, 2, 3);
$buchstaben = array('a', 'b', 'c', 'd');
array_multisort($zahlen, SORT_NUMERIC, SORT_DESC,
                $buchstaben, SORT_STRING , SORT_DESC);
```

In diesem Beispiel wird die Reihenfolge der Arrays umgekehrt.

Siehe auch

Rezept 4.16 zum einfachen Sortieren und Rezept 4.17 zum Sortieren mit einer benutzerdefinierten Funktion; die Dokumentation zu `array_multisort()` unter *http://www.php.net/array-multisort*.

4.19 Ein Array mithilfe einer Methode statt einer Funktion sortieren

Problem

Sie möchten eine benutzerdefinierte Sortierroutine zum Sortieren eines Arrays definieren, aber Sie wollen anstelle einer Funktion eine Objekt-Methode verwenden.

Lösung

Übergeben Sie statt des Funktionsnamens ein Array, das einen Klassennamen und eine Methode enthält:

```
usort($access_times, array('dates', 'compare'));
```

Diskussion

Wie die benutzerdefinierte Sortierfunktion muss auch die Objekt-Methode zwei Eingabeargumente übernehmen und 1, 0 oder –1 zurückgeben, je nachdem, ob der erste Parameter größer, gleich oder kleiner als der zweite ist:

```
class pc_sort {
    // Stringvergleich in umgekehrter Reihenfolge
    function strrcmp($a, $b) {
```

```
            return strcmp($b, $a);
        }
    }

    usort($words, array('pc_sort', 'strrcmp'));
```

Siehe auch

Kapitel 7 mit weiteren Informationen über Klassen und Objekte; Rezept 4.17 mit mehr Informationen über benutzerdefiniertes Sortieren von Arrays.

4.20 Ein Array in eine zufällige Reihenfolge bringen

Problem

Sie möchten die Elemente eines Arrays vermischen, sodass sie eine zufällige Reihenfolge aufweisen.

Lösung

Wenn Sie mit PHP 4.3 oder höher arbeiten, verwenden Sie `shuffle()`:

```
shuffle($array);
```

Wenn Sie eine ältere Version haben, nehmen Sie die in Beispiel 4-4 dargestellte Funktion `pc_array_shuffle()`.

Beispiel 4-4: pc_array_shuffle()

```
function pc_array_shuffle($array) {
    $i = count($array);

    while(--$i) {
        $j = mt_rand(0, $i);

        if ($i != $j) {
            // Elemente vertauschen.
            $tmp = $array[$j];
            $array[$j] = $array[$i];
            $array[$i] = $tmp;
        }
    }

    return $array;
}
```

Hier folgt ein Beispiel:

```
$karten = range(1,52); // gibt 52 "Karten" aus
$karten = pc_array_shuffle($karten);
```

Diskussion

Es gibt in PHP bereits eine Funktion `shuffle()` zum Mischen von Arrays, aber bis zur Version PHP 4.2.2 tut sie ihren Dienst nicht korrekt. Der eingebaute Misch-Algorithmus tendiert dazu, bestimmte Permutationen gegenüber anderen zu bevorzugen. Am Ende sehen die Elemente zwar zufällig angeordnet aus, aber da nicht jedes Element die gleiche Chance hat, an alle Positionen zu gelangen, ist es kein wirkliches Mischen. Seit PHP 4.3 ist dieser Mangel behoben.

Die Funktion `pc_array_shuffle()`, auch als Fisher-Yates-Mischung bekannt, verteilt die Elemente gleichmäßig im Array. Verwenden Sie sie, wenn Sie eine PHP-Version haben, die älter als 4.3 ist. Anders als `shuffle()` gibt diese Funktion das durchgerührte Array zurück und ändert es nicht an Ort und Stelle. Auch erfordert sie ein dicht gepacktes Array mit Integer-Schlüsseln.

Siehe auch

Rezept 4.21 mit einer Funktion, die das Mischen eines Kartenstapels simuliert; die Dokumentation zu `shuffle()` unter *http://www.php.net/shuffle*.

4.21 Einen Kartenstapel mischen

Problem

Sie möchten einen Stapel Spielkarten mischen und ausgeben.

Lösung

Legen Sie ein Array mit 52 Integer-Zahlen an, mischen Sie es, und bilden Sie es auf die Spielkarten ab:

```
$farben = array('Kreuz', 'Pik', 'Herz', 'Karo');
$karten = array('Ass', 2, 3, 4, 5, 6, 7, 8, 9, 10, 'Bube', 'Dame', 'König');

$stapel = pc_array_shuffle(range(0, 51));

while (($zug = array_pop($stapel)) != NULL) {
    print  $farben[$zug % 4] . $karten[$zug / 4] . "\n";
}
```

Dieses Programm verwendet die Funktion `pc_array_shuffle()` aus Rezept 4.20.

Diskussion

Hier werden die zwei Arrays `$farben` und `$karten` angelegt, die deutsche Spielkarten repräsentieren sollen. Die Zahlen `0` bis `51` werden zufällig angeordnet und dem Array

$deck zugewiesen. Um eine Karte auszugeben, nehmen Sie sie oben aus dem Array und behandeln damit das Array buchstäblich wie einen Kartenstapel.

Es ist notwendig, innerhalb der while-Schleife die Prüfung auf NULL hinzuzufügen, da andernfalls die Schleife beim Ziehen der nullten Karte beendet wird. Wenn Sie den Stapel so ändern, dass er die Zahlen 1 bis 52 enthält, werden die Berechnungen für die Entscheidung, welche Zahl zu welcher Karte gehört, komplexer.

Um mehrere Karten auf einmal auszugeben, können Sie array_slice() verwenden:

```
array_slice($stapel, count($karten) * -1);
```

Siehe auch

Rezept 4.20 mit einer Funktion zum zufälligen Anordnen eines Arrays; die Dokumentation zu array_slice() unter *http://www.php.net/array-slice*.

4.22 Doppelte Elemente aus einem Array entfernen

Problem

Sie möchten Duplikate in einem Array eliminieren.

Lösung

Wenn das Array bereits vollständig ist, verwenden Sie array_unique(); diese Funktion gibt ein neues Array zurück, das keine doppelten Werte enthält:

```
$unique = array_unique($array);
```

Wenn Sie das Array erzeugen, indem Sie Ergebnisse verarbeiten, können Sie bei numerischen Arrays die folgende Technik verwenden:

```
foreach ($_REQUEST['obst'] as $frucht) {
    if (!in_array($array, $frucht)) { $array[] = $frucht; }
}
```

Und dies ist eine Möglichkeit für assoziative Arrays:

```
foreach ($_REQUEST['obst'] as $frucht) {
    $array[$frucht] = $frucht;
}
```

Diskussion

Wenn die Verarbeitung bereits abgeschlossen ist, stellt array_unique() die beste Möglichkeit zum Eliminieren von Duplikaten dar. Innerhalb einer Schleife können Sie dagegen das Entstehen von Duplikaten dadurch verhindern, dass Sie prüfen, ob die Elemente bereits im Array enthalten sind.

Eine noch schnellere Möglichkeit als `in_array()` ist ein Hybrid-Array, bei dem der Schlüssel und der Wert jedes Elements gleich sind. Dies macht die lineare Prüfung von `in_array()` überflüssig, ermöglicht Ihnen aber trotzdem, die Familie der Array-Funktionen zu nutzen, die über die Werte eines Arrays operieren und nicht über die Schlüssel.

Tatsächlich geht es schneller, wenn Sie die Methode mit dem assoziativen Array verwenden und dann mit dem Ergebnis `array_values()` aufrufen (oder in diesem Fall auch `array_keys()`, aber `array_values()` ist etwas schneller), als wenn Sie direkt ein numerisches Array verwenden und dabei die Mehrbelastung durch das `in_array()` in Kauf nehmen.

Siehe auch

Die Dokumentation zu `array_unique()` unter *http://www.php.net/array-unique*.

4.23 Die Vereinigungs-, Schnitt- oder Differenzmenge zweier Arrays ermitteln

Problem

Sie haben zwei Arrays, bei denen Sie die Vereinigungsmenge (alle Elemente), die Schnittmenge (Elemente, die in beiden, nicht nur in einem vorhanden sind) oder die Differenzmenge (Elemente, die in einem, aber nicht in beiden vorhanden sind) herausfinden möchten.

Lösung

Um die Vereinigungsmenge zu bilden:

```
$union = array_unique(array_merge($a, $b));
```

Um die Schnittmenge zu bilden:

```
$intersection = array_intersect($a, $b);
```

Um die einfache Differenzmenge zu finden:

```
$difference = array_diff($a, $b);
```

Und für die symmetrische Differenz:

```
$difference = array_merge(array_diff($a, $b), array_diff($b, $a));
```

Diskussion

Viele erforderliche Komponenten für solche Berechnungen sind in PHP eingebaut, es ist alles nur eine Frage der Kombination in der richtigen Reihenfolge.

Um die Vereinigungsmenge zu finden, mischen Sie die beiden Arrays und erhalten damit ein Riesen-Array mit allen Werten. Aber die Funktion `array_merge()` lässt doppelte Werte zu, wenn sie zwei numerische Arrays mischt, daher rufen Sie zusätzlich noch `array_unique()` auf, um diese herauszufiltern. Allerdings können dabei Lücken zwischen den Einträgen entstehen, denn `array_unique()` komprimiert das Array nicht. Dies ist allerdings kein Problem, da `foreach` und `each()` mit lückenhaft gefüllten Arrays ohne Störung umgehen können.

Die Funktion zum Berechnen der Schnittmenge hat einfach den Namen `array_intersection()` und erfordert keine weiteren Maßnahmen von Ihrer Seite.

Die Funktion `array_diff()` gibt ein Array zurück, das alle eindeutigen Elemente in `$alt` enthält, die nicht in `$neu` sind. Dies wird als *einfache Differenz* bezeichnet:

```
$alt = array('To', 'be', 'or', 'not', 'to', 'be');
$neu = array('To', 'be', 'or', 'whatever');
$differenz = array_diff($alt, $neu);
Array
(
    [3] => not
    [4] => to
)
```

Das resultierende Array `$differenz` enthält `'not'` und `'to'`, denn `array_diff()` unterscheidet zwischen Klein- und Großschreibung. Das Element `'whatever'` ist nicht im Ergebnis enthalten, da es nicht in `$alt` vorkommt.

Um die umgekehrte Differenz zu finden, also die eindeutigen Elemente in `$neu`, die in `$alt` fehlen, vertauschen Sie die Argumente:

```
$alt = array('To', 'be', 'or', 'not', 'to', 'be');
$neu = array('To', 'be', 'or', 'whatever');
$umgekehrte_diff = array_diff($neu, $alt);
Array
(
    [3] => whatever
)
```

Das Array `$umgekehrte_diff` enthält nur `'whatever'`.

Wenn Sie eine Funktion oder einen anderen Filter auf `array_diff()` anwenden möchten, benötigen Sie einen eigenen Algorithmus zur Differenzbildung:

```
// Eine Differenzbildung implementieren, die unabhängig
// von Groß- und Kleinschreibung ist; diff -i.

$gesehen = array();
foreach ($neu as $n) {
    $gesehen[strtolower($n)]++;
}

foreach ($alt as $a) {
    $a = strtolower($a);
```

```
        if (!$gesehen[$a]) { $diff[$a] = $a; }
    }
```

Das erste `foreach` baut eine Nachschlagetabelle als assoziatives Array auf. Dann durchlaufen Sie in einer Schleife `$alt` und, wenn Sie keinen Eintrag in der Tabelle finden, fügen das Element zu `$diff` hinzu.

Es kann etwas schneller gehen, wenn man `array_diff()` mit `array_map()` kombiniert:

```
$diff = array_diff(array_map('strtolower', $alt), array_map('strtolower', $neu));
```

Die symmetrische Differenz besteht aus allen Elementen, die in `$a`, aber nicht in `$b` sind sowie die in `$b`, aber nicht in `$a` sind:

```
$differenz = array_merge(array_diff($a, $b), array_diff($b, $a));
```

Einmal klargelegt, ist der Algorithmus verständlich. Sie rufen zweimal `array_diff()` auf und finden damit die beiden Differenzen. Dann mischen Sie diese zusammen in ein Array. Der Aufruf von `array_unique()` ist hier nicht erforderlich, denn Sie haben ja diese Arrays gerade absichtlich so aufgebaut, dass sie nichts gemeinsam haben.

Siehe auch

Die Dokumentationen zu `array_unique()` unter *http://www.php.net/array-unique*, `array_intersect()` unter *http://www.php.net/array-intersect*, `array_diff()` unter *http://www.php.net/array-diff*, `array_merge()` unter *http://www.php.net/array-merge* und `array_map()` unter *http://www.php.net/array-map*.

4.24 Alle Elementkombinationen eines Arrays finden

Problem

Sie möchten alle Mengenkombinationen herausfinden, die einige oder alle Elemente eines Arrays enthalten; dies wird auch als *Potenzmenge* bezeichnet.

Lösung

Verwenden Sie die in Beispiel 4-5 dargestellte Funktion `pc_array_power_set()`.

Beispiel 4-5: pc_array_power_set()

```
function pc_array_power_set($array) {
    // Initialisieren durch Hinzufügen einer leeren Menge
    $results = array(array());

    foreach ($array as $element)
        foreach ($results as $combination)
            array_push($results, array_merge(array($element), $combination));
```

Beispiel 4-5: pc_array_power_set() (Fortsetzung)

```
    return $results;
}
```

Diese Funktion gibt ein Array aus Arrays zurück, das jede Kombination von Elementen einschließlich der leeren Menge enthält. Ein Beispiel:

```
$menge = array('A', 'B', 'C');
$potenzmenge = pc_array_power_set($menge);
```

`$potenzmenge` enthält acht Arrays:

```
array();
array('A');
array('B');
array('C');
array('A', 'B');
array('A', 'C');
array('B', 'C');
array('A', 'B', 'C');
```

Diskussion

Zuerst fügen Sie die leere Menge {} zu den Ergebnissen hinzu. Schließlich besteht eine mögliche Kombination einer Menge darin, dass keines seiner Elemente übernommen wird.

Der Rest dieser Funktion beruht auf der Natur von Kombinationen und der PHP-Implementierung von `foreach`. Jedes neue Element, das zu dem Array hinzugefügt wird, erhöht die Anzahl der Kombinationen. Die neuen Kombinationen bestehen immer aus allen alten Kombinationen und dem neuen Element. Ein Array mit den beiden Elementen `A` und `B` ergibt vier mögliche Kombinationen: {}, {A}, {B} und {A, B}. Wenn man zu dieser Menge `C` hinzufügt, bleiben die vier vorangehenden Kombinationen erhalten, aber es kommen noch vier neue Kombinationen hinzu: {C}, {A, C}, {B, C} und {A, B, C}.

Daher arbeitet die äußere `foreach`-Schleife alle Elemente der Liste ab, während das innere `foreach` alle vorhandenen, aus den vorherigen Elementen erzeugten Kombinationen durchläuft. Das ist die knifflige Stelle; Sie müssen genau wissen, wie sich PHP während eines `foreach` verhält.

Die Funktion `array_merge()` kombiniert das Element mit den vorherigen Kombinationen. Beachten Sie aber, dass das Array `$results`, dem das neue Array mit `array_push()` hinzugefügt wird, dasjenige ist, das in dem `foreach` durchlaufen wird. Normalerweise würde es zu einer Endlosschleife führen, wenn Sie Einträge zu `$results` hinzufügen, bei PHP ist dies aber nicht der Fall, denn PHP arbeitet mit einer Kopie der ursprünglichen Liste. Wenn Sie aber eine Ebene höher in die äußere Schleife zurückspringen und das `foreach` mit dem nächsten `$element` erneut ausführen, wird es zurückgesetzt. Daher können Sie direkt mit `$results` arbeiten und das Array als Stapelspeicher für Ihre Kombina-

tionen verwenden. Indem Sie alles als Array speichern, bekommen Sie mehr Flexibilität, wenn Sie die Daten ausgeben oder die Kombinationen später weiter unterteilen möchten.

Um die leere Menge zu entfernen, ersetzen Sie die Zeile am Anfang:

```
// Initialisieren durch Hinzufügen einer leeren Menge
$results = array(array());
```

durch:

```
// Initialisieren durch Hinzufügen des ersten Elements
$results = array(array(array_pop($array)));
```

Da es zu einem Array mit einem Element nur eine einzige Kombination gibt, nämlich das Array selbst, hat die Entnahme des ersten Elements den gleichen Effekt wie der erste Durchgang durch die Schleife. Die beiden `foreach`-Anweisungen wissen nicht, dass sie ihre Verarbeitung in Wirklichkeit mit dem zweiten Element des Arrays beginnen.

Um die Ergebnisse mit Tabulatoren zwischen den Elementen innerhalb der Kombinationen und Zeilenwechseln zwischen den Kombinationen auszugeben, verwenden Sie folgenden Code:

```
$array = array('Adam', 'Bret', 'Ceff', 'Dave');

foreach (pc_array_power_set($array) as $kombination) {
    print join("\t", $kombination) . "\n";
}
```

Hier sehen Sie, wie Sie nur die Kombinationen mit jeweils drei Elementen ausgeben:

```
foreach (pc_array_power_set($set) as $kombination) {
    if (3 == count($kombination)) {
        print join("\t", $kombination) . "\n";
    }
}
```

Der Durchlauf dauert bei einer großen Menge von Elementen sehr lange. Aus einer Menge mit n Elementen entstehen 2^{n+1} Mengen. Mit anderen Worten, wenn `n` um 1 erhöht wird, verdoppelt sich die Anzahl der Elemente.

Siehe auch

Rezept 4.25 mit einer Funktion, die alle Permutationen eines Arrays findet.

4.25 Alle Permutationen eines Arrays finden

Problem

Sie haben ein Array mit Elementen und möchten alle Möglichkeiten berechnen, wie diese unterschiedlich angeordnet werden können.

Lösung

Verwenden Sie einen der beiden Permutationsalgorithmen, die im Folgenden behandelt werden.

Diskussion

Die in Beispiel 4-6 dargestellte Funktion `pc_permute()` ist die PHP-Modifikation einer grundlegenden rekursiven Funktion.

Beispiel 4-6: pc_permute()

```
function pc_permute($items, $perms = array()) {
    if (empty($items)) {
        print join(' ', $perms) . "\n";
    }  else {
        for ($i = count($items) - 1; $i >= 0; --$i) {
             $newitems = $items;
             $newperms = $perms;
             list($foo) = array_splice($newitems, $i, 1);
             array_unshift($newperms, $foo);
             pc_permute($newitems, $newperms);
         }
    }
}
```

Ein Beispiel:

```
pc_permute(split(' ', 'she sells seashells'));
she sells seashells
she seashells sells
sells she seashells
sells seashells she
seashells she sells
seashells sells she
```

Diese Rekursion ist zwar elegant, aber ineffizient, denn sie erzeugt ständig neue Kopien. Außerdem kann man die Funktion nicht ohne weiteres so verändern, dass sie die Werte zurückgibt anstatt sie auszugeben, wenn man dabei nicht auf eine globale Variable zurückgreifen möchte.

Die Funktion `pc_next_permutation()` in Beispiel 4-7 ist dagegen etwas raffinierter. Sie kombiniert eine Idee von Mark-Jason Dominus aus dem *Perl Kochbuch* von Tom Christiansen und Nathan Torkington (O'Reilly Verlag) mit einem Algorithmus aus Edsger Dijkstras Klassiker *A Discipline of Programming* (Prentice-Hall).

Beispiel 4-7: pc_next_permutation()

```
function pc_next_permutation($p, $size) {
    // Durch das Array gehen und nach der Stelle suchen,
    // an der wir kleiner als das nächste Element sind.
    for ($i = $size - 1; $p[$i] >= $p[$i+1]; --$i) { }
```

Beispiel 4-7: pc_next_permutation() (Fortsetzung)

```
    // Wenn dies nicht geschieht, ist die Permutation fertig.
    // Das Array ist umgekehrt: (1, 2, 3, 4) => (4, 3, 2, 1)
    if ($i == -1) { return false; }

    // Durch das Array gehen und nach einer Zahl suchen,
    // die größer als die zuvor gefundene ist.
    for ($j = $size; $p[$j] <= $p[$i]; --$j) { }

    // Die Elemente vertauschen.
    $tmp = $p[$i]; $p[$i] = $p[$j]; $p[$j] = $tmp;

    // Nun die Elemente dazwischen umkehren, indem die Enden vertauscht werden.
    for (++$i, $j = $size; $i < $j; ++$i, --$j) {
         $tmp = $p[$i]; $p[$i] = $p[$j]; $p[$j] = $tmp;
    }

    return $p;
}

$set = split(' ', 'she sells seashells'); // wie array('she', 'sells', 'seashells')
$size = count($set) - 1;
$perm = range(0, $size);
$j = 0;

do {
     foreach ($perm as $i) { $perms[$j][] = $set[$i]; }
} while ($perm = pc_next_permutation($perm, $size) and ++$j);

foreach ($perms as $p) {
    print join(' ', $p) . "\n";
}
```

Dominus' Idee besteht darin, Permutationen von Integer-Zahlen zu erzeugen, anstatt das Array selbst zu manipulieren. Dann werden die umpositionierten Integer-Werte zurück auf die Elemente des Arrays abgebildet, um die eigentlichen Permutationen zu bilden – eine clevere Idee.

Allerdings hat auch diese Technik noch einige Nachteile. Der wichtigste für Sie als PHP-Programmierer ist, dass häufig Pop-, Push und Splice-Operationen mit Arrays durchführt werden; dies ist sehr Perl-orientiert. Außerdem ist zur Berechnung der Integer-Permutationen eine Reihe von Schritten erforderlich, um eine neue Permutation zu erhalten, da die vorherigen Permutationen nicht erhalten bleiben. Jedes Mal beginnt also die ursprüngliche Permutation von neuem. Aber warum die Arbeit erneut ausführen, wenn dies nicht nötig ist?

Dijkstras Algorithmus löst das Problem, indem er eine Permutation einer Reihe von Integer-Zahlen nimmt und die nächstgrößere Permutation zurückgibt. Aufgrund dieser Annahme ist der Code optimiert. Wenn Sie mit dem kleinsten Muster beginnen (das sind einfach die Integer-Zahlen in aufsteigender Reihenfolge) und sich hocharbeiten, können

Sie alle Permutationen eine nach der anderen durchlaufen, indem Sie die vorherige Permutation zurück in die Funktion geben, um die nächste zu bekommen. Tauschvorgänge sind dabei kaum noch nötig, nicht einmal in der letzten Schleife, in der Sie das Ende austauschen.

Dabei gibt es noch eine positive Nebenwirkung. Das Rezept von Dominus benötigt die Gesamtzahl der Permutationen für ein gegebenes Muster. Da dies die Fakultät der Anzahl der Elemente in der Menge ist, ist es eine potenziell aufwendige Berechnung, auch wenn man Memoisierung verwendet. Es ist schneller, wenn Sie diese Zahl nicht berechnen, sondern in `pc_next_permutation()` den Wert `false` zurückgeben, sobald `$i == -1` ist. Wenn dies geschieht, kommen Sie aus dem Array heraus, und die Permutationen für die Phrase sind erschöpft.

Zwei abschließende Anmerkungen zur Implementierung: Da die Größe der Menge unveränderlich ist, ermitteln Sie diese einmal mit `count()` und geben sie an `pc_next_permutation()` weiter; dies geht schneller als der wiederholte Aufruf von `count()` innerhalb der Funktion. Außerdem brauchen Sie innerhalb der ersten beiden `for`-Schleifen nicht auf Gleichheit zu prüfen, da die Menge konstruktionsbedingt garantiert nur eindeutige Elemente hat, d.h., es gibt genau ein Exemplar von jeder Integer-Zahl. Diese Prüfung sollten Sie aber einbauen, falls Sie dieses Rezept auf andere numerische Mengen anwenden möchten, bei denen Duplikate vorkommen könnten.

Siehe auch

Rezept 4.24 mit einer Funktion zur Ermittlung der Potenzmenge eines Arrays; Rezept 4.20 im *Perl Kochbuch* (O'Reilly Verlag); Kapitel 3 in *A Discipline of Programming* (Prentice-Hall).

4.26 Eine Funktion auf jedes Element eines Arrays anwenden

Problem

Sie möchten eine Funktion oder Methode auf jedes Element in einem Array anwenden. Auf diese Weise können Sie die Eingabedaten der gesamten Menge auf einmal transformieren.

Lösung

Nutzen Sie `array_walk( )`:

```
function escape_data(&$value, $key) {
    $value = htmlentities($value, ENT_QUOTES);
}
```

```
$names = array('firstname' => "Baba",
               'lastname'  => "O'Riley");

array_walk($names, 'escape_data');
foreach ($names as $name) {
    print "$name";
}
Baba
O&#039;Riley
```

Und für geschachtelte Daten `array_walk_recursive( )`:

```
function escape_data(&$value, $key) {
    $value = htmlentities($value, ENT_QUOTES);
}
$names = array('firstnames' => array("Baba", "Bill"),
               'lastnames'  => array("O'Riley", "O'Reilly"));
array_walk_recursive($names, 'escape_data');

foreach ($names as $nametypes) {
    foreach ($nametypes as $name) {
        print "$name";
    }
}
Baba
Bill
O&#039;Riley
O&#039;Reilly
```

Diskussion

Häufig ist es nützlich, alle Elemente in einem Array zu durchlaufen. Das kann man machen, indem man die Daten mit `foreach` durchläuft. Als gute Alternative dazu bietet sich auch die Funktion `array_walk( )` an.

Diese Funktion erwartet eine Array-Referenz und den Namen einer Callback-Funktion, mit der die Elemente im Array verarbeitet werden. Die Callback-Funktion muss zwei Parameter akzeptieren, einen Wert und einen Schlüssel. Sie kann auch einen optionalen dritten Parameter akzeptieren, über den zusätzliche Daten übergeben werden können, die Sie der Callback-Funktion zur Verfügung stellen wollen.

Hier sehen Sie ein Beispiel, das sichert, dass alle Daten im Array `$names` korrekt HTML-codiert werden. Die Callback-Funktion `escape_data( )` nimmt die Array-Werte, übergibt sie an `htmlentities( )`, um die wichtigsten HTML-Entities zu codieren, und weist das Ergebnis `$value` zu:

```
function escape_data(&$value, $key) {
    $value = htmlentities($value, ENT_QUOTES);
}
$names = array('firstname' => "Baba",
               'lastname'  => "O'Riley");
```

```
array_walk($names, 'escape_data');
foreach ($names as $name) {
    print "$name";
}
Baba
O&#039;Riley
```

Da `array_walk` vor Ort arbeitet, statt eine veränderte Kopie des Arrays zurückzuliefern, müssen Sie die Werte per Referenz übergeben, wenn Sie diese verändern wollen. Dazu müssen Sie wie in diesem Beispiel dem Parameternamen ein `&` voranstellen. Erforderlich ist das aber nur, wenn Sie das Array verändern wollen.

Bei einer Gruppe geschachtelter Arrays nutzen Sie stattdessen die Funktion `array_walk_recursive( )`:

```
function escape_data(&$value, $key) {
    $value = htmlentities($value, ENT_QUOTES);
}
$names = array('firstnames' => array("Baba", "Bill"),
               'lastnames'  => array("O'Riley", "O'Reilly"));
array_walk_recursive($names, 'escape_data');

foreach ($names as $nametypes) {
    foreach ($nametypes as $name) {
        print "$name\n";
    }
}
Baba
Bill
O&#039;Riley
O&#039;Reilly
```

Die Funktion `array_walk_recursive( )` übergibt nur die Elemente, die keine Arrays sind. Sie müssen Ihre Callback-Funktion also nicht überarbeiten, wenn Sie sie zuvor bereits mit `array_walk( )` verwendet haben.

Siehe auch

Die Dokumentationen zu `array_walk( )` unter *http://www.php.net/array-walk*, zu `array_walk_recursive( )` unter *http://www.php.net/array_walk_recursive* und zu `htmlentities( )` unter *http://www.php.net/htmlentities*.

4.27 Echte Objekte als Schlüssel von Arrays verwenden

Problem

Sie möchten echte Objekte als Schlüssel von Arrays verwenden, was PHP nicht erlaubt. Ein Workaround, wie die Serialisierung der Objekte in eine Zeichenkette, die dann als eindeutiger Identifier genutzt wird, kommt für Sie nicht infrage.

Lösung

Die Standard PHP Library (SPL) stellt ab PHP 5.3 die Klasse `SplObjectStorage` zur Verfügung. Diese können Sie als Map-Datenstruktur nutzen und Objekte als Schlüssel verwenden:

```
class Foo {}
class Bar {}

$s1 = new SplObjectStorage();
$o1 = new StdClass;
$o2 = new Foo;
$o3 = new Bar;
$s1[$o1] = 'Wert $o1';
var_dump($s1->contains($o1));
var_dump($s1->contains($o2));
var_dump($s1->contains($o3));
bool(true)
bool(false)
bool(false)

$s2 = new SplObjectStorage();
$s2[$o2] = 'Wert $o2';
$s2[$o3] = 'Wert $o3';
$s1->addAll($s2);
$s1->detach($o1);
var_dump($s1->contains($o1));
var_dump($s1->contains($o2));
var_dump($s1->contains($o3));
bool(false)
bool(true)
bool(true)
```

Das Codebeispiel zeigt, wie Sie die Klasse `SplObjectStorage` instantiieren und Objekte als Schlüssel ablegen. Die Methoden `attach()` und `detach()` ermöglichen das Hinzufügen bzw. Entfernen von Einträgen. Mit der Methode `addAll()` können die Einträge eines weiteren `SplObjectStorage` hinzugefügt werden, und `contains()` prüft, ob ein Objekt bereits als Schlüssel verwendet wird.

Diskussion

Bei normalen PHP-Arrays können Objekte nicht als Schlüssel verwendet werden. Oft behilft man sich mit einem Workaround, indem man ein Objekt erst in einen String serialisiert und es dann als Schlüssel verwendet.

```
$a = array();
$a[$o1] = "Wert";
print_r($a);
$a[md5(serialize($o1))] = "Wert";
print_r($a);
Warning: Illegal offset type in [...] on line [...]
Array
(
```

```
)
Array
(
    [f7827bf44040a444ac855cd67adfb502] => Wert
)
```

Der verwendete Schlüssel berücksichtigt nun zwar den inneren Zustand des Objekts, allerdings nicht die Objektidentität. Besser funktioniert dann schon die Funktion `spl_object_hash()`, die ab PHP 5.2.0 verfügbar ist:

```
$o1 = new StdClass;
$o2 = new StdClass;
var_dump($o1 == $o2);
var_dump($o1 === $o2);
var_dump($h1 = spl_object_hash($o1));
var_dump($h2 = spl_object_hash($o2));
var_dump($h1 == $h2);
bool(true)
bool(false)
string(32) "000000006dfa4081000000004d3a55a4"
string(32) "000000006dfa4085000000004d3a55a4"
bool(false)
```

Eine weitere interessante Möglichkeit der Anwendung von `SplObjectStorage` besteht in der Nutzung als Datenstruktur *Menge*, oft auch *Set* genannt. Man kann eine ungeordnete Sammlung von Objekten vorhalten, wobei jedes Objekt nur einmal im Set enthalten ist – unabhängig davon, wie oft es hinzugefügt wurde:

```
$set = new SplObjectStorage;
$o1 = new StdClass;
$o2 = new StdClass;
$o3 = new StdClass;
$set->attach($o1);
$set->attach($o2);
$set->attach($o2);
$set->attach($o3);

foreach ($set as $o) {
    var_dump($o);
}
object(stdClass)#8 (0) {
}
object(stdClass)#6 (0) {
}
object(stdClass)#2 (0) {
}
```

Siehe auch

Weitere Informationen zu `SplObjectStorage` finden Sie im PHP-Manual unter *http://php.net/manual/function.spl-object-hash.php*. Informationen über weitere SPL-Datenstrukturen liefert Rezept 8.8.

4.28 Ein Objekt wie ein Array auftreten lassen

Problem

Sie haben ein Objekt, möchten es aber wie ein Array behandeln können. Das ermöglicht Ihnen, die Vorteile eines objektorientierten Entwurfs und die Vertrautheit mit der einfachen Array-Syntax zu kombinieren.

Lösung

Implementieren Sie die SPL-Schnittstelle `ArrayAccess`:

```
class FakeArray implements ArrayAccess {
    private $elements;

    public function __construct() {
        $this->elements = array();
    }

    public function offsetExists($offset) {
        return isset($this->elements[$offset]);
    }
    public function offsetGet($offset) {
        return $this->elements[$offset];
    }

    public function offsetSet($offset, $value) {
        return $this->elements[$offset] = $value;
    }

    public function offsetUnset($offset) {
        unset($this->elements[$offset]);
    }
}

$array = new FakeArray;

// What's Opera, Doc?[2]
$array['animal'] = 'Ninchen';

// Still, ich jage Ninchen.
if (isset($array['animal']) &&
    // Ninchenspuren!!!
    $array['animal'] == 'Ninchen') {

    // Ninchen schnappen, Ninchen killen, Ninchen grillen.
    unset($array['animal']);
```

2 Frei nach Bugs Bunnys berühmtem Abenteuer »What's Opera, Doc?«, siehe Wikipedia-Eintrag *http://en.wikipedia.org/wiki/What%27s_Opera,_Doc%3F*; siehe auch entsprechende Videos auf YouTube.

```
    // Weg, Schmeck, Lecker ...
}

// Was nur habe ich getan. Das arme Ninchen ist nicht mehr ...
// Armes kleines Häschen, kleines armes Häschen ...
if (!isset($array['animal'])) {
    print "Was sonst haben Sie in einer Oper erwartet. Ein Happy End etwa?\n";
}
Was sonst haben Sie in einer Oper erwartet. Ein Happy End etwa?
```

Diskussion

Die Schnittstelle `ArrayAccess` ermöglicht Ihnen, die Daten in einem Objekt über die gleichen Syntaxformen zu manipulieren wie bei Arrays. So können Sie die Vorteile eines objektorientierten Entwurfs wie den Einsatz von Klassenhierarchien oder die Implementierung zusätzlicher Methoden nutzen und Ihren Anwendern dennoch ermöglichen, mit dem Objekt über die vertraute Array-Syntax zu arbeiten. Alternativ wird Ihnen ermöglicht, ein »Array« zu erstellen, dessen Daten an einem externen Ort wie einem geteilten Speicher oder einer Datenbank gespeichert werden.

Die Implementierung von `ArrayAccess` erfordert vier Methoden: `offsetExists()`, die anzeigt, ob ein Element definiert ist, `offsetGet()`, die den Wert eines Elements liefert, `offsetSet()`, die ein Element auf einen neuen Wert setzt, und `offsetUnset()`, die ein Element und seinen Wert entfernt.

In folgendem Beispiel werden die Daten lokal in Objekteigenschaften gespeichert:

```
class FakeArray implements ArrayAccess {
    private $elements;

    public function __construct() {
        $this->elements = array();
    }

    public function offsetExists($offset) {
        return isset($this->elements[$offset]);
    }
    public function offsetGet($offset) {
        return $this->elements[$offset];
    }

    public function offsetSet($offset, $value) {
        return $this->elements[$offset] = $value;
    }

    public function offsetUnset($offset) {
        unset($this->elements[$offset]);
    }
}
```

Der Objektkonstruktor initialisiert die Eigenschaft `$elements` mit einem neuen Array. Dieses dient als Ort, an dem die Schlüssel und Werte Ihres Arrays gespeichert werden. Die

Eigenschaft ist als `private` deklariert, damit auf ihre Daten nur über die Zugriffsmethoden zugegriffen werden kann, die die Schnittstelle definiert.

Die anschließenden vier Methoden implementieren alles, was man zur Manipulation eines Arrays benötigt. Da `offsetExists( )` prüft, ob ein Array-Element gesetzt ist, liefert die Methode den Wert von `isset($this->elements[$offset])`.

Die Methoden `offsetGet( )` und `offsetSet( )` interagieren mit der Eigenschaft `$elements` über die Syntaxformen, die man für den gewöhnlichen Array-Zugriff einsetzt.

`offsetUnset( )` schließlich ruft einfach `unset( )` auf dem Element auf. Anders als die drei vorangehenden Methoden liefert sie den Rückgabewert der ausgeführten Operation nicht zurück. Der Grund dafür ist, dass `unset( )` eine Anweisung und keine Funktion ist und daher keinen Wert zurückliefert.

Jetzt können Sie eine `FakeArray`-Instanz erzeugen und manipulieren wie ein gewöhnliches Array:

```
$array = new FakeArray;
// What's Opera, Doc??
$array['animal'] = 'Ninchen';
// Still, ich jage Ninchen.
if (isset($array['animal']) &&
    // Ninchenspuren!!!
    $array['animal'] == 'Ninchen') {

    // Ninchen schnappen, Ninchen killen, Ninchen grillen.
    unset($array['animal']);
    // Weg, Schmeck, Lecker ...
}
// Was nur habe ich getan. Das arme Ninchen ist nicht mehr ...
// Armes kleines Häschen, kleines armes Häschen ...
if (!isset($array['animal'])) {
    print "Was sonst haben Sie in einer Oper erwartet. Ein Happy End etwa?\n";
}
Was sonst haben Sie in einer Oper erwartet. Ein Happy End etwa?
```

Jede der Operationen ruft eine der Methoden auf: Wird `$array['animal']` ein Wert zugewiesen, wird `offsetSet( )` aufgerufen, `isset($array['animal'])` ruft `offsetExists( )` auf, `offsetGet( )` tritt auf, wenn Sie den Vergleich `$array['animal'] == 'ninchen'` durchführen, und `offsetUnset( )` wird für `unset($array['animal'])` aufgerufen.

Wie Sie sehen können, ist das süße Ninchen nach Durchführung dieser Operationen nicht mehr.

Siehe auch

Weitere Informationen zu Objekten bietet Kapitel 7; `ArrayAccess`-Referenzseite unter *http://www.php.net/~helly/php/ext/spl/interfaceArrayAccess.html*; Wikipedia-Eintrag zu »What's Opera, Doc?« unter *http://en.wikipedia.org/wiki/What%27s_Opera%2C_Doc*.

4.29 Programm: Ein Array horizontal angeordnet in einer HTML-Tabelle ausgeben

Wenn Sie ein Array in eine Tabelle mit horizontal angeordneten Spalten konvertieren, platzieren Sie eine festgelegte Anzahl von Elementen in eine Zeile. Der erste Satz kommt in die Tabellenzeile am Anfang, der zweite Satz in die nächste Zeile und so weiter. Schließlich erreichen Sie die letzte Zeile, die Sie möglicherweise optional mit leeren Zellen auffüllen müssen.

Bei der in Beispiel 4-8 dargestellte Funktion `pc_grid_horizontal()` können Sie ein Array und die Anzahl der Spalten angeben. Sie geht davon aus, dass die Tabellenbreite 100% beträgt; dies können Sie aber ändern, indem Sie die Variable `$table_width` modifizieren.

Beispiel 4-8: pc_grid_horizontal()

```
function pc_grid_horizontal($array, $size) {

    // Prozentwerte für die <td>-Breite berechnen.
    $table_width = 100;
    $width = intval($table_width / $size);

    // Definieren, wie die Tags <tr> und <td> erscheinen.
    // Bei sprintf() müssen Sie %% angeben, um das % als Literal zu erhalten.
    $tr = '<tr align="center">';
    $td = "<td width=\"$width%%\">%s</td>";

    // Tabelle beginnen.
    $grid = "<table width=\"$table_width%\">$tr";

    // Einträge in Schleife durchlaufen und in Spalten der Breite $size ausgeben.
    // Mit $i wird berechnet, wann eine neue Tabellenzeile beginnt.
    $i = 0;
    foreach ($array as $e) {
        $grid .= sprintf($td, $e);
        $i++;

        // Ende einer Zeile.
        // Zeile abschließen und neue beginnen.
        if (!($i % $size)) {
            $grid .= "</tr>$tr";
        }
    }

    // Übrige Zellen auffüllen, wenn leer.
    while ($i % $size) {
        $grid .= sprintf($td, ' ');
        $i++;
    }

    // Wenn erforderlich, </tr> hinzufügen.
```

Beispiel 4-8: pc_grid_horizontal() (Fortsetzung)

```
    $end_tr_len = strlen($tr) * -1;
    if (substr($grid, $end_tr_len) != $tr) {
        $grid .= '</tr>';
    } else {
        $grid = substr($grid, 0, $end_tr_len);
    }

    // Tabelle abschließen.
    $grid .= '</table>';

    return $grid;
}
```

Die Funktion beginnt damit, die Breite jedes `<td>` als Prozentwert der gesamten Tabellenbreite zu berechnen. Abhängig von der Anzahl der Spalten und der Gesamtgröße, kann es sein, dass die Summe der `<td>`-Breiten nicht mit der Breite der `<table>` übereinstimmt; dies sollte aber die angezeigte HTML-Seite nicht in auffälliger Weise beeinträchtigen. Danach werden mithilfe der Formatierung im `printf`-Stil die `<td>`- und `<tr>`-Tags ausgegeben. Um das für die Angabe der `<td>`-Breite in Prozent benötigte Literal `%` zu erhalten, verwenden Sie ein doppeltes `%%`.

Der Kern der Funktion besteht in der `foreach`-Schleife durch das Array, in der Sie die einzelnen `<td>` an das `$grid` hängen. Wenn das Ende einer Zeile erreicht ist, was der Fall ist, wenn die Gesamtzahl der verarbeiteten Elemente ein Vielfaches der Anzahl der Elemente in einer Zeile ist, schließen Sie die Tabellenzeile und öffnen eine neue.

Wenn alle Elemente hinzugefügt sind, müssen Sie die letzte Zeile mit leeren `<td>`-Elementen auffüllen. Schreiben Sie ein ` ` in die Datenzellen, anstatt sie leer zu lassen, damit die Tabelle im Browser korrekt angezeigt wird. Nun stellen Sie noch sicher, dass sich am Ende der Tabelle kein überflüssiges `<tr>` befindet, was vorkommen kann, wenn die Anzahl der Elemente genau ein Vielfaches der Breite ist (das heißt, wenn Sie keine Zellen zum Auffüllen hinzufügen mussten). Am Ende können Sie die Tabelle schließen.

Als Beispiel werden hier die Namen der 50 US-Staaten in einer sechsspaltigen Tabelle ausgegeben:

```
// Verbindung mit der Datenbank herstellen.
$dsn = 'mysql://user:password@localhost/table';
$dbh = DB::connect($dsn);
if (DB::isError($dbh)) { die ($dbh->getMessage()); }

// Datenbank nach den 50 Staaten abfragen.
$sql = "SELECT state FROM states";
$sth = $dbh->query($sql);

// Daten aus der Datenbank in ein Array laden.
while ($row = $sth->fetchRow(DB_FETCHMODE_ASSOC)) {
  $states[] = $row['state'];
}
```

```
// HTML-Tabelle generieren...
$grid = pc_grid_horizontal($states, 6);

// ... und  ausgeben.
print $grid;
```

Wenn die Tabelle in einem Browser dargestellt wird, sieht sie wie in Abbildung 4-1 gezeigt aus.

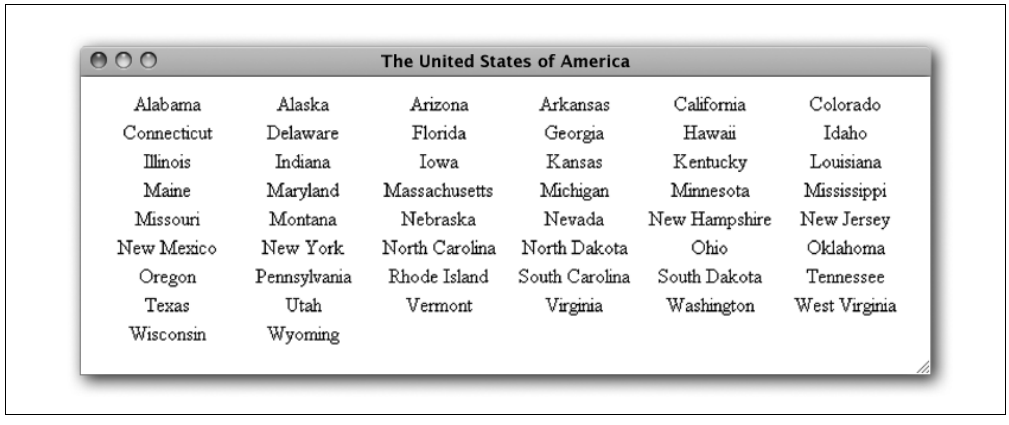


Abbildung 4-1: Die Vereinigten Staaten von Amerika

Da sich 50 nicht ohne Rest durch 6 teilen lässt, ist die letzte Zeile mit vier zusätzlichen Zellen aufgefüllt.

KAPITEL 5

Variablen

5.0 Einführung

Zusammen mit der Bedingungslogik bilden Variablen den Kern dessen, was Computer-Programme mächtig und flexibel macht. Wenn Sie sich eine Variable als einen Behälter mit einem Namen vorstellen, der einen Wert enthält, gibt es bei PHP neben den gewöhnlichen einfachen Behältern auch solche, die den Namen eines anderen Behälters enthalten, Behälter, die Zahlen oder Strings enthalten, Behälter mit Arrays aus anderen Behältern sowie Behälter, die voller Objekte sind. Diese Liste können Sie mit ziemlich jeder weiteren denkbaren Variante dieser Analogie fortsetzen.

Eine Variable ist entweder gesetzt oder nicht gesetzt. Eine Variable ist gesetzt, wenn ihr irgendein Wert zugewiesen wurde; dabei kann sie `true` oder `false`, leer oder nicht leer sein. Die Funktion `isset()` gibt `true` zurück, wenn man ihr eine gesetzte Variable übergibt. Man kann eine gesetzte Variable nur in eine nicht gesetzte verwandeln, wenn man die Funktion `unset()` mit dieser Variablen aufruft. An `unset()` können sowohl Skalare als auch Arrays und Objekte übergeben werden. Sie können `unset()` auch mehrere Variablen auf einmal übergeben, um sie alle auf einmal zurückzusetzen:

```
unset($gemuese);
unset($gemuese[12]);
unset($sonne, $mond, $sterne);
```

Wenn eine Variable im Query-String einer URL enthalten ist, ist sie ebenfalls gesetzt; dies gilt auch, wenn ihr kein Wert zugewiesen wurde. Daher setzt die folgende Anfrage die Variable `$_GET['affen']` auf `12` und `$_GET['schimpansen']` auf einen Leer-String:

```
http://www.example.com/set.php?schimpansen=&affen=12
```

Alle nicht gesetzten Variablen sind zugleich auch leer. Gesetzte Variablen können leer oder nicht leer sein. Leere Variablen haben einen Wert, der, wenn er als Boolescher Wert ausgewertet wird, `false` ergibt: die Integer-Zahl 0, die Double-Zahl 0.0, der leere String (aber auch der gesetzte String »0«), der Boolesche Wert `false`, ein Array ohne Elemente, ein Objekt ohne Variablen und Methoden (in PHP-Versionen vor PHP 5) sowie `NULL`. Alles andere ist nicht-leer. Dazu gehören der String »00« und der String » «, der nur ein Leerzeichen enthält.

Variablen ergeben entweder `true` oder `false`. Die oben aufgeführten Werte, die als Boolesche Werte `false` ergeben, stellen die komplette Liste all dessen dar, was in PHP `false` ist. Jeder andere Wert ist `true`. Der Unterschied zwischen leer und falsch besteht darin, dass nur Variablen leer sein können. Konstanten und Rückgabewerte von Funktionen können zwar `false` sein, aber nicht leer. Der folgende Ausdruck ist zum Beispiel gültig, weil `$vorname` eine Variable ist:

```
if (empty($vorname)) { /* ... */ }
```

Diese beiden Ausdrücke dagegen führen zu einem Parser-Fehler bzw. zu einem Laufzeitfehler, da weder `0` (eine Konstante) noch der Rückgabewert von `get_vorname()` leer sein können:

```
if (empty(0)) { /* ... */ }
if (empty(get_vorname())) { /* ... */ }
```

5.1 Die Verwechslung von == und = vermeiden

Problem

Sie möchten beim Vergleich (Vergleichsoperator: `==`) von Variablen mit Konstanten nicht versehentlich Werte zuweisen (Zuweisungsoperator: `=`).

Lösung

Schreiben Sie:

```
if (12 == $zwerge) { /* ... */ }
```

anstelle von:

```
if ($zwerge == 12) { /* ... */ }
```

Die Konstante auf der linken Seite löst einen Parser-Fehler aus, wenn sie mit dem Zuweisungsoperator verwendet wird. Mit anderen Worten: PHP beschwert sich, wenn Sie schreiben:

```
if (12 = $zwerge) { /* ... */ }
```

führt aber:

```
if ($zwerge = 12) { /* ... */ }
```

stillschweigend aus; dabei wird der Variablen `$zwerge` der Wert `12` zugewiesen und anschließend der Code im Block ausgeführt (`$zwerge = 12` ergibt `12`, und das ist `true`.)

Diskussion

Wenn Sie die Konstante auf der linken Seite des Vergleichs eintragen, erfolgt der Vergleich mit dem Typ der Konstanten. Das kann zu Problemen führen, wenn Sie eine Inte-

ger-Zahl mit einer Variablen vergleichen, die eine Integer-Zahl oder ein String sein kann. `0 == $zwerge` ist nicht nur `true`, wenn `$zwerge` den Wert `0` hat, sondern auch, wenn `$zwerge` den Wert `schlafen` hat. Da sich auf der linken Seite des Vergleichs ein Integer-Wert (`0`) befindet, konvertiert PHP vor dem Vergleich den Wert auf der rechten Seite (also den String `schlafen`) zu einem Integer (`0`). Um dies zu vermeiden, verwenden Sie stattdessen besser den Identitätsoperator: `0 === $zwerge`.

Siehe auch

Die Dokumentation zu `=` unter *http://www.php.net/manual/en/language.operators.assignment.php* sowie zu `==` und `===` unter *http://www.php.net/manual/en/language.operators.comparison.php*.

5.2 Einen Vorgabewert festlegen

Problem

Sie möchten einer Variablen einen Vorgabewert zuweisen, sofern diese nicht schon einen Wert hat. Es kommt häufig vor, dass Sie einen fest programmierten Wert für eine Variable verwenden, der durch eine Formulareingabe oder eine Umgebungsvariable überschrieben werden kann.

Lösung

Verwenden Sie `isset()`, um einer Variablen einen Vorgabewert zuzuweisen, die möglicherweise schon einen Wert hat:

```
if (! isset($autos)) { $autos = $vorgabe_autos; }
```

Verwenden Sie den ternären Operator (`a ? b : c`), um einer neuen Variablen einen Wert (gegebenenfalls die Vorgabe) zu geben:

```
$autos = isset($_REQUEST['autos']) ? $_REQUEST['autos'] : $vorgabe_autos;
```

Diskussion

Sie müssen vor der Zuweisung von Vorgabewerten `isset()` aufrufen, da andernfalls ein nicht vorgegebener Wert nicht `0` oder etwas anderes sein kann, dessen Auswertung `false` ergeben würde. Sehen Sie sich die folgende Zuweisung an:

```
$autos = $_REQUEST['autos'] ? $_REQUEST['autos'] : $vorgabe_autos;
```

Wenn `$_REQUEST['autos']` den Wert `0` hat, wird `$autos` auch dann auf `$vorgabe_autos` gesetzt, wenn `0` ein gültiger Wert für `$autos` ist.

Eine alternative Syntax, um Arrays auf gesetzte Inhalte zu überprüfen, kann mit der Funktion array_key_exists() aufgebaut werden:

```
$autos = array_key_exists('autos',$_REQUEST) ? $_REQUEST['autos'] : $vorgabe_autos;
```

Der einzige Unterschied zwischen isset() und array_key_exists() besteht darin, dass, wenn ein Schlüssel im Array existiert, aber dessen Wert null ist, die Funktion array_key_exists()trotzdem true zurückgibt, während isset() mit false antwortet.

Mithilfe eines Arrays von Vorgaben können Sie auf einfache Weise mehrfache Vorgabewerte setzen. Die Schlüssel im Vorgabe-Array sind Variablennamen, und die Werte im Array sind die Vorgabewerte für die einzelnen Variablen:

```
$defaults = array('kaiser'  => array('Rudolf II','Caligula'),
                  'gemuese' => 'Sellerie',
                  'felder'  => 15);

foreach ($defaults as $k => $v) {
    if (! isset($GLOBALS[$k])) { $GLOBALS[$k] = $v; }
}
```

Da die Variablen im globalen Namensraum gesetzt werden, kann man mit dem obigen Code keine funktionsinternen Vorgaben setzen. Um dies zu tun, benötigen Sie variable Variablen:

```
foreach ($defaults as $k => $v) {
    if (! isset($$k)) { $$k = $v; }
}
```

Siehe auch

Die Dokumentation zu isset() unter *http://www.php.net/isset*; variable Variablen werden in Rezept 5.4 behandelt sowie unter *http://www.php.net/manual/en/language.variables.variable.php*.

5.3 Werte ohne Hilfe von temporären Variablen austauschen

Problem

Sie möchten die Werte von zwei Variablen vertauschen, ohne dabei zusätzliche Variablen als Zwischenspeicher zu benutzen.

Lösung

Zum Vertauschen von $a und $b verwenden Sie:

```
list($a,$b) = array($b,$a);
```

Diskussion

Mithilfe des PHP-Sprachkonstrukts `list()` können Sie die Werte eines Arrays einzelnen Variablen zuweisen. Das Gegenstück auf der rechten Seite des Ausdrucks, `array()`, ermöglicht Ihnen, Arrays aus einzelnen Werten zu bilden. Indem Sie das Array, das von `array()` zurückgegeben wird, den Variablen in `list()` zuweisen, jonglieren Sie mit der Anordnung dieser Werte. Dies funktioniert auch mit mehr als zwei Werten:

```
list($gestern,$heute,$morgen) = array($heute,$morgen,$gestern);
```

Diese Methode ist nicht schneller als die Verwendung temporärer Variablen, Sie sollten sie der Klarheit, nicht der Geschwindigkeit wegen einsetzen.

Siehe auch

Die Dokumentationen zu `list()` unter *http://www.php.net/list* und `array()` unter *http://www.php.net/array*.

5.4 Einen dynamischen Variablennamen erzeugen

Problem

Sie möchten einen Variablennamen dynamisch zusammensetzen. Beispielsweise möchten Sie Variablennamen verwenden, die mit den Feldnamen aus einer Datenbankabfrage übereinstimmen.

Lösung

Verwenden Sie die PHP-Syntax der variablen Variablen, indem Sie ein `$` vor die Variable stellen, deren Wert der gewünschte Variablenname ist:

```
$tier = 'schildkroeten';
$schildkroeten = 103;
print $$tier;
103
```

Diskussion

Das obige Beispiel gibt `103` aus. Weil `$tier = 'schildkroeten'` ist, ist `$$tier` dasselbe wie `$schildkroeten`, und dies ist gleich 103.

Mithilfe geschweifter Klammern können Sie kompliziertere Ausdrücke bilden, die Variablennamen bezeichnen:

```
$stooges      = array('Moe','Larry','Curly');
$stooge_moe   = 'Moses Horwitz';
$stooge_larry = 'Louis Feinberg';
$stooge_curly = 'Jerome Horwitz';

foreach ($stooges as $s) {
  print "$s hieß in Wirklichkeit ${'stooge_'.strtolower($s)}.\n";
}
Moe hieß in Wirklichkeit Moses Horwitz.
Larry hieß in Wirklichkeit Louis Feinberg.
Curly hieß in Wirklichkeit Jerome Horwitz.
```

PHP wertet den Ausdruck zwischen den geschweiften Klammern aus und verwendet ihn als Variablennamen. Der Ausdruck kann auch Funktionsaufrufe wie etwa `strtolower()` enthalten.

Variable Variablen sind ebenfalls nützlich, um eine Reihe ähnlich benannter Variablen zu durchlaufen. Angenommen, Sie fragen eine Datenbanktabelle mit den Feldern `titel_1`, `titel_2` usw. ab. Wenn Sie prüfen möchten, ob ein Titel mit einem dieser Werte übereinstimmt, durchlaufen Sie sie am einfachsten in einer Schleife, wie im folgenden Beispiel zu sehen:

```
for ($i = 1; $i <= $n; $i++) {
    $t = "titel_$i";
    if ($titel == $$t) { /* Übereinstimmung */ }
}
```

Natürlich wäre es geradliniger, diese Werte in einem Array zu speichern, aber bei der Pflege alter Programme, in denen diese Technik angewendet wird (und bei denen Sie dies nicht ändern können), sind variable Variablen hilfreich.

Die Syntax mit den geschweiften Klammern wird auch benötigt, um Mehrdeutigkeiten bei Array-Elementen aufzulösen. Die variable Variable `$$esel[12]` kann zwei Bedeutungen haben. Die erste ist: »Nimm das 12. Element des Arrays `$esel` und verwende es als Variablennamen.« Dies schreiben Sie in der Form `${$esel[12]}`. Die zweite Bedeutung ist: »Verwende den Inhalt des Skalars `$esel` als Array-Namen und siehe im 12. Element dieses Arrays nach.« Dies formulieren Sie als `${$esel}[12]`.

Es können hier übrigens auch drei oder mehr Dollarzeichen kombiniert werden. In der Praxis begegnet man aber kaum mehr als zwei Stufen der Indirektion.

Siehe auch

http://www.php.net/manual/en/language.variables.variable.php mit der Dokumentation zu variablen Variablen.

5.5 Statische Variablen verwenden

Problem

Sie benötigen eine lokale Variable, die ihren Inhalt zwischen zwei Aufrufen einer Funktion behält.

Lösung

Deklarieren Sie die Variable als statisch (`static`):

```
function track_times_called() {
    static $i = 0;
    $i++;
    return $i;
}
```

Diskussion

Wenn eine Variable als `static` deklariert ist, erinnert sich eine Funktion an sie. Wenn es also nachfolgende Aufrufe der Funktion gibt, können Sie auf den Wert der gespeicherten Variablen zugreifen. Die in Beispiel 5-1 dargestellte Funktion `pc_check_the_count()` verwendet statische Variablen, um die Strikes und Bälle in einem Baseball-Spiel zu verfolgen.

Beispiel 5-1: pc_check_the_count()

```
function pc_check_the_count($pitch) {
    static $strikes = 0;
    static $balls   = 0;

    switch ($pitch) {
    case 'foul':
        if (2 == $strikes) break; // bei 2 Strikes geschieht nichts
        // andernfalls wie bei einem Strike verfahren
    case 'strike':
        $strikes++;
        break;
    case 'ball':
        $balls++;
        break;
    }

    if (3 == $strikes) {
        $strikes = $balls = 0;
        return 'strike out';
    }
    if (4 == $balls) {
        $strikes = $balls = 0;
        return 'walk';
```

Beispiel 5-1: pc_check_the_count() (Fortsetzung)

```
    }
    return 'at bat';
}

$what_happened = pc_check_the_count($pitch);
```

Die `switch`-Anweisung innerhalb der Funktion `pc_check_the_count()` enthält die Logik dafür, was mit dem Batter in Abhängigkeit vom Pitch-Count geschieht. Sie können stattdessen auch die Anzahl der Strikes und Bälle zurückgeben, allerdings müssen Sie dann an mehreren Stellen im Code die Überprüfung auf Strike-Out, Walk und Verbleiben an der Plate durchführen.

Statische Variablen bewahren zwar ihre Werte zwischen Funktionsaufrufen, jedoch tun sie dies nur im Laufe der Ausführung eines Skripts. Eine `static`-Variable, auf die in einer Anfrage zugegriffen wird, behält ihren Wert nicht bis zum nächsten Abruf derselben Seite.

Siehe auch

Die Dokumentation zu `static`-Variablen unter *http://www.php.net/manual/en/language.variables.scope.php*.

5.6 Variablen in mehreren Prozessen gemeinsam verwenden

Problem

Sie suchen eine Möglichkeit für die Verwendung von Informationen in mehreren Prozessen bzw. über mehrere Skriptaufrufe hinweg, die einen schnellen Zugriff auf die gemeinsam genutzten Daten bietet.

Lösung

Speichern Sie die Daten in einem Shared-Memory-Segment und sichern Sie den exklusiven Zugriff auf den gemeinsam genutzten Speicher durch eine Semaphore ab:

```
$semaphore_id = 100;
$segment_id   = 200;
// Eine Semaphore holen, die mit dem gewünschten Shared-
// Memory-Segment verknüpft wird.
$sem = sem_get($semaphore_id,1,0600);
// Exklusiven Zugriff auf die Semaphore sicherstellen.
sem_acquire($sem) or die("Kann keine Semaphore erlangen");
// Handle für Shared-Memory-Segment holen.
$shm = shm_attach($segment_id,16384,0600);
```

```
// Einen Wert aus dem Shared-Memory-Segment lesen.
$bevoelkerung = shm_get_var($shm,'bevoelkerung');
// Den Wert verändern.
$bevoelkerung += ($geburten + $immigranten - $todesfaelle - $emigranten);
// Den Wert in das Shared-Memory-Segment zurückschreiben.
shm_put_var($shm,'bevoelkerung',$bevoelkerung);
// Handle auf Shared-Memory-Segment freigeben.
shm_detach($shm);
// Semaphore freigeben, sodass sie sich ein anderer Prozess holen kann.
sem_release($sem);
```

Diskussion

Ein Shared-Memory-Segment ist ein kleiner Bereich im RAM Ihres Rechners, auf den verschiedene Prozesse (zum Beispiel mehrere Webserver-Prozesse, die Anfragen bearbeiten) zugreifen können. Eine Semaphore stellt sicher, dass sich die verschiedenen Prozesse beim Zugriff auf das Shared-Memory-Segment nicht gegenseitig stören. Bevor ein Prozess das Segment benutzen kann, muss er sich die Kontrolle über die Semaphore verschaffen. Wenn er mit der Arbeit an dem Segment fertig ist, gibt er die Semaphore frei, sodass ein anderer Prozess sie sich holen kann.

Sie bekommen die Kontrolle über die Semaphore, indem Sie `sem_get()` aufrufen und dabei die ID der Semaphore erhalten. Das erste Argument für `sem_get()` ist ein ganzzahliger Semaphoren-Schlüssel. Der Schlüssel kann ein beliebiger Integer-Wert sein, solange alle Programme, die auf diese spezielle Semaphore zugreifen müssen, den gleichen Schlüssel verwenden. Wenn noch keine Semaphore mit dem angegebenen Schlüssel existiert, wird sie erzeugt. Mit dem zweiten Argument von `sem_get()` wird die maximale Anzahl der Prozesse festgelegt, die gleichzeitig auf die Semaphore zugreifen können (in diesem Fall `1`), und mit dem dritten Argument für `sem_get()` werden die Zugriffsrechte der Semaphore (`0600`) gesetzt. Diese Rechte funktionieren genauso wie Zugriffsrechte auf Dateien; `0600` bedeutet also, dass der Benutzer, der die Semaphore erzeugt hat, sie lesen und schreiben kann. In diesem Kontext ist unter Benutzer nicht nur der Prozess zu verstehen, der die Semaphore angelegt hat, sondern jeder Prozess, der unter der gleichen Benutzer-ID läuft. Die Zugriffsrechte `0600` sollten in den meisten Fällen angemessen sein, wenn die Webserver-Prozesse unter dem gleichen Benutzernamen laufen.

`sem_get()` gibt einen Identifikator zurück, der auf die zugrunde liegende System-Semaphore verweist. Verwenden Sie diese ID, um mit `sem_acquire()` die Kontrolle über die Semaphore zu erhalten. Diese Funktion wartet, bis die Semaphore erworben werden kann (weil sie vielleicht gerade von anderen Prozessen gesperrt ist), und gibt dann `true` zurück. Im Fehlerfall gibt sie `false` zurück. Zu den möglichen Fehlern gehören ungültige Zugriffsrechte oder der Mangel an Speicher zum Anlegen der Semaphore. Sobald Sie die Semaphore erhalten haben, können Sie aus dem Shared-Memory-Segment lesen.

Als Erstes etablieren Sie mit `shm_attach()` eine Verbindung mit dem speziellen Shared-Memory-Segment. Wie bei `sem_get()` ist das erste Argument für `shm_attach()` ein ganz-

zahliger Schlüssel. Diesmal identifiziert dieser jedoch das gewünschte Segment, nicht die Semaphore. Wenn das Segment mit dem angegebenen Schlüssel nicht existiert, wird es aufgrund der anderen Argumente angelegt. Das zweite Argument (`16384`) ist die Segmentgröße in Bytes, und das letzte Argument (`0600`) gibt die Zugriffsrechte für das Segment an. Mit `shm_attach(200,16384,0600)` wird also ein 16 KByte großes Shared-Memory-Segment erzeugt, das nur von dem Benutzer beschrieben und gelesen werden kann, der es angelegt hat. Die Funktion gibt einen Identifikator zurück, den Sie benötigen, um das Shared-Memory-Segment zu lesen oder in ihm zu schreiben.

Nachdem Sie mit dem Segment verbunden sind, entnehmen Sie ihm mit `shm_get_var($shm, 'bevoelkerung')` Variablen. Diese Funktion wirft einen Blick in das durch `$shm` identifizierte Shared-Memory-Segment und liest die Variable mit dem Namen `bevoelkerung` aus. Sie können im gemeinsamen Speicher jeden beliebigen Variablentyp ablegen. Nachdem die Variable ausgelesen worden ist, kann mit ihr wie mit jeder anderen Variablen gearbeitet werden. `shm_put_var($shm,'bevoelkerung',$bevoelkerung)` schreibt den Inhalt von `$bevoelkerung` als eine Variable namens `bevoelkerung` in das Shared-Memory-Segment zurück.

Nun sind Sie fertig mit der Shared-Memory-Anweisung. Lösen Sie sich davon mit `shm_detach()` und geben Sie die Semaphore mit `sem_release()` frei, damit andere Prozesse sie verwenden können.

Der größte Vorteil des gemeinsam genutzten Speichers besteht darin, dass er schnell ist. Da er sich aber im RAM befindet, kann er nicht allzu viele Daten aufnehmen, und er übersteht es nicht, wenn der Rechner neu gestartet wird (es sei denn, Sie unternehmen zusätzlich noch etwas, um die Informationen aus dem Shared-Memory vor dem Herunterfahren auf die Platte zu schreiben und nach dem Starten wieder in den Speicher zu laden). Außerdem ist Shared-Memory unter Windows nicht verfügbar.

Siehe auch

Rezept 10.23 enthält ein Programm, das Shared-Memory verwendet; die Dokumentation zu Shared-Memory und Semaphoren unter *http://www.php.net/sem*.

5.7 Komplexe Daten als String kapseln

Problem

Sie benötigen eine String-Repräsentation eines Arrays oder eines Objekts, um es in einer Datei oder Datenbank zu speichern. Aus diesem String soll das ursprüngliche Array oder Objekt wieder leicht zu rekonstruieren sein.

Lösung

Verwenden Sie serialize(), um Variablen und deren Inhalte in Textform zu serialisieren:

```
$speisekammer = array('Zucker' => '2 kg','Butter' => '3 Stück');
$fp = fopen('/tmp/speisekammer','w') or die ("Kann Speisekammer nicht öffnen");
fputs($fp,serialize($speisekammer));
fclose($fp);
```

Zum Wiederherstellen der Variablen verwenden Sie unserialize():

```
$neue_speisekammer = unserialize(join('',file('/tmp/speisekammer')));
```

Diskussion

Der serialisierte String, aus dem $speisekammer wiederhergestellt werden kann, sieht folgendermaßen aus:

```
a:2:{s:6:"Zucker";s:4:"2 kg";s:6:"Butter";s:7:"3 Stück";}
```

Darin stecken genug Informationen, um alle Werte des Arrays zurückzubekommen, aber der Variablenname selbst ist in der serialisierten Repräsentation nicht gespeichert.

Wenn Sie serialisierte Daten in einer URL von einer Seite zur nächsten weitergeben, rufen Sie urlencode() mit den Daten auf, um sicherzustellen, dass darin enthaltene URL-Metazeichen durch Escape-Sequenzen ersetzt werden:

```
$einkaufswagen = array('Mohn-Bagel'      => 2,
                       'Einfacher Bagel' => 1,
                       'Lachs'           => 4);
print '<a href="next.php?cart='.urlencode(serialize($einkaufswagen)).'">Weiter</a>';
```

Die Konfigurationseinstellungen magic_quotes_gpc und magic_quotes_runtime beeinflussen die an unserialize() übergebenen Daten. Wenn magic_quotes_gpc auf on gestellt ist, müssen in URLs, POST-Variablen oder Cookies übergebene Daten vor dem Deserialisieren mit stripslashes() verarbeitet werden:

```
$new_cart = unserialize(stripslashes($cart)); // wenn magic_quotes_gpc auf on steht
$new_cart = unserialize($cart);               // wenn magic_quotes_gpc auf off steht
```

Wenn magic_quotes_runtime auf on steht, müssen in einer Datei gespeicherte, serialisierte Daten mit addslashes() beim Schreiben und stripslashes() beim Lesen verarbeitet werden:

```
$fp = fopen('/tmp/cart,'w');
fputs($fp,addslashes(serialize($a)));
fclose($fp);

// wenn magic_quotes_runtime auf on steht
$new_cart = unserialize(stripslashes(join('',file('/tmp/cart'))));
// wenn magic_quotes_runtime auf off steht
$new_cart = unserialize(join('',file('/tmp/cart')));
```

Auch serialisierte Daten, die aus einer Datenbank gelesen werden, müssen mit `stripslashes()` verarbeitet werden, wenn `magic_quotes_runtime` auf on steht:

```
mysql_query(
    "INSERT INTO cart (id,data) VALUES (1,'".addslashes(serialize($cart))."')");

$r = mysql_query('SELECT data FROM cart WHERE id = 1');
$ob = mysql_fetch_object($r);
// wenn magic_quotes_runtime auf on steht
$new_cart = unserialize(stripslashes($ob->data));
// wenn magic_quotes_runtime auf off steht
$new_cart = unserialize($ob->data);
```

Wenn Sie serialisierte Daten in eine Datenbank schreiben möchten, müssen Sie immer `addslashes()` aufrufen (oder eine andere für die Datenbank geeignete Escape-Methode), um sie korrekt speichern zu können.

Siehe auch

Rezept 12.10 mit Informationen über Escape-Sequenzen in Daten für eine Datenbank. Details zur Objektserialisierung finden Sie in Rezept 7.22.

5.8 Variableninhalte als Strings ausgeben

Problem

Sie möchten sich die in einer Variablen gespeicherten Werte ansehen. Dabei kann es sich um ein kompliziertes, geschachteltes Array handeln, sodass Sie die Daten nicht einfach anzeigen oder mit einer Schleife durchlaufen können.

Lösung

Verwenden Sie `print_r()` oder `var_dump()`:

```
$array = array("name" => "frank", 12, array(3, 4));

print_r($array);
Array
(
    [name] => frank
    [0] => 12
    [1] => Array
        (
            [0] => 3
            [1] => 4
        )
)
var_dump($array);
array(3) {
```

```
  ["name"]=>
  string(5) "frank"
  [0]=>
  int(12)
  [1]=>
  array(2) {
    [0]=>
    int(3)
    [1]=>
    int(4)
  }
}
```

Diskussion

Die Ausgabe von `print_r()` ist prägnanter und besser lesbar. Dagegen gibt `var_dump()` auch die Datentypen und die Länge jeder Variablen an.

Diese Funktionen suchen sich rekursiv ihren Weg durch die Variablen, daher kann es zu einer Endlosschleife kommen, wenn Sie innerhalb einer Variablen eine Referenz haben, die zurück zu der Variablen selbst führt. Keine der beiden Funktionen gibt allerdings ohne Ende Variableninformationen aus. Sobald `print_r()` eine Variable schon einmal gesehen hat, gibt die Funktion `*RECURSION*` aus, anstatt erneut Informationen über die Variable anzuzeigen, und fährt damit fort, die übrigen noch auszugebenden Informationen zu durchlaufen. Die Funktion `var_dump()` gibt die Meldung `*RECURSION*` erst aus, wenn sie zum dritten Mal auf dieselbe Variable stößt. Betrachten Sie die Arrays `$user_1` und `$user_2`, die sich gegenseitig über ihre `friend`-Elemente referenzieren:

```
$user_1 = array('name'     => 'Max Bialystock',
                'username' => 'max');

$user_2 = array('name'     => 'Leo Bloom',
                'username' => 'leo');

// Max und Leo sind Freunde.
$user_2['friend'] = &$user_1;
$user_1['friend'] = &$user_2;

// Max und Leo haben Berufe.
$user_1['job'] = 'Schwindler';
$user_2['job'] = 'Buchhalter';
```

Die Ausgabe von `print_r($user_2)` ist:

```
Array
(
    [name] => Leo Bloom
    [username] => leo
    [friend] => Array
        (
            [name] => Max Bialystock
            [username] => max
```

```
            [friend] => Array
                (
                    [name] => Leo Bloom
                    [username] => leo
                    [friend] => Array
 *RECURSION*
                    [job] => Buchhalter
                )

            [job] => Schwindler
        )

    [job] => Buchhalter
)
```

Wenn die Funktion `print_r()` die Referenz auf `$user_1` zum zweiten Mal findet, gibt sie `*RECURSION*` aus und geht nicht in das Array hinein. Dann fährt sie auf ihrem Weg fort und gibt die übrigen Elemente von `$user_1` und `$user_2` aus

Der Aufruf von `var_dump($user_2)` sieht ein wenig anders aus, wenn er mit der Rekursion konfrontiert wird:

```
array(4){
  ["name"]=>
  string(9) "Leo Bloom"
  ["username"]=>
  string(3) "leo"
  ["friend"]=>
  &array(4) {
    ["name"]=>
    string(14) "Max Bialystock"
    ["username"]=>
    string(3) "max"
    ["friend"]=>
    &array(4) {
      ["name"]=>
      string(9) "Leo Bloom"
      ["username"]=>
      string(3) "leo"
      ["friend"]=>
      &array(4) {
        ["name"]=>
        string(14) "Max Bialystock"
        ["username"]=>
        string(3) "max"
        ["friend"]=>
        &array(4) {
          ["name"]=>
          string(9) "Leo Bloom"
          ["username"]=>
          string(3) "leo"
          ["friend"]=>
          *RECURSION*
          ["job"]=>
```

```
            string(10) "Buchhalter"
          }
          ["job"]=>
          string(10) "Schwindler"
        }
        ["job"]=>
        string(10) "Buchhalter"
      }
      ["job"]=>
      string(10) "Schwindler"
    }
    ["job"]=>
    string(10) "Buchhalter"
}
```

Die Funktion `var_dump()` beendet die Rekursion nicht vor dem dritten Erscheinen der Referenz zu `$user_1`.

Obwohl `print_r()` und `var_dump()` ihre Ergebnisse direkt anzeigen und standardmäßig nicht zurückgeben, können Sie die Daten alternativ abfangen und in Variablen ablegen. Die Funktion `print_r()` bietet seit PHP 4.3.0 einen zweiten, Boole'schen Parameter an. Wenn Sie über diesen der Funktion den Wert `true` übermitteln, gibt `print_r()` die Ausgabe zurück, statt diese am Bildschirm anzuzeigen:

```
$dump = print_r($user_2, true);
```

Leider besitzt die Funktion `var_dump()` diese Möglichkeit nicht. Trotzdem können Sie hier mithilfe des Ausgabepuffers die Ausgabe abfangen, ohne dass sie ausgegeben wird:

```
ob_start();
var_dump($user_2);
$dump = ob_get_contents();
ob_end_clean();
```

Auf diese Weise wird das Ergebnis von `var_dump($user_2)` in `$dump` geschrieben.

Siehe auch

Die Ausgabepufferung wird in Rezept 10.12 behandelt; die Dokumentationen zu `print_r()` unter *http://www.php.net/print-r* und `var_dump()` unter *http://www.php.net/var-dump*.

KAPITEL 6

Funktionen

6.0 Einführung

Funktionen unterstützen Sie dabei, gut organisierten und wiederverwendbaren Code zu schreiben. Mit ihrer Hilfe können Sie Details »heraus-abstrahieren«, sodass Ihre Programme flexibler und lesbarer werden. Ohne Funktionen ist es unmöglich, leicht wartbare Programme zu schreiben, denn Sie müssen immer wieder identische Programmteile an verschiedenen Stellen in vielen Dateien aktualisieren.

Sie können einer Funktion mehrere Argumente übergeben und einen Wert zurückbekommen.

```
// Addiere zwei Zahlen.
function add($a, $b) {
    return $a + $b;
}

$total = add(2, 2);    // 4
```

Sie deklarieren eine Funktion mithilfe des Schlüsselworts `function`, dem der Name der Funktion sowie gegebenenfalls die Parameter in Klammern folgen. Um eine Funktion aufzurufen, verwenden Sie einfach den Funktionsnamen und geben Argumentwerte für alle Parameter der Funktion an. Wenn die Funktion einen Rückgabewert liefert, können Sie das Ergebnis der Funktion einer Variablen zuweisen, wie es im obigen Beispiel zu sehen ist.

Sie müssen eine Funktion nicht vorab deklariert haben, bevor Sie sie benutzen. PHP verarbeitet die gesamte Datei, bevor die Ausführung beginnt; daher können Sie Funktionsdeklarationen und -aufrufe beliebig mischen. Allerdings können Sie in PHP keine Funktionen umdefinieren. Wenn der PHP-Prozessor auf eine Funktion stößt, deren Name identisch mit dem einer zuvor gefundenen Funktion ist, löst er einen schweren Fehler aus und verabschiedet sich.

Gelegentlich kann es vorkommen, dass das standardmäßige Vorgehen, bei dem eine festgelegte Anzahl von Argumenten übergeben und ein Wert zurückgeliefert wird, nicht zu

der besonderen Situation in Ihrem Programm passt. Möglicherweise wissen Sie im Vorhinein nicht genau, wie viele Parameter Ihre Funktion übernehmen muss. Oder Sie kennen zwar die Parameter, aber diese haben fast immer den gleichen Wert, sodass es lästig ist, sie immer wieder erneut zu übergeben. Oder Sie möchten aus Ihrer Funktion mehr als einen Wert zurückgeben.

Dieses Kapitel hilft Ihnen, PHP so zu verwenden, dass Sie derartige Probleme lösen können. Wir beginnen mit der detaillierten Darstellung verschiedener Möglichkeiten, Argumente an Funktionen zu übergeben. Die Rezepte 6.1 bis 6.5 behandeln die Übergabe von Argumenten als Werte, als Referenzen und als benannte Parameter, die Zuweisung von Vorgabeparametern sowie Funktionen mit einer variablen Anzahl von Parametern.

In den vier daran anschließenden Rezepten geht es um die Rückgabe von Werten aus einer Funktion. Rezept 6.6 beschreibt die Rückgabe als Referenz, Rezept 6.7 behandelt die Rückgabe von mehr als einer Variablen, Rezept 6.8 beschreibt, wie man bestimmte Rückgabewerte überspringt, und in Rezept 6.9 erfahren Sie, wie man am besten aus einer Funktion zurückkehrt und das Ergebnis auf Fehler überprüft. Die letzten drei Rezepte zeigen, wie man variable Funktionen aufruft, mit Geltungsbereichsproblemen umgeht, und wie man eine Funktion dynamisch erzeugt. Es gibt noch ein Rezept zu Funktionsvariablen in Kapitel 5; wenn Sie möchten, dass eine Variable zwischen zwei Funktionsaufrufen ihren Wert behält, schauen Sie in Rezept 5.5 nach.

6.1 Auf Funktionsparameter zugreifen

Problem

Sie möchten auf die Werte zugreifen, die an eine Funktion übergeben worden sind.

Lösung

Verwenden Sie die Namen aus dem Funktionsprototyp:

```
function commercial_sponsorship($letter, $number) {
    print "Diese Episode der Sesamstraße wurde Ihnen präsentiert von ";
    print "dem Buchstaben $letter und der Zahl $number.\n";
}

commercial_sponsorship('G', 3);
commercial_sponsorship($another_letter, $another_number);
```

Diskussion

Innerhalb der Funktion ist es unwesentlich, ob die übergebenen Werte als Strings, Zahlen, Arrays oder andere Variablenarten übergeben werden. Sie können sie alle gleich behandeln und sie ansprechen, indem Sie die Namen aus dem Prototyp verwenden.

Anders als in C müssen Sie nicht den Typ der zu übergebenden Variablen festlegen. PHP achtet für Sie darauf.

Außerdem werden, wenn nicht anders angegeben, alle an und aus einer Funktion übergebenen Werte als Wert und nicht als Referenz übergeben (dies gilt nicht für übergebene Objekte. Ab PHP 5 werden hierbei standardmäßig nur noch Referenzen ausgetauscht). Dies bedeutet, dass PHP eine Kopie des Werts anlegt und Ihnen diese Kopie übergibt, damit Sie auf den Wert zugreifen und ihn manipulieren können. Daher verändern Änderungen, die Sie an Ihrer Kopie vornehmen, nicht den Originalwert. Hier sehen Sie ein Beispiel dazu:

```
function add_one($number) {
    $number++;
}

$number = 1;
add_one($number);
print "$number\n";
1
```

Wenn die Variable als Referenz übergeben worden wäre, würde der Wert von `$number` am Ende 2 sein.

In vielen Sprachen hat die Übergabe von Variablen als Referenz auch den Vorteil, dass dies deutlich schneller vonstatten geht als die Übergabe als Wert. Dies gilt zwar auch für PHP, allerdings ist der Geschwindigkeitsunterschied nur marginal. Aus diesem Grund empfehlen wir die Variablenübergabe als Referenz nur für den Fall, dass dies tatsächlich erforderlich ist, und nicht als Trick zur Performance-Verbesserung.

Siehe auch

Rezept 6.3 zur Übergabe von Werten als Referenzen und Rezept 6.6 zur Rückgabe von Werten als Referenzen.

6.2 Vorgabewerte für Funktionsparameter festlegen

Problem

Sie möchten, dass ein Parameter einen Vorgabewert für den Fall hat, dass der Aufrufer der Funktion ihn nicht übergibt. Beispielsweise könnte eine Funktion zum Zeichnen einer Tabelle einen Parameter für die Randbreite haben, der als Vorgabe den Wert 0 hat, wenn keine Breite angegeben wird.

Lösung

Weisen Sie innerhalb des Funktionsprototyps den Parametern Vorgabewerte zu:

```
function wrap_html_tag($string, $tag = 'b') {
    return "<$tag>$string</$tag>";
}
```

Diskussion

Das Beispiel in der Lösung setzt den Vorgabewert für `$tag` auf `b` (Fettschrift). Dieses Beispiel:

```
$string = 'Ich bin etwas HTML';
wrap_html_tag($string);
```

ergibt:

```
<b>Ich bin etwas HTML</b>
```

Dieses Beispiel:

```
wrap_html_tag($string, 'i');
```

liefert:

```
<i>Ich bin etwas HTML</i>
```

Zwei wichtige Dinge müssen Sie beim Zuweisen von Vorgabewerten bedenken: Erstens müssen alle Parameter mit Vorgabewerten nach den Parametern ohne Vorgabewerte erscheinen. Andernfalls kann PHP nicht wissen, welche Parameter ausgelassen wurden und den Vorgabewert annehmen sollen und welche Argumente die Vorgabe überschreiben. Daher kann `wrap_html_tag()` nicht so definiert werden wie hier:

```
function wrap_html_tag($tag = 'i', $string)
```

Wenn Sie dies tun und dann an `wrap_html_tag()` nur ein einzelnes Argument übergeben, weist PHP diesen Wert `$tag` zu und gibt eine Warnung über ein fehlendes Argument aus.

Zweitens muss der zugewiesene Wert eine Konstante sein – ein String oder eine Zahl. Er kann keine Variable sein. Wenn Sie noch einmal `wrap_html_tag()` als Beispiel nehmen, können Sie Folgendes nicht tun:

```
$my_favorite_html_tag = 'i';

function wrap_html_tag($string, $tag = $my_favorite_html_tag) {
    // ...
}
```

Wenn Sie eine leere Vorgabe zuweisen möchten, haben Sie die Möglichkeit, Ihrem Parameter den Leer-String zuzuweisen:

```
function wrap_html_tag($string, $tag = '') {
    if (empty($tag)) return $string;
    return "<$tag>$string</$tag>";
}
```

Diese Funktion gibt den Original-String zurück, wenn kein Wert für `$tag` übergeben wurde. Andernfalls gibt sie, wenn ein (nicht leerer) Tag-Wert übergeben wurde, den mit Tags umgebenen String zurück.

Unter Umständen kann eine andere Möglichkeit für den $tag-Vorgabewert auch 0 oder NULL sein. Bei wrap_html_tag() möchten Sie kein leeres Tag zulassen. Aber in manchen Fällen kann ein leerer String eine akzeptable Möglichkeit sein. Zum Beispiel wird join() häufig mit einem Leer-String aufgerufen, wenn zuvor file() aufgerufen wurde, um eine Datei in einen String einzufügen. Sie können auch, wie das folgende Programmstück zeigt, eine vorgegebene Mitteilung verwenden, wenn kein Argument angegeben ist, aber eine leere Nachricht bei Übergabe eines Leer-Strings:

```
function pc_log_db_error($message = NULL) {
    if (is_null($message)) {
        $message = 'Keine Verbindung zur Datenbank';
    }

    error_log("[DB] [$message]");
}
```

Siehe auch

Rezept 6.5 zu Funktionen, die eine variable Anzahl von Argumenten annehmen.

6.3 Werte als Referenzen übergeben

Problem

Sie möchten an eine Funktion eine Variable übergeben, die alle innerhalb der Funktion an ihr vorgenommenen Änderungen beibehält.

Lösung

Damit eine Funktion ein übergebenes Argument als Referenz und nicht als Wert annimmt, setzen Sie ein & vor den Parameternamen im Funktionsprototyp:

```
function wrap_html_tag(&$string, $tag = 'b') {
    $string = "<$tag>$string</$tag>";
}
```

Nun muss der String nicht zurückgegeben werden, da das Original an Ort und Stelle modifiziert wird.

In PHP 5 können Sie alternativ einfach beim Funktionsaufruf eine Referenz übergeben:

```
function wrap_html_tag($string, $tag = 'b') {
    $string = "<$tag>$string</$tag>";
}

$x = "Hello!";
wrap_html_tag(&$x,"b");
echo $x; // "<b>Hello!</b>"
```

Ab PHP 5.3 führt diese Variante allerdings zu einer Warnung:

```
PHP Deprecated:  Call-time pass-by-reference has been deprecated; If you would like to
pass it by reference, modify the declaration of wrap_html_tag().  If you would like to
enable call-time pass-by-reference, you can set allow_call_time_pass_reference to true in
your INI file ...
```

Diskussion

Wenn Sie eine Variable an eine Funktion als Referenz übergeben, können Sie sich die Mühe sparen, die Variable zurückzugeben und den Rückgabewert der ursprünglichen Variablen zuzuweisen. Dies ist auch nützlich, wenn die Funktion einen Booleschen Wert `true` oder `false` für den Erfolg zurückgeben soll, trotzdem aber Argumentwerte durch die Funktion bearbeitet werden sollen.

In PHP 5 können Sie bei der Argumentübergabe zwischen Übergabe als Parameter und Übergabe als Referenz wählen. Es ist immer beides möglich, solange der Parameter in der Funktion nicht mit `&` markiert ist. Ist Letzteres der Fall, wird der übergebene Parameter als Referenz behandelt. Ab PHP 5.3 gilt jedoch die Einschränkung, dass die Übergabe als Referenz zur Aufrufzeit (*call-time pass-by-reference*) zu einer Warnung führt. Entweder deklarieren Sie die Übergabe als Referenz im Funktionskopf, oder Sie erlauben die Übergabe zur Laufzeit, indem Sie in der *php.ini* die Direktive `allow_call_time_pass_reference` auf den Wert `true` setzen.

Wenn Sie ein Objekt an eine Funktion übergeben, ist die Variable in PHP 5 automatisch eine Referenz, d.h., die Funktion arbeitet mit dem eigentlichen Objekt. Hier dürfen Sie kein `&` vor die Variable stellen.

Außerdem sollten Sie beachten, dass Sie an einen Parameter, der so deklariert ist, dass er einen Wert als Referenz übernimmt, keinen konstanten String (bzw. keine konstante Zahl usw.) übergeben können. Andernfalls bricht PHP mit einem schweren Fehler ab.

Siehe auch

Rezept 6.6 zur Rückgabe von Werten als Referenzen.

6.4 Benannte Parameter verwenden

Problem

Sie möchten die Argumente für einen Funktionsaufruf über Namen zuordnen und nicht nur über ihre Position.

Lösung

Definieren Sie die Funktion mit nur einem Parameter, der aber ein assoziatives Array ist:

```
function image($img) {
    $tag  = '<img src="' . $img['src'] . '" ';
    $tag .= 'alt="' . ($img['alt'] ? $img['alt'] : '') .'">';
    return $tag;
}

$image = image(array('src' => 'cow.png', 'alt' => 'Kuh sagt Muh'));
$image = image(array('src' => 'pig.jpeg'));
```

Diskussion

Zwar wird der Code Ihrer Funktionen komplizierter, wenn Sie benannte Parameter verwenden, aber er wird dadurch auch besser lesbar. Da sich eine Funktion nur an einer Stelle befindet, aber an vielen verwendet wird, erhalten Sie auf diese Weise ein insgesamt verständlicheres Programm.

Bei Verwendung dieser Technik beschwert sich PHP nicht mehr, wenn Sie versehentlich einen Parameternamen falsch schreiben. Sie müssen daher besonders sorgfältig sein, denn der Parser kann derartige Fehler nicht abfangen. Auch können Sie nicht die Fähigkeit von PHP nutzen, Parametern Vorgabewerte zuzuweisen. Zum Glück lässt sich aber dieses Defizit mit ein wenig einfachem Code am Anfang der Funktion ausgleichen:

```
function image($img) {
    if (! isset($img['src']))    { $img['src']    = 'cow.png';      }
    if (! isset($img['alt']))    { $img['alt']    = 'Molkerei';     }
    if (! isset($img['height'])) { $img['height'] = 100;            }
    if (! isset($img['width']))  { $img['width']  = 50;             }
    // ...
}
```

Mit der `isset()`-Funktion prüfen Sie, ob die Werte aller Parameter gesetzt sind, und weisen gegebenenfalls einen Vorgabewert zu.

Als Alternative können Sie zu diesem Zweck auch eine kurze Funktion schreiben:

```
function pc_assign_defaults($array, $defaults) {
    $a = array();
    foreach ($defaults as $d => $v) {
        $a[$d] = isset($array[$d]) ? $array[$d] : $v;
    }

    return $a;
}
```

Diese Funktion durchläuft in einer Schleife eine Reihe von Schlüsseln, die sich in einem Array von Vorgabewerten befinden, und prüft jeweils, ob in dem übergebenen Array `$array` ein solcher Wert gesetzt ist. Wenn nicht, weist die Funktion einen Vorgabewert aus `$defaults` zu. Um diese Funktion in den obigen Programmausschnitt einzusetzen, tauschen Sie die oberen Zeilen gegen die folgenden aus:

```
function image($img) {
    $defaults = array('src'    => 'cow.png',
```

```
                    'alt'    => 'Molkerei',
                    'height' => 100,
                    'width'  => 50
                   );
    $img = pc_assign_defaults($img, $defaults);
    // ...
}
```

Diese Lösung ist schöner, weil der Code dadurch flexibler wird. Wenn Sie den Mechanismus der Zuweisung von Vorgabewerten ändern möchten, brauchen Sie diesen nur innerhalb von `pc_assign_defaults()` zu ändern und nicht in hunderten von Programmzeilen. Außerdem ist es klarer, wenn Sie ein Array von Name/Wert-Paaren verwenden und die Vorgabewerte in einer einzigen Zeile zuweisen, anstatt beides in einer Reihe fast identischer, sich wiederholender Zeilen zu vermischen.

Siehe auch

Rezept 6.5 zu Funktionen, die eine variable Anzahl von Argumenten annehmen.

6.5 Funktionen mit einer variablen Anzahl von Argumenten verwenden

Problem

Sie möchten eine Funktion definieren, die eine variable Anzahl von Argumenten übernimmt.

Lösung

Übergeben Sie ein Array und platzieren Sie die variablen Argumente in das Array:

```
// Den "Durchschnitt" einer Reihe von Zahlen ermitteln.
function mean($numbers) {
    // Initialisieren, um Warnungen zu vermeiden.
    $sum = 0;

    // Die Anzahl der Elemente im Array.
    $size = count($numbers);

    // Das Array durchlaufen und die Zahlen addieren.
    for ($i = 0; $i < $size; $i++) {
        $sum += $numbers[$i];
    }

    // Durch die Anzahl der Zahlen dividieren.
    $average = $sum / $size;
```

```
    // Den Durchschnitt zurückgeben
    return $average;
}

$mean = mean(array(96, 93, 97));
```

Diskussion

Abhängig von Ihrem Programmierstil und Ihren Vorlieben gibt es zwei gute Lösungen. Die eher traditionelle PHP-Methode ist die in der obigen Lösung beschriebene. Wir bevorzugen dieses Vorgehen, da Arrays in PHP häufig verwendet werden und daher alle Programmierer mit Arrays und deren Verhalten vertraut sind.

Zwar erzeugt diese Methode einen gewissen Mehraufwand, aber das Bündeln von Variablen ist ein alltäglicher Vorgang. Es dient in Rezept 6.4 der Verwendung benannter Parameter und in Rezept 6.7 der Rückgabe von mehr als einem Wert aus einer Funktion. Außerdem erfordert die Syntax für den Zugriff auf das Array und seine Manipulation nur elementare Befehle wie `$array[$i]` und `count($array)`.

Dies mag aber etwas umständlich erscheinen, daher bietet PHP eine Alternative an, die den direkten Zugriff auf die Argumentliste ermöglicht:

```
// Den "Durchschnitt" einer Reihe von Zahlen ermitteln.
function mean() {
    // Initialisieren, um Warnungen zu vermeiden.
    $sum = 0;

    // Die Anzahl der an die Funktion übergebenen Argumente.
    $size = func_num_args();

    // Die Argumente durchlaufen und die Zahlen addieren.
    for ($i = 0; $i < $size; $i++) {
        $sum += func_get_arg($i);
    }

    // Durch die Anzahl der Zahlen dividieren.
    $average = $sum / $size;

    // Den Durchschnitt zurückgeben.
    return $average;
}

$mean = mean(96, 93, 97);
```

Dieses Beispiel verwendet eine Reihe von Funktionen, die Daten anhand der Argumente liefern, die der Funktion übergeben wurden, aus der sie aufgerufen werden. Die erste, `func_num_args()`, gibt eine Integer-Zahl mit der Anzahl der Argumente zurück, die an die aufrufende Funktion übergeben worden sind – in diesem Fall also an `mean()`. Danach können Sie dann `func_get_arg()` aufrufen, um den spezifischen Argumentwert für jede einzelne Position zu ermitteln.

Wenn Sie `mean(96, 93, 97)` aufrufen, gibt `func_num_args()` den Wert `3` zurück. Das erste Argument befindet sich an der Position `0`; Sie müssen also von `0` bis `2` zählen und nicht von `1` bis `3`. Genau dies geschieht in der `for`-Schleife, in der `$i` die Werte `0` bis `$size` minus eins durchläuft. Wie Sie sehen, ist dies die gleiche Logik, die wir bereits im ersten Beispiel angewendet haben, bei dem ein Array übergeben wurde. Wegen der möglichen Mehrbelastung durch die Verwendung von `func_get_arg()` innerhalb einer Schleife brauchen Sie sich keine Sorgen zu machen – tatsächlich ist diese Version immer noch schneller als die mit der Übergabe eines Arrays.

Es gibt noch eine dritte Version dieser Funktion, die mithilfe von `func_num_args()` ein Array mit allen an die Funktion übergebenen Werten liefert. Diese sieht gewissermaßen wie eine Kreuzung aus den vorangegangenen beiden Funktionen aus:

```
// Den "Durchschnitt" einer Reihe von Zahlen ermitteln.
function mean() {
    // Initialisieren, um Warnungen zu vermeiden.
    $sum = 0;

    // Die Argumente in $numbers laden.
    $numbers = func_get_args();

    // Die Anzahl der Argumente im Array.
    $size = count($numbers);

    // Das Array durchlaufen und die Zahlen addieren.
    for ($i = 0; $i < $size; $i++) {
        $sum += $numbers[$i];
    }

    // Durch die Anzahl der Zahlen dividieren.
    $average = $sum / $size;

    // Den Durchschnitt zurückgeben.
    return $average;
}

$mean = mean(96, 93, 97);
```

Hier haben Sie den doppelten Vorteil, dass Sie die Zahlen nicht in ein temporäres Array speichern müssen, um sie an `mean()` zu übergeben, und trotzdem können Sie sie weiterhin so behandeln wie zuvor. Leider ist diese Methode jedoch etwas langsamer als die ersten beiden.

Siehe auch

Rezept 6.7 zur Rückgabe von Mehrfachwerten aus einer Funktion; die Dokumentationen zu `func_num_args()` unter *http://www.php.net/func-num-args*, `func_get_arg()` unter *http://www.php.net/func-get-arg* sowie `func_get_args()` unter *http://www.php.net/func-get-args*.

6.6 Werte per Referenz zurückgeben

Problem

Sie möchten einen Wert als Referenz und nicht als Wert zurückgeben. Dadurch können Sie das zweifache Kopieren einer Variablen vermeiden.

Lösung

Die Syntax für die Rückgabe einer Variablen per Referenz ähnelt der Parameterübergabe per Referenz. Allerdings fügen Sie hier das `&` nicht vor dem Parameternamen, sondern vor dem Funktionsnamen ein:

```
function &wrap_html_tag($string, $tag = 'b') {
    $result = "<$tag>$string</$tag>";
    return $result;
}
```

Außerdem müssen Sie beim Aufruf der Funktion den Zuweisungsoperator `=&` anstelle des einfachen `=` verwenden:

```
$html =& wrap_html_tag($string);
```

Diskussion

Während bei der Übergabe von Werten an Funktionen die Argumente entweder als Wert oder als Referenz übergeben werden müssen, können Sie hier auch, anstatt die Referenz zuzuweisen, einfach den zurückgegebenen Wert übernehmen. Setzen Sie einfach `=` anstelle von `=&` ein, und PHP weist den Wert und nicht die Referenz zu.

Wie bei der Übergabe von Parametern werden Objekte allerdings immer per Referenz zurückgegeben.

Siehe auch

Rezept 6.3 zur Übergabe von Werten per Referenz.

6.7 Mehr als einen Wert zurückgeben

Problem

Sie möchten aus einer Funktion mehr als nur einen Wert zurückgeben.

Lösung

Geben Sie ein Array zurück und trennen Sie dessen Elemente mit `list()`:

```
function averages($stats) {
    // ...
    return array($median, $mean, $mode);
}

list($median, $mean, $mode) = averages($stats);
```

Diskussion

Aus Performance-Sicht ist dies keine besonders gute Idee. Es führt zu einer zusätzlichen Belastung, wenn PHP gezwungen wird, ein Array anzulegen und gleich wieder wegzuwerfen. Und dies geschieht in dem folgenden Beispiel:

```
function time_parts($time) {
    return explode(':', $time);
}

list($hour, $minute, $second) = time_parts('12:34:56');
```

Sie übergeben einen Zeit-String, wie man ihn auf einer Digitaluhr sehen kann, und rufen `explode()` auf, um ihn in Array-Elemente aufzuteilen. Nach der Rückkehr von `time_parts()` übernehmen Sie mithilfe von `list()` jedes einzelne Element und speichern es in einer Skalar-Variablen ab. Dies ist zwar etwas ineffizient, aber die anderen möglichen Lösungen sind noch schlimmer, da sie zu verwirrendem Code führen können.

Eine Alternative besteht darin, die Werte per Referenz zu übergeben. Das ist jedoch ziemlich umständlich und wirkt nicht intuitiv, denn es ist nicht immer sinnvoll, die notwendigen Variablen an die Funktion zu übergeben. Ein Beispiel:

```
function time_parts($time, &$hour, &$minute, &$second) {
    list($hour, $minute, $second) = explode(':', $time);
}

time_parts('12:34:56', $hour, $minute, $second);
```

Wenn man den Funktionsprototyp nicht kennt, kann man beim Betrachten dieser Zeile kaum erkennen, dass `$hour`, `$minute` und `$second` eigentlich die Rückgabewerte von `time_parts()` sind.

Sie können auch globale Variablen benutzen, aber dies bläht den globalen Namensraum auf, und man kann nicht mehr ohne weiteres erkennen, welche Variablen durch die Funktion stillschweigend verändert werden. Dazu ein Beispiel:

```
function time_parts($time) {
    global $hour, $minute, $second;
    list($hour, $minute, $second) = explode(':', $time);
}

time_parts('12:34:56');
```

Hier ist es nur klar, weil die Funktion unmittelbar über dem Aufruf steht, aber wenn sich die Funktion in einer anderen Datei befindet oder von einer anderen Person geschrieben wurde, ist diese Form ziemlich mysteriös und kann somit leicht zu subtilen Fehlern führen.

Wir empfehlen, wenn Sie einen Wert innerhalb einer Funktion verändern, diesen als Rückgabewert zu übergeben und einer Variablen zuzuweisen, es sei denn, Sie haben einen sehr guten Grund, wie etwa signifikante Performance-Probleme. Diese Lösung ist sauberer und einfacher zu verstehen und zu pflegen.

Siehe auch

Rezept 6.3 zur Übergabe von Werten per Referenz und Rezept 6.11 für Informationen über die Geltungsbereiche von Variablen.

6.8 Bestimmte Rückgabewerte überspringen

Problem

Eine Funktion liefert mehrere Werte, aber Sie interessieren sich nur für einen Teil davon.

Lösung

Lassen Sie Variablen innerhalb von `list()` aus:

```
// Nur die Minuten sind von Interesse.
function time_parts($time) {
    return explode(':', $time);
}

list(, $minute,) = time_parts('12:34:56');
```

Diskussion

Auch wenn es wie ein Fehler aussieht: Der Code zu dieser Lösung ist gültiges PHP. So etwas sieht man häufig, wenn ein Programmierer mit `each()` ein Array durchläuft und dabei nur die Werte des Arrays betrachten will:

```
while (list(,$value) = each($array)) {
    process($value);
}
```

Mithilfe von `foreach` kann dies jedoch klarer formuliert werden:

```
foreach ($array as $value) {
    process($value);
}
```

Um die Verwirrung zu verringern, verwenden wir diese Möglichkeit nicht häufig, aber wenn eine Funktion viele Werte liefert und Sie nur einen oder zwei von ihnen benötigen, ist diese Technik sehr praktisch. Ein Beispiel dafür ist das Einlesen von Feldern mit

fgetcsv(); diese Funktion gibt ein Array mit Feldern aus der Zeile zurück. In diesem Fall können Sie die folgenden Zeilen verwenden:

```
while ($fields = fgetcsv($fh, 4096)) {
    print $fields[2] . "\n";  // das dritte Feld
}
```

Wenn es eine intern geschriebene und keine eingebaute Funktion ist, können Sie auch dafür sorgen, dass das zurückgegebene Array String-Schlüssel hat, denn man kann sich nur schwer merken, dass beispielsweise das zweite Array-Element mit 'rank' verknüpft ist:

```
while ($fields = read_fields($filename)) {
    $rank = $fields['rank']; // das dritte Feld heißt nun "rank"
    print "$rank\n";
}
```

Die folgende Methode ist allerdings am effizientesten:

```
while (list(,,$rank,,) = fgetcsv($fh, 4096)) {
    print "$rank\n";         // $rank direkt zuweisen
}
```

Achten Sie darauf, dass Sie sich bei der Anzahl der Kommata nicht verzählen, sonst haben Sie einen Bug.

Siehe auch

Rezept 1.9 mit weiteren Informationen über das Lesen von Dateien mit fgetcsv().

6.9 Fehlermeldungen zurückgeben

Problem

Sie möchten anzeigen, dass eine Funktion auf einen Fehler gestoßen ist.

Lösung

Werfen Sie eine Exception:

```
function lookup($name) {
    if (empty($name)) { throw new Exception("Name ist leer!"); }
    // ...
}
```

Diskussion

Eine saubere Variante, Fehler- bzw. Ausnahmesituationen zu signalisieren, steht ab PHP 5 in der Form von Ausnahmen (Exceptions) zur Verfügung. Hier können Sie Fehlermeldungen künstlich erzeugen:

```
function lookup($name) {
    if (empty($name)) { throw new Exception("Name ist leer!", 1); }
    // ...
}
```

Eine solche Exception führt unbehandelt zum Skriptabbruch – wie andere Fatal Error-Fehlermeldungen auch. Wenn Sie nicht wollen, dass Ihr Skript durch einen solchen Fehler gleich abstürzt, müssen Sie den Funktionsaufruf mit einem try/catch-Statement umgeben und die Ausnahme abfangen:

```
try {
    $address = lookup($name);
}
catch (Exception $e)
{
    echo "Der Name darf nicht leer sein!";
    // Sonstige Maßnahmen in dieser Situation.
    // ...
}
// Mit $address weiterarbeiten.
// ...
```

Exceptions sind ein derart umfangreiches Thema, dass wir ihnen ein eigenes Kapitel (Kapitel 8) gewidmet haben. Der Hauptvorteil hier besteht darin, dass Ihre Funktion auch im Normalfall alle möglichen Werte zurückgeben kann und keinen Wert für eine Fehlermeldung reservieren muss. In diesem Beispiel könnte `lookup()` auch dann `false` zurückgeben, wenn `$name` nicht gefunden wird.

In PHP-Versionen vor 5.0 müssen Sie Fehlersituationen durch den Rückgabewert einer Funktion signalisieren. Daher ist es am besten, wenn alle Ihre Funktionen das definierte Schlüsselwort `false` zurückgeben. Auf diese Weise kann man am besten einen logischen Wert prüfen.

Andere Möglichkeiten sind `''` oder `0`. Alle drei Werte gelten zwar innerhalb eines `if` als nicht wahr, jedoch sind sie unterschiedlich. Außerdem kann bisweilen der Rückgabewert `0` ein sinnvoller Wert sein, trotzdem möchten Sie aber auch den Fehlerzustand zurückgeben können.

Beispielsweise gibt `strpos()` die Position des ersten Teil-Strings innerhalb eines Strings zurück. Wenn der Teil-String nicht gefunden werden kann, liefert `strpos()` den Wert `false`. Wird er gefunden, gibt die Funktion eine Integer-Zahl mit der Position zurück. Daher können Sie, um die Position eines Teil-Strings zu ermitteln, Folgendes schreiben:

```
if (strpos($string, $substring)) { /* gefunden! */ }
```

Wenn allerdings `$substring` genau am Anfang von `$string` gefunden wird, ist der Rückgabewert `0`. Leider wird dies innerhalb eines `if` als `false` ausgewertet, und somit wird die bedingte Verzweigung nicht ausgeführt. Hier ist das korrekte Vorgehen, um mit dem Rückgabewert von `strpos()` umzugehen:

```
if (false !== strpos($string, $substring)) { /* gefunden! */ }
```

Außerdem ist `false` garantiert immer falsch – in der aktuellen PHP-Version und für alle Zeiten. Bei anderen Werten ist dies nicht so sicher. So ergab in PHP 3 `empty('0')` das Ergebnis `true`; in PHP 4 wurde daraus jedoch `false`.

Siehe auch

Kapitel 8 über Exceptions; die Einführung zu Kapitel 5 mit weiteren Informationen über Wahrheitswerte bei Variablen; die Dokumentationen zu `strpos()` unter *http://www.php.net/strpos* und `empty()` unter *http://www.php.net/empty*.

6.10 Variable Funktionen aufrufen

Problem

Sie möchten in Abhängigkeit vom Wert einer Variablen unterschiedliche Funktionen aufrufen.

Lösung

Verwenden Sie die Funktion `call_user_func()`:

```
function get_file($filename) { return file_get_contents($filename); }

$function = 'get_file';
$filename = 'graphic.png';

call_user_func($function, $filename); // Ruft get_file('graphic.png') auf.
```

Wenn Ihre Funktionen unterschiedliche Anzahlen von Argumenten entgegennehmen, verwenden Sie die Funktion `call_user_func_array()`:

```
function get_file($filename) { return file_get_contents($filename); }
function put_file($filename, $data) { return file_put_contents($filename, $data); }

if ($action == 'get') {
    $function = 'get_file';
    $args = array('graphic.png');
} elseif ($action == 'put') {
    $function = 'put_file';
    $args = array('graphic.png', $graphic);
}

// Ruft abhängig vom Inhalt von $action
// die Funktion get_file('graphic.png')
// oder put_file('graphic.png', $graphic) auf.
call_user_func_array($function, $args);
```

Alternativ verwenden Sie variable Variablen:

```
function eat_fruit($fruit) { print "zerkaue $fruit."; }

$function = 'eat_fruit';
$fruit = 'Kiwi';

$function($fruit); // Ruft eat_fruit() auf.
```

Diskussion

Die Funktionen `call_user_func()` und `call_user_func_array()` unterscheiden sich in ihrer Funktionsweise von normalen PHP-Funktionen. Ihr erstes Argument enthält keinen String für eine Ausgabe, auch keine Zahl für eine Addition, sondern den Namen einer aufzurufenden Funktion. Das Konzept vom Durchreichen des Namens einer Funktion, die dann indirekt aufgerufen wird, bezeichnet man als *Callback*. Die indirekt aufgerufene Funktion ist also die Callback-Funktion.

Während die Funktion `call_user_func()` nach dem ersten Parameter beliebig viele weitere Parameter entgegennimmt, um diese der aufzurufenden Funktion durchzureichen, akzeptiert die Funktion `call_user_func_array()` zu diesem Zweck als zweiten Parameter nur ein Array.

Wenn es mehrere Aufrufmöglichkeiten gibt, bietet es sich an, ein assoziatives Array mit Funktionsnamen aufbauen. Im Beispiel mit dem Einsatz variabler Variablen würde das dann wie folgt aussehen:

```
$dispatch = array(
    'add'      => 'do_add',
    'commit'   => 'do_commit',
    'checkout' => 'do_checkout',
    'update'   => 'do_update'
);

$cmd = (isset($_REQUEST['command']) ? $_REQUEST['command'] : '');

if (array_key_exists($cmd, $dispatch)) {
    $function = $dispatch[$cmd];
    $function(); // Rufe die Funktion auf.
} else {
    error_log("Unbekannter Befehl $cmd");
}
```

Dieses Programmstück übernimmt einen Befehlsnamen aus der Anfrage und führt diese Funktion aus. Beachten Sie die Überprüfung, ob sich das Wort in der Liste der akzeptierten Befehle befindet. Diese verhindert, dass Ihr Code jede beliebige mit einer Anfrage übergebene Funktion ausführt, zum Beispiel `phpinfo()`. Dadurch wird Ihr Programm sicherer, und Sie haben die Möglichkeit, auf einfache Weise Fehler zu protokollieren.

Ein weiterer Vorteil besteht darin, dass Sie mehrere Befehle auf dieselbe Funktion abbilden und damit einen langen und einen kurzen Namen verwenden können:

```
$dispatch = array(
    'add'      => 'do_add',
    'commit'   => 'do_commit',   'ci' => 'do_commit',
    'checkout' => 'do_checkout', 'co' => 'do_checkout',
    'update'   => 'do_update',   'up' => 'do_update'
);
```

Siehe auch

Rezept 5.4 mit weiteren Informationen über variable Variablen. Die Funktion `call_user_func( )` wird unter *http://www.php.net/call-user-func* beschrieben. Für die Funktion `call_user_func_array( )` liefert die Adresse *http://www.php.net/call-user-func-array* detaillierte Informationen.

6.11 Innerhalb einer Funktion auf eine globale Variable zugreifen

Problem

Sie müssen innerhalb einer Funktion auf eine globale Variable zugreifen.

Lösung

Bringen Sie die globale Variable mit dem Schlüsselwort `global` in den lokalen Geltungsbereich:

```
function eat_fruit($fruit) {
   global $chew_count;

   for ($i = $chew_count; $i > 0; $i--) {
       // ...
   }
}
```

oder referenzieren Sie die globale Variable direkt in `$GLOBALS`:

```
function eat_fruit($fruit) {
   for ($i = $GLOBALS['chew_count']; $i > 0; $i--) {
       // ...
   }
}
```

Diskussion

Wenn Sie innerhalb einer Funktion mehrere globale Variablen verwenden, wird die Syntax der Funktion durch das Schlüsselwort `global` möglicherweise verständlicher, insbesondere wenn die globalen Variablen in Strings interpoliert werden.

Mit dem Schlüsselwort `global` können Sie mehrere globale Variablen zugleich in den lokalen Geltungsbereich bringen; geben Sie dazu die Variablen in einer kommaseparierten Liste an:

```
global $age,$gender,$shoe_size;
```

Sie können auch die Namen der globalen Variablen mithilfe von variablen Variablen angeben:

```
$which_var = 'age';
global $$which_var; // meint die globale Variable $age
```

Wenn Sie allerdings `unset()` für eine Variable aufrufen, die mithilfe des Schlüsselworts `global` in den lokalen Geltungsbereich gebracht wurde, wird die Variable nur innerhalb der Funktion gelöscht. Um eine Variable aus dem globalen Geltungsbereich zu löschen, müssen Sie `unset()` mit dem entsprechenden Element des Arrays `$GLOBALS` aufrufen:

```
$food =  'Pizza';
$drink = 'Bier';

function party() {
    global $food, $drink;

    unset($food);             // iss Pizza
    unset($GLOBALS['drink']); // trinke Bier
}

print "$food: $drink\n";
party();
print "$food: $drink\n";
```

```
Pizza: Bier
Pizza:
```

Sie sehen, dass `$food` gleichgeblieben ist, während `$drink` gelöscht wurde. Die Deklaration einer Variablen als `global` innerhalb einer Funktion ähnelt der Zuweisung einer Referenz auf die globale Variable an die lokale Variable:

```
$food = $GLOBALS['food'];
```

Siehe auch

Die Dokumentation zu Geltungsbereichen von Variablen unter *http://www.php.net/variables.scope* und zu Referenzen auf Variablen unter *http://www.php.net/language.references*.

6.12 Dynamische Funktionen erzeugen

Problem

Sie möchten eine Funktion anlegen und definieren, während das Programm läuft.

Lösung

Seit PHP 5.3 können Sie hierfür eine *Lambda-Funktion* einsetzen:

```
$add = function($a, $b) {return $a+$b;};

$add(1, 1); // gibt 2 zurück
```

In PHP-Versionen vor 5.3 verwenden Sie `create_function()`:

```
$add = create_function('$i,$j', 'return $i+$j;');

$add(1, 1); // gibt 2 zurück
```

Diskussion

Mit PHP 5.3 wurden Lambda-Funktionen eingeführt. Eine Lambda-Funktion ist eine anonyme Funktion, die an der Stelle ihrer Verwendung deklariert wird. Sie können eine Lambda-Funktion einer Variablen zuweisen und sie somit auch als Argument an andere Funktionen übergeben. Lambda-Funktionen existieren nur in dem Geltungsbereich der Variablen, der sie zugewiesen werden. Oftmals bezeichnet man diese Funktionen daher auch als »Wegwerffunktionen«.

In PHP-Versionen vor 5.3 können Sie eine dynamische Funktion durch den Aufruf von `create_function()` erzeugen. Der erste Parameter für `create_function()` ist ein String, der die Argumente der Funktion enthält, und der zweite ist der Funktionsrumpf. Die Verwendung von `create_function()` ist außerordentlich langsam; wenn Sie daher eine Funktion vorab definieren können, sollten Sie dies besser tun.

Meistens werden dynamische Funktionen zur Definition angepasster Sortierfunktionen für `usort()` und `array_walk()` verwendet:

```
// Dateien in der umgekehrten natürlichen Anordnung sortieren.
usort($files, create_function('$a, $b', 'return strnatcmp($b, $a);'));
// ... oder als Lambda-Funktion ...
usort($files, function($a,$b) {return strnatcmp($b, $a);});
```

Die Möglichkeit, ab PHP 5.3 Lambda-Funktionen deklarieren zu können, ist zwar syntaktisch eleganter als die Nutzung der Funktion `create_function()`, allerdings bieten sie keinen Zugewinn an neuer Funktionalität. An dieser Stelle kommen die *Closures* ins Spiel, die ebenfalls seit PHP 5.3 zur Verfügung stehen.

Ein Closure hat die gleichen Eigenschaften wie eine Lambda-Funktion. Genauer gesagt, ist ein Closure eine spezielle Lambda-Funktion. Closures werden an der Stelle ihrer Ver-

wendung deklariert und können ebenfalls Variablen zugewiesen werden. Darüber hinaus haben Closures allerdings die Möglichkeit der Interaktion mit Variablen aus der Umgebung, in der sie deklariert wurden. Diese Variablen müssen dazu bei der Deklaration eines Closures lediglich an das Closure gebunden werden. Das nachfolgende Beispiel zeigt ein Closure, das eine Variable aus seiner Umgebung importiert:

```
$string  = 'Hallo Welt!';
$closure = function() use ($string) { echo $string; };
$closure();
Hallo Welt!
```

Sie können Variablen an ein Closure auch per Referenz übergeben. Das bedeutet, dass sich Veränderungen an der Variablen innerhalb des Closures auch auf die Variable außerhalb der Funktion auswirken:

```
$zaehler = 1;
$closure = function() use (&$zaehler) { $zaehler++; };
echo $zaehler;
$closure();
echo $zaehler;
$closure();
echo $zaehler;
1
2
3
```

Siehe auch

Rezept 4.17 mit Informationen über `usort()`; die Dokumentationen zu anonymen Funktionen unter *http://docs.php.net/functions.anonymous*, zu `create_function()` unter *http://www.php.net/create-function* und zu `usort()` unter *http://www.php.net/usort*.

6.13 Objekt-Datentypen für Funktionsparameter vorschreiben

Problem

Sie möchten sicherstellen, dass ein übergebener Funktionsparameter immer einen bestimmten Objekt-Datentyp (eine Klasse) hat. Das schützt Sie gegen Bugs, weil auf diese Weise frühzeitig auffällt, wenn etwas anderes übergeben wird, als Sie erwarten.

Lösung

Verwenden Sie *Type Hinting*:

```
function printBusinessPhoneNumber(business $business) {
    echo $business->phone;
```

```
}

$shop = new business;
$shop->phone = "35972355";
printBusinessPhoneNumber($shop); // Gibt die Telefonnummer aus.

$fred = new person;
$fred->phone = "38549865";
printBusinessPhoneNumber($fred); // Gibt eine Fehlermeldung aus.
```

Diskussion

Es kann leicht passieren, dass Ihrer Funktion ein Objekt mit falschem Datentyp übergeben wird oder auch gar kein Objekt, beispielsweise wenn Sie vergessen, eine globale Variable mit `global` in eine Funktion zu importieren und diese dann versehentlich als undefinierte Variable einer anderen Funktion übergeben wird. Oder wenn Sie Code gemeinsam mit anderen entwickeln und ein Anwender Ihrer Funktion plötzlich einen Datentyp übergibt, den die Funktion nicht verdauen kann oder soll. Da PHP im Allgemeinen versucht, hilfreich zu sein und Daten so weit wie möglich einfach in den passenden Typ umwandelt, kann es manchmal schwer sein, den Fehler zu finden.

Type Hinting ist neu in PHP 5 und ermöglicht es Ihnen, PHP vorzuschreiben, was es in Ihre Funktion hereinlassen darf und was nicht. Im obigen Fall bestehen wir beispielsweise darauf, ein Objekt der Klasse `business` überreicht zu bekommen. Damit kommen wir dem Missbrauch der Funktion mit einem `person`-Objekt auf die Spur, obwohl es ebenfalls eine Eigenschaft namens `name` hat.

Siehe auch

Die PHP-Dokumentation zu Type Hinting unter *http://www.php.net/manual/en/language.oop5.typehinting.php*.

KAPITEL 7

Klassen und Objekte

7.0 Einführung

PHP 5 bietet eine erheblich verbesserte Unterstützung für die objektorientierte Programmierung (OOP). Das ist eine bedeutsame Änderung und ein wesentlicher Grund für einen Wechsel von PHP 4 zu PHP 5. Wenn Sie bereits ein OOP-Fan sind, werden Sie mit den Mitteln, die PHP 5 bietet, sehr zufrieden sein.

Frühe PHP-Versionen waren rein prozedural: Sie konnten Funktionen definieren, aber keine Objekte. PHP 3 führte dann eine äußerst rudimentäre Form von Objekten ein, die nichts anderes war als ein flüchtiger Hack. Das war 1997, und da konnte noch niemand vorhersehen, in welchem Maße die Zahl der PHP-Programmierer explodieren würde oder dass jemand damit beginnen könnte, mit PHP großformatige Anwendungen zu entwerfen. Niemand betrachtete diese Einschränkungen deswegen als ein Problem.

Mit den Jahren erhielt PHP weitere objektorientierte Funktionalitäten. Das Entwicklungsteam entschloss sich allerdings nie dazu, den Kern des OO-Codes so zu überarbeiten, dass Objekte und Klassen angemessen integriert wurden. Die Folge war, dass PHP 4 die Leistung im Allgemeinen zwar erheblich verbesserte, das Schreiben komplexer OO-Programme aber weiterhin schwierig, wenn nicht gar unmöglich blieb.

PHP 5 behob diese Probleme, indem es die Zend Engine 2 (ZE2) einsetzte. ZE2 ermöglichte es PHP, fortgeschrittenere objektorientierte Funktionalitäten einzuschließen, bewahrte aber immer noch ein großes Maß an Rückwärtskompatibilität mit den Millionen von PHP-Skripten, die bereits geschrieben wurden.

Wenn Sie noch keine Erfahrung mit der objektorientierten Programmierung in anderen Sprachen als PHP haben, können Sie sich auf eine kleine Überraschung gefasst machen. Mit einigen der neuen Funktionalitäten können Sie tatsächlich manches leichter bewältigen, mit vielen anderen können Sie dagegen überhaupt nichts Neues machen. In mancherlei Hinsicht beschränken sie sogar, was Sie machen können.

Obgleich das der Intuition zu widersprechen scheint, helfen Ihnen diese Einschränkungen tatsächlich, sicheren Code schneller zu schreiben, weil sie Sie zur Codewiederverwendung und Datenkapselung auffordern. Diese Grundtechniken der OO-Programmierung werden in diesem Kapitel erläutert. Vorab wollen wir Ihnen aber zunächst eine Einführung in die objektorientierte Programmierung, ihre Terminologie sowie ihre Konzepte bieten.

Eine Klasse ist ein Paket, das zwei Dinge enthält: Daten und Methoden zum Zugriff und zur Manipulation dieser Daten. Die erste Komponente einer Klasse, die Daten, wird über Variablen festgehalten, die als Eigenschaften bezeichnet werden. Die zweite Komponente ist ein Satz von Funktionen, die diese Eigenschaften verwenden können und als Methoden bezeichnet werden.

Wenn Sie eine Klasse definieren, definieren Sie damit nicht unmittelbar ein Objekt, auf das Sie zugreifen und das Sie manipulieren können, sondern eine Vorlage für ein Objekt. Auf Basis dieser Blaupause erzeugen Sie in einem Instantiierung genannten Vorgang die formbaren Objekte. Ein Programm kann mit mehreren Objekten einer Klasse arbeiten, ebenso wie ein Mensch mehrere Bücher haben oder ein Baum viele Früchte tragen kann.

Klassen existieren in einer festen Hierarchie. Die Klassen auf den jeweils untergeordneten Ebenen sind dabei spezifischer als die auf den übergeordneten Ebenen. Diese spezialisierteren Klassen nennt man Kindklassen, während die Klasse, die sie modifizieren, als Elternklasse bezeichnet wird. Eine Elternklasse könnte beispielsweise Gebäude sein. Gebäude selbst können dann in Wohnhäuser und Bürogebäude gegliedert sein. Wohnhäuser können wiederum in Einfamilienhäuser, Mehrfamilienhäuser und so weiter gegliedert sein. Die oberste Elternklasse bezeichnet man als Basisklasse.

Einfamilienhäuser und Mehrfamilienhäuser haben Eigenschaften, die alle Wohnhäuser charakterisieren. Ebenso haben Wohnhäuser und Bürogebäude einige Dinge gemeinsam. Werden Klassen eingesetzt, um derartige Eltern-Kind-Beziehungen auszudrücken, erbt die Kindklasse die Eigenschaften und Methoden, die in der Elternklasse definiert werden. Das ermöglicht Ihnen, Code aus der Elternklasse wiederzuverwenden. Neuen Code müssen Sie nur schreiben, um die spezifischen Eigenschaften der neuen Kindklasse zu beschreiben. Das bezeichnet man als *Vererbung*. Diese ist einer der wichtigsten Vorteile, den Klassen im Vergleich zu einer unstrukturierten Sammlung von Funktionen bietet. Definiert man eine Kindklasse auf Basis einer Elternklasse, bezeichnet man das als Ableitung oder Erweiterung.

Klassen lassen sich in PHP leicht definieren und erstellen:

```
class Gaestebuch {
 public $kommentare;
 public $letzter_gast;

 function aktualisieren($kommentar, $gast) {
 ...
 }

}
```

Das Schlüsselwort `class` deklariert eine Klasse genau so, wie `function` eine Funktion deklariert. Eigenschaften werden mit dem Schlüsselwort `public` deklariert. Eine Methodendeklaration entspricht einer Funktionsdeklaration.

Mit dem Schlüsselwort `new` werden Objekte instantiiert:

```
$gb = new Gaestebuch;
```

Die Objektinstantiierung wird in Rezept 7.1 ausführlicher behandelt.

Optional können Sie in einer Klasse mithilfe des Schlüsselworts `public` Eigenschaften definieren. Das müssen Sie nicht tun, es ist allerdings nützlich, um die sichtbaren Eigenschaften der Klasse kenntlich zu machen. Da PHP Sie nicht zwingt, Variablen vorab zu deklarieren, können Sie Eigenschaften nachträglich in der Klassendefinition (und auch bei der Verwendung von Objekten) erzeugen, ohne dass PHP einen Fehler meldet oder Sie auf andere Weise darauf hinweist. Das kann dazu führen, dass die Variablenliste am Anfang einer Klasse irreführend ist, weil sie nicht alle Eigenschaften angibt, die es in der Klasse tatsächlich gibt.

In PHP 4 wurden Eigenschaften mit `var` statt mit `public` deklariert. Sie können `var` immer noch verwenden, aber `public` wurde in PHP 5 als Synonym eingeführt, weil PHP 5 drei verschiedene Arten von Eigenschaften kennt, die mit `public`, `protected` oder `private` deklariert werden. Mit `public` deklarierte öffentliche Eigenschaften entsprechen den aus PHP 4 bekannten Eigenschaften, aber die beiden anderen Arten verhalten sich auch anders. Das werden wir uns in Rezept 7.4 ausführlicher ansehen.

Sie können Eigenschaften bei der Deklaration zugleich auch mit einem Wert initialisieren:

```
public $letzterGast = 'Donald';
```

Als Initialisierungswerte sind allerdings nur konstante Werte erlaubt:

```
public $letzterGast = 'Donald';       // okay
public $letzterGast = 9;              // okay
public $letzterGast = array('Jesus'); // okay
public $letzterGast = gastWaehlen();  // nicht okay
public $letzterGast = 'Karl' . '9';   // nicht okay
```

Versuchen Sie, bei der Initialisierung einen anderen Wert zuzuweisen, bricht PHP mit einem Parser-Fehler ab.

Möchten Sie einer Eigenschaft einen nicht konstanten Wert zuweisen, tun Sie das aus einer Methode in der Klasse:

```
class Gaestebuch {
  public $letzterGast;

  public function aktualisieren($kommentar, $gast) {
    if (!empty($kommentar)) {
      array_unshift($this->kommentare, $kommentar);
      $this->letzterGast = $gast;
    }
  }
}
```

Wenn der Gast eine Nachricht hinterlassen hat, wird diese an den Anfang des Arrays mit den Kommentaren geschoben, und der Gast wird als letzter Besucher des Gästebuchs gesetzt. Die Variable `$this` ist eine spezielle Variable, die auf das aktuelle Objekt verweist. Wenn Sie die `$letzterGast`-Eigenschaft eines Objekts im Code der Klasse selbst referenzieren wollen, verwenden Sie also `$this->letzterGast`.

Sollen Eigenschaften bei der Instantiierung nicht konstante Werte zugewiesen werden, weisen Sie diese im Klassenkonstruktor zu. Der Konstruktor ist eine Methode, die automatisch aufgerufen wird, wenn ein neues Objekt erstellt wird. Er trägt, wie in Beispiel 7-1 gezeigt, den Namen `__construct()`.

Beispiel 7-1: Im Klassenkonstruktor Eigenschaften Werte zuweisen

```
class Gaestebuch {
  public $kommentare;
  public $letzterGast;

  public function __construct($benutzer) {
     $dbh = mysqli_connect('localhost', 'benutzername', 'passwort', 'sites');
     $user = mysqli_real_escape_string($dbh, $benutzer);
     $sql = "SELECT kommentare, letzter_gast FROM gaestebuch WHERE benutzer='$benutzer'";
     $r = mysqli_query($dbh, $sql);

     if ($obj = mysqli_fetch_object($dbh, $r)) {
       $this->kommentare = $obj->kommentare;
       $this->letzterGast = $obj->letzter_gast;
     }
  }
}

$gb = new Gaestebuch('stefan');
```

Konstruktoren behandeln wir in Rezept 7.2. Beachten Sie, dass Konstruktoren in PHP 4 den gleichen Namen wie die Klasse hatten. Hier wäre das also `Gaestebuch` gewesen.

Geben Sie acht, dass Sie nicht versehentlich `$this->$kommentare` schreiben. Das ist zwar zulässig, bewirkt aber nicht das Gleiche wie `$this->kommentare`. Stattdessen greift es auf die Eigenschaft des Objekts zu, deren Name dem in der Variablen `$kommentare` gespeicherten Wert entspricht. In der Regel wird `$kommentare` nicht definiert sein, `$this->$kommentare` also leer erscheinen. Mehr zu variablen Eigenschaftsnamen erfahren Sie in Rezept 5.4.

Sie können nicht nur `->` verwenden, um auf Methoden oder Member-Variablen zuzugreifen, Sie können auch `::` nehmen. Mit dieser Syntax greifen Sie auf die statischen Methoden in einer Klasse zu. Diese Methoden sind für alle Instanzen einer Klasse gleich, weil sie sich nicht auf instanzspezifische Daten stützen dürfen. In einer statischen Methode gibt es kein `$this`. Ein Beispiel:

```
class Konvertierer {
 // Von Celsius in Fahrenheit konvertieren.
 public static function c2f($grad) {
   return (1.8 * $grad) + 32;
 }
}
$f = Konvertierer::c2f(100); // 212
```

Vererbung durch Erweiterung einer bestehenden Klasse implementieren Sie mithilfe des Schlüsselworts `extends` :

```
class XHtml extends Xml {
  // ...
}
```

Kindklassen erben Elternmethoden und können optional eigene Versionen dieser Methoden implementieren, wie Beispiel 7-2 zeigt.

Beispiel 7-2: Elternmethoden überschreiben

```
class DB {
 public $result;

 function getResult() {
  return $this->result;
 }

 function query($sql) {
  error_log("query() muss durch eine datenbankspezifische Methode überschrieben werden.");
  return false;
 }
}

class MySQL extends DB {
 function query($sql) {
  $this->result = mysql_query($sql);
 }
}
```

Die `MySQL`-Klasse oben erbt die `getResult()`-Methode unverändert von der Elternklasse `DB`, hat aber eine eigene, MySQL-spezifische `query()`-Methode.

Stellen Sie dem Methodennamen `parent::` voran, um explizit eine Elternmethode aufzurufen, wie Sie es in Beispiel 7-3 sehen.

Beispiel 7-3: Elternmethoden explizit aufrufen

```
function escape($sql) {
 $safe_sql = mysql_real_escape_string($sql); // Sonderzeichen maskieren
 $safe_sql = parent::escape($safe_sql); // Elternmethode umschließt $sql mit ''
 return $safe_sql;
}
```

Rezept 7.18 behandelt überschriebene Methoden.

Mit Version 5.3 hat PHP noch einmal einen großen Sprung bezüglich Objektorientierung gemacht, denn seit dieser Version gibt es eine Unterstützung für Namensräume. Namensräume sind ein Konzept zur Vermeidung von Kollisionen zwischen Klassennamen. Das Konzept von Namensräumen kann man auch abstrahieren und auf andere Bereiche anwenden. In einem Dateisystem kann es zum Beispiel mehrere Dateien mit dem Namen *kochbuch.txt* geben. Solange diese Dateien in unterschiedlichen Verzeichnissen liegen, gibt es keine Namenskonflikte. Wenn Sie die Dateien aber in dasselbe Verzeichnis legen wollen, wird die bereits existierende gegebenenfalls überschrieben. Das Verzeichnis in diesem kleinen Beispiel ist mit einem Namensraum gleichzusetzen.

Namensräume ermöglichen es, dass zwei Klassen in PHP den gleichen Namen verwenden. Diese Klassen müssen nur in unterschiedlichen Namensräumen existieren. Ohne Namensräume führt die Deklaration zweier Klassen mit dem gleichen Namen zu folgender Fehlermeldung:

```
class Foo {}
class Foo {}
Fatal error: Cannot redeclare class Foo ...
```

Ab PHP 5.3 können Sie zwei Klassen mit dem Namen `Foo` deklarieren, wenn diese in unterschiedlichen Namensräumen liegen:

```
namespace de\oreilly\phpckbk {
    class Foo {}
}
namespace com\zend\framework {
    class Foo {}
}
namespace meinprojekt {
    $foo1 = new \de\oreilly\phpckbk\Foo;
    $foo2 = new \com\zend\framework\Foo;
}
```

Durch diese lang ersehnte Erweiterung der Sprachfeatures von PHP ist es fortan unnötig, lange Präfixe vor den eigentlichen Namen einer Klasse zu setzen, um Namenskonflikte zwischen eigenen und Klassen von Drittanbietern zu vermeiden. Klassennamen wie beispielsweise `Zend_Controller_Dispatcher_Standard` (eine Klasse aus dem Zend Framework) werden damit sicher bald der Vergangenheit angehören. Beim Einsatz von Namensräumen könnte man diese z.B. in den Namespace `com\zend\framework\controller` legen und `StandardDispatcher` nennen. Weitere Informationen zur Vermeidung von Namenskonflikten liefert Rezept 7.9.

Wie Sie bereits im vorherigen Beispiel sehen konnten, ist es möglich, mehrere Namensräume in einer Datei zu deklarieren. Es gibt unterschiedliche Syntaxvarianten zur Deklaration von Namespaces – ohne geschweifte Klammern:

```
namespace ProjektA;

class Foo {}
```

```
function foobar() {}

namespace ProjektB;

class Foo {}
function foobar() {}
```

und mit geschweiften Klammern:

```
namespace ProjektA {
    class Foo {}
    function foobar() {}
}

namespace ProjektB {
    class Foo {}
    function foobar() {}
}
```

Bei der Variante ohne geschweifte Klammern befindet sich sämtlicher Code, der nach einer Namensraumdeklaration folgt, im entsprechenden Namensraum, bis ein anderer deklariert wird. Bei Verwendung der geschweiften Klammern ist die Namensraumzugehörigkeit durch den Codeblock zwischen den geschweiften Klammern klar abgegrenzt. Die beiden Syntaxvarianten können nicht innerhalb einer Datei gemischt werden. Das PHP-Manual empfiehlt die Verwendung der geschweiften Klammern, die der Lesbarkeit des Quellcodes deutlich zugute kommt.

Bei der Nutzung von Namespaces muss sich aller Code in einem Namensraum befinden. Folgendes ist nicht möglich:

```
namespace ProjektA {
    class Foo {}
    function foobar() {}
}
echo "Hello World";
```

Das `echo`-Statement muss sich entweder im globalen oder einem anderen Namensraum befinden. Um die Anweisung im globalen Namensraum auszuführen, muss der Code wie folgt aussehen:

```
namespace ProjektA {
    class Foo {}
    function foobar() {}
}
namespace {
    echo "Hello World";
}
```

Weiterführende Informationen zu Namensräumen liefern die Rezepte 7.9, 7.10, 7.11 sowie das PHP-Manual unter *http://www.php.net/namespaces*.

7.1 Objekte instantiieren

Problem

Sie möchten eine neue Instanz einer Klasse erstellen.

Lösung

Definieren Sie die Klasse und nutzen Sie dann `new`, um eine Instanz der Klasse zu erstellen:

```
class Benutzer {
    function datenLaden($benutzername) {
       // Profildaten aus Datenbank laden.
    }
}
$benutzer = new Benutzer;
$benutzer->datenLaden($_GET['benutzername']);
```

Diskussion

Sie können mehrere Instanzen der gleichen Klasse instantiieren:

```
$adam = new Benutzer;
$adam->datenLaden('adam');
$david = new Benutzer;
$david->datenLaden('adam');
```

Das sind zwei unabhängige Objekte, die zufälligerweise die gleichen Daten kapseln. Sie sind so identisch wie Zwillinge: Auch wenn sie anfangs gleich sind, können sie jeweils ein eigenes Leben führen.

Sollten Sie PHP ab Version 5.3 einsetzen und Namensräume nutzen, müssen Sie bei der Instantiierung einer Klasse angeben, in welchem Namensraum diese deklariert ist. Man spricht häufig vom *voll qualifizierten Klassennamen*:

```
namespace de\oreilly\phpckbk {
    class Foo {}
}
namespace {
    $foo1 = new \de\oreilly\phpckbk\Foo;
}
```

Wenn Sie eine Klasse in dem Namensraum instantiieren wollen, in dem diese deklariert ist, müssen Sie nicht den voll qualifizierten Klassennamen verwenden:

```
namespace de\oreilly\phpckbk {
    class Foo {}
    $foo1 = new Foo;
}
```

Sie können Objekte auch dynamisch instantiieren, indem der Name der Klasse nicht zur Entwicklungszeit feststeht, sondern zur Laufzeit aus einer Variablen ausgelesen wird:

```
class Foo{}
$klassenname = 'Foo';
$foo = new $klassenname;
```

Wenn Sie mit Namensräumen arbeiten, muss die Variable `$klassenname` zusätzlich die entsprechende Namensrauminformation enthalten.

Siehe auch

In Rezept 7.13 finden Sie weitere Informationen zum Kopieren und Klonen von Objekten; die Dokumentation zu Klassen und Objekten unter *http://www.php.net/oop*.

7.2 Objektkonstruktoren definieren

Problem

Sie wollen eine Methode definieren, die aufgerufen wird, wenn ein Objekt instantiiert wird. Sie möchten beispielsweise bei der Erstellung automatisch Daten aus einer Datenbank in ein Objekt laden.

Lösung

Definieren Sie eine Methode namens `__construct( )`:

```
class Benutzer {
    function __construct($benutzername, $passwort) {
        ...
    }
}
```

Diskussion

Die Methode namens `__construct( )` (vor dem Wort `construct` stehen zwei Unterstriche) dient als Konstruktor, wie Beispiel 7-4 zeigt.

Beispiel 7-4: Einen Konstruktor definieren

```
class Benutzer {
  public $benutzername;

  function __construct($benutzername, $passwort) {
     if ($this->benutzerValidieren($benutzername, $passwort)) {
       $this->benutzername = $benutzername;
     }
  }
}

$benutzer = new Benutzer('Luis', 'Ziffer'); // den eingebauten Konstruktor verwenden
```

Unter PHP 4 trugen Konstruktoren den gleichen Namen wie die Klasse. Wie das aussah, sehen Sie in Beispiel 7-5.

Beispiel 7-5: Konstruktoren definieren unter PHP 4

```
class Benutzer {
  function Benutzer($benutzername, $passwort) {
  ...
  }
}
```

Wenn PHP 5 keinen Konstruktor namens `__construct()` findet, aber eine Methode mit dem gleichen Namen wie die Klasse vorhanden ist (die Namenskonvention für Konstruktoren unter PHP 4), nutzt es diese Methode als Konstruktor, um die Rückwärtskompatibilität zu gewährleisten.

Gibt es einen Standardnamen für Konstruktoren wie den, den PHP 5 implementiert, erleichtert das den Aufruf des Elternkonstruktors (weil man den Namen der Elternklasse nicht kennen muss). Außerdem müssen Sie den Konstruktor nicht mehr ändern, wenn Sie der Klasse einen neuen Namen geben.

Siehe auch

Rezept 7.18 bietet weitere Informationen zum Aufruf von Elternkonstruktoren; die Dokumentation zu Konstruktoren unter *http://www.php.net/oop.constructor*.

7.3 Destruktoren definieren

Problem

Sie möchten eine Methode definieren, die aufgerufen wird, wenn ein Objekt zerstört wird. Sie könnten beispielsweise automatisch Daten aus einem Objekt in einer Datenbank speichern wollen, wenn es zerstört wird.

Lösung

Objekte werden automatisch zerstört, wenn die Ausführung eines Skripts beendet wird. Erzwingen können Sie die Zerstörung eines Objekts mit `unset()`, wie Sie in Beispiel 7-6 sehen.

Beispiel 7-6: Ein Objekt löschen

```
$auto = new Auto; // neues Auto kaufen
...
unset($auto);      // Auto verschrotten
```

Soll PHP selbsttätig eine bestimmte Methode aufrufen, wenn ein Objekt zerstört wird, müssen Sie eine Methode namens `__destruct()` definieren, wie in Beispiel 7-7 gezeigt.

Beispiel 7-7: Einen Destruktor definieren

```
class Auto {
    function __destruct() {
        // den Wagen dem Schrotthändler übergeben
    }
}
```

Diskussion

In der Regel ist es nicht notwendig, Objekte manuell zu beseitigen. Aber wenn Ihr Code eine umfangreiche Schleife nutzt, kann `unset()` eingesetzt werden, damit der Speicherbedarf nicht außer Kontrolle gerät.

PHP 5 unterstützt Destruktoren. Destruktoren sind das Gegenstück zu Konstruktoren: Sie werden aufgerufen, wenn das Objekt gelöscht wird. Selbst wenn Sie das Objekt nicht selbst mit `unset()` löschen, ruft PHP den Destruktor auf, wenn es feststellt, dass das Objekt nicht mehr verwendet wird. Das könnte zu dem Zeitpunkt erfolgen, da das Skript beendet wird, aber auch schon viel früher.

Destruktoren setzen Sie ein, um hinter einem Objekt aufzuräumen. Der `Database`-Destruktor würde beispielsweise eine Verbindung zur Datenbank beenden und die für sie benötigten Ressourcen freigeben. Anders als Konstruktoren können Sie Destruktoren keine Daten übergeben, weil Sie selbst keinen Einfluss darauf haben, wann sie aufgerufen werden.

Wenn Ihr Destruktor instanzspezifische Daten benötigt, müssen Sie diese deswegen, wie Sie in Beispiel 7-8 sehen, in einer Eigenschaft speichern.

Beispiel 7-8: In einem Destruktor auf instanzspezifische Daten zugreifen

```
// Destruktor
class Database {
    function __destruct() {
         db_close($this->handle); // die Datenbankverbindung schließen
    }
}
```

Destruktoren werden ausgeführt, bevor PHP die Antwort und das Versenden der Daten abschließt. Deswegen können Sie in Destruktoren Ausgaben generieren, in eine Datenbank schreiben oder gar einen entfernten Server anpingen.

Sie können allerdings nicht voraussetzen, dass PHP Objekte in einer bestimmten Reihenfolge zerstört. Deswegen sollten Sie in Ihren Destruktoren keine anderen Objekte referenzieren, da es sein könnte, dass PHP diese bereits zerstört hat. Wenn Sie es dennoch tun,

führt das zwar nicht zu einem Programmabsturz, kann aber dafür verantwortlich sein, dass sich Ihr Code auf nicht vorhersehbare (und fehlerhafte) Weise verhält.

Bei Destruktoren gibt es keine Probleme mit der Rückwärtskompatibilität. In PHP 4 stehen sie einfach nicht zur Verfügung. Das heißt allerdings nicht, dass nicht der eine oder andere versucht hätte, die entsprechende Funktionalität mit anderen Sprachfunktionen nachzubauen. Wenn Sie Destruktoren emulieren, sollten Sie Ihren Code portieren, weil die Destruktoren von PHP 5 effizienter sind und sich leichter einsetzen lassen.

Siehe auch

Die Dokumentation zu `unset()` unter *http://www.php.net/unset*.

7.4 Zugriffskontrolle implementieren

Problem

Sie möchten Methoden und Funktionen eine bestimmte Sichtbarkeit geben, damit sie nur in Klassen zugreifbar sind, die eine bestimmte Beziehung zu dem Objekt haben.

Lösung

Nutzen Sie die Schlüsselwörter `public`, `protected` und `private`, wie in Beispiel 7-9 gezeigt.

Beispiel 7-9: Eine Klasse mit Zugriffskontrolle

```
class Person {
     public $name;     // überall zugreifbar
     protected $alter; // nur in dieser Klasse und in Kindklassen zugreifbar
     private $gehalt;  // nur in dieser Klasse zugreifbar

    public function __construct() {
     // ...
    }

    protected function setAlter() {
     // ...
    }

    private function setGehalt() {
     // ...
    }
}
```

Diskussion

PHP ermöglicht Ihnen, den Zugriff auf Methoden und Eigenschaften einzuschränken. Es gibt drei Stufen der Sichtbarkeit:

- `public` (öffentlich)
- `protected` (geschützt)
- `private` (privat)

Machen Sie eine Methode oder Eigenschaft `public`, bedeutet das, dass jeder sie aufrufen oder verändern kann. Das ist das gleiche Verhalten wie in PHP-Versionen vor PHP 5.

Sie können eine Methode oder Eigenschaft auch als `protected` markieren. Das schränkt den Zugriff auf die aktuelle Klasse und alle Kindklassen ein, die diese Klasse erweitern.

Die letzte und eingeschränkteste Sichtbarkeit ist `private`. Auf Methoden und Eigenschaften, die als `private` markiert sind, kann nur in der jeweiligen Klasse zugegriffen werden.

Wem dieses Konzept nicht vertraut ist, dem mag die Zugriffskontrolle seltsam erscheinen. Aber ihr Einsatz macht Ihren Code robuster, da er die Datenkapselung fördert, die eins der Grundprinzipien der OO-Programmierung ist.

Beim Programmieren gibt es immer irgendeinen Teil – die Art und Weise der Datenspeicherung, die Parameter, die Funktionen übernehmen, der Aufbau der Datenbank –, der nicht funktioniert, wie er sollte. Er ist zu langsam, ungeschickt oder verhindert, dass Sie eine gewünschte neue Funktionalität einbauen. Also machen Sie sich daran, diesen Code zu reparieren.

Dass Sie den Code reparieren, ist eigentlich eine gute Sache – bis zu dem Moment, an dem Sie versehentlich dabei andere Teile Ihres Systems zerbrechen. Wenn beim Entwurf eines Programms eine strenge Kapselung implementiert wurde, wird auf die zugrunde liegenden Datenstrukturen und Datenbanktabellen nicht direkt zugegriffen. Stattdessen definieren Sie einen Satz von Funktionen und leiten ihre gesamten Anfragen über diese Funktionen.

Nehmen wir beispielsweise an, Sie haben eine Datenbanktabelle, die Namen und E-Mail-Adressen festhält. Ein Programm mit ungenügender Kapselung greift an jedem Punkt direkt auf die Tabelle zu, wenn es die E-Mail-Adresse einer Person abrufen muss, wie Sie es in Beispiel 7-10 sehen können:

Beispiel 7-10: Eine E-Mail-Adresse abrufen

```
$name   = 'Rasmus Lerdorf';
$db     = mysqli_connect();
$ergebnis = mysqli_query($db, "SELECT email FROM benutzer
                           WHERE name  LIKE '$name'");
$zeile    = mysqli_fetch_assoc($db, $ergebnis);
$email  = $zeile['email'];
```

Ein Programm mit besserer Kapselung nutzt stattdessen eine Funktion:

Beispiel 7-11: Eine E-Mail-Adresse über eine Funktion abrufen

```
function getEmail($name) {
    $db = mysqli_connect();
    $ergebnis = mysqli_query($db, "SELECT email FROM benutzer
```

Beispiel 7-11: Eine E-Mail-Adresse über eine Funktion abrufen (Fortsetzung)

```
                                WHERE name  LIKE '$name'");
    $zeile    = mysqli_fetch_assoc($db, $ergebnis);
    $email  = $zeile['email'];
    return $email;
}

$email = getEmail('Rasmus Lerdorf');
```

Es bietet viele Vorteile, `getEmail()` zu nutzen. Einer davon ist, dass Sie weniger Code schreiben müssen, um eine E-Mail-Adresse abzurufen. Außerdem können Sie problemlos das Datenbankschema ändern, weil Sie dann nur die eine Abfrage in `getEmail()` anpassen und nicht sämtliche Zeilen in sämtlichen Dateien nach Stellen durchsuchen müssen, in denen mit `SELECT` Daten aus der Tabelle `benutzer` abgerufen werden.

Es ist recht schwer, mit Funktionen gut gekapselte Programme zu schreiben, da man anderen nur durch Kommentare und Programmierkonventionen mitteilen kann, dass sie von bestimmten Dingen die Finger lassen sollen.

Aber Klassen ermöglichen Ihnen, Implementierungsinterna vor Zugriffen von außen zu schützen. Das verhindert, dass sich jemand auf Code stützt, der sich ändern kann, und erzwingt, dass alle die verfügbaren Methoden nutzen, um auf die Daten zuzugreifen. Methoden dieser Art bezeichnet man als Akzessormethoden (oder Zugriffsmethoden), weil sie den Zugriff auf ansonsten geschützte Daten ermöglichen. Wenn Sie bei der Überarbeitung Ihres Codes die Akzessormethoden so überarbeiten, dass sie weiterhin funktionieren wie zuvor, wird die Funktionsfähigkeit des sie nutzenden Codes nicht beeinträchtigt.

PHP verhindert, dass von außerhalb der Klasse eine private Methode aufgerufen oder eine private Eigenschaft gelesen wird. Aus externer Perspektive könnte es diese Methoden und Eigenschaften genauso gut nicht geben, da es keinerlei Möglichkeit gibt, auf sie zuzugreifen.

Bei der objektorientierten Programmierung gibt es einen impliziten Vertrag zwischen dem Autor und den Benutzern einer Klasse. Die Benutzer versichern, dass sie sich nicht um Implementierungsdetails kümmern. Der Autor versichert, dass Code auch nach Umarbeitungen der Klasse noch funktioniert, wenn der Benutzer nur die öffentlichen Methoden nutzt.

`protected` und `private` sorgen beide dafür, dass ein Zugriff von außerhalb der Klasse unmöglich ist. Welche dieser beiden Sichtbarkeiten Sie wählen, hängt also davon ab, ob Sie erwarten, dass jemand diese Methode oder Eigenschaft aus einer Kindklasse aufrufen muss, und ob Sie den Zugriff aus einer Kindklasse zulassen wollen.

Da man eigentlich nie wissen kann, in welcher Weise Anwender eine Klasse nutzen, ist es ratsam, eher `protected` als `private` zu wählen, sofern man sich nicht völlig sicher ist, dass `private` die richtige Wahl ist und es absolut keinen Grund gibt, dass ein anderer diese Methode oder Eigenschaft je benötigt oder einsetzt.

7.5 Änderungen an Klassen und Methoden verhindern

Problem

Sie möchten verhindern, dass andere Entwickler bestimmte Methoden in einer Kindklasse neu definieren oder überhaupt Kindklassen von der Klasse ableiten.

Lösung

Markieren Sie die entsprechenden Methoden oder Klassen als `final`:

```
final public function connect($server, $username, $password) {
    // Methodendefinition
}
```

und:

```
final class MySQL {
    // Klassendefinition
}
```

Diskussion

Eigentlich ist die Vererbung eine gute Sache. Trotzdem kann es sinnvoll sein, sie einzuschränken.

Einer der besten Gründe dafür, eine Methode als `final` zu deklarieren, ist, dass tatsächlich eine Gefahr besteht, wenn jemand sie überschreibt – beispielsweise Datenkorruption, eine Race-Condition oder ein möglicher Absturz oder Aufhänger, weil jemand vergisst, einen Semaphor zu schließen (oder freizugeben).

Ein weiterer verbreiteter Grund dafür, eine Methode als `final` zu deklarieren, ist, dass die Methode »perfekt« ist. Wenn Sie der Meinung sind, dass es überhaupt keine Möglichkeit gibt, die Methode zu verbessern, deklarieren Sie sie unter Verwendung von `final`. Das verhindert, dass Unterklassen sie ruinieren, indem sie sie auf schlechtere Weise neu implementieren.

Bei einem solchen Fall sollten Sie sich allerdings genau überlegen, ob Sie tatsächlich `final` verwenden wollen. Es ist unmöglich, alle Gründe zu erwägen, die jemand haben könnte, eine Methode zu überschreiben. Wenn Sie eine Bibliothek (wie ein PEAR-Paket) veröffentlichen, könnten Sie ziemliche Probleme verursachen, wenn Sie eine Methode fälschlich als `final` markieren.

Wie Sie eine Methode final machen, indem Sie am Anfang der Methodendeklaration das Schlüsselwort `final` angeben, sehen Sie hier:

Beispiel 7-12: Eine finale Methode definieren

```
final public function connect($server, $benutzername, $passwort) {
    // Methodendefinition
}
```

Das verhindert, dass jemand, der die Klasse erweitert, eine andere `connect()`-Methode erstellt.

Wenn Sie verhindern wollen, dass eine Klasse überhaupt erweitert wird, markieren Sie nicht die einzelnen Methoden `final`, sondern machen die Klasse selbst final, wie Sie es in Beispiel 7-13 sehen.

Beispiel 7-13: Eine finale Klasse deklarieren

```
final class MySQL {
     // Klassendefinition
}
```

Eine finale Klasse kann nicht erweitert werden. Das ist etwas anderes als eine Klasse, in der alle Methoden als `final` deklariert sind, weil eine solche Klasse, auch wenn keine der bestehenden Methoden geändert werden kann, immer noch erweitert und mit zusätzlichen Methoden ausgestattet werden kann.

7.6 String-Darstellungen für Objekte definieren

Problem

Sie möchten steuern, wie PHP ein Objekt anzeigt, wenn Sie es ausgeben.

Lösung

Implementieren Sie eine `__toString()`-Methode, wie in Beispiel 7-14 gezeigt.

Beispiel 7-14: Die String-Repräsentation einer Klasse definieren

```
class Person {
     // Rest der Klasse

    public function __toString() {
        return "$this->name <$this->email>";
    }
}
```

Diskussion

PHP stattet Objekte mit einer Möglichkeit aus, ihre Konvertierung in einen String zu steuern. Das ermöglicht Ihnen, Objekte lesbar auszugeben, ohne dass Sie dazu Mengen an zusätzlichem Code schreiben müssten.

PHP ruft die `__toString()`-Methode eines Objekts auf, wenn Sie das Objekt mit `echo` oder `print` ausgeben, wie Sie es in Beispiel 7-15 sehen.

Beispiel 7-15: Die String-Repräsentation einer Klasse definieren

```
class Person {
    protected $name;
    protected $email;

    public function setName($name) {
        $this->name = $name;
    }

    public function setEmail($email) {
        $this->email = $email;
    }

    public function __toString() {
        return "$this->name <$this->email>";
    }
}
```

Dann können Sie mit Code wie diesem arbeiten:

```
$rasmus = new Person;
$rasmus->setName('Rasmus Lerdorf');
$rasmus->setEmail('rasmus@php.net');
print $rasmus;
Rasmus Lerdorf <rasmus@php.net>
```

Das veranlasst PHP, im Hintergrund die `__toString()`-Methode aufzurufen, die die String-Darstellung des Objekts zurückliefert.

Ihre Methode muss einen String zurückliefern, andernfalls meldet PHP einen Fehler. Auch wenn das hier selbstverständlich scheint, können Ihnen die Auto-Casting-Funktionen von PHP dabei gelegentlich ein Bein stellen, die sich hier nicht auswirken.

Beispielsweise ist es kein Problem, den String `'9'` und den Integer `9` auf gleiche Weise zu behandeln, weil PHP je nach Kontext nahtlos zwischen beidem wechselt und dabei fast immer die richtigen Ergebnisse liefert.

Aber aus `__toString()` dürfen Sie keine Integer zurückliefern. Befürchten Sie, dass es passieren kann, dass aus dieser Methode ein Nicht-String-Wert geliefert wird, sollten Sie überlegen, ob es nicht besser wäre, das Ergebnis, wie in Beispiel 7-16 gezeigt, explizit auf einen String zu casten.

Beispiel 7-16: Einen Rückgabewert casten

```
class TextFeld {
     // Rest der Klasse

    public function __toString() {
        return (string) $this->label;
    }
}
```

Wenn Sie `$this->label` auf einen String casten, müssen Sie sich keine Gedanken mehr machen, ob jemand diesem Textfeld ein Label mit einer Zahl gegeben hat.

In PHP-Versionen vor PHP 5.2 unterlag die `__toString()`-Funktionalität einer Reihe von Einschränkungen. Beispielsweise funktionierte sie bei interpolierten und verketteten Strings nicht (siehe Beispiel 7-17).

Beispiel 7-17: __toString() aufrufen

```
print  "PHP wurde von $rasmus geschaffen";
print  'PHP wurde von  ' . $rasmus . " geschaffen";
printf('PHP wurde von %s geschaffen', $rasmus);
```

Die einzige Ausnahme bildet eine verstaubte Ecke von PHP, die `echo` nutzt und ein Komma (,) anstelle eines Punkts (.), um Elemente zu kombinieren, wie hier gezeigt:

Beispiel 7-18: String-Darstellung von Objekten und String-Verkettung

```
echo   'PHP wurde von ', $rasmus, ' geschaffen';
PHP wurde von Rasmus Lerdorf <rasmus@php.net> geschaffen
```

Außerdem wandeln frühere Versionen von PHP 5 Objekte nicht automatisch in Strings um, wenn Sie sie an Funktionen übergeben, die String-Argumente erwarten. Sie sollten auf ihnen deswegen `__toString()` aufrufen, wie es Beispiel 7-19 zeigt.

Beispiel 7-19: __toString() direkt aufrufen

```
print htmlentities($rasmus);              // schlecht
print htmlentities($rasmus->__toString());    // gut
```

Das gilt auch, wenn Sie:

- das Objekt in doppelte Anführungszeichen oder ein Here-Dokument stecken,
- das Objekt über den Punktoperator (.) verketten,
- das Objekt mit `(string)` oder `strval()` in einen String umwandeln,
- das Objekt in `printf()` als String behandeln, indem Sie angeben, dass es mit `%s` formatiert werden soll.

Wenn Sie sich gezwungen sehen, in Ihrem Code intensiv `__toString()` zu verwenden, sollten Sie also besser PHP 5.2 oder neuer verwenden.

7.7 Interfaces definieren

Problem

Sie möchten sichern, dass eine Klasse eine Anzahl von Methoden mit bestimmten Namen, Sichtbarkeiten und Prototypen implementiert.

Lösung

Definieren Sie ein Interface (eine Schnittstelle) und deklarieren Sie, das die Klasse dieses Interface implementiert:

```
interface Benennbar {
    public function getName();
    public function setName($name);
}

class Buch implements Benennbar {
    private $name;

    public function getName() {
         return $this->name;
    }

    public function setName($name) {
         return $this->name = $name;
    }
}
```

Das Interface `Benennbar` definiert zwei Methoden, die genutzt werden können, wenn Objekte einen Namen haben. Da Bücher einen Namen haben, sagt die Klasse `Buch` mit dem Schlüsselwort `implements`, dass sie das Interface `Benennbar` implementiert, und definiert dann im Klassenbody die beiden Methoden.

Diskussion

Wenn Sie objektorientiert programmieren, müssen Ihre Objekte zusammenarbeiten. Deswegen sollte man verlangen können, dass eine Klasse (oder mehrere Klassen) Methoden implementieren, die notwendig sind, damit die Klassen in Ihrem System richtig interagieren können.

Eine E-Commerce-Anwendung muss beispielsweise bestimmte Informationen zu jeder Ware bereitstellen, die verkauft wird. Diese Waren können durch unterschiedliche Klassen repräsentiert werden: `Buch`, `CD`, `DVD` usw. Zumindest aber müssen Sie wissen, dass jede Ware in Ihrem Katalog unabhängig von ihrem spezifischen Typ einen Namen hat. (Wahrscheinlich wollen Sie, da Sie gerade dabei sind, auch dafür sorgen, dass sie einen Preis und eine ID hat.)

Den Mechanismus, über den man Klassen dazu zwingt, den gleichen Satz Methoden zu unterstützen, bezeichnet man als Interfaces. Ein Interface definieren Sie auf ähnliche Weise wie eine Klasse, wie Sie in Beispiel 7-20 sehen.

Beispiel 7-20: Ein Interface definieren

```
interface Benennbar {
    public function getName();
    public function setName($name);
}
```

Anstelle des Schlüsselworts `class` nutzen Sie bei einem Interface das Schlüsselwort `interface`. Im Interface deklarieren Sie die Prototypen der Methoden, bieten aber keine Implementierung.

Der Code oben definiert ein Interface namens `Benennbar`. Jede Klasse, die `Benennbar` implementiert, muss die beiden Methoden implementieren, die im Interface aufgeführt werden: `getName( )` und `setName( )`.

Wenn eine Klasse alle Methoden unterstützt, die das Interface deklariert, sagt man, dass sie das Interface implementiert. Dass Sie ein Interface implementieren, geben Sie in der Klassendefinition an (siehe Beispiel 7-21).

Beispiel 7-21: Ein Interface implementieren

```
class Buch implements Benennbar {
    private $name;

    public function getName() {
    return $this->name;
    }

    public function setName($name) {
    return $this->name = $name;
    }
}
```

Werden nicht alle Methoden implementiert, die im Interface aufgeführt werden, oder werden sie mit einem anderen Prototyp implementiert, meldet PHP einen fatalen Fehler.

Eine Klasse kann so viele Interfaces implementieren, wie sie will. Beispielsweise möchten Sie eventuell ein `Abspielbar`-Interface definieren, das angibt, wie man den Audioclip für etwas abruft. Die Klassen `CD` und `DVD` würden dann auch `Abspielbar` implementieren, die Klasse `Buch` nicht (es sei denn, es ist ein Hörbuch).

Wenn Sie mit Interfaces arbeiten, ist es wichtig, dass Sie Ihre Klassen deklarieren, bevor Sie Objekte instantiieren. Andernfalls kann es passieren, dass PHP 5 verwirrt ist, wenn eine Klasse ein Interface implementiert. Damit sich das nicht auf die Funktionsfähigkeit bestehender Klassen auswirkt, wird diese Anforderung nicht erzwungen. Verlassen sollten Sie sich auf dieses Verhalten aber besser nicht.

Wenn Sie prüfen wollen, ob eine Klase ein bestimmtes Interface implementiert, können Sie, wie Beispiel 7-22 zeigt, `class_implements( )` einsetzen.

Beispiel 7-22: Prüfen, ob eine Klasse ein Interface implementiert

```
class Buch implements  {
     // restlicher Code
}

$interfaces = class_implements('Buch');
```

Beispiel 7-22: Prüfen, ob eine Klasse ein Interface implementiert (Fortsetzung)

```
if (isset($interfaces['Benennbar'])) {
    // Buch implementiert Benennbar
}
```

Sie können auch wie in Beispiel 7-23 die Reflection-Klassen nutzen.

Beispiel 7-23: Mit den Reflection-Klassen prüfen, ob eine Klasse ein Interface implementiert

```
class Buch implements Benennbar {
     // restlicher Code
}
$rc = new ReflectionClass('Buch');
if ($rc->implementsInterface('Benennbar')) {
  print "Buch implementiert Benennbar";
}
```

Siehe auch

Rezept 7.23 bietet weitere Informationen zu den Reflection-Klassen; Dokumentationen zu `class_implements()` unter *http://www.php.net/class_implements* und zu Interfaces unter *http://www.php.net/interfaces*.

7.8 Eine abstrakte Basisklasse definieren

Problem

Sie möchten eine »abstrakte« Basisklasse erstellen, d.h. eine Klasse, die nicht direkt instantiiert werden kann, aber als gemeinsame Basis für Kindklassen dient.

Lösung

Markieren Sie die Klasse mit `abstract`:

```
abstract class Database {
  // ...
}
```

Geben Sie also vor der Klassendeklaration das Schlüsselwort `abstract` an.

Außerdem müssen Sie in Ihrer Klasse mindestens eine abstrakte Methode deklarieren. Das tun Sie, indem Sie der Methodendeklaration das Schlüsselwort `abstract` voranstellen:

```
abstract class Database {
    abstract public function connect();
    abstract public function query();
    abstract public function fetch();
    abstract public function close();
    public function setOption();
    public function getOption();}
```

Diskussion

Abstrakte Klassen sind am geeignetsten, wenn man eine Gruppe von Objekten verwendet, die in einer »Ist-ein-Beziehung« stehen. Es ist also logisch, sie von einem gemeinsamen Elter abstammen zu lassen. Aber während die Kinder greifbar sind, ist der Elter abstrakt.

Nehmen wir beispielsweise eine `Database`-Klasse. Da eine Datenbank ein reales Objekt ist, ist eine `Database`-Klasse sinnvoll. Aber obwohl es Oracle, MySQL, Postgres, MSSQL und Hunderte anderer Datenbanken gibt, können Sie keine generische Datenbank herunterladen und installieren. Sie müssen sich für eine bestimmte Datenbank entscheiden.

PHP bietet Ihnen ein Mittel, Klassen zu erstellen, die nicht instantiiert werden können. Eine solche Klasse bezeichnet man als abstrakt. Schauen Sie sich beispielsweise die Klasse `Database` in folgendem Beispiel an:

Beispiel 7-24: Eine abstrakte Klasse definieren

```
abstract class Database {
    abstract public function connect();
    abstract public function query();
    abstract public function fetch();
    abstract public function close();
    public function setOption();
    public function getOption();}
}
```

Sie machen eine Klasse abstrakt, indem Sie vor `class` das Schlüsselwort `abstract` angeben.

Abstrakte Klassen müssen mindestens eine Methode enthalten, die ebenfalls mit `abstract` markiert ist. Derartige Methoden bezeichnet man als abstrakte Methoden. `Database` enthält vier abstrakte Methoden: `connect()`, `query()`, `fetch()` und `close()`. Diese vier Methoden repräsentieren Grundfunktionalitäten, die für die Verwendung einer Datenbank erforderlich sind.

Wenn eine Klasse eine abstrakte Methode enthält, muss die Klasse ebenfalls abstrakt deklariert sein. Abstrakte Klassen können allerdings nicht abstrakte Methoden enthalten. In `Database` sind das die Methoden `getOption()` und `setOption()`.

Abstrakte Methoden werden wie die in einem Interface aufgeführten Methoden nicht in der abstrakten Klasse implementiert. Stattdessen werden abstrakte Methoden in den Kindklassen implementiert, die die abstrakte Elternklasse erweitern. Das könnte beispielsweise aussehen wie bei der `MySQL`-Klasse in Beispiel 7-23.

Beispiel 7-25: Eine auf einer abstrakten Klasse basierende Klasse implementieren

```
class MySQL extends Database {
    protected $dbh;
    protected $query;

    public function connect($server, $benutzername, $passwort, $datenbank) {
        $this->dbh = mysqli_connect($server, $benutzername,
```

Beispiel 7-25: Eine auf einer abstrakten Klasse basierende Klasse implementieren (Fortsetzung)

```
                                        $password, $datenbank);
        }

        public function query($sql) {
            $this->query = mysqli_query($this->dbh, $sql);
        }

        public function fetch() {
            return mysqli_fetch_row($this->dbh, $this->query);
        }

        public function close() {
            mysqli_close($this->dbh);
        }
    }
```

Wenn eine Kindklasse eine der abstrakten Methoden in der Elternklasse nicht implementiert, ist sie selbst abstrakt. Dann brauchen Sie noch eine weitere Klasse, die diese erste Kindklasse erweitert. Sie könnten so vorgehen, wenn Sie beispielsweise zwei MySQL-Klassen erstellen wollen: eine, die Daten als Objekte abruft, und eine, die Daten als Arrays liefert.

Es gibt zwei Anforderungen für abstrakte Methoden:

- Abstrakte Methoden können nicht `private` sein, weil sie vererbt werden müssen.
- Abstrakte Methoden dürfen nicht `final` sein, weil sie überschrieben werden müssen.

Abstrakte Klassen und Interfaces sind ähnliche, aber keine identischen Konzepte. Beispielsweise können Sie mehrere Interfaces implementieren, aber nur eine abstrakte Klasse erweitern. Außerdem können Sie in einem Interface nur die Prototypen von Methoden deklarieren, die Methoden selbst aber nicht implementieren. Eine abstrakte Klasse hingegen muss nur eine abstrakte Methode haben und kann daneben beliebig viele nicht abstrakte Methoden oder gar Eigenschaften haben.

Abstrakte Klassen sollten Sie verwenden, wenn eine »Ist-ein-Regel« anwendbar ist. Da man »MySQL ist eine Datenbank« sagen kann, ist es sinnvoll, dass die Klasse `Database` abstrakt ist. Weil man aber nicht »Buch ist ein Benennbares« oder »Buch ist ein Name« sagen kann, sollte `Benennbar` ein Interface sein.

7.9 Mit Namespaces Kollisionen zwischen Klassennamen verhindern

Problem

Sie nutzen in Ihrer Applikation diverse Bibliotheken oder Frameworks und wollen Konflikte zwischen eigenen Klassen und Framework-Klassen mit identischen Namen vermei-

den. Sie möchten dabei aber nicht den in PHP häufig angewandten Kompromiss eingehen und den Klassennamen verkomplizieren (z.B. `MeinProjekt_MeineTeilanwendung_MeineKlasse`).

Lösung

Durch die Verwendung von Namespaces, die ab PHP 5.3 zur Verfügung stehen, können Sie eine saubere Trennung der Namensräume erreichen, ohne den Klassennamen künstlich zu verkomplizieren. Das folgende Codebeispiel zeigt, wie das aussehen könnte:

```
namespace net\php\pear\Date {
    class DateTime {
        public function __construct() {
            print __CLASS__ . " created\n";
        }
    }
}

namespace de\oreilly\phpckbk {
    class DateTime {
        public function __construct() {
            print __CLASS__ . " created\n";
        }
    }


    $oreillyDateTime    = new DateTime();

    use net\php\pear\Date;
    $pearDateTime       = new Date\DateTime();

    $phpDateTime        = new \DateTime();
    print_r($phpDateTime);
}
```

Wie Sie sehen, werden im gezeigten Codebeispiel drei Klassen mit identischem Namen (`DateTime`) verwendet. Wir selbst bewegen uns im Namensraum `de\oreilly\phpckbk` und haben dort die Klasse `DateTime` definiert. Diese kann also ohne weitere Namensraumangabe instantiiert werden. Durch Angeben des entsprechenden Namensraums können wir genauso einfach die PEAR-Klasse `DateTime` (Namespace `net\php\pear\Date`[1]) wie auch die von *ext/date* im globalen Namensraum bereitgestellte Klasse `DateTime` nutzen. Das angeführte Skript läuft in PHP 5.3 ohne Fehler durch und erzeugt die folgende Ausgabe:

```
de\oreilly\phpckbk\DateTime created
net\php\pear\Date\DateTime created
DateTime Object
(
```

1 Wir tun hier so, als würde das PEAR-Paket *PEAR::Date* bereits von Namespace-Features Gebrauch machen.

```
    [date] => 2009-04-26 13:45:01
    [timezone_type] => 3
    [timezone] => Europe/Berlin
)
```

Diskussion

In PHP-Versionen vor 5.3 gab es keine Unterstützung für Namensräume. Um Konflikte zwischen Klassennamen zu vermeiden, wichen Entwickler in der Regel auf unbequem lange Klassennamen aus. So heißt im Zend Framework eine Klasse `Zend_Controller_Dispatcher_Standard`. Beim Einsatz von PHP-Namespaces könnte man diese z.B. in den Namensraum `com\zend\framework\controller` legen und `StandardDispatcher` nennen.

Dass auch die umständliche Benennung von Klassen nicht die perfekte Lösung ist, zeigte sich im November 2005, als das Release von PHP 5.1.0 die Klasse `Date` der Erweiterung *ext/date* einführte. Das zog einen Namenskonflikt mit der gleichnamigen Klasse `Date` aus dem *PEAR::Date*-Paket nach sich, und alle Skripten, die das PEAR-Paket nutzten, funktionierten auf einen Schlag nicht mehr. Die Verwendung von Namensräumen löst dieses Problem.

Siehe auch

Es gibt keine einheitliche Regelung für die Benennung von Namensräumen. In XML-Sprachen verwendet man üblicherweise URLs (siehe *http://de.wikipedia.org/wiki/Liste_der_XML-Namensräume*). In der Programmiersprache Java befinden sich alle Klassen im Namensraum *java.**, in Open Source-Projekten werden oft URLs verwendet, jedoch in umgekehrter Reihenfolge. Die Volltext-Suchmaschine Lucene benutzt als Namensraum z.B. *org.apache.lucene.**. Letztendlich liegt die Wahl des Namensraums bei Ihnen, die Verwendung einer URL als Namensraum hat jedoch den Vorteil, dass die Verbindung zwischen Ihrem Quellcode und der zugehörigen Webseite hergestellt werden kann (z.B. `net\php\pear` für *http://pear.php.net*).

7.10 Namespace-Aliase – weniger Tipparbeit bei Verwendung von Namensräumen

Problem

Sie nutzen Klassen aus unterschiedlichen Namensräumen (ab PHP 5.3). Sie wollen sich Tipparbeit ersparen und beim Instantiieren von Klassen nicht immer den voll qualifizierten Klassennamen angeben.

Lösung

Mithilfe des use-Statements können Sie Namensräume oder Klassen in Ihren Namensraum importieren. Ein langer Name eines zu importierenden Namensraums kann so auf einen kurzen Alias gemappt werden:

```
namespace de\oreilly\phpckbk {
    class Foo {
        public function hallo() {
            echo 'Hallo Welt';
        }
    }
}

namespace {
    use de\oreilly\phpckbk;
    $foo = new phpckbk\Foo();
    $foo->hallo();
}
```

Durch den Import von `de\oreilly\phpckbk` in den globalen Namensraum kann die Klasse `Foo` durch `new phpckbk\Foo` anstelle von `new \de\oreilly\phpckbk\Foo` instantiiert werden. Die Zeitersparnis bei der Tipparbeit kann je nach Länge des importierten Namespace-Namens signifikant sein.

Diskussion

Es gibt zwei Möglichkeiten des Importierens/Aliasings[2]: Sie können einen Alias für einen Namensraum oder für einen Klassennamen festlegen. Beide Varianten erlauben es, einen konkreten Alias festzulegen oder den ursprünglichen Namen als Alias zu verwenden. Folgende Zeilen für den Import eines Namensraums haben das gleiche Ergebnis:

```
use de\oreilly\phpckbk;
use de\oreilly\phpckbk as phpckbk;
```

In beiden Fällen können Sie danach durch `$x = new phpckbk\Klasse();` eine beliebige Klasse aus dem importierten Namensraum instantiieren. Bezogen auf den Import von Klassen, haben die folgenden zwei Zeilen die gleiche Wirkung:

```
use de\oreilly\phpckbk\Klasse;
use de\oreilly\phpckbk\Klasse as Klasse;
```

Nach dem so erfolgten Import in Ihren Namensraum können Sie `Klasse` durch `$k = new Klasse();` instantiieren. Sie müssen als Alias natürlich nicht den ursprünglichen Namensraum- oder Klassennamen beibehalten, sondern können genauso gut einen eigenen Namen vergeben:

2 Die Begriffe Importieren und Aliasing werden von den Autoren synonym verwendet. Ebenso wird dies im PHP-Manual gehandhabt.

```
use de\oreilly\phpckbk\KlasseFoo as KlasseBar;
$k = new KlasseBar;
```

Sollten Sie mehrere Importe durchführen wollen, müssen Sie nicht jeden Import in eine eigene Zeile schreiben, sondern können die Import-Statements gruppieren:

```
use de\oreilly\phpckbk\Foo as Bar, de\oreilly\phpckbk\anothernamespace;
```

Zudem ist es wichtig zu wissen, dass die Auflösung von Aliasen von PHP zur Kompilierzeit vorgenommen wird. Das bedeutet, dass sich diese Aliase nicht auf dynamische Funktions-, Klassen- oder Konstantennamen auswirken:

```
use de\oreilly\phpckbk\x\Foobar as Foo;
$klasse = 'Foo';
$foo = new Foo; // funktioniert
$foo = new $klasse; // Fatal error: Class 'Foo' not found
```

Siehe auch

Weitere Informationen zum Aliasing finden Sie im PHP-Manual unter *http://php.net/manual/language.namespaces.importing.php* und zu Namensräumen allgemein unter *http://php.net/namespaces*.

7.11 Funktionen und Klassen aus dem globalen Namensraum verwenden

Problem

In Ihrem Programm machen Sie vom Namespace-Feature Gebrauch (ab PHP 5.3). Sie haben eine Funktion geschrieben, die auch PHP im globalen Namensraum unter dem gleichen Funktionsnamen bereitstellt. Sie wollen nun diese globale Funktion und nicht die in Ihrem Quellcode deklarierte lokale Funktion aufrufen.

Lösung

Wenn Sie einem Namen (z.B. einem Funktions- oder Klassennamen) einen Backslash voranstellen, wird der Name im globalen Namensraum gesucht, unabhängig davon, in welchem Subnamespace sich Ihr Code befindet:

```
namespace de\oreilly\phpckbk {
    function strlen($str) {
        return 99;
    }
    print strlen('Hallo Welt!');
    print \strlen('Hallo Welt!');
}
99
11
```

Diskussion

In obigem Beispiel wird im Namensraum `de\oreilly\phpckbk` eine Funktion `strlen()` deklariert, die immer den ganzzahligen Wert `99` zurückgibt. Diese Funktion existiert auch im globalen Namensraum von PHP. Rufen Sie innerhalb des Namensraums `de\oreilly\phpckbk` die Funktion `strlen()` auf, führt PHP die dort deklarierte eigene Funktion aus. Stellen Sie dem Funktionsnamen einen Backslash voran, wird die von PHP global deklarierte Funktion `strlen()` ausgeführt.

Wenn Sie sich im globalen Namensraum befinden oder ohne Namespaces arbeiten, ist es egal, ob Sie einem Namen einen Backslash voranstellen. Es wird immer die PHP-eigene Funktion `strlen()` aufgerufen:

```
print strlen('Hallo Welt!') . "\n";
print \strlen('Hallo Welt!') . "\n";
11
11
```

Mit diesem Wissen im Hinterkopf können Sie in Ihrem eigenen Namensraum Funktionen schreiben, die wiederum auf die globalen Funktionen von PHP aufsetzen:

```
namespace de\oreilly\phpckbk {
    function strlen($str) {
        return \strlen($str) + 1;
    }
}
```

Siehe auch

Informationen zu Namensräumen liefert das PHP-Manual unter *http://php.net/namespaces*.

7.12 Objektreferenzen zuweisen

Problem

Sie möchten zwei Variablen auf das gleiche Objekt zeigen lassen, damit Änderungen, die Sie über die eine Referenz vornehmen, von der anderen gespiegelt werden.

Lösung

Nutzen Sie `=`, um einer Variablen per Referenz das Objekt zuzuweisen, auf das die andere Variable zeigt:

```
$adam = new Benutzer;
$david = $adam;
```

Diskussion

Wenn Sie ein Objekt mit = zuweisen, wird keine neue Kopie eines Objekts erstellt. Stattdessen greift die neue Variable auf das gleiche Objekt zu wie die andere Variable. Eine Änderung des Objekts über die eine Variable wird also von der anderen Variablen widergespiegelt.

Das unterscheidet sich erheblich davon, wie PHP 5 Variablen behandelt, die andere Datentypen festhalten. Wird eine Variable, die einen Nicht-Objekttyp festhält, einer anderen Variablen zugewiesen oder einer Methode/Funktion übergeben, wird einfach eine Kopie des Werts zugewiesen oder übergeben. Es unterscheidet sich ebenfalls von der Behandlung von Variablen unter PHP 4: Dort wird bei alle Arten von Variablen, egal welchen Typ Wert sie festhalten, der Wert kopiert.

Wo Sie in PHP 4 =& einsetzen mussten, um zwei Variablen auf das gleiche Objekt zeigen zu lassen, müssen Sie jetzt also nur noch = nutzen:

```
$adam = new Benutzer;
$adam->datenLaden('adam');

$david = $adam;
```

`$david` und `$adam` sind also jetzt zwei Variablen, die als separate Referenzen auf ein und dasselbe Objekt dienen.

Siehe auch

Rezept 7.13 enthält mehr zum Klonen von Objekten; Dokumentation zu Referenzen unter *http://www.php.net/references*.

7.13 Objekte klonen

Problem

Sie möchten ein Objekt kopieren.

Lösung

Kopieren Sie Objekte per Referenz mit =:

```
$rasmus = $zeev;
```

Klonen Sie Objekte per Wert (by value) mit `clone`:

```
$rasmus = clone $zeev;
```

Diskussion

PHP 5 kopiert Objekte per Referenz, nicht per Wert. Weisen Sie ein durch eine Variable festgehaltenes, bestehendes Objekt einer weiteren Variablen zu, ist diese neue Variable nichts anderes als ein neuer Name für das bestehende Objekt. Ob Sie den alten oder den neuen Namen für den Zugriff auf das Objekt einsetzen, ist egal: Sie erhalten immer das gleiche Ergebnis.

Wenn Sie eine neue, unabhängige Instanz des Objekts mit dem gleichen Inhalt erstellen wollen – auch als Klonen bezeichnet –, müssen Sie das Schlüsselwort `clone` verwenden. Ansonsten greift die zweite Variable einfach auf das gleiche Objekt zu wie die erste.

Beim Klonen werden alle Eigenschaften des ursprünglichen Objekts in das zweite kopiert, Eigenschaften, die auf Objekte zugreifen, eingeschlossen. Es kann also passieren, dass das geklonte Objekt die gleichen Objektreferenzen verwendet wie das Original.

Häufig ist dieses Verhalten nicht erwünscht. Betrachten Sie beispielsweise die zusammengesetzte Version von `Person` in Beispiel 7-26, die ein `Adresse`-Objekt festhält.

Beispiel 7-26: Eine zusammengesetzte Klasse verwenden

```
class Adresse {
  protected $stadt;
  protected $staat;

  public function setStadt($Stadt) { $this->stadt = $stadt; }
  public function getStadt() { return $this->stadt; }
  public function setStaat($staat) { $this->staat = $staat; }
  public function getStaat() { return $this-> staat;}
}

class Person {
 protected $name;
 protected $adresse;

 public function __construct() { $this->adresse = new Adresse; }
 public function setName($name) { $this->name = $name; }
 public function getName() { return $this->name; }
 public function __call($methode, $argumente) {
  if (method_exists($this->adresse, $methode)) {
   return call_user_func_array( array($this->adresse, $methode), $argumente);
  }
 }
}
```

Eine zusammengesetzte Klasse ist eine Klasse, die eine andere Klasse so einbettet, dass man leicht auf die ursprüngliche und die eingebettete Klasse zugreifen kann. Das Wichtigste ist, sich zu merken, dass die Eigenschaft `$adresse` ein `Adresse`-Objekt festhält.

Beispiel 7-27 zeigt unter Verwendung dieser Klasse, was beim Klonen eines Objekts passiert.

Beispiel 7-27: Eine zusammengesetzte Klasse klonen

```
$rasmus = new Person;
$rasmus->setName('Rasmus Lerdorf');
$rasmus->setStadt('Sunnyvale');

$zeev = clone $rasmus;
$zeev->setName('Zeev Suraski');
$zeev->setStadt('Tel Aviv');

print $rasmus->getName() . ' lebt in ' . $rasmus->getStadt() . '.';
print $zeev->getName() . ' lebt in ' . $zeev->getStadt() . '.';

Rasmus Lerdorf lebt in Tel Aviv.

Zeev Suraski lebt in Tel Aviv.
```

Interessant. Der Aufruf von `setName( )` funktionierte wie erwartet, da die Eigenschaft `$name` ein String ist und folglich als Wert kopiert wird. Aber `$adresse` wird per Referenz kopiert, da es ein Objekt ist. `getStadt( )` liefert also nicht das richtige Ergebnis und lässt Rasmus nach Tel Aviv umziehen.

Einen derartigen Klon eines Objekts bezeichnet man als flachen Klon oder flache Kopie. Den Gegensatz dazu, einen »tiefen Klon«, erhält man, wenn alle beteiligten Objekte geklont werden. Auf diese Weise klont PHP 4.

Wie PHP 5 Objekte klont, können Sie steuern, indem Sie in Ihrer Klasse eine `__clone( )`-Methode implementieren. Wenn diese Methode existiert, überschreibt sie das PHP-Standardverhalten beim Klonen, wie Sie in Beispiel 7-28 sehen.

Beispiel 7-28: Tiefes Klonen für zusammengesetzte Klassen implementieren

```
class Person {
 // ... alles wie zuvor
 public function __clone() {
  $this->adresse = clone $this->adresse;
 }
}
```

In `__clone( )` steht Ihnen über `$this` automatisch eine flache Kopie des Objekts zur Verfügung, das dem entspricht, was PHP liefert, wenn es kein `__clone( )` gibt.

Da PHP bereits alle Eigenschaften kopiert hat, müssen Sie nur die überschreiben, die Ihren Anforderungen nicht genügen. Hier ist `$name` okay, während `$adresse` explizit geklont werden muss.

Anschließend verhalten sich die Klone wie gewünscht, wie nachfolgend gezeigt.

Beispiel 7-29: Eine zusammengesetzte Klasse klonen

```
$rasmus = new Person;
$rasmus->setName('Rasmus Lerdorf');
$rasmus->setStadt('Sunnyvale');
```

Beispiel 7-29: Eine zusammengesetzte Klasse klonen (Fortsetzung)

```
$zeev = clone $rasmus;
$zeev->setName('Zeev Suraski');
$zeev->setStadt('Tel Aviv');

print $rasmus->getName() . ' lebt in ' . $rasmus->getStadt() . '.';
print $zeev->getName() . ' lebt in ' . $zeev->getStadt() . '.';

Rasmus Lerdorf lebt in Sunnyvale.

Zeev Suraski lebt in Tel Aviv.
```

Wenn Sie den `clone`-Operator auf Objekte anwenden, die in Eigenschaften gespeichert sind, prüft PHP, ob diese Objekte eine `__clone()`-Methode anbieten. Gibt es eine, ruft PHP diese auf. Das wiederholt PHP für alle noch weiter geschachtelten Objekte.

Mit diesem Verfahren wird das vollständige Objekt geklont, und es erklärt auch, warum man das Ergebnis als eine tiefe Kopie bezeichnet.

Siehe auch

In Rezept 7.12 finden Sie mehr zur Zuweisung von Objekten per Referenz.

7.14 Callback-Funktionen mit einem Zustandsgedächtnis programmieren

Problem

Sie benötigen eine Callback-Funktion. Eine einfache Funktion (z.B. ein Closure oder eine Lambda-Funktion) reicht aber für Ihre Zwecke nicht aus, da die Callback-Funktion einen eigenen Zustand halten – also ein Zustandsgedächtnis haben – soll.

Lösung

Seit PHP 5.3 haben Sie die Möglichkeit, von Funktionsobjekten Gebrauch zu machen. Dabei handelt es sich um Objekte, die wie Funktionen behandelt und demnach also auch wie Funktionen aufgerufen werden können:

```
class Funktor {
    public function __invoke($wert) {
        print $wert . "\n";
    }
}
$f = new Funktor;
$f('Hallo Welt!');
Hallo Welt!
```

Diskussion

In Rezept 6.12 haben Sie gesehen, wie Lambda-Funktionen verwendet werden können, um z.B. ein Array mit der Funktion `usort()` zu sortieren. Diese Funktionen werden oft als Callback-Funktionen oder kurz Callbacks bezeichnet. Ein solches Callback übergibt man an eine beliebige Funktion, und diese ruft es mit diversen Parameterwerten auf. Die `usort()`-Funktion ruft ein Callback beispielsweise mit zwei Werten auf, die miteinander verglichen werden sollen. Die Callback-Funktion ist dabei nicht in der Lage, einen Zustand über mehrere Aufrufe hinweg zu halten. Einfach formuliert, könnte man sagen, sie hat kein »Gedächtnis«. Das an `usort()` übergebene Callback weiß nicht, welche Werte es im vorherigen Aufruf verglichen hat.

PHP bietet ab Version 5.3 mit den Funktoren die Möglichkeit, Callbacks zu implementieren, die in der Lage sind, einen Zustand zu halten. Ein Funktor (auch Funktionsobjekt genannt) ist eigentlich keine Methode, sondern eine vollwertige Klasse, die die magische Methode `__invoke()` implementiert. Durch die Implementierung dieser Methode ist es möglich, ein Objekt der Klasse wie eine normale Funktion aufzurufen. Demnach kann man ein solches Funktionsobjekt auch problemlos an Funktionen wie `usort()` oder `array_map()` übergeben, die eine Callback-Funktion als Parameter erwarten. Im Vergleich zu einer einfachen Callback-Funktion kann ein Funktionsobjekt allerdings einen internen Zustand haben, der zwischen den einzelnen Callback-Aufrufen erhalten bleibt:

```
$ar = array(3, 21, 9);
class Funktor {
    private $summe;
    public function __invoke($wert) {
        $this->summe += $wert;
        print $wert;
    }
    public function summe() {
        return $this->summe;
    }
}
$f = new Funktor;
array_map($f, $ar);
echo $f->summe();
3
21
9
33
```

Das Beispiel zeigt, wie das Funktionsobjekt `$f` als Callback an `array_map()` übergeben wird und jeden einzelnen Array-Wert ausgibt. Die Werte werden aber nicht nur ausgegeben, sondern der Funktor addiert bei jedem Aufruf den übergebenen Wert zur `$summe`, die als Instanzvariable angelegt ist. Nachdem `array_map()` beendet ist, können Sie über die Methode `summe()` des Funktionsobjekts die Summe der einzeln übergebenen Werte abfragen. Die Ausgabe des Werts `33` zeigt, dass das Funktionsobjekt den Zustand korrekt über die einzelnen Aufrufe hinweg transportiert hat.

Die Anzahl der Parameter, die beim Aufruf eines Funktionsobjekts übergeben werden können bzw. müssen, hängt davon ab, wie die `__invoke()`-Methode implementiert ist. Wenn Sie `__invoke()` so deklarieren, dass diese drei Parameter erwartet, müssen beim Aufruf des Funktionsobjekts natürlich auch drei Werte übergeben werden:

```
class Funktor {
    public function __invoke($p1, $p2, $p3) {
        var_dump($p1);
        var_dump($p2);
        var_dump($p3);
    }
}
$f = new Funktor;
$f('foo', 3, false);
}
string(3) "foo"
int(3)
bool(false)
```

Siehe auch

Informationen zur magischen `__invoke()`-Methode finden Sie im PHP-Manual unter *http://php.net/manual/language.oop5.magic.php#language.oop5.magic.invoke*, zu `usort()` unter *http://php.net/manual/function.usort.php* und zu `array_map()` unter *http://php.net/manual/function.array-map.php*.

7.15 Den Zugriff auf Eigenschaften abfangen

Problem

Sie möchten, dass bestimmte Handler-Methoden ausgeführt werden, wenn Eigenschaften gelesen oder geschrieben werden. Das ermöglicht Ihnen, allgemeinen Code für den Eigenschaftszugriff in Ihrer Klasse zu schreiben.

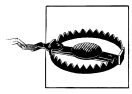
Lösung

Fangen Sie den Eigenschaftszugriff mit den magischen Methoden `__get()` und `__set()` ab. Diese Methoden werden oft auch als Interzeptor-Methoden bezeichnet.

Diese Abstraktion können Sie noch verbessern, indem Sie ebenfalls die Methoden `__isset()` und `__unset()` implementieren, damit sich die Klasse korrekt verhält, wenn Sie mit `isset()` eine Eigenschaft prüfen oder mit `unset()` löschen.

Diskussion

Das Abfangen des Zugriffs auf Eigenschaften ermöglicht Ihnen, den tatsächlichen Ort der Eigenschaften Ihres Objekts und die zu ihrer Speicherung eingesetzten Datenstrukturen nahtlos zu verbergen.



Wenn man im Kontext von PHP über Interzeptor-Methoden liest, kommt es vor, dass vom »Überladen von Eigenschaften« geschrieben wird. Der Begriff »Überladen« wird in einigen Programmiersprachen anders verwendet als in PHP. In Java spricht man oft vom Überladen von Methoden. Das bedeutet, dass es erlaubt ist, zwei unterschiedliche Methoden mit demselben Namen zu definieren. Der Compiler unterscheidet diese Methoden anhand ihrer Signatur (Methodenname und Parameterliste). In PHP führt die Deklaration zweier Funktionen oder Methoden mit demselben Namen zu einem Fehler (*Fatal error: Cannot redeclare ...*).

Die in Beispiel 7-30 präsentierte `Person`-Klasse speichert Variablen im Array `$__daten`.

Beispiel 7-30: Die magische Getter-/Setter-Methoden implementieren

```
class Person {
    private $__daten = array();

    public function __get($eigenschaft) {
        if (isset($this->__daten[$eigenschaft])) {
            return $this->__daten[$eigenschaft];
        } else {
            return false;
        }
    }

    public function __set($eigenschaft, $wert) {
        $this->__daten[$eigenschaft] = $wert;
    }
}
```

Beispiel 7-31 zeigt, wie man die Klasse `Person` verwendet.

Beispiel 7-31: Die magischen Getter-/Setter-Methoden überschreiben

```
$johnwood = new Person;
$johnwood->email = 'jonathan@wopr.mil'; // setzt $user->__daten['email']
print $johnwood->email;                 // liest $user->__daten['email']
jonathan@wopr.mil
```

Beim Setzen von Daten ändert `__set()` das entsprechende Element in `$__daten`. `__get()` wird auf parallele Weise eingesetzt, um den Eigenschaftsaufruf abzufangen und das entsprechende Array-Element zu liefern.

Nutzt man diese Methoden und ein Array als alternative Speicherquelle für Daten, erleichtert das die Implementierung von Objektkapselung. Sie müssen nicht mehr ein Paar von Zugriffsmethoden für jede Eigenschaft schreiben, sondern verwenden stattdessen `__get()` und `__set()`.

Wenn Sie `__get()` und `__set()` einsetzen, können Sie Dinge wie `$johnwood->name` verwenden, die öffentliche Eigenschaften zu sein scheinen, ohne dabei die Kapselung zu verlet-

zen. Das liegt daran, dass die Eigenschaften nicht direkt gelesen und geschrieben werden, sondern der Zugriff über die Zugriffsmethoden abgewickelt wird.

`__get()` erhält als einziges Argument den Namen der Eigenschaft. In der Methode selbst prüfen Sie dann, ob diesem Eigenschaftsnamen ein Wert in `$__daten` entspricht. Ist das der Fall, liefert die Methode diesen Wert zurück, andernfalls false.

Wenn Sie `$johnwood->name` lesen, rufen Sie eigentlich `__get('name')` auf und erhalten `$__daten['name']`, aber aus externer Perspektive ist das irrelevant.

`__set()` erhält zwei Argumente: den Namen der Eigenschaft und den neuen Wert. Ansonsten ist die Logik in der Methode ähnlich wie in `__get()`.

Diese magischen Methoden reduzieren nicht nur die Anzahl von Methoden in Ihren Klassen, sie vereinfachen es auch, eine Zentralstelle für die Validierung von Aus-/Eingaben zu implementieren.

Beispiel 7-32 zeigt zusätzlich, wie Sie exakt erzwingen können, welche Eigenschaften für eine bestimmte Klasse zulässig und welche unzulässig sind.

Beispiel 7-32: Eigenschaftszugriff mit den magischen Zugriffsmethoden erzwingen

```
class Person {
    // name und email als gültige Eigenschaften vordefinieren.
    protected $__daten = array('name', 'email');

    public function __get($eigenschaft) {
        if (isset($this->__daten[$eigenschaft])) {
            return $this->__daten[$eigenschaft];
        } else {
            return false;
        }
    }

     // Erzwingen, dass nur die vordefinierten Eigenschaften
     // gesetzt werden können.
    public function __set($eigenschaft, $wert) {
        if (array_key_exists($eigenschaft, $this->__daten)) {
            $this->__daten[$eigenschaft] = $wert;
        }
    }
}
```

In dieser aktualisierten Version des Codes führen Sie explizit die gültigen Eigenschaftsnamen an, wenn Sie die Eigenschaft `$__daten` definieren. In `__set()` nutzen Sie dann `isset()`, um zu prüfen, ob tatsächlich versucht wird, eine der zulässigen Eigenschaften zu setzen.

`$__daten` ist `protected`, nicht `public`, damit keine unzulässigen Lese- und Schreiboperationen auf Eigenschaften möglich sind. Wäre das nicht der Fall, könnte jemand Folgendes machen:

```
$person = new Person;
$person->__daten['beliebige_eigenschaft'] = 'beliebige_daten';
```

Der Grund dafür ist, dass die magischen Zugriffsmethoden für bestehende Eigenschaften nicht verwendet werden.

Achten Sie auf diese wichtigen Implementierungsdetails. Wenn Sie davon ausgehen, dass andere Ihre Klasse erweitern, müssen Sie berücksichtigen, dass sie eine Eigenschaft einführen könnten, die mit einer Eigenschaft kollidiert, die bei Ihnen durch `__get()` und `__set()` verwaltet wird. Aus diesem Grund trägt die Eigenschaft in Beispiel 7-32 den Namen `$__daten` mit zwei Unterstrichen am Anfang.

Sie sollten erwägen, allen »richtigen« Eigenschaften in Klassen, in denen Sie die magischen Zugriffsmethoden verwenden, zwei Unterstriche voranzustellen, um Kollisionen zwischen Eigenschaften, auf die ganz normal zugegriffen werden soll, und Eigenschaften, die über `__get()` und `__set()` gesteuert werden, zu verhindern.

Der Einsatz von `__get()` und `__set()` hat drei Haken. Erstens kommen diese Methoden nur bei Eigenschaften auf den Plan, die nicht definiert sind. Ist eine Eigenschaft in Ihrer Klasse definiert, ruft PHP `__get()` und `__set()` nicht auf, wenn auf diese Eigenschaft zugegriffen wird.

Zweitens zerstören diese Methoden die Vererbung von Eigenschaften. Wenn eine Elternklasse eine `__get()`-Methode hat und Sie in ihrer Kindklasse eine eigene Version von `__get()` implementieren, arbeiten Ihre Objekte nicht mehr richtig, weil die `__get()`-Methode der Elternklasse nicht aufgerufen wird.

Das können Sie beheben, indem Sie `parent::__get()` aufrufen. Sie müssen es aber explizit selbst machen und erhalten es nicht »kostenlos« als Teil des OO-Entwurfs.

Drittens ist die Vorspiegelung der Eigenschaften unvollständig, weil sie sich nicht auf die Methoden `isset()` und `unset()` erstreckt. Versuchen Sie beispielsweise, mit `isset()` eine überladene Eigenschaft zu prüfen, erhalten Sie keine korrekte Antwort, weil PHP dann nicht `__get()` aufruft.

Beheben Sie das, indem Sie in Ihrer Klasse die Methoden `__isset()` und `__unset()` definieren. Wie das geht, zeigt Beispiel 7-33.

Beispiel 7-33: Die magischen Methoden für isset() und unset() implementieren

```
class Person {
    // name und email als zulässige Eigenschaften vordefinieren.
    protected $__daten = array('person', 'email');

    public function __get($eigenschaft) {
```

Beispiel 7-33: Die magischen Methoden für isset() und unset() implementieren (Fortsetzung)

```
            if (isset($this->__daten[$eigenschaft])) {
                return $this->__daten[$eigenschaft];
            } else {
                return false;
            }
        }

        // Erzwingen, dass nur die vordefinierten Eigenschaften
        // gesetzt werden können.
        public function __set($eigenschaft, $wert) {
            if (array_key_exists($eigenschaft, $this->__daten)) {
                $this->__daten[$eigenschaft] = $wert;
            }
        }

        public function __isset($eigenschaft) {
            if (isset($this->__daten[$eigenschaft])) {
                return true;
            } else {
                return false;
        }

        public function __unset($eigenschaft) {
            if (isset($this->__daten[$eigenschaft])) {
                unset($eigenschaft);
            }
        }
    }
```

Die Methode `__isset()` prüft das Element in `$__daten` und liefert je nach Status der geprüften Eigenschaft `true` oder `false`.

Ähnlich liefert `__unset()` den Wert von `unset()` für die »tatsächliche« Eigenschaft zurück oder `false`, wenn diese nicht definiert ist.

Die Implementierung dieser beiden Methoden ist nicht erforderlich, wenn Sie `__get()` und `__set()` verwenden, aber empfehlenswert, da man im Voraus nie vollständig weiß, wie Eigenschaften eventuell verwendet werden. Definieren Sie diese Methoden nicht, kann das zu Verwirrungen führen, wenn jemand (vielleicht sogar Sie selbst) nicht weiß (oder vergisst), dass diese Klasse die magischen Zugriffsmethoden verwendet.

Die Methoden `__isset()` und `__unset()` sind allerdings erst ab PHP 5.1 verfügbar.

Andere Gründe, die gegen den Einsatz der magischen Zugriffsmethoden sprechen, sind:

- Sie sind recht langsam. Sie sind langsamer als ein direkter Eigenschaftszugriff, und sie sind auch langsamer als der Eigenschaftszugriff über explizite Zugriffsmethoden.

- Sie machen es den Reflection-Klassen und Werkzeugen wie phpDocumentor unmöglich, Ihren Code automatisch zu dokumentieren.
- Sie können nicht für statische Eigenschaften eingesetzt werden.

Siehe auch

Die Dokumentation zu magischen Methoden unter *http://www.php.net/manual/en/language.oop5.magic.php*.

7.16 Methoden auf Objekten aufrufen, die von einer anderen Methode geliefert werden

Problem

Sie müssen eine Methode auf einem Objekt aufrufen, das von einer anderen Methode geliefert wird.

Lösung

Verketten Sie die Methodenaufrufe:

```
$orange = $frucht->get('zitrus')->schaelen();
```

Diskussion

PHP ist clever genug, erst `$frucht->get('zitrus')` aufzurufen und `schaelen()` dann auf dem Rückgabewert. Man bezeichnet dieses Konstrukt der Methodenverkettung als *Fluent Interfaces*.

Das ist eine Verbesserung gegenüber PHP 4, wo Sie eine temporäre Variable verwenden mussten:

```
$orange = $frucht->get('zitrus');
$orange->schaelen();
```

Ein weiterer Punkt für PHP 5!

7.17 Zusammengesetzte Klassen verschmelzen

Problem

Sie möchten zwei oder mehr Klassen so zusammensetzen, dass sie sich wie eine einzige Klasse zu verhalten scheinen.

Lösung

Setzen Sie die Objekte zusammen und nutzen Sie die magische Methode `__call()`, um Methodenaufrufe abzufangen und entsprechend weiterzuleiten:

```
class Adresse {
    protected $stadt;

    public function setStadt($stadt) {
        $this->stadt = $stadt;
    }

    public function getStadt() {
        return $this->stadt;
    }
}

class Person {
    protected $name;
    protected $adresse;

    public function __construct() {
        $this->adresse = new Adresse;
    }

    public function setName($name) {
        $this->name = $name;
    }

    public function getName() {
        return $this->name;
    }

    public function __call($methode, $argumente) {
        if (method_exists($this->adresse, $methode)) {
            return call_user_func_array(
                array($this->adresse, $methode), $argumente);
        }
    }
}

$rasmus = new Person;
$rasmus->setName('Rasmus Lerdorf');
$rasmus->setStadt('Sunnyvale');

print $rasmus->getName() . ' lebt in ' . $rasmus->getStadt() . '.';
Rasmus Lerdorf lebt in Sunnyvale.
```

Wird ein neues `Person`-Objekt konstruiert, wird parallel eine Instanz der Klasse `Adresse` erzeugt. Rufen Sie Methoden auf, die in `Person` nicht definiert sind, werden diese von der Methode `__call()` abgefangen und, falls anwendbar, über `call_user_func_array()` weitergeleitet.

Diskussion

Bei der Beispielklasse für dieses Rezept kann man nicht sagen, dass `Person` eine `Adresse` »ist« oder umgekehrt. Deswegen ist es nicht sinnvoll, die eine Klasse die andere erweitern zu lassen.

Sinnvoll ist es hingegen, die jeweiligen Daten und die entsprechenden Funktionalitäten in separate Klassen zu stecken, um Flexibilität und Wiederverwendbarkeit zu maximieren sowie Codeverdopplung zu minimieren. Sie prüfen deswegen, ob sich eine andere Regel – die »Hat ein«-Regel – anwenden lässt. Da eine `Person` eine `Adresse` hat, ist es sinnvoll, diese Klassen zusammenzusetzen.

Verwenden Sie Zusammensetzung oder Aggregation, dient ein Objekt als Container für weitere Objekte. Das ist wiederum eine Möglichkeit, das Problem der Mehrfachvererbung zu lösen, weil auf diese Weise leicht Klassen aus kleineren Komponenten zusammengesetzt werden können.

Ein `Person`-Objekt kann zum Beispiel ein `Adresse`-Objekt enthalten. Dass Personen Adressen haben, ist keine Frage. Adressen haben aber nicht nur Personen, sondern auch Unternehmen und andere Kooperationen. Deswegen ist es sinnvoll, Adressdaten nicht fest in eine `Person`-Klasse einzubinden, sondern eine separate `Adresse`-Klasse zu erstellen, die von mehreren Klassen verwendet werden kann.

Beispiel 7-34 zeigt, wie das in der Praxis funktioniert.

Beispiel 7-34: Klassen zusammensetzen

```
class Adresse {
    protected $stadt;

    public function setStadt($stadt) {
        $this->stadt = $stadt;
    }

    public function getStadt() {
        return $this->stadt;
    }
}

class Person {
    protected $name;
    protected $adresse;

    public function __construct() {
        $this->adresse = new Adresse;
    }

    public function setName($name) {
        $this->name = $name;
    }
```

Beispiel 7-34: Klassen zusammensetzen (Fortsetzung)

```
        public function getName() {
            return $this->name;
        }

        public function __call($methode, $argumente) {
            if (method_exists($this->adresse, $methode)) {
                return call_user_func_array(
                    array($this->adresse, $methode), $argumente);
            }
        }
    }
```

Die Klasse `Adresse` hat ein `stadt`-Feld und zwei Zugriffsmethoden zur Manipulation der darin gespeicherten Daten: `setStadt()` und `getStadt()`.

`Person` hat `setName()` und `getName()`, ganz ähnlich wie `Adresse`, bietet aber noch zwei weitere Methoden: `__construct()` und `__call()`.

Der Konstruktor instantiiert ein `Adresse`-Objekt und speichert es in der Eigenschaft `protected $adresse`. Das gestattet Methoden in `Person`, auf `$adresse` zuzugreifen, verhindert aber, dass andere direkt mit dieser Eigenschaft arbeiten.

Optimal wäre es, wenn PHP einen Aufruf einer Methode aus `Adresse` automatisch ausführen würde. Das passiert aber nicht, da `Person` `Adresse` nicht erweitert. Den Code, der diese Aufrufe an die entsprechenden Methoden weiterleitet, müssen Sie selbst schreiben.

Eine Möglichkeit wären Wrapper-Methoden, zum Beispiel:

```
public function setStadt($stadt) {
        $this->adresse->setStadt($stadt);
}
```

Diese `setStadt()`-Methode ruft mit ihrem Argument die `setStadt()`-Methode auf dem `Adresse`-Objekt in `$adresse` auf. Das ist einfach, aber auch recht mühsam, da man Wrapper für jede einzelne Methode schreiben muss.

Mit `__call()` können Sie diesen Vorgang automatisieren, indem Sie die Aufrufweiterleitungen an einem Ort vereinen, wie Sie es in Beispiel 7-35 sehen.

Beispiel 7-35: Zentralisierter Methodenaufruf in __call()

```
public function __call($methode, $argumente) {
        if (method_exists($this->adresse, $methode)) {
            return call_user_func_array(
                array($this->adresse, $methode), $argumente);
        }
}
```

`__call()` fängt alle Aufrufe an in einer Klasse nicht definierte Methoden ab. Es wird mit zwei Argumenten aufgerufen: dem Namen der Methode und einem Array mit den Argumenten, die beim Aufruf angegeben wurden. Über das erste Argument sehen Sie, welche

Methode aufzurufen versucht wurde, und können dann prüfen, ob eine Weiterleitung an `$adresse` korrekt ist.

Hier wollen Sie den Aufruf weiterleiten, wenn die Methode eine gültige Methode der Klasse `Adresse` ist. Das prüfen Sie mit `method_exists()`, indem Sie als ersten Parameter das Objekt und als zweiten Parameter den Methodennamen angeben.

Liefert diese Funktion `true`, wissen Sie, dass die Methode gültig ist, und können sie folglich aufrufen. Unglücklicherweise müssen Sie sich immer noch darum kümmern, dass die Argumente aus dem Array `$argumente` ausgepackt werden, und das kann eine ziemliche Qual werden.

Die selten verwendete und seltsam benannte Funktion `call_user_func_array()` löst dieses Problem. Über diese Funktion können Sie eine Benutzerfunktion aufrufen und ihr Argumente in einem Array übergeben. Der erste Parameter ist die Funktion, der zweite das Array mit ihren Argumenten.

Hier möchten Sie aber statt einer Funktion eine Methode aufrufen. Es gibt eine spezielle Syntax, die für diesen Verwendungszweck geeignet ist. Statt einen Funktionsnamen zu übergeben, übergeben Sie ein Array mit zwei Argumenten. Das erste Element ist das Objekt und das zweite der Methodenname.

Das bewirkt, dass `call_user_func_array()` die Methode auf Ihrem Objekt aufruft. Dann müssen Sie das Ergebnis des `call_user_func_array()`-Aufrufs an den ursprünglichen Aufrufer zurückliefern, da Rückgabewerte ansonsten schweigend verworfen werden.

Hier ist ein `Person`-Beispiel, in dem jeweils Methoden aufgerufen werden, die in `Person` und `Adresse` definiert sind:

```
$rasmus = new Person;
$rasmus->setName('Rasmus Lerdorf');
$rasmus->setStadt('Sunnyvale');
print $rasmus->getName() . ' lebt in ' . $rasmus->getStadt() . '.';
Rasmus Lerdorf lebt in Sunnyvale.
```

Obwohl `setStadt()` und `getStadt()` keine Methoden von `Person` sind, haben Sie sie in diese Klasse eingebunden.

Sie können weitere Objekte zu einer einzigen Klasse zusammensetzen oder genauer auswählen, welche Methoden einem externen Nutzer verfügbar gemacht werden sollen. In dem Fall ist eine Filterung auf Grundlage des Methodennamens erforderlich.

Ab PHP 5.3 können Sie auch statische Methodenaufrufe einer Klasse abfangen. Hierzu dient die magische Methode `__callStatic()`, die analog zur bereits vorgestellten `__call()`-Methode funktioniert:

```
class Adresse {
    protected static $planet = 'Erde';
    [...]
    public static function getPlanet() {
        return self::$planet;
```

```
    }
    public static function setPlanet($planet) {
        self::$planet = $planet;
    }
}
class Person {
    [...]
    public static function __callStatic($methode, $argumente) {
        if (method_exists('Adresse', $methode)) {
            return call_user_func_array(
                array('Adresse', $methode), $argumente);
        }
    }
}
```

Siehe auch

Die Dokumentation zu magischen Methoden unter *http://www.php.net/manual/en/language.oop5.magic.php*.

7.18 Auf überschriebene Methoden zugreifen

Problem

Sie möchten auf eine Methode in der Elternklasse zugreifen, die in der Kindklasse überschrieben wurde.

Lösung

Stellen Sie dem Methodennamen `parent::` voran:

```
class Figur {
    function zeichnen() {
        // Auf den Bildschirm zeichnen.
    }
}

class Kreis extends Figur {
   function zeichnen($mittelpunkt, $radius) {
      // Daten prüfen.
      if ($radius > 0) {
          parent::zeichnen();
          return true;
      }

      return false;
   }
}
```

Diskussion

Wenn Sie eine Elternmethode überschreiben, indem Sie sie in der Kindklasse definieren, wird die Methode der Elternklasse nur aufgerufen, wenn Sie sie explizit aufrufen.

In der Lösung haben wir die Methode `zeichnen()` in der Kindklasse `Kreis` überschrieben, weil wir kreisspezifische Parameter akzeptieren und die Daten validieren wollten. Es soll aber immer noch die allgemeine `Figur::zeichnen()`-Aktion durchgeführt werden, die den tatsächlichen Zeichenvorgang abwickelt. Wir rufen also `parent::zeichnen()` in der Methode auf, wenn `$radius` größer `0` ist.

Nur der Code in der Klasse kann `parent::` einsetzen. Wird `parent::zeichnen()` von außerhalb der Klasse aufgerufen, erhalten Sie einen Parser-Fehler. Folgendes Beispiel würde nicht funktionieren, wenn `Kreis::zeichnen()` nur den Radius prüfen würde, Sie aber auch `Figur::zeichnen()` aufrufen wollten:[3]

```
$kreis = new Kreis;
if ($kreis->zeichnen($mittelpunkt, $radius)) {
    $kreis->parent::zeichnen();
}
```

Das gilt auch für Objektkonstruktoren. Man sieht also recht häufig Konstruktionen wie diese:

```
class Kreis {
    function __construct($x, $y, $r) {
        // Erst den Figur-Konstruktor aufrufen.
        parent::__construct();
        // Jetzt kreisspezifischen Krempel erledigen.
    }
}
```

Dass ein Aufruf des Elternkonstruktors ein derartiges Kinderspiel ist, ist einer der Vorteile des konsistenten Namensschemas für Konstruktoren, das PHP 5 einführt. In PHP 4 mussten Sie noch durch eine Reihe von Reifen springen, um so etwas auf robuste Weise zu implementieren.

Siehe auch

Rezept 7.2 bietet mehr zu Konstruktoren; die Dokumentationen zu Elternklassen unter *http://www.php.net/keyword.parent* und zu `get_parent_class()` unter *http://www.php.net/get-parent-class*.

3 Es scheitert mit der Meldung `unexpected T_PAAMAYIM_NEKUDOTAYIM`, was auf Hebräisch Doppelpunkt heißt.

7.19 Methodenpolymorphie einsetzen

Problem

Sie möchten in Abhängigkeit von der Anzahl und dem Typ der an eine Methode übergebenen Argumente jeweils unterschiedlichen Code ausführen.

Lösung

Methodenpolymorphie unterstützt PHP nicht als eingebaute Funktionalität. Sie lässt sich aber mithilfe verschiedener Funktionen zur Typprüfung emulieren. Die folgende `kombinieren( )`-Methode nutzt `is_numeric( )`, `is_string( )`, `is_array( )` und `is_bool( )`:

```
// kombinieren() addiert Zahlen, verkettet Strings, verschmilzt Arrays
// und verknüpft Boolesche Argumente mit einem bitweisen UND.
function kombinieren($a, $b) {
    if (is_int($a) && is_int($b))     {
        return $a + $b;
    }

    if (is_float($a) && is_float($b))   {
        return $a + $b;
    }

    if (is_string($a) && is_string($b))  {
        return "$a$b";
    }

    if (is_array($a) && is_array($b))   {
        return array_merge($a, $b);
    }

    if (is_bool($a) && is_bool($b))    {
        return $a & $b;
    }

    return false;
}
```

Diskussion

Da Sie in PHP im Prototyp von Methoden keine Variablentypen angeben können, kann es Methoden nicht in Abhängigkeit von der Methodensignatur ausführen wie Java oder C++. Aber Sie können stattdessen eine Methode erstellen und eine `switch`-Anweisung einsetzen, um diese Funktionalität manuell zu konstruieren.

In PHP können Sie beispielsweise mit der Bibliothek GD Bilder bearbeiten. In einer Grafikklasse kann es hilfreich sein, wenn man entweder den Ort des Bilds (entfernt oder lokal)

übergeben kann oder ein Handle, das PHP einem bestehenden Stream zugewiesen hat. Beispiel 7-36 zeigt eine pk_Image-Klasse, die genau das macht.

Beispiel 7-36: pk_Image-Klasse

```
class pk_Image {

    protected $handle;

    function ImageCreate($image) {
        if (is_string($image)) {
            // Einfache Dateitypermittlung.

            // Dateinamenserweiterung abrufen.
            $info = pathinfo($image);
            $extension = strtolower($info['extension']);
            switch ($extension) {
            case 'jpg':
            case 'jpeg':
                $this->handle = ImageCreateFromJPEG($image);
                break;
            case 'png':
                $this->handle = ImageCreateFromPNG($image);
                break;
            default:
                die('Bilder müssen JPEGs oder PNGs sein.');
            }
        } elseif (is_resource($image)) {
            $this->handle = $image;
        } else {
            die('Variablen müssen Strings oder Ressourcen sein.');
        }
    }
}
```

Dieser Code behandelt übergebene Strings als Pfad einer Datei. Wir nutzen also `pathinfo()`, um die Dateierweiterung abzurufen. Kennen wir diese, versuchen wir zu ermitteln, welche `ImageCreateFrom()`-Funktion das Bild öffnet und ein Handle liefert.

Ist das, was übergeben wurde, kein String, haben wir es direkt mit einem GD-Stream zu tun. Ein solcher hat den Typ `resource`. Da keine Umwandlung erforderlich ist, weisen wir den Stream direkt `$handle` zu. Wenn Sie diese Klasse in einer Produktionsumgebung einsetzen, sollten Sie natürlich eine stabilere Fehlerbehandlung einbauen.

Methodenpolymorphie umfasst auch Methoden mit einer unterschiedlichen Anzahl von Argumenten. Der Code, mit dem Sie in einer Methode die Anzahl von Argumenten ermitteln können, entspricht dem, den Sie in einer Funktion, die eine variable Anzahl von Argumenten verarbeiten kann, mit `func_num_args()` implementieren. Das wird in Rezept 6.5 beschrieben.

Siehe auch

Rezept 6.5 beschreibt Funktionen mit einer variablen Anzahl von Argumenten; die Dokumentationen zu `is_string()` unter *http://www.php.net/is-string*, zu `is_resource()` unter *http://www.php.net/is-resource* und zu `pathinfo()` unter *http://www.php.net/pathinfo*.

7.20 Klassenkonstanten definieren

Problem

Sie möchten Konstanten auf Klassenbasis definieren, nicht auf globaler Basis.

Lösung

Definieren Sie sie wie Eigenschaften, verwenden Sie dabei aber die Markierung `const`:

```
class Math {
    const pi = 3.14159; // universelle
    const  e = 2.71828; // Konstanten
}

$flaeche = math::pi * $radius * $radius;
```

Diskussion

PHP nutzt sein Konzept globaler Konstanten und wendet es auf Konstanten an. Diese sind im Wesentlichen finale Eigenschaften.

Konstanten deklarieren Sie mit dem Label `const`:

```
class Math {
    const pi = 3.14159; // universelle
    const  e = 2.71828; // Konstanten
}

$flaeche = math::pi * $radius * $radius;
```

Wie auf statische Eigenschaften können Sie auch auf Konstanten zugreifen, ohne dazu Ihre Klasse zu instantiieren. Dazu wird ebenfalls die Doppel-Doppelpunkt-Notation (`::`) eingesetzt. Stellen Sie dem Namen der Konstanten `self::` voran, wenn Sie sie in der Klasse selbst referenzieren wollen.

Anders als Eigenschaften wird Konstanten kein Dollarzeichen (`$`) vorangestellt:

```
class Kreis {
    const pi = 3.14159;
    protected $radius;

    public function __construct($radius) {
        $this->radius = $radius;
```

```
    }

    public function umfang() {
        return 2 * self::pi * $this->radius;
    }
}

$kreis = new Kreis(1);
print $kreis->umfang();
6.28318
```

Dieses Beispiel erzeugt einen Kreis mit einem Radius von `1` und ruft dann die Methode `umfang` auf, um den Umfang des Kreises zu berechnen. Damit im folgenden Beispiel die `pi`-Konstante der Klasse verwendet wird, müssen Sie sie über `self::pi` referenzieren, da PHP den Umfang andernfalls unter Verwendung der definierten globalen Konstante `pi` berechnet:

```
define('pi', 10); // globale Konstante pi

class Kreis {
    const pi = 3.14159; // Klassenkonstante pi
    protected $radius;

    public function __construct($radius) {
        $this->radius = $radius;
    }

    public function umfang() {
        return 2 * pi * $this->radius;
    }

}

$kreis = new Kreis(1);
print $kreis->umfang()
20
```

Oops! PHP hat für `pi` `10` statt `3.14159` verwendet und deswegen `20` statt `6.28318` als Antwort geliefert.

Obwohl es eher unwahrscheinlich ist, dass Sie `pi` versehentlich neu definieren (da Sie vermutlich ohnehin die eingebaute Konstante `M_PI`) verwenden), können Sie sich so dennoch ein Bein stellen.

Einer Konstanten können Sie weder den Wert eines Ausdrucks zuweisen noch Daten, die Ihrem Skript übergeben wurden:

```
// ungültig
class Berechtigungen {
    const    read = 1 << 2;
    const   write = 1 << 1;
    const execute = 1 << 0;
}
```

```
// unsicher und ungültig
class Datenbank {
    const debug = $_REQUEST['debug'];
}
```

Weder die Konstanten in `Berechtigungen` noch die Konstante `debug` in `Datenbank` ist zulässig, weil keine festen Werte verwendet werden. Selbst das erste Beispiel, `1 << 2`, ist nicht erlaubt, obwohl PHP dort keine externen Daten einlesen muss.

Da Sie auf Konstanten über einen expliziten Namen zugreifen müssen – entweder `self::` oder den Namen der Klasse –, können Sie den Klassennamen nicht dynamisch zur Laufzeit berechnen. Er muss im Voraus deklariert sein, zum Beispiel:

```
class Konstanten {
    const pi = 3.14159;

    // Rest der Klasse
}

$klasse = 'Konstanten';

print $klasse::pi;
```

Das führt in PHP-Versionen vor 5.3 zu einem Parser-Fehler, obwohl diese Art Konstrukt bei nicht konstanten Ausdrücken wie `$klasse->pi` zulässig ist. In diesen PHP-Versionen darf der Scope Resultion Operator (`::`) nicht nach einer Variablen folgen. Ab PHP 5.3 ist das kein Problem, der Variablenwert darf jedoch kein Keyword sein (z.B. `self`, `parent` oder `static`).

Siehe auch

Die Dokumentation zu Klassenkonstanten finden Sie unter *http://www.php.net/manual/en/language.oop5.constants.php*.

7.21 Statische Eigenschaften und Methoden definieren

Problem

Sie möchten in einer Klasse Methoden definieren, auf die Sie zugreifen können, ohne ein Objekt zu instantiieren.

Lösung

Deklarieren Sie die Methode als `static`:

```
class Format {
    public static function number($number, $decimals = 2,
                                  $decimal = ',', $thousands = '.') {
        return number_format($number, $decimals, $decimal, $thousands);
    }
```

```
}

print Format::number(1234.567);
1,234.57
```

Diskussion

Gelegentlich will man in einer Klasse einen Satz von Methoden definieren, die man aufrufen können möchte, ohne dazu ein Objekt zu instantiieren. In PHP 5 können Sie Methoden direkt aufrufen, wenn Sie sie als statisch deklarieren:

```
class Format {
    public static function number($number, $decimals = 2,
                                  $decimal = ',', $thousands = '.') {
        return number_format($number, $decimals, $decimal, $thousands);
    }
}

print Format::number(1234.567);
1,234.57
```

Da statische Methoden keine Objektinstanz verlangen, können Sie den Klassennamen statt der Referenz auf ein Objekt verwenden. Geben Sie vor dem Klassennamen kein Dollarzeichen (`$`) an.

Statische Methoden werden nicht über einen Pfeil (`->`) referenziert, sondern über einen doppelten Doppelpunkt (`::`) – das zeigt PHP an, dass es die Methode statisch aufrufen soll. Auf die `number()`-Methode der *Format*-Klasse im Beispiel oben wird also mit `Format::number()` zugegriffen.

Die Formatierung von Zahlen ist von keinen anderen Eigenschaften oder Methoden eines Objekts abhängig. Deswegen ist es vernünftig, eine entsprechende Methode als statisch zu deklarieren. Auf diese Weise könnte man beispielsweise in einer Warenkorbanwendung die Preise der Waren mit einer einzigen Zeile formatieren, ohne anstelle von Objekten wieder globale Funktionen zu verwenden.

Statische Methoden operieren nicht auf einer bestimmten Instanz der Klasse, in der sie definiert werden. PHP »konstruiert« kein temporäres Objekt, das Sie innerhalb einer statische Methode nutzen können. Deswegen können Sie `$this` in statischen Methoden nicht verwenden – es gibt kein `$this`, auf dem Sie operieren können. Statische Methoden rufen Sie auf wie gewöhnliche Funktionen.

PHP 5 kennt außerdem statische Eigenschaften. Derartige Eigenschaften teilen alle Instanzen einer Klasse. Sie fungieren daher als globale Variablen, die auf den Geltungsbereich der Klasse eingeschränkt sind.

Ein Grund, eine statische Eigenschaft zu verwenden, wäre beispielsweise, eine Datenbankverbindung von mehreren `Database`-Objekten teilen zu lassen. Der Effizienz halber sollten Sie darauf verzichten, bei jeder Instantiierung von `Database` eine neue Verbindung

zu erstellen. Stattdessen sollten Sie die Verbindung bei der Instantiierung der ersten Instanz aushandeln und diese dann in allen weiteren Instanzen wiederverwenden, wie es Beispiel 7-37 zeigt.

Beispiel 7-37: Statische Eigenschaften in mehreren Instanzen nutzen

```
class Database {
    private static $dbh = NULL;

    public function __construct($server, $username, $password) {
        if (self::$dbh == NULL) {
            self::$dbh = db_connect($server, $username, $password);
        } else {
            // Bestehende Verbindung wiederverwenden
        }
    }
}

$db  = new Database('db.example.com', 'web', 'jsd6w@2d');
// Ein paar Abfragen durchführen.

$db2 = new Database('db.example.com', 'web', 'jsd6w@2d');
// Ein paar Abfragen durchführen.
```

Auf statische Eigenschaften greift man wie auf statische Methoden mit dem doppelten Doppelpunkt zu. Wollen Sie innerhalb der Klasse selbst auf eine statische Eigenschaft verweisen, nutzen Sie das Präfix `self`. `self` ist für statische Eigenschaften und Methoden das Gleiche wie `$this` für Instanzeigenschaften und Methoden.

Der Konstruktor in unserem Beispiel nutzt `self::$dbh`, um auf die statische Eigenschaft zuzugreifen, die die Verbindung festhält. Bei der Erstellung der ersten Instanz, `$db`, ist `$dbh` noch `NULL`. Der Konstruktor ruft also `db_connect( )` auf, um eine neue Datenbankverbindung zu erstellen.

Wird die zweite Instanz erstellt, `$db2`, passiert das nicht, da `$dbh` bereits ein Datenbank-Handle zugewiesen wurde.

Ab Version 5.3 unterstützt PHP spätes statisches Binden (*Late Static Binding*). Neben `self` können Sie in einer statischen Methode nun zusätzlich auf das Schlüsselwort `static` zurückgreifen. Das folgende Beispiel soll den Sachverhalt näher beleuchten:

```
class A {
    static $greeting = "Hallo Welt";
    static function gruesse() { print self::$greeting . "\n"; }
}

class B extends A {
    static $greeting = "Guten Tag Welt";
}

B::gruesse();
Hallo Welt
```

Wenn Sie den Code ausführen, wird wie ersichtlich »Hallo Welt« ausgegeben, obwohl in der Klasse `B` die Variable den Wert `"Guten Tag Welt"` hat. Das liegt daran, dass `self` an die Klasse gebunden wird, in der es geschrieben steht – in diesem Fall `A`. Das Binden erfolgt also bei `self` zur Entwicklungszeit des Skripts. Bei spätem statischem Binden sieht das Codebeispiel wie folgt aus:

```
class A {
    static $greeting = "Hallo Welt";
    static function gruesse() { print static::$greeting . "\n"; }
}

class B extends A {
    static $greeting = "Guten Tag Welt";
}

B::gruesse();
Guten Tag Welt
```

Jetzt lautet die Ausgabe »Guten Tag Welt«. Das bedeutet, dass `static` zur Laufzeit des Skripts an die Klasse gebunden wird, die den Methodenaufruf ausführt. Das ist in diesem Fall die Klasse `B`.

Siehe auch

Die Dokumentation zum Schlüsselwort `static` unter *http://www.php.net/manual/en/language.oop5.static.php*.

7.22 Die Objektserialisierung steuern

Problem

Sie möchten steuern, wie sich ein Objekt verhält, wenn Sie es serialisieren und deserialisieren. Das ist hilfreich, wenn Sie Verbindungen zu entfernten Ressourcen wie Datenbanken, Dateien und Webservices herstellen und schließen müssen.

Lösung

Definieren Sie die magischen Methoden `__sleep()` und `__wakeUp()`, wie Sie es in Beispiel 7-38 sehen.

Beispiel 7-38: Mit __sleep() und __wakeUp() die Serialisierung steuern

```
<?php
class LogFile {
    protected $dateiname;
    protected $handle;

    public function __construct($dateiname) {
```

Beispiel 7-38: Mit __sleep() und __wakeUp() die Serialisierung steuern (Fortsetzung)

```
            $this->dateiname = $dateiname;
            $this->open();
        }

        private function open() {
            $this->handle = fopen($this->dateiname, 'a');
        }

        public function __destruct($dateiname) {
            fclose($this->handle);
        }

        // Wird aufgerufen, wenn ein Objekt serialisiert wird, und sollte
        // ein Array mit zu serialisierenden Objekteigenschaften liefern.
        public function __sleep() {
            return array('dateiname');
        }

        // Wird aufgerufen, wenn ein Objekt deserialisiert wird.
        public function __wakeUp() {
            $this->open();
        }
    }
    ?>
```

Diskussion

Wenn Sie in PHP ein Objekt serialisieren, bewahrt es alle Eigenschaften Ihres Objekts. Verbindungen oder Handles auf externe Ressourcen wie Datenbanken, Dateien oder Webservices schließt das allerdings nicht ein.

Diese müssen erneut hergestellt werden, wenn Sie das Objekt deserialisieren, da sich Ihr Objekt ansonsten nicht richtig verhält. Das können Sie explizit in Ihrem Code machen. Aber es ist besser, Ihren Code allgemeiner zu halten und diese Dinge von PHP im Hintergrund abwickeln zu lassen.

Verwenden Sie dazu daher die magischen Methoden `__sleep()` und `__wakeUp()`. Wenn Sie auf einem Objekt `serialize()` aufrufen, ruft PHP `__sleep()` auf, wenn Sie `unserialize()` aufrufen, `__wakeUp()`.

Die `LogFile`-Klasse in Beispiel 7-38 hat fünf einfache Methoden. Der Konstruktor erwartet einen Dateinamen und speichert ihn, damit er später wiederverwendet werden kann. Die Methode `open()` öffnet die entsprechende Datei und speichert das Datei-Handle, das im Destruktor des Objekts geschlossen wird.

Die `__sleep()`-Methode liefert ein Array mit den Eigenschaften, die bei der Serialisierung eines Objekts gespeichert werden sollen. Da Datei-Handles bei einer Serialisierung nicht bewahrt werden, liefert sie `array('filename')`, weil das alles ist, was gespeichert werden soll.

Deswegen müssen Sie die Datei neu öffnen, wenn das Objekt deserialisiert wird. Das wird in `__wakeUp()` abgewickelt. Diese Methode ruft erneut die schon im Konstruktor aufgerufene `open()`-Methode auf. Da Sie der `__wakeUp()`-Methode keine Argumente übergeben können, muss sie den Dateinamen von anderer Stelle erhalten. Glücklicherweise kann die Methode auf die Objekteigenschaften zugreifen, in denen wir aus genau diesem Grund den Dateinamen gespeichert haben.

Machen Sie sich stets bewusst, dass eine einzige Instanz bei einem einzigen Aufruf mehrfach serialisiert oder nach der Serialisierung noch weiterverwendet werden kann. Deswegen sollten Sie in `__sleep()` nichts machen, was eins dieser Dinge verhindert. `__sleep()` sollte nur verwendet werden, um Eigenschaften auszuschließen, die nicht serialisiert werden sollten, weil sie zu viel Speicherplatz in Anspruch nehmen oder auf Basis von anderen Daten berechnet werden und neu berechnet oder auf andere Weise aufgefrischt werden sollten, wenn das Objekt deserialisiert wird.

Deswegen erfolgt der Aufruf von `fclose()` im Destruktor und nicht in `__sleep()`.

Siehe auch

Die Dokumentationen zu den magischen Methoden unter *http://www.php.net/manual/en/language.oop5.magic.php*, zu `unserialize()` unter *http://www.php.net/unserialize* und zu `serialize()` unter *http://www.php.net/serialize*.

7.23 Objektintrospektion

Problem

Sie möchten ein Objekt untersuchen, um zu sehen, welche Methoden und Eigenschaften es hat, damit Sie Code schreiben können, der mit jedem allgemeinen Objekt arbeitet, unabhängig davon, welchen Typ es hat.

Lösung

Nutzen Sie die Reflection-Klassen, um Informationen zu einer Klasse abzufragen.

Einen Kurzüberblick über die Klasse erhalten Sie mit `Reflection::export()`:

```
// Informationen zur Klasse Wagen abfragen.
Reflection::export(new ReflectionClass('Wagen'));
```

Spezifische Informationen erhalten Sie mit:

```
$wagen = new ReflectionClass('Wagen');
if ($car->hasMethod('verdeckOeffnen')) {
    // Wagen ist ein Cabrio.
}
```

Diskussion

Sie müssen nur selten mit Klassen arbeiten, deren Code Sie nicht einsehen können, um sich ihre Definition anzusehen. Aber mit den Reflection-Klassen können Sie innerhalb eines Programms ebenso Informationen zu OO-Funktionalitäten wie Klassen, Methoden und Eigenschaften abrufen wie zu Nicht-OO-Funktionalitäten wie Funktionen und Erweiterungen.

Das ist bei Projekten nützlich, die Sie auf viele unterschiedliche Klassen anwenden wollen, beispielsweise zur Erstellung automatisierter Dokumentationen zu Klassen, allgemeiner Objekt-Debugger oder Zustandsspeicherern wie `serialize( )`.

Wie die Reflection-Klassen funktionieren, zeigt Beispiel 7-39 anhand des Beispiels einer `Person`-Klasse, die viele der PHP 5-OO-Funktionalitäten nutzt.

Beispiel 7-39: Person-Klasse

```
class Person {
    public $name;
    protected $frau;
    private $passwort;

    public function __construct($name) {
        $this->name = $name;
    }

    public function getName() {
        return $name;
    }

    protected function setFrau(Person $frau) {
        if (!isset($this->frau)) {
            $this->frau = $frau;
        }
    }

    private function setPasswort($passwort) {
        $this->passwort = $passwort;
    }
}
```

Um sich schnell einen Überblick über die Klasse zu verschaffen, rufen Sie `Reflection::export( )` auf:

```
Reflection::export(new ReflectionClass('Person'));
Class [ <user> class Person ] {
  @@ /www/reflection.php 3-25

  - Constants [0] {
  }

  - Static properties [0] {
```

```
  }

  - Static methods [0] {
  }

  - Properties [3] {
    Property [ <default> public $name ]
    Property [ <default> protected $frau ]
    Property [ <default> private $passwort ]
  }

  - Methods [4] {
    Method [ <user> <ctor> public method _ _construct ] {
      @@ /www/reflection.php 8 - 10

      - Parameters [1] {
        Parameter #0 [ $name ]
      }
    }

    Method [ <user> public method getName ] {
      @@ /www/reflection.php 12 - 14
    }

    Method [ <user> protected method setFrau] {
      @@ /www/reflection.php 16 - 20

      - Parameters [1] {
        Parameter #0 [ Person or NULL $frau ]
      }
    }

    Method [ <user> private method setPasswort] {
      @@ /www/reflection.php 22 - 24

      - Parameters [1] {
        Parameter #0 [ $passwort ]
      }
    }
  }
}
```

Die statische Methode `Reflection::export()` erwartet eine Instanz von `ReflectionClass` und liefert eine üppige Menge an Informationen. Wie Sie sehen, wird die Anzahl an Konstanten, statischen Eigenschaften, statischen Methoden, Eigenschaften und Methoden in der Klasse angegeben. Jedes Element wird in seine Bestandteile aufgelöst. Beispielsweise enthalten die Einträge alle Angaben zur Sichtbarkeit (`private`, `protected` oder `public`) und Methoden unter ihrer Definition eine Liste mit den von ihnen erwarteten Parametern.

`Reflection::export()` meldet nicht nur die Datei, in der die Dinge definiert werden, sondern gibt sogar die Zeilennummer an! Das ermöglicht Ihnen, Code aus einer Datei herauszuziehen und ihn in eine Dokumentation einzufügen.

Beispiel 7-40 zeigt ein kurzes Kommandozeilenskript, das nach einem Dateinamen und der Nummer der Zeile sucht, an der eine Methode oder Funktion beginnt.

Beispiel 7-40: Mit Reflection Funktions- und Methodendefinitionen suchen

```
<?php
if ($argc < 2) {
    print "$argv[0]: Funktion/Methode, klasse1.php [, ... klasseN.php]\n";
    exit;
}

// Den Funktionsnamen abrufen.
$funktion = $argv[1];
// Die Dateien einlesen.
foreach (array_slice($argv, 2) as $dateiname) {
    include_once $dateiname;
}

try {
    if (strpos($funktion, '::')) {
        // Es ist eine Methode.
        list ($klasse, $methode) = explode('::', $funktion);
        $reflect = new ReflectionMethod($klasse, $methode);
    } else {
        // Es ist eine Funktion.
        $reflect = new ReflectionFunction($funktion);
    }

    $datei = $reflect->getFileName();
    $zeile = $reflect->getStartLine();

    printf ("%s | %s | %d\n", "$funktion()", $datei, $zeile);
} catch (ReflectionException $e) {
    printf ("%s nicht gefunden.\n", "$funktion()");
}

?>
```

Übergeben Sie als erstes Argument einen Funktions- oder Methodennamen und als weitere Argumente die Namen der Dateien. Die Dateien werden mit `included` eingelesen. Achten Sie also darauf, dass sie nichts ausgeben.

Im nächsten Schritt wird ermittelt, ob das erste Argument eine Methode oder eine Funktion ist. Da Methoden in der Form `Klasse::Methode` angegeben werden, können Sie `strpos( )` einsetzen, um sie auseinanderzuhalten.

Handelt es sich um eine Methode, nutzen Sie `explode( )`, um die Klasse von der Methode zu trennen, und übergeben beides an `ReflectionMethod`. Handelt es sich um eine Funktion, können Sie direkt und ohne Umstände ein `ReflectionFunction`-Objekt instantiieren.

Da `ReflectionMethod` `ReflectionFunction` erweitert, können Sie dann auf Objekten beider Klassen `getFileName( )` und `getStartLine( )` aufrufen. Damit sammeln Sie die Informationen, die Sie ausgeben möchten. Die Ausgabe selbst erfolgt mit `printf( )`.

Versuchen Sie, ein ReflectionMethod- oder ReflectionFunction-Objekt anhand des Namens einer nicht definierten Methode/Funktion aufzurufen, lösen die Klassen eine ReflectionException aus. Diese fangen Sie ab und lassen eine Fehlermeldung ausgeben.

Ein komplexeres Skript, das Informationen gleicher Art für alle benutzerdefinierten Methoden und Funktionen ausgibt, finden Sie in Rezept 7.28.

Wenn Sie nur schnell einen Blick auf ein Objekt werfen müssen, sollten Sie sich nicht mit den Reflection-Klassen herumschlagen. Nutzen Sie einfach var_dump( ), var_export( ) oder print_r( ), um die Werte des Objekts auszugeben. Diese drei Funktionen geben die Daten jeweils auf leicht unterschiedliche Weise aus. var_export( ) kann die Informationen optional auch zurückliefern, statt sie auszugeben.

Siehe auch

Rezept 5.8 liefert weitere Informationen zur Ausgabe von Variablen; die Dokumentationen zu Reflection unter *http://www.php.net/manual/en/language.oop5.reflection.php*, zu var_dump( ) unter *http://www.php.net/var-dump*, zu var_export( ) unter *http://www.php.net/var-export* und zu print_r( ) unter *http://www.php.net/print-r*.

7.24 Prüfen, ob ein Objekt eine Instanz einer bestimmten Klasse ist

Problem

Sie möchten prüfen, ob ein Objekt eine Instanz einer bestimmten Klasse ist.

Lösung

Wenn Sie sicherstellen wollen, dass ein Wert, der einer Funktion übergeben wird, eine Instanz einer bestimmten Klasse ist, geben Sie den Klassennamen in der Argumentliste des Funktionsprototyps an:

```
public function add(Person $person) {
        // $person dem Adressbuch hinzufügen.
    }
}
```

In anderen Kontexten nutzen Sie den instanceof-Operator:

```
<?php
$media = etwas_aus_Katalog abrufen();
if ($media instanceof Buch) {
  // Buch-Sachen machen.
} else if ($media instanceof DVD) {
  // Film anzeigen.
}
?>
```

Diskussion

Eine Möglichkeit, zu erzwingen, welche Arten von Objekten verwendet werden, sind *Typhinweise*. Mit einem Typhinweis sagen Sie PHP, dass ein einer Funktion oder Methode übergebenes Objekt eine bestimmte Klasse haben muss.

Dazu geben Sie den Klassennamen im Prototyp Ihrer Funktion oder Methode an. Ab PHP 5.1 können Sie mit dem Schlüsselwort `array` auch verlangen, dass das Argument ein Array ist. Das funktioniert allerdings nur bei Klassen und Arrays, bei anderen Arten von Typen nicht. Sie können beispielsweise nicht angeben, dass Sie einen String oder Integer haben wollen.

Um zu fordern, dass das erste Argument der `add()`-Methode Ihrer `Adressbuch`-Klasse den Typ `Person` hat, können Sie beispielsweise Folgendes einsetzen:

```
class Adressbuch {

    public function add(Person $person) {
        // $person dem Adressbuch hinzufügen.
    }
}
```

Übergeben Sie beim Aufruf von `add( )` einen String, erhalten Sie einen fatalen Fehler:

```
$buchk = new Adressbuch;

$person = 'Rasmus Lerdorf';

$buchk->add($person);
PHP Fatal error:  Argument 1 must be an object of class Person in...
```

Der `Person`-Typhinweis auf dem ersten Argument in der Funktionsdeklaration hat die gleiche Auswirkungen wie eine Erweiterung Ihrer Funktion mit folgendem PHP-Code:

```
public function add($person) {
        if (!($person instanceof Person)) {
                die("Argument 1 muss eine Instanz von Person sein");
        }
}
```

Der `instanceof`-Operator prüft, ob ein Objekt eine Instanz einer bestimmten Klasse ist. Dieser Code prüft, ob `$person` ein `Person`-Objekt ist.

PHP 4 besitzt keinen `instanceof`-Operator. Sie müssen die Funktion `is_a( )` einsetzen, die unter PHP 5 nicht mehr verwendet werden sollte.

Der `instanceof`-Operator liefert auch bei Klassen `true`, die Kindklassen der Klasse sind, auf die Sie prüfen, zum Beispiel:

```
class Person { /* ... */ }

class Kind extends Person { /* ... */ }

$kind = new Kind;
```

```
if ($kind instanceof Person) {
    print "Auch Kinder sind Menschen.\n";
}
```

```
Auch Kinder sind Menschen.
```

Schließlich können Sie `instanceof` einsetzen, um zu schauen, ob eine Klasse ein bestimmtes Interface implementiert:

```
interface Benennbar {
    public function getName();
    public function setName($name);
}

class Buch implements Benennbar {
    private $name;

    public function getName() {
        return $this->name;
    }

    public function setName($name) {
        return $this->name = $name;
    }
}

$buch = new Buch;
if ($Buch instanceof Benennbar) {
    print "Ein Buch kann einen Namen haben.\n";
}
```

```
Ein Buch kann einen Namen haben.
```

Typhinweise haben den zusätzlichen Vorteil, dass mit ihnen die API-Dokumentation direkt in die Klasse eingebettet wird. Wenn Sie sehen, dass ein Klassenkonstruktor ein Argument des Typs `Event` erwartet, wissen Sie genau, was Sie der Methode übergeben müssen. Außerdem wissen Sie, dass Code und »Dokumentation« immer übereinstimmen, da die »Dokumentation« direkt in die Klassendefinition eingraviert ist.

Typhinweise können Sie auch in Interface-Definitionen verwenden, um so ganz präzise Informationen zur Verwendung Ihres Interface anzugeben.

Allerdings führen Typhinweise dazu, dass man an Flexibilität verliert. Es gibt keine Möglichkeit, für einen Parameter mehrere Typen von Objekten zuzulassen, und das führt dazu, dass Sie beim Entwurf Ihrer Klassenhierarchie einigen Einschränkungen unterliegen.

Außerdem ist die Strafe für die Verletzung eines Typhinweises drakonisch – das Skript bricht mit einem fatalen Fehler ab. Bei einer Webanwendung möchten Sie eventuell präziser steuern, wie Fehler verarbeitet werden, damit Fehler dieser Art nicht gleich zu einem Programmabbruch führen. Wenn Sie in Ihren Methoden eine eigene Typprüfung implementieren, können Sie gegebenenfalls auch eine Fehlerseite ausgeben.

Und anders als in vielen anderen Sprachen können Sie Typhinweise nicht für Rückgabewerte verwenden. Es gibt also keine Möglichkeit, zu verlangen, dass eine bestimmte Funktion immer Objekte eines bestimmten Typs zurückliefert.

Siehe auch

Die Dokumentationen zu Typhinweisen unter *http://www.php.net/manual/language.oop5.typehinting.php* und zu `instanceof` unter *http://www.php.net/manual/language.operators.type.php*.

7.25 Klassendateien bei der Instantiierung von Objekten automatisch laden

Problem

Sie möchten Ihre Klassendefinitionen nicht in alle Seiten einschließen, stattdessen sollen dynamisch nur die geladen werden, die in der jeweiligen Seite benötigt werden.

Lösung

Nutzen Sie die magische Methode `__autoload()`:

```
function __autoload($klassenname) {
    include "$klassenname.php";
}

$person = new Person;
```

PHP-Versionen ab 5.3 verfügen über eine Unterstützung von Namensräumen. Wenn Sie eine Klasse aus einem bestimmten Namensraum instantiieren wollen, hat das einen Einfluss auf den Klassennamen, der `__autoload()` übergeben wird:

```
function __autoload($klassenname) {
    // Backslashes in Namespace durch Slashes ersetzen
    $datei = str_replace('\\', '/', $klassenname) . '.php';
    include $datei;
}

$person = new de\oreilly\phpckbk\NamespacePerson();
```

Im gezeigten Beispiel wird der String `de\oreilly\phpckbk\NamespacePerson` an `__autoload()` übergeben. Die Backslashs werden durch Slashes ersetzt, um den Namensraum auf ein Verzeichnis zu mappen, in dem die Datei *NamespacePerson.php* liegt.

Diskussion

Versuchen Sie, eine Klasse zu instantiieren, die nicht definiert ist, bricht PHP normalerweise die Ausführung mit einem fatalen Fehler ab, weil es nicht finden kann, was Sie haben wollen. Deswegen lädt man üblicherweise in einer Seite vorab alle Klassen, die eventuell in ihr verwendet werden könnten, egal ob sie nun verwendet werden oder nicht.

Das führt dazu, dass die Verarbeitungszeit wächst, da PHP alle Klassen parsen muss, auch die, die nie verwendet werden. Eine Lösung ist, fehlenden Code nach Bedarf zu laden, indem man die `__autoload()`-Methode nutzt, die aufgerufen wird, wenn Sie Klassen instantiieren, die nicht definiert sind.

Auf folgende Weise können Sie mit `include` alle Klassen einlesen, die in Ihrem Skript tatsächlich verwendet werden:

```
function __autoload($klassenname) {
    include "$klassenname.php";
}

$person = new Person;
```

`__autoload()` erhält als einziges Argument den Klassennamen. Im ersten Beispiel wird an diesen Namen die Erweiterung *.php* angehängt, und anschließend wird versucht, auf Basis von `$klassenname` eine Datei zu laden. Das zweite Beispiel zeigt eine Variante unter Berücksichtigung von Namespaces. Instantiieren Sie ein `Person`- oder `NamespacePerson`-Objekt, sucht die Funktion also *Person.php* in ihrem `include_path` bzw. *NamespacePerson.php* in einem Verzeichnis *de/oreilly/phpckbk* ausgehend vom `include_path`.

Kann `__autoload()` keine Klassendefinition für die Klasse laden, die Sie zu instantiieren versuchen, bricht PHP die Ausführung mit einem fatalen Fehler ab, es verhält sich also so, wie es sich verhält, wenn eine Klassendefinition nicht gefunden werden kann und kein Autoload verwendet wird.

Wenn Sie die PEAR-Namenskonvention übernehmen, zwischen den Wörtern einen Unterstrich einzuschieben, um die Dateihierarchie widerzuspiegeln, können Sie den Code in Beispiel 7-41 nutzen.

Beispiel 7-41: Klassen unter PEAR-Namenskonventionen automatisch laden

```
function __autoload($paketname) {
    // Auf dem Unterstrich trennen.
    $ordner = split('_', $paketname);
    // Auf Basis der Ordnerstruktur wieder zusammenfügen und dabei
    // die Konstante DIRECTORY_SEPARATOR verwenden, damit der Code
    // plattformübergreifend funktioniert.
    $pfad    =  join(DIRECTORY_SEPARATOR, $ordner);
    // Erweiterung anhängen.
    $pfad   .= '.php';

    include $pfad;
}
```

Mit dem Code in Beispiel 7-41 können Sie Folgendes machen:

```
$person = new Tiere_Person;
```

Ist die Klasse nicht definiert, wird `Tiere_Person` an `__autoload( )` übergeben. Die Funktion trennt den Klassennamen auf dem Unterstrich (`_`) auf und verknüpft ihn wieder mit dem `DIRECTORY_SEPARATOR`. Das wandelt den Namen auf Unix-Systemen in den String `Tiere/Person` um (unter Windows in `Tiere\Person`).

Dann wird die Erweiterung *.php* angehängt und schließlich die Datei *Tiere/Person.php* mit `include` eingelesen.

`__autoload( )` erhöht die Verarbeitungszeit, während die Klasse hinzugefügt wird, wird pro Klasse aber nur einmal aufgerufen. Mehrere Instanzen einer Klasse führen nicht dazu, dass `__autoload( )` mehrfach aufgerufen wird.

Bevor Sie in einer Produktionsumgebung `__autoload( )` einsetzen, sollten Sie messen, ob die Mehrarbeit, die durch das Öffnen, Lesen und Schließen mehrerer Dateien entsteht, keine größere Auswirkungen auf die Leistung hat als die Zeit, die für das zusätzliche Parsen nicht benötigter Klassen erforderlich ist.

Insbesondere wenn Sie einen Opcode-Cache wie APC oder Zend Accelerator einsetzen, können `__autoload( )` und `include_once` die Leistung negativ beeinflussen. Die besten Ergebnisse erhalten Sie dann, wenn Sie alle Ihre Dateien zu Anfang des Skripts mit `include` einlesen und achtgeben, dass Sie keine Datei doppelt einlesen.

Siehe auch

Die Dokumentationen zum Autoloading finden Sie unter *http://www.php.net/manual/language.oop5.autoload.php*.

7.26 Mehrere Autoload-Handler definieren

Problem

Sie nutzen in Ihrem Projekt intensiv Klassen aus Bibliotheken und Frameworks von Drittanbietern. In jedem Ihrer Skripten wären unzählige `include/require`-Statements notwendig, und daher sind Sie auf die `__autoload()`-Funktion von PHP umgestiegen, um so den Code für die Inkludierung zu kapseln. Die `__autoload()`-Funktion ist jedoch durch viele notwendige Fallunterscheidungen sehr komplex und unübersichtlich, und Sie möchten diese vereinfachen.

Lösung

Verwenden Sie die Autoloading-Funktionalität der Standard PHP Library (SPL), um den Quellcode für das Laden der unterschiedlichen Gruppen von Klassen auf mehrere Funk-

tionen – sogenannte Autoload-Handler – aufzuteilen. Sie sparen sich dadurch die Fallunterscheidungen, da die SPL nacheinander die Autoload-Handler aufruft und probiert, die gesuchte Klasse zu laden:

```
// includes/SomeClass.my.php
// --
// namespace MyProject {
//     class SomeClass {}
// }
//
// includes/SomeClass.lib.php
// --
// namespace TheLibrary {
//     class SomeClass {}
// }

function projectClassLoader($classname)
{
    echo __FUNCTION__ . " wurde aufgerufen \n";
    $file = str_replace('MyProject\\', 'includes/classes/', $classname) . '.my.php';
    if (file_exists($file)) {
        include $file;
        return true;
    }
    return false;
}

function libLoader($classname)
{
    echo __FUNCTION__ . " wurde aufgerufen \n";
    $file = str_replace('TheLibrary\\', 'includes/libs/', $classname) . '.lib.php';
    if (file_exists($file)) {
        include $file;
        return true;
    }
    return false;
}

spl_autoload_register('projectClassLoader');
spl_autoload_register('libLoader');

$myclass = new \MyProject\SomeClass();
print_r($myclass);

$library = new \TheLibrary\SomeClass();
print_r($library);
projectClassLoader wurde aufgerufen
MyProject\SomeClass Object
(
)
projectClassLoader wurde aufgerufen
libLoader wurde aufgerufen
TheLibrary\SomeClass Object
(
)
```

Diskussion

PHP erlaubt nur die Definition einer einzigen `__autoload()`-Funktion. Aller Quellcode, der beim Autoloading ausgeführt werden soll, um nicht gefundene Klassen zu laden, muss also zwangsläufig in diese Funktion geschrieben werden. Dadurch kann sie schnell unübersichtlich werden.

Das Codebeispiel zeigt zwei Funktionen, auf die das Autoloading aufgeteilt wurde: `projectClassLoader()` lädt Klassen, die wir für unser aktuelles Projekt erstellt haben, und `libLoader()` lädt Klassen der Bibliothek eines Drittanbieters. Jede Funktion implementiert also eine eigene Autoloading-Strategie und kapselt diese. Mit der Funktion `spl_autoload_register()` werden diese beiden Funktionen registriert, sodass sie ausgeführt werden, wenn PHP das Autoloading anstößt. Die Übersichtlichkeit des Quellcodes wird hierdurch deutlich erhöht.

Das gezeigte Vorgehen bietet sich zum Beispiel auch dann an, wenn man es mit einer Kombination aus PHP-Skripten zu tun hat, die zu einem Teil bereits von Namensräumen Gebrauch machen, zum anderen Teil aber noch nicht.

Siehe auch

Weitere Informationen zum Autoloading in PHP bietet die Seite *http://php.net/manual/language.oop5.autoload.php*. Die erweiterten Features der SPL werden auf der Seite *http://php.net/manual/ref.spl.php* beschrieben.

7.27 Objekte dynamisch instantiieren

Problem

Sie wollen ein Objekt instantiieren, kennen den Namen der Klasse aber erst zur Laufzeit. Beispielsweise möchten Sie Ihre Seite lokalisieren, indem Sie ein Objekt instantiieren, das einer bestimmten Sprache entspricht. Welche Sprache gewählt werden soll, erfahren Sie aber erst, wenn die Klasse angefordert wird.

Lösung

Geben Sie den Klassennamen über eine Variable an:

```
$sprache = $_REQUEST['language'];
$sprachen = array('en_US' => 'US English',
                  'en_UK' => 'British English',
                  'es_US' => 'US Spanish',
                  'fr_CA' => 'Canadian French');

if (isset($sprachen[$sprache]) && class_exists($sprache)) {
```

```
    $sprache = new $sprache;
}
```

In PHP-Versionen ab 5.3 haben Sie die Möglichkeit, von Namensräumen Gebrauch zu machen. Sollten Sie das tun, müssen Sie darauf achten, dass Sie dem Klassennamen den entsprechenden Namensraum voranstellen:

```
namespace de\oreilly\phpckbk {
    class Foo {}
}

namespace MeinProjekt {
    $classname = "de\oreilly\phpckbk\Foo";
    $f = new $classname;
}
```

Diskussion

Gelegentlich kennen Sie den Namen der Klasse nicht, die Sie zur Laufzeit instantiieren wollen, oder nur zu einem Teil. Um Ihrer Klassenhierarchie einen Pseudo-Namensraum zu geben, können Sie allen Klassennamen beispielsweise eine bestimmte Zeichenfolge voranstellen (in PHP-Versionen vor 5.3 ohne Namensraumunterstützung war dies gängige Praxis). Wir nutzen in diesem Buch gelegentlich `pk_` für *PHP-Kochbuch*. PEAR nutzt beispielsweise `Net_` vor allen netzwerkbezogenen Klassen.

Aber während Folgendes in PHP zulässig ist:

```
$klassenname = 'Net_Ping';
$klasse = new $klassenname;               // new Net_Ping
```

ist dies nicht zulässig:

```
$partieller_klassenname = 'Ping';
$klasse = new "Net_$partieller_klassenname"; // new Net_Ping
```

Folgendes hingegen ist wieder okay:

```
$partieller_klassenname = 'Ping';
$klassenpraefix = 'Net_';

$klassenname = "$klassenpraefix$partieller_klassenname";
$klasse = new $klassenname;               // new Net_Ping
```

Sie können ein Objekt also nicht instantiieren, wenn der Name der entsprechenden Klasse im gleichen Schritt erst über Variablenverkettung konstruiert wird. Aber da Sie Variablen verwenden können, auf die keine Operationen angewandt werden, müssen Sie den Klassennamen einfach nur zuvor aufbauen.

Siehe auch

Die Dokumentationen zu `class_exists()` unter *http://www.php.net/class-exists* und zu Namensräumen unter *http://php.net/namespaces*.

7.28 Eine Anwendung: whereis

Obwohl Werkzeuge wie phpDocumentor ziemlich ausführliche Informationen zu einem ganzen Satz von Klassen liefern, kann es nützlich sein, sich schnell eine Aufstellung zu verschaffen, die alle Funktionen und Methoden aufführt, die in einer Liste von Dateien definiert werden.

Das Programm in Beispiel 7-42 durchläuft eine Liste mit Dateien, liest sie ein und nutzt dann die Reflection-Klassen, um Informationen zu ihnen zu sammeln. Ist die Master-Liste zusammengestellt, werden die Funktionen und Methoden alphabetisch sortiert und ausgegeben.

Beispiel 7-42: whereis

```
<?php
if ($argc < 2) {
    print "$argv[0]: klasse1.php [, ...]\n";
    exit;
}

// Die Dateien einlesen.
foreach (array_slice($argv, 1) as $dateiname) {
    include_once $dateiname;
}

// Alle Informationen zu Methoden und Funktionen abrufen.
// Zunächst die Klassen ...
$methoden = array();
foreach (get_declared_classes() as $klasse) {
    $r = new ReflectionClass($klasse);
    // Eingebaute Klassen streichen.
    if ($r->isUserDefined()) {
        foreach ($r->getMethods() as $methode) {
            // Geerbte Methoden streichen.
            if ($methode->getDeclaringClass()->getName() == $klasse) {
                $signatur = "$klasse::" . $methode->getName();
                $methoden[$signatur] = $methode;
            }
        }
    }
}

// Dann die Funktionen ...
$funktionen = array();
$definierte_funktionen = get_defined_functions();
foreach ($definierte_funktionen['user'] as $funktion) {
    $funktionen[$funktion] = new ReflectionFunction($funktion);
}

// Methoden alphabetisch über die Klasse sortieren.
function methoden_sortieren($a, $b) {
```

Beispiel 7-42: whereis (Fortsetzung)

```
    list ($a_klasse, $a_methode) = explode('::', $a);
    list ($b_klasse, $b_methode) = explode('::', $b);

    if ($cmp = strcasecmp($a_klasse, $b_klasse)) {
        return $cmp;
    }

    return strcasecmp($a_methode, $b_methode);
}
uksort($methoden, 'methoden_sortieren');

// Funktionen alphabetisch sortieren.
// Das ist nicht sehr kompliziert, Sie dürfen allerdings nicht vergessen,
// die Funktion zum Sortieren der Methoden zu entfernen.
unset($funktionen['methoden_sortieren']);
// Sortieren.
ksort($funktionen);

// Die gesammelten Daten ausgeben.
foreach (array_merge($funktionen, $methoden) as $name => $reflect) {
    $datei = $reflect->getFileName();
    $zeile = $reflect->getStartLine();

    printf ("%-25s | %-40s | %6d\n", "$name()", $datei, $zeile);
}
?>
```

Dieser Code nutzt die Reflection-Klassen und einige PHP-Funktionen, `get_declared_classes( )` und `get_declared_functions( )`, die nicht Teil der Reflection-Klassen sind, uns bei der Untersuchung des Codes aber helfen.

Es ist wichtig, dass Sie alle eingebauten PHP-Klassen und -Funktionen herausfiltern, da der Bericht sich sonst weniger mit Ihrem Code als mit Ihrer PHP-Installation befasst. Das erfolgt auf zweierlei Weise. Da `get_declared_classes( )` nicht zwischen benutzerdefinierten und internen Klassen unterscheidet, ruft der Code `ReflectionClass::isUserDefined( )`, um das zu prüfen. Der `get_defined_function( )`-Aufruf hingegen kümmert sich bereits für Sie um diese Angelegenheit, indem er die für Sie relevanten Informationen in das Array-Element `user` packt.

Da man die Ausgabe einer sortierten Liste leichter erfassen kann, sortiert das Skript die Arrays mit den Methoden und Funktionen. Weil mehrere Klassen Methoden mit dem gleichen Namen haben können, müssen wir eine eigene Sortierfunktion, `methoden_sortieren( )`, entwerfen, die zwei Methoden erst anhand ihres Klassennamens und dann anhand ihres Methodennamens vergleicht.

Sind die Daten einmal sortiert, ist es recht leicht, die verschmolzenen Arrays zu durchlaufen und dabei die Dateinamen und die Nummern der Anfangszeilen zu sammeln und einen Bericht auszugeben.

Folgendes Ergebnis erhält man, wenn man das Skript über die PEAR-Klasse HTTP laufen lässt:

```
HTTP::Date()              | /usr/lib/php/HTTP.php                    |     38
HTTP::head()              | /usr/lib/php/HTTP.php                    |    144
HTTP::negotiateLanguage() | /usr/lib/php/HTTP.php                    |     77
HTTP::redirect()          | /usr/lib/php/HTTP.php                    |    186
```

KAPITEL 8

Effizienter Umgang mit Daten

8.0 Einführung

Gute Programme bestehen aus sauberem Code, den Sie (oder andere) warten und ausbauen können, ohne dass daraus ein Chaos entsteht. Dazu gehört, dass Code nicht unnötig dupliziert wird. Es ist aber auch wichtig, Details so unterzubringen, dass sie nicht den Blick aufs Ganze verstellen.

Wenn Sie zum Beispiel mit Funktionen in PHP arbeiten, gruppieren Sie Codeanweisungen, die eine bestimmte Funktionalität ausführen. Später reicht es dann, die Funktion aufzurufen, statt alle Anweisungen noch einmal zu schreiben. Damit wird Ihr Code viel lesbarer. Bei Objekten ist es ähnlich: Sie gruppieren Eigenschaften und Methoden, die zusammengehören. Dann arbeiten Sie mit dem Objekt statt mit einer unübersehbaren Menge von Einzelteilen.

Dieses Kapitel behandelt vier neue Aspekte von PHP, die in Version 5 dazugekommen sind: *Iteratoren*, *Streams*, *Wrapper* und *Filter*. Einigen dieser Konzepte, z.B. Iteratoren, sind Sie vielleicht schon in einem der anderen Kapitel begegnet, wo sie in konkreten Situationen eingesetzt werden, z.B. in den Rezepten 22.7, 22.8, 22.9 und 22.10. Zusammen haben die vier eines gemeinsam: Sie ermöglichen es Ihnen, sich in Ihrem Code aufs Wesentliche zu konzentrieren und die Details an andere Stellen zu verbannen.

Iteratoren sind die Antwort auf ein häufiges Problem: Sie haben eine Datenstruktur, die Sie gern durchlaufen würden: ein Objekt mit einem internen Array, das Sie öfter durchlaufen müssen, oder ein Verzeichnis oder auch ein Datenbankresultat. Ohne Iteratoren geht das z.B. bei einem Objekt mit internem Array so:

```
foreach ($meinObjekt->internesArray as $schluessel => $arrayElement) {
    // Etwas mit $arrayElement anfangen.
}
```

Dabei fangen wir aber schon wieder an, verschiedene Aspekte zu mischen: Anstatt das `foreach`-Statement direkt mit einem Array zu beschicken, holen wir uns das Array vor

Ort eben schnell noch aus dem Objekt. Wenn das häufiger passiert, wäre es einfacher, es so zu machen:

```
foreach ($meinObjekt as $schluessel => $arrayElement) {
    // Etwas mit $arrayElement anfangen.
}
```

Hier müssten wir uns nicht um das Dereferenzieren des Objekts kümmern und könnten es einfach wie ein Array behandeln. Aber wie weiß PHP, über welche Array-Eigenschaft von `$meinObjekt` es mit `foreach` iterieren soll? Das können Sie durch die Implementierung eines Iterators festlegen.

Ein Iterator in PHP ist ein Objekt einer Klasse, die das `Iterator`-Interface implementiert. Es besteht aus fünf als `public` deklarierten Methoden: `current()`, `key()`, `next()`, `rewind()` und `valid()`. Wie bei anderen Schnittstellen muss Ihre Objektklasse auch hier diese Methoden selbst definieren; nur Name und Parameter (in diesem Fall keine) sind vorgegeben.

Die Aufgaben der fünf Methoden liegen auf der Hand: `current()` liefert das aktuelle Element zurück, also den Wert für `$arrayElement` im obigen Beispiel. Die Methode `key()` liefert den zugehörigen Wert für `$schluessel`. Um zum nächsten Element vorzurücken, ruft `foreach` die `next()`-Methode des Interface auf. Mit `rewind()` sollten Sie auf das erste Element zurückspringen – diese Methode wird aufgerufen, bevor `foreach` mit seiner Iteration beginnt. Sie wird außerdem von `reset()` verwendet. Die `valid()`-Methode überprüft vor dem Aufruf von `current()`, ob das gegenwärtige Element überhaupt existiert. Wie das im Zusammenspiel funktioniert, sehen Sie hier:

```
// Klassen-Deklaration mit Iterator-Interface
class TemperatureMeasurements implements Iterator {
    public $month;
    public $temperatures = array(); // Temperaturen für jeden Monat

    // Iterator-Interface
    private $n;

    public function current() { // Aktuelles Element zurückliefern.
        return $this->temperatures[$this->n];
    }

    public function key() { // Schlüssel des aktuellen Elements zurückliefern.
        return $this->n;
    }

    public function next() { // Zum nächsten Element vorrücken.
        $this->n++;
    }

    public function rewind() { // Array zurücksetzen.
        $this->n = 1;
    }
```

```
    public function valid() { // Aktuelles Element zurückliefern.
        return isset($this->temperatures[$this->n]);
    }
}

$temp = new TemperatureMeasurements;
$temp->month = "Januar";
$temp->temperatures = array(1 => 4, 2 => 9, 3 => 7);

foreach ($temp as $day => $tempDeg) {
    echo "Die Temperatur am $day. $temp->month  war $tempDeg Grad<br/>";
}
```

Wenn Sie wie hier nur über ein internes Array des Objekts iterieren wollen, zeigt Ihnen Rezept 8.2 einen noch einfacheren Weg, über den Sie sich das gesonderte Implementieren der fünf Methoden sparen und durch einen Dreizeiler ersetzen können. Rezept 8.1 zeigt Ihnen, wie Sie über die Eigenschaften eines Objekts iterieren können, und Rezept 8.3 gibt Ihnen die Möglichkeit, die Spielregeln beim Iterieren komplett neu bestimmen zu können.

Streams sind eine andere Neuerung in PHP 5. Sie haben sie vielleicht schon in anderen Kapiteln erspäht: Jedes Rezept in diesem Buch, das mit `fopen()`, `file_get_contents()` oder `file_put_contents()` arbeitet, verwendet einen Stream. Die Grundidee von Streams besteht darin, dass es eigentlich egal sein sollte, ob Sie Ihre Daten aus einer Datei, über das Netz, aus gemeinsam genutzten Speicherbereichen oder auch einfach aus einem selbst deklarierten Objekt beziehen. Mit Streams können Sie beim Lesen und Schreiben von Daten immer dieselbe Schnittstelle verwenden – egal wo Ihre Daten herkommen oder hinsollen. Sie können also z.B. Ihren Code erst mit simulierten Daten aus einer Datei testen und dann ohne große Änderung mit echten Daten verwenden, die Sie über das Web beziehen.

Um Ihre eigenen Objekte als Datenquelle für Streams nehmen zu können, brauchen Sie einen Wrapper. Ein Wrapper ist für eine Stream-Funktion wie `file_get_contents()` das, was ein Iterator für `foreach` ist: die Schnittstelle, über die sie an die Daten kommen oder die Daten schreiben kann. Rezept 8.4 zeigt Ihnen, wie Sie einen eigenen Wrapper implementieren.

Filter sind eine sehr nützliche Ergänzung zu Streams. Wenn Sie einen Stream mit einem Filter versehen, können Sie beim Schreiben oder Lesen nur die Daten durchlassen, die Sie wollen. Rezept 8.5 zeigt Ihnen, wie Sie Filter an Ihren Stream anhängen, Rezept 8.6 beschreibt den Eigenbau von Filtern.

Seit PHP 5.3 bietet die Standard PHP Library (SPL) eine Handvoll nativ implementierter Datenstrukturen mit objektorientierter Schnittstelle. Die Rezepte 8.7 und 8.8 zeigen, wie Sie diese nutzen können.

8.1 Über die Eigenschaften eines Objekts iterieren

Problem

Sie wollen über die öffentlichen Eigenschaften eines Objekt iterieren können, um die Eigenschaftsnamen und -werte auszugeben. Das kann beispielsweise zur Diagnose nützlich sein.

Lösung

Sofern Ihre Klasse keinen speziellen Iterator implementiert, ist die Iteration über die Eigenschaften das Default-Verhalten der Objekte dieser Klasse:

```
class Person {
    public $name;
    public $address;
    private $pin = "3958";

    // ...
}

$fred = new Person;
$fred->name = "Fred";
$fred->address = "Stonehenge";

foreach ($fred as $property => $value) {
    echo "$property:$value<br/>";
}
```

Diskussion

Das obige Beispiel gibt aus:

```
name:Fred
address:Stonehenge
```

Wie Sie sehen können, wird nur über die als `public` deklarierten Eigenschaften iteriert. Eigenschaften, die im Sichtbarkeitsbereich der `foreach`-Schleife nicht sichtbar sind, werden auch nicht ausgegeben. Ebenso können Sie nicht über Eigenschaften iterieren, die mit `__set()` gesetzt und/oder mit `__get()` gelesen werden.

Siehe auch

Rezept 8.2 zur Änderung des Iterationsverhaltens eines Objekts; Rezept 8.3 zur Implementierung eines eigenen Iterators; die Dokumentation zur Objekt-Iteration in PHP unter *http://www.php.net/manual/en/language.oop5.iterations.php*.

8.2 Einfache Objekt-Iteration mit IteratorAggregate und ArrayObject

Problem

Sie würden gern über eine Array-Eigenschaft in Ihrem Objekt iterieren. Die Implementierung der fünf Methoden des `Iterator`-Interface sind Ihnen aber zu aufwendig.

Lösung

Verwenden Sie das `IteratorAggregate`-Interface und geben Sie über seine `getIterator()`-Methode ein `ArrayObject` Ihres Arrays zurück:

```
class TemperatureMeasurements implements IteratorAggregate {
    public $month;
    public $temperatures = array(); // Temperaturen für jeden Monat

    // IteratorAggregate-Interface
    public function getIterator() {
        return new ArrayObject($this->temperatures);
    }
}

$temp = new TemperatureMeasurements;
$temp->month = "Januar";
$temp->temperatures = array(1 => 4, 2 => 9, 3 => 7);

foreach ($temp as $day => $tempDeg) {
    echo "Die Temperatur am $day. $temp->month  war $tempDeg Grad<br/>";
}
```

Diskussion

Das `IteratorAggregate`-Interface stellt Ihnen eine Alternativlösung zur Verfügung: Wenn Sie bereits einen Iterator oder ein Array – irgendetwas Iterierbares eben – in Ihrem Objekt vorliegen haben, brauchen Sie das Rad ja nicht noch einmal neu zu erfinden. Stattdessen können Sie z.B. `foreach` einfach die bestehende Datenstruktur nutzen lassen. Dazu müssen Sie diese Datenstruktur über `getIterator()` zugänglich machen. Weil Ihr Objekt `IteratorAggregate` implementiert, weiß PHP, dass diese Methode vorhanden sein muss.

Die von `getIterator()` zurückgegebene Datenstruktur muss entweder ein Objekt sein, das die `Iterator`-Schnittstelle implementiert, oder es muss ein `ArrayObject` sein. Ein normales Array konvertieren Sie einfach in ein `ArrayObject`, indem Sie es dem Konstruktor von `ArrayObject` übergeben:

```
$myArrayObject = ArrayObject($myArray);
```

So einfach ist das. Damit wird es dann auch ein Kinderspiel, das Objekt entscheiden zu lassen, über was genau iteriert wird:

```
public function getIterator() {
    switch ($this->what) {
        case "air": return new ArrayObject($this->airTemperatures);
        case "water": return new ArrayObject($this->waterTemperatures);
    }
}
```

Siehe auch

Rezept 8.3 zur Implementierung Ihres eigenen Iterators; die Dokumentationen zu `IteratorAggregate` und `ArrayObject` unter *http://www.php.net/~helly/php/ext/spl/* und *http://www.php.net/spl*.

8.3 Einen eigenen Iterator implementieren

Problem

Sie wollen eine Reihe von Werten aus einem Objekt in einer `foreach`-Schleife auslesen. Die Werte sind aber nicht in einer Array-Eigenschaft gespeichert. Zum Beispiel wollen Sie die Werte erst zum Zeitpunkt der Iteration berechnen. Beispiel: Sie wollen die Namen der nächsten fünf Tage ausgeben.

Lösung

Implementieren Sie das `Iterator`-Interface, bestehend aus den Methoden `rewind()`, `valid()`, `next()`, `key()` und `current()`:

```
class DayIterator implements Iterator{

    public $n; // Zahl der auszugebenden Tage
    public $today;
    private $days = array(
        0 => "Sonntag",
        1 => "Montag",
        2 => "Dienstag",
        3 => "Mittwoch",
        4 => "Donnerstag",
        5 => "Freitag",
        6 => "Samstag"
    );
    private $currentDay;
    private $daysLeft;

    function __construct($n = 7) {
        $this->n = $n;
```

```
            $this->today = date("w");
        }

        // Iterator-Interface:
        function rewind() { // Zurücksetzen des Iterators
            $this->currentDay = $this->today;
            $this->daysLeft = $this->n;
        }

        function valid() { // Noch weitere Tage auszugeben?
            return $this->daysLeft > 0;
        }

        function next() { // Zum nächsten Tag vorrücken.
            $this->currentDay++;
            if ($this->currentDay > 6) $this->currentDay = 0;
            $this->daysLeft--;
        }

        function key() { // Tageszahl des momentanen Tages ausgeben.
            return $this->currentDay;
        }

        function current() { // Momentanen Tag ausgeben.
            return $this->days[$this->currentDay];
        }
    }

    $dayIterator = new DayIterator(5);
    foreach ($dayIterator as $dayNo => $dayName) {
        echo "$dayNo:$dayName<br />";
    }
```

Diskussion

Das vollständige `Iterator`-Interface besteht aus fünf Methoden:

- `rewind()`: Diese Methode wird zu Beginn der Iteration aufgerufen und setzt den Iterator auf seinen Ausgangswert (das erste Array) zurück. Im obigen Beispiel wird dafür gesorgt, dass das aktuelle Element des Iterators auf den heutigen Tag gesetzt wird.
- `valid()`: Mit dieser Methode testet PHP, ob noch weitere Elemente vorliegen. Sie wird jeweils vor `current()` aufgerufen, damit nur auf Elemente zugegriffen wird, die auch tatsächlich existieren (bzw., wie in diesem Fall, berechnet werden sollen). Wenn weitere Elemente vorliegen, wird `true` zurückgegeben, ansonsten `false`.
- `next()`: Wenn der Iterator zum nächsten Element vorrücken soll, wird `next()` aufgerufen. Beachten Sie, dass `next()` erst dann zum ersten Mal aufgerufen wird, wenn das erste Element bereits ausgelesen worden ist. `next()` wird auch dann noch einmal aufgerufen, wenn keine weiteren Elemente mehr vorliegen – achten Sie darauf, dass

hier keine Probleme entstehen können (z.B. Zugriffsversuch auf nicht existierende Ressourcen innerhalb von `next()`).

- `key()`: Mit dieser Methode lesen Sie den Schlüssel des Elements aus. Sie wird übrigens erst nach `current()` aufgerufen, auch wenn Ihr `foreach`-Statement etwas anderes vermuten lässt!
- `current()` gibt das aktuelle Element zurück.

Die Aufrufsequenz der Methoden im obigen Beispiel ist also `rewind()`, `valid()`, `current()` (Ausgabe des heutigen Tages), `key()` (Ausgabe der heutigen Tagesnummer, z.B. 0 für Sonntag, 1 für Montag usw.), `next()`, `valid()`, `current()` (Ausgabe des folgenden Tages), usw. Die Iteration endet mit einem Aufruf von `valid()`, der `false` zurückliefert.

Eine etwas ungewöhnliche Anwendung eines Iterators bietet die unten gezeigte Klasse `SoundGenerator`. Sie verwendet einen Iterator, mit dem sich die Komponenten einer Wave-Datei (*.wav*, Standard-Audioformat unter Windows) in einer `foreach`-Schleife der Reihenfolge nach auslesen lassen. Der Header der Datei wird vom Iterator byteweise ausgegeben. Anschließend folgen die Signalwerte in jeweils zwei Bytes:

```
class SoundGenerator implements Iterator {
    private $header;
    private $length;
    private $frequencyFactor;
    private $modFrequencyFactor;
    private $amplitude;
    private $samplingRate;
    public $fileLength;
    private $readouts;

    // Konstruktor: Tonlänge in Sek., Grundfrequenz, Amplitude, Samplingrate,
    // Modulationsfrequenz
    public function __construct($length = 5, $frequency = 440, $amplitude = 1,
                                $samplingRate = 8000, $modFrequency = 0.7) {
        $this->length = $length;
        $this->samplingRate = $samplingRate;
        $this->frequencyFactor = $frequency * 2 * pi() / $samplingRate;
        $this->modFrequencyFactor = $modFrequency * 2 * pi() / $samplingRate;
        $this->amplitude = 32767 * $amplitude;
        $this->fileLength = 36 + $length * 2 * $samplingRate; // Dateilänge minus 8 Bytes
        $this->readouts = $length * $samplingRate + 1; // Anzahl der auslesbaren Werte
    }

    // Iterator-Interface
    private $n;

    public function current() { // Aktuelles Element zurückgeben.
        if ($this->n == 0) {
            return $this->header; // Header ausgeben.
        }
        else
        {
```

```
            // Sample berechnen und ausgeben.
            return pack("s",$this->amplitude *
                            sin($this->n * $this->frequencyFactor) *
                            sin($this->n * $this->modFrequencyFactor));
        }
    }

    public function key() { // Index des ausgelesenen Werts zurückgeben.
        return $this->n;
    }

    public function next() { // Zum nächsten Ausgabewert vorrücken.
        $this->n++;
    }

    public function rewind() { // Array zurücksetzen.
        // Header bilden:
        $this->header = "RIFF"; // Schema: 4 Bytes
        $this->header .= pack("V",$this->fileLength); // Datenlänge: 4 Bytes
        $this->header .= "WAVEfmt "; // Typ & Format-Header: 8 Bytes
        $this->header .= pack("V",16); // Länge Formatinfo: 4 Bytes
        $this->header .= pack("v",1); // Kompressionscode (1 = keine)
        $this->header .= pack("v",1); // Anz. der Kanäle (1 = mono)
        $this->header .= pack("V",$this->samplingRate); // Samples pro Sek.
        $this->header .= pack("V",2 * $this->samplingRate); // Bytes pro Sek.
        $this->header .= pack("v",2); // Bytes pro Sample * Nr. der Kanäle
        $this->header .= pack("v",16); // Bits pro Sample
        $this->header .= "data"; // Daten-Header gefolgt von Datenlänge
        $this->header .= pack("V",2 * $this->samplingRate * $this->length);
        // Index zurücksetzen.
        $this->n = 0;
    }

    public function valid() { // true zurückgeben, wenn noch Werte vorliegen.
        return $this->n < $this->readouts;
    }
}

$sound = new SoundGenerator;
header("Content-type: audio/wav");
foreach ($sound as $data) {
    echo $data;
}
```

Sie können die Klasse wie gezeigt zum Download der gesamten Datei verwenden oder aber auch zum Auslesen und Weiterverwenden einzelner Samples oder des Headers innerhalb einer `foreach()`-Schleife.

Und so funktioniert's: Der Konstruktor nimmt die Tonparameter entgegen: Länge des Tons in Sekunden, Grundfrequenz (Default: Kammerton A), Amplitude (1 = maximale Amplitude), Samplingfrequenz, Modulationsfrequenz (wie oft der Ton pro Sekunde an-

und abschwillt). Daraus errechnet er Längenparameter für den Header unseres virtuellen Wave-Dokuments und Konstanten für die Tonerzeugungsformel.

Beim Iterieren wird zunächst `rewind()` aufgerufen. Die Methode berechnet den Header des Tondokuments aus den aktuellen Parametern und setzt den Zähler `$n` auf 0, der als Zeiger für die Ausgabeposition dient. Dann wird `valid()` aufgerufen und überprüft, ob `$n` die Zahl der vorgesehenen Samples übersteigt.

Wenn `valid()` `true` zurückgibt, liegen noch Daten vor. Das iterierende Statement (z.B. `foreach`) ruft dann `current()` auf. Steht die Ausgabeposition auf 0, wird der Header ausgegeben, in allen anderen Fällen das `$n`-te Tonsample. Das Tonsample wird jeweils direkt aus seiner durch `$n` angegebenen Position in der Datei berechnet. Anschließend rücken wir mit `next()` zur nächsten Sample-Position weiter.

Wenn Sie im Digitalzeitalter das Rauschen Ihres guten alten Dampfradios vermissen, können Sie diese Klasse im Handumdrehen zu einem Rauschgenerator umfunktionieren. Ändern Sie einfach die Zeile mit der Formel in `current()` zu:

```
return pack("s",$this->amplitude * rand(-32768,32767));
```

und die Zeile im Konstruktor, in der `$this->amplitude` gesetzt wird, zu:

```
$this->amplitude = $amplitude;
```

Außerdem können Sie sich in diesem Fall sämtliche Angaben zu Frequenzen im Konstruktor sparen.

Siehe auch

Die Dokumentationen zu Iteratoren unter *http://www.php.net/~helly/php/ext/spl/* und *http://www.php.net/spl*. Eine gute Dokumentation zu Wave-Dateien findet sich u.a. unter *http://www.sonicspot.com/guide/wavefiles.html*.

8.4 Einen eigenen Wrapper für Streams schreiben

Problem

Sie haben ein Objekt, aus dem Sie Daten mit einer Stream-Funktion auslesen wollen. Zum Beispiel wollen Sie die Wave-Daten aus Rezept 8.3 mit `file_get_contents()` auslesen können.

Lösung

Implementieren Sie eine Wrapper-Klasse mit den Wrapper-Methoden `stream_open()`, `stream_close()`, `stream_eof()`, `stream_flush()`, `stream_read()` und `stream_stat()`:

```
class SoundWrapper {
    private $soundGenerator;
    private $buffer;

    public function stream_open($path, $mode, $options, &$opened_path) {
        // Parameterpaare aus Pfad extrahieren.
        list($protocol,$specs) = explode("://",$path);
        $pairs = explode("/",$specs);
        // Default-Werte
        $length = 5;
        $frequency = 440;
        $amplitude = 1;
        $samplingRate = 8000;
        $modFrequency = 0.7;
        // Parameterpaare durchlaufen und testen.
        foreach ($pairs as $pair) {
            list($name,$value) = explode("=",$pair);
            if (!is_numeric($value))
                if ($options & STREAM_REPORT_ERRORS) {
                    trigger_error("Invalid parameter value \"$value\".", E_USER_WARNING);
                }
                return false;
            }
            switch ($name) {
                case "len": $length = $value; break;
                case "freq": $frequency = $value; break;
                case "ampl": $amplitude = $value; break;
                case "samp": $samplingRate = $value; break;
                case "mod": $modFrequency = $value; break;
            }
        }
        if (($mode != "r") && ($mode != "rb")) {
                if ($options & STREAM_REPORT_ERRORS) {
                    trigger_error("Ungültiger Modus (Modus muss \'r\' oder \'rb\'
                                sein). ", E_USER_WARNING);
                }
                return false;
           }
        $this->soundGenerator = new SoundGenerator($length, $frequency, $amplitude,
                                                   $samplingRate, $modFrequency);
        $this->soundGenerator->rewind();
        return true;
    }

    public function stream_close() {
        unset($this->soundGenerator);
        $this->output = "";
    }

    public function stream_eof() {
        return !($this->soundGenerator->valid() || strlen($this->buffer));
    }
```

```
    public function stream_flush() {
        return true;
    }

    public function stream_read($bytes) {
        while ((strlen($this->buffer) < $bytes) && ($this->soundGenerator->valid())){
            $value = $this->soundGenerator->current();
            $this->soundGenerator->next();
            $this->buffer .= $value;
        }
        $output = substr($this->buffer,0,$bytes);
        $this->buffer = substr($this->buffer,$bytes);
        return $output;
    }

    public function stream_stat() {
        return array("size",$this->soundGenerator->fileLength + 8);
    }

    public function __call($name, $a) {
        echo "Function $name not found!";
    }
}

header("Content-type: audio/wav");
// Wrapper registrieren ...
stream_wrapper_register("soundgen","SoundWrapper");
// ... und anwenden.
echo file_get_contents("soundgen://length=8");
```

Diskussion

Im Gegensatz zu anderen Features in PHP 5 (wie z.B. Iteratoren) haben Wrapper kein fest definiertes Interface. Das liegt daran, dass Wrapper mit sehr vielen unterschiedlichen Datenquellen bzw. Datenzielen umgehen können müssen. Auf einige kann zum Beispiel nicht geschrieben werden (z.B. auf `http`), andere sind u.U. nicht lesbar. Stattdessen definieren Sie einfach die Methoden, die von den für Sie wichtigen Stream-Funktionen gebraucht werden.

In diesem Fall ist die Stream-Funktion, die wir anwenden wollen, `file_get_contents()`. Der Wrapper muss daher alle die Methoden bereithalten, auf die `file_get_contents()` zugreift.

Die erste dieser Methoden ist `stream_open()`. Sie nimmt vier Parameter, den Pfad der zu öffnenden Ressource, den Modus, einen Optionswert und einen Pfadparameter, über den ggf. der tatsächlich geöffnete Pfad einer Datei zurückgegeben werden kann. Der für Sie wichtigste Parameter ist der Pfad. In ihm bringen Sie die Parameter für die Tongeneration unter, z.B.:

```
soundgen://len=8/freq=1000/samp=22100/mod=0.1
```

Dieser Pfad z.B. stellt eine Wave-Datei von 8 Sekunden Länge mit einer Grundfrequenz von 1.000 Hz, einer Samplingfrequenz von 22.100 pro Sekunde und einer Modulationsfrequenz von 0,1 Hz dar. Der soundgen-Teil identifiziert den Wrapper – dazu gleich mehr. stream_open() extrahiert die Parameter und überprüft, ob der Modus ein Lesemodus ist. Bei Fehlern in Modus oder Parameter werden mit trigger_error() Fehlermeldungen ausgegeben, wenn STREAM_REPORT_ERRORS in $options gesetzt ist. Mit den überprüften Parametern wird ein soundGenerator erzeugt und zurückgesetzt. stream_open() gibt true zurück, wenn die Methode erfolgreich war, false, wenn nicht.

Als nächste Methode ruft file_get_contents() stream_stat() auf, um herauszufinden, wie groß die Datei ist. Wir geben hier die fileLength-Eigenschaft des Soundgenerators zurück, plus 8 Bytes für den Header der Datei, die in fileLength nicht enthalten sind. Danach versucht file_get_contents() die Bytes auszulesen. Um einen eventuellen Puffer nicht dauernd zu überfordern, tut file_get_contents() das in kleinen Häppchen von jeweils 8.192 Bytes, die es über stream_read() anfordert.

stream_read() hat eine wichtige Aufgabe: Die Methode muss sicherstellen, dass genügend Daten aus dem Soundgenerator ausgelesen werden, um die Wünsche von file_get_contents() zu erfüllen. Nun können wir Daten aus dem Soundgenerator nur in Stücken, aber nicht in Bytes anfordern, so wie file_get_contents() es will. Außerdem sind die Stücke (Header oder Samples) ungleichmäßig groß. stream_read() schreibt die Daten darum in einen Buffer, dessen Inhalt dann bis zur gewünschten Anzahl Bytes ausgegeben wird. Dabei werden aber immer nur so viele Daten ausgelesen, dass der Buffer gerade genug für file_get_contents() hat – bei großen Dateien würde sonst viel Zwischenspeicher unnötig belegt.

Nach jedem Häppchen fragt file_get_contents() über stream_eof() nach, ob noch Daten zur Verfügung stehen. Der Grund dafür ist, dass sich bei bestimmten Datenquellen die Anzahl der zur Verfügung stehenden Bytes seit dem Aufruf von stream_stat() geändert haben könnte. Wenn noch Daten da sind, wird wieder stream_read() aufgerufen, und die Prozedur wiederholt sich.

Abschließend wird dann stream_flush() aufgerufen, um etwaige ungeschriebene Daten vor dem Schließen des Streams noch zu sichern, gefolgt von stream_close(). Fertig ist unser Wrapper – bis auf eine Kleinigkeit: Wir müssen ihn vor der Verwendung noch registrieren:

```
stream_wrapper_register("soundgen","soundWrapper");
```

Damit weiß PHP, dass es bei Pfadangaben, die mit soundgen:// anfangen, ein soundWrapper-Objekt erzeugen soll.

In manchen Situationen kann man Parameter für einen Wrapper nicht einfach im Pfad der Ressource unterbringen. Hier helfen Kontextoptionen weiter:

```
$options = array("soundgen" => array("len" => 12, "freq" => 880))
$context = stream_context_create($options);
file_get_contents("soundgen://", false, $context);
```

Wird ein Kontext übergeben, fügt PHP dem Wrapper-Objekt eine zusätzliche Eigenschaft namens `$context` hinzu. Aus dieser können Sie dann in `stream_open()` (oder anderswo in Ihrem Wrapper) wieder an Ihre Kontextoptionen kommen:

```
$options =  stream_context_get_options($this->context);
$length = empty($options["soundgen"]["length"]) ? 5 : $options["soundgen"]["length"];
...
```

Wenn Sie Ihre eigenen Wrapper implementieren wollen, ist es oft nicht ganz einfach herauszufinden, welche Wrapper-Methoden Sie implementieren müssen (insgesamt gibt es 17 mögliche Methoden, siehe *http://www.php.net/stream-wrapper-register*), um Ihre gewünschte(n) Stream-Funktion(en) zu unterstützen. Wenn Sie eine `__call()`-Methode in Ihre Wrapper-Klasse einfügen, die Ihnen sagt, welche Methoden nicht gefunden worden sind, brauchen Sie nicht zu raten.

Siehe auch

Rezept 8.3 beschreibt die `SoundGenerator`-Klasse; die Dokumentation zu Streams unter *http://www.php.net/manual/en/ref.stream.php*; die Dokumentationen zu Wrappern und Wrapper-Methoden sowie `stream_wrapper_register()` unter *http://www.php.net/stream-wrapper-register*, zu `stream_context_create()` unter *http://www.php.net/stream-context-create* und zu `stream_context_get_options()` unter *http://www.php.net/stream-context-get-options*.

8.5 Einen Stream filtern

Problem

Sie wollen die Daten eines Streams filtern oder modifizieren.

Lösung

Hängen Sie einen Filter an den Ein- oder Ausgang des Streams mit `stream_filter_prepend()` oder `stream_filter_append()`. Wenn Sie direkt auf das Stream-Handle zugreifen können, gehen Sie beim Anhängen an die Stream-Ausgabe so vor:

```
// Datei öffnen, deren Inhalt zu Großbuchstaben konvertiert werden soll.
$fh = fopen("myText.txt","r");
// Filter an Stream-Ausgabe anhängen.
stream_filter_append($fh,"string.toupper");
// Daten aus Stream lesen und Stream schließen.
while (!feof($fh)) {
    echo fgets($fh,256);
}
fclose($fh);
```

Bei Schreiboperationen hängen Sie den Filter an die Stream-Eingabe:

```
// Unbearbeiteter HTML-Text
$html = "I am <b>bold</b> and <i>italic</i>.";
// Datei, in die Klartext ohne HTML geschrieben werden soll.
$fh = fopen("myText.txt","w");
// Filter vor den Stream hängen.
stream_filter_prepend($fh,"string.strip_tags");
// Daten in den Stream schreiben und Stream schließen.
fwrite($fh, $html);
fclose($fh);
```

Wenn Sie nicht direkt auf das Stream-Handle zugreifen können, verwenden Sie den Wrapper `php://filter`, z.B. beim Lesen:

```
// Filter mit php://filter hinter den Stream hängen.
echo file_get_contents("php://filter/read=string.toupper/resource=myText.txt");
```

und beim Schreiben:

```
// Unbearbeiteter HTML-Text
$html = "I am <b>bold</b> and <i>italic</i>.";
// Filter mit php://filter vor den Stream hängen.
file_put_contents("php://filter/write=string.strip_tags/resource=myText.txt",$html);
```

Diskussion

Mit Filtern können Sie die Modifikationsoperationen direkt in Ihr Datei-Handle integrieren. Damit werden sie gegenüber Funktionen, die das Handle verwenden, transparent. Sie können das Handle also z.B. einer Funktion übergeben, die HTML-Daten portionsweise auf das Handle schreibt, und diese Funktion sowohl zum Schreiben von HTML als auch von Klartext verwenden. Im letzteren Fall hängen Sie einfach den Filter `string.strip_tags` vor den Stream. Ihre Funktion merkt davon gar nichts.

Welche Filter Ihnen auf Ihrem System zur Verfügung stehen, erfahren Sie über:

```
print_r(stream_get_filters());
```

Normalerweise sollten dies sechs Filter sein: `convert.iconv.*`, `string.rot13`, `string.toupper`, `string.tolower`, `string.strip_tags` und `convert.*`.

Sie können auch mehrere Filter auf einen Stream loslassen:

```
$html = "I am <b>bold</b> and <i>italic</i>.";
$filter1 = "php://filter/write=string.strip_tags/resource=myText.txt";
$filter2 = "php://filter/write=string.toupper/resource=$filter1";
file_put_contents($filter2,$html);
```

Siehe auch

Rezept 8.6 zum Schreiben Ihres eigenen Filters; die Dokumentation zu Streams unter *http://www.php.net/manual/en/ref.stream.php*; die Dokumentationen zu `stream_filter_`

append() unter *http://www.php.net/stream-filter-append* und zu stream_filter_append() unter *http://www.php.net/stream-filter-append*; die Dokumentation zum php://filter-Wrapper unter *http://www.php.net/manual/en/wrappers.php.php*.

8.6 Eigene Filter schreiben

Problem

Sie würden gern einen Stream filtern, aber PHP hat nicht den passenden Filter dafür. Zum Beispiel möchten Sie überzählige Leerzeichen und anderen Whitespace aus einem String entfernen, bevor Sie ihn nach php://output ausgeben.

Lösung

Implementieren Sie Ihren eigenen Filter:

```
class WhitespaceFilter extends php_user_filter {
    function filter($in, $out, &$consumed, $closing) {
        while ($bucket = stream_bucket_make_writeable($in)) {
            $consumed += $bucket->datalen;
            $bucket->data = preg_replace("/(\s)\s+/","$1",$bucket->data);
            $bucket->datalen = strlen($bucket->data);
            stream_bucket_append($out, $bucket);
        }
        return PSFS_PASS_ON;
    }
}

stream_filter_register("convert.whitespace","WhitespaceFilter");

echo file_put_contents("php://filter/write=convert.whitespace/resource=php://output",
    $stringWithWhitespace);
```

Diskussion

Viele Internetsites entfernen überzähligen Whitespace aus ihren Seiten. Das verringert die Download-Zeiten, insbesondere für Benutzer mit Telefonmodems.

Ein Filter ist eine Klasse, die die Klasse php_user_filter erweitert und in der Sie die Methode filter() implementieren. Sie hat vier Parameter: zwei sogenannte bucket brigades (Eimerketten) $in und $out, mit denen Sie die Daten in der Form von bucket-Objekten angeliefert bekommen und sie nach dem Filtern wieder ausliefern, eine Variable $consumed, über die Sie kommunizieren, wie viele Daten bisher insgesamt verarbeitet wurden, und eine Variable $closing, die Ihnen über true den letzten Filterdurchgang mitteilt.

Sie bearbeiten dann die Buckets der bucket brigade $in der Reihe nach, erhöhen dabei $consumed immer um die Anzahl der bearbeiteten Zeichen und übergeben die bearbeiteten Zeichen an die Eimerkette $out. Konnten Sie alle Zeichen bearbeiten (was hier kein Problem ist), geben Sie die Konstante PSFS_PASS_ON zurück.

Um den Filter einzusetzen, müssen Sie ihn dann nur noch mit stream_filter_register() registrieren.

Manchmal lässt sich die Bearbeitung in filter() aber nicht vollständig durchführen, weil noch Daten fehlen. Das ist meist dann der Fall, wenn ein Filter größere zusammenhängende Muster erkennen und ersetzen soll. Dann kann es sein, dass die letzten Daten, die dem Filter in einem Durchgang übergeben werden, den Anfang eines solchen Musters darstellen. Ob das tatsächlich so ist, lässt sich aber erst beim nächsten Aufruf der filter()-Methode feststellen. Unter Umständen kann sich ein Muster auch über mehrere Eimerketten erstrecken. Dann ist es am besten, wenn Sie die eintreffenden Daten erst einmal abspeichern, bis Sie über $closing erfahren, dass alle Daten vorliegen.

Nehmen wir mal an, dass Ihre Firma sich – ganz im Trend liegend – von *Computer Experts* in *Compaganza* umbenennt. Sie sollen die Website umstellen. Was liegt näher, als das mit einem PHP-Filter zu machen? Er sollte dazu so aussehen:

```
class CompanyNameFilter extends php_user_filter {

    public function onCreate() {
        $this->data = "";
        $this->open = true;
        return true;
    }

    public function filter($in, $out, &$consumed, $closing) {
        while ($bucket = stream_bucket_make_writeable($in)) {
            $consumed = 0;
            $this->data .= $bucket->data;
            $this->bucket = $bucket;
        }
        if ($closing) {
            if ($this->open) {
                $consumed += strlen($this->data);
                $this->data = str_replace("Computer Experts","Compaganza",$this->data);
                $this->bucket->data = $this->data;
                $this->bucket->datalen = strlen($this->data);
                stream_bucket_append($out, $this->bucket);
                $this->open = false;
            }
            return PSFS_PASS_ON;
        }
        return PSFS_FEED_ME;
    }
}
```

Hier gibt es zwei Neuigkeiten, die Sie vielleicht nicht erwartet haben: die Methode `onCreate()`, die beim Erstellen des Filters aufgerufen wird, um ein paar Eigenschaften zu initialisieren, und ein `if`-Statement, mit dem überprüft wird, ob die Eigenschaft `$open` gesetzt ist. Es stellt sicher, dass der Codeblock mit der Namenskonversion nur einmal ausgeführt wird. Bei einigen Funktionen, z.B. bei `file_get_contents()`, kann es vorkommen, dass filter() mehrfach mit `$closing = true` aufgerufen wird. Ohne das `if`-Statement kann das den Stream derart durcheinander bringen, dass sogar Ihr Webserver abschmiert.

Siehe auch

Die Dokumentation zu *http://www.php.net/manual/en/function.stream-filter-register.php*.

8.7 Performancegewinn mit Arrays fester Größe erzielen

Problem

Sie arbeiten mit Arrays und verwenden ausschließlich numerische Indizes. Nun möchten Sie die Effizienz und Performance Ihrer Applikation steigern.

Lösung

Nutzen Sie die Klasse `SplFixedArray`, die von der Standard PHP Library (SPL) bereitgestellt wird. Sie bietet nahezu die gleiche Funktionalität wie ein normales PHP-Array, arbeitet dabei aber schneller:

```
$array = new SplFixedArray(100);

$array[0] = "Wert eins";
$array->offsetSet(99, "Letzter Wert");

foreach ($array as $idx => $value) {
    print "Index: $idx - Value: ";
    var_dump($value);
}

print $array->offsetGet(0) . "\n";

try {
    $array["key"] = "value";
} catch (Exception $e) {
    print $e->getMessage();
}
```

```
Index: 0 - Value: string(9) "Wert eins"
Index: 1 - Value: NULL
Index: 2 - Value: NULL
```

```
[...]
Index: 98 - Value: NULL
Index: 99 - Value: string(11) "Letzter Wert"
Wert Eins
Index invalid or out of range
```

Diskussion

Für die meisten PHP-Anwendungen reicht die Performance der normalen PHP-Arrays mit hoher Wahrscheinlichkeit aus. Wenn es aber in Ihrer Applikation auf jede Mikrosekunde ankommt, sollten Sie über die Verwendung von `SplFixedArray` nachdenken. Abgesehen von der Beschränkung auf numerische Indizes bietet es sämtliche Funktionalitäten eines normalen Arrays, jedoch bei besserer Performance.

Auch wenn es nicht so sehr auf Geschwindigkeit ankommt, kann der Einsatz der hier vorgestellten Klasse durchaus sinnvoll sein. Wenn Sie die maximale Anzahl der Elemente in einem Array kennen und nur numerische Indizes verwenden möchten, sollten Sie `SplFixedArray` nutzen. Das nimmt Ihnen die Arbeit der Konsistenzprüfung ab, denn die Datenstruktur selbst kümmert sich darum sicherzustellen, dass nur numerische Indizes verwendet werden bzw. die festgelegte Größe eingehalten wird. Werden die Konsistenzbedingungen durch Programmierfehler verletzt, wird die Klasse entsprechende Exceptions werfen.

Siehe auch

Weitere Informationen zu `SplFixedArray` finden Sie im PHP-Manual unter *http://php.net/manual/class.splfixedarray.php*. Informationen über weitere SPL-Datenstrukturen liefert Rezept 8.8.

8.8 Standard-Datenstrukturen nicht neu erfinden – Queues, Stacks und Co.

Problem

Sie benötigen in Ihrem Programm Datenstrukturen, die PHP nicht out-of-the-box bietet (z.B. Stack, Linked-List oder Queue). Sie könnten diese mithilfe von Arrays als eigene Klasse implementieren, sind sich aber nicht sicher, ob Sie dabei nicht unnötig das Rad neu erfinden.

Lösung

Seit PHP 5.3 bietet die Standard PHP Library (SPL) einige dieser Datenstrukturen an. Die Klasse `SplStack` implementiert eine LIFO-Datenstruktur (Last-In-First-Out) und somit eine Datenstruktur, die sich wie ein Stack (engl. für Stapel) verhält:

```
$stack = new SplStack();

$stack->push('a');
$stack->push('b');
$stack->push('c');

echo $stack->pop()."\n";
echo $stack->pop()."\n";
echo $stack->pop()."\n";
```
```
c
b
a
```

Die Klassen `SplHeap`, `SplMinHeap` und `SplMaxHeap` implementieren eine Heap-Datenstruktur. `SplHeap` ist dabei eine abstrakte Klasse und kann nicht direkt instantiiert werden. In einem Heap werden die einzelnen Elemente anhand ihrer Schlüsselwerte in einer Baumstruktur abgelegt. `SplMinHeap` und `SplMaxHeap` implementieren jeweils die von `SplHeap` als abstrakt deklarierte Funktion `compare()`. `SplMinHeap` legt die hinzugefügten Elemente in aufsteigend sortierter Reihenfolge ab, `SplMaxHeap` dagegen in absteigend sortierter Reihenfolge:

```
$heap = new SplMaxHeap();
$heap->insert('b');
$heap->insert('a');
$heap->insert('c');

echo $heap->extract()."\n";
echo $heap->extract()."\n";
echo $heap->extract()."\n";
```
```
c
b
a
```
```
$heap = new SplMinHeap();
$heap->insert('b');
$heap->insert('a');
$heap->insert('c');

echo $heap->extract()."\n";
echo $heap->extract()."\n";
echo $heap->extract()."\n";
```
```
a
b
c
```

Mit der Klasse `SplPriorityQueue` kann eine Queue erstellt werden, die ihre Elemente nach einer definierten Priorität sortiert und nicht nach der FIFO-(First-In-First-Out-)Reihenfolge einer »normalen« Queue:

```
$queue = new SplPriorityQueue();
$queue->insert('niedrig',    1);
$queue->insert('hoch',    3);
$queue->insert('mittel', 2);
```

```
echo 'TOP Element: '.$queue->top()."\n";
echo $queue->extract()."\n";
echo $queue->extract()."\n";
echo $queue->extract()."\n";
TOP Element: hoch
hoch
mittel
niedrig
```

Diskussion

Sämtliche SPL-Datenstrukturen sind in der Sprache C implementierte PHP-Klassen. Dadurch sind sie sehr effizient. Neben den gezeigten Klassen existieren noch diverse weitere Datenstrukturen. Die nachfolgende Tabelle gibt Ihnen einen Überblick über die verfügbaren Klassen:

Tabelle 8-1: Übersicht über die SPL-Datenstrukturen

Klasse	Beschreibung
`SplDoublyLinkedList`	Doppelt verkettete Liste.
`SplStack`	Stapel bzw. Stack-Datenstruktur.
`SpleQueue`	Schlange bzw. Queue-Datenstruktur.
`SplHeap`	Haufen bzw. Heap-Datenstruktur. Diese Klasse definiert die abstrakte Methode `compare()` und kann nicht direkt instantiiert werden.
`SplMaxHeap`	Heap-Datenstruktur, deren größtes Element immer an oberster Stelle liegt.
`SplMinHeap`	Heap-Datenstruktur, deren kleinstes Element immer an oberster Stelle liegt.
`SplPriorityQueue`	Eine Queue-Datenstruktur, deren Elementen Prioritäten zugeordnet sind, woraus sich die Sortierreihenfolge ergibt.
`SplFixedArray`	Eine Array-Datenstruktur fester Größe. Es sind nur Integer-Werte als Keys erlaubt, deren Wert zwischen 0 und der angegebenen maximalen Größe liegt.
`SplObjectStorage`	Eine Map-Datenstruktur, deren Schlüssel Objekte sind. Erlaubt ein Mapping von Objekten auf zugeordnete Daten oder kann als Mengen-Datenstruktur (Set) für Objekte genutzt werden.

Siehe auch

Wie auch PHP entwickelt sich die SPL ständig weiter, und es kommen eventuell neue Datenstrukturen hinzu. Aktuelle Informationen finden Sie im PHP-Manual unter *http://de.php.net/manual/spl.datastructures.php*. Die Rezepte 8.7 und 4.27 zeigen detailliert den Umgang mit `SplFixedArray` und `SplObjectStorage`.

KAPITEL 9

Fehlerbehandlung mit Exceptions

9.0 Einführung

Auch in den besten Programmen tauchen Fehler auf. Der Unterschied zu schlechten Programmen besteht darin, dass das Fehlerszenario vorhersehbar ist und die Fehler so abgefangen werden können, dass kein großer Schaden entsteht. Manchmal ist das einfach. In anderen Situationen kann theoretisch so viel schiefgehen, dass eine umfassende Fehlerbehandlung sehr aufwendig wird. Wie Ihr Code dann aussieht, wissen Sie vermutlich selbst: viele verschachtelte `if`-Statements. Einfach unüberschaubar, insbesondere wenn Sie den Normalfall nachvollziehen wollen, in dem ja alles klappen soll.

In vielen Programmiersprachen haben Sie ein Werkzeug an der Hand, mit dem Sie die `if`-Inflation zumindest teilweise unter Kontrolle kriegen können: die Behandlung von Ausnahmen (*Exception Handling*). Die Grundidee ist das erfrischend einfache Prinzip des Trial-and-Error: Zunächst wird versucht, den Code wie geplant auszuführen, ohne dabei Fehler von vornherein zu umgehen. Wenn ein Fehler (d.h. eine Exception) auftritt – bzw. *geworfen* wird –, fangen Sie die Exception einfach ab und kümmern sich um die Folgen.

In Ihrem Code packen Sie einfach alles das, was unter Umständen fatale Fehler produzieren kann, in einen `try`-Block und verarzten etwaige Probleme in einem dazugehörigen `catch`-Block:

```
try {
    // Code, in dem Fehler auftreten können.
}
catch (Exception $e) {
    // Code zur Fehlerbehandlung (Meldung, Loggen, Alternativmaßnahmen).
}
```

Dieses Werkzeug gibt es jetzt auch in PHP. Der kleine Wermutstropfen: Nur wenige Umstände in PHP produzieren bisher echte Exceptions – auch die meisten Fehler nicht. Es wird überwiegend lediglich eine Warnmeldung ausgegeben, der Code wird jedoch so gut es geht weiter ausgeführt. Selbst wenn ein zum Skriptabbruch führender Fehler (Fatal Error) vorliegt, gibt es in der Regel keine Exception.

Das gilt auch für Fälle, die in anderen Programmiersprachen als Paradebeispiele für Exceptions in der Dokumentation angegeben sind. Zum Beispiel verursacht eine Division durch null in PHP nur eine Warnung statt einer Exception. Das liegt daran, dass viele existierende PHP-Skripten Warnmeldungen unterdrücken (@-Zeichen vor dem Funktionsnamen) und anschließend Probleme ermitteln. So können Sie z.B. bei nicht vorhandenen Dateien einen Default-Inhalt angeben:

```
if (($file = @file("existiertNicht.txt")) === false) {
    $string = "Hier sollte eigentlich was anderes stehen. Tut es aber nicht.";
} else { $string = implode("\n", $file); }
echo $string;
```

Stellen Sie sich vor, dass `file()` hier eine Exception wirft. Dann versagt diese Fehlererkennungsmethode komplett.

Eingebaute Exceptions finden sich deshalb nur in zwei Bereichen in PHP: in ein paar wenigen neueren Modulen (allen voran DOM, aber auch da nur, wenn unbedingt nötig) sowie in Objektkonstruktoren von vordefinierten Objekten:

```
try {
    $foo = new DirectoryIterator("/pfad/nach/nirgendwo");
}
catch (Exception $e) {
    echo "Kann keinen DirectoryIterator für /pfad/nach/nirgendwo erzeugen!";
}
```

Bei Objektkonstruktoren ist dies sinnvoll: Mit einem Konstruktor wird immer ein Objekt zurückgegeben. Selbst wenn Sie explizit `return false` in einen Konstruktor schreiben, wird immer noch ein Objekt zurückgeliefert. Damit kann nicht über `false` signalisiert werden, dass etwas schiefgelaufen ist. Mit anderen Worten, die folgenden Zeilen sind nutzlos:

```
if ($foo = new DirectoryIterator("/pfad/nach/nirgendwo")) {
    // Etwas mit $foo machen.
}
else {
    echo "Kann keinen DirectoryIterator für /pfad/nach/nirgendwo erzeugen!";
}
```

Hier bricht die Codeausführung bereits in der Bedingung des `if`-Statements ab – wir kommen also um `try/catch` nicht herum.

In den Rezepten 9.1 bis 9.5 geht es um Themen der allgemeinen Fehlerbehandlung, darunter die Festlegung, wo Fehler ausgegeben werden, wie Sie benutzerdefinierte Funktionen zur Fehlerbehandlung schreiben und wie Sie Ihre Programme mit Debugging-Informationen ergänzen können.

Wenn Ihr Code sowohl Statements enthält, die Warnungen produzieren, als auch Statements, die Exceptions werfen, müssten Sie demnach sowohl `if`-Statements als auch `try/catch`-Blöcke zur Fehlerbehandlung einsetzen. Rezept 9.6 zeigt Ihnen, dass es auch

anders geht. Rezept 9.7 demonstriert, wie Sie Ausnahmen abfangen und damit umgehen können.

In Ihrem eigenen Code können Sie nach Belieben neue Exceptions definieren und werfen. Rezept 9.8 zeigt Ihnen, wie. Mit unterschiedlichen Ausnahmetypen umzugehen, demonstrieren die Rezepte 9.9, 9.10 und 9.12. Rezept 9.11 zeigt, wie Sie eine Behandlung aller Fehler durch eine zentrale Fehlerbehandlungsinstanz realisieren können. Die allgemeine Dokumentation zu Ausnahmen in PHP finden Sie unter *http://www.php.net/manual/en/language.exceptions.php*.

9.1 Fehlermeldungen vor Anwendern verbergen

Problem

Sie möchten, dass PHP-Fehlermeldungen für Anwender nicht sichtbar sind.

Lösung

Setzen Sie die folgenden Werte in Ihrer *php.ini* oder in der Konfigurationsdatei des Webservers:

```
display_errors =off
log_errors     =on
```

Sie können diese Variablen auch im Programmcode mit der Funktion `ini_set()` verändern, wenn Sie die Erlaubnis haben, diese Werte auf dem Server zu verändern:

```
ini_set('display_errors', 'off');
ini_set('log_errors', 'on');
```

Diese Einstellungen veranlassen PHP dazu, Fehlermeldungen nicht in HTML-Form an den Browser zu senden, sondern sie in das Fehlerprotokoll des Servers zu schreiben.

Diskussion

Wenn `log_errors` auf `on` gesetzt ist, werden Fehlermeldungen in das Fehlerprotokoll des Servers geschrieben. Sollen die PHP-Fehler in eine gesonderte Datei geschrieben werden, setzen Sie die Konfigurationsdirektive `error_log` auf den Namen dieser Datei:

```
error_log   = /var/log/php.error.log
```

oder:

```
ini_set('error_log', '/var/log/php.error.log');
```

Ist `error_log` auf `syslog` gesetzt, werden PHP-Fehlermeldungen unter Unix mit `syslog(3)` an den System-Logger gesendet bzw. unter Windows an den Event-Log.

Es gibt viele Fehlermeldungen, die Sie Ihren Anwendern zeigen möchten, zum Beispiel um ihnen mitzuteilen, dass sie ein Formular falsch ausgefüllt haben, aber Sie sollten die Benutzer vor Fehlern abschirmen, die möglicherweise eine Folge von Problemen in Ihrem Code sind. Es gibt zwei Gründe dafür: Erstens erscheinen solche Fehler (den Experten unter den Anwendern) unprofessionell oder (den Neulingen) verwirrend. Wenn beim Speichern von Formulardaten in eine Datenbank etwas schiefgeht, prüfen Sie den Rückgabecode aus der Datenbankabfrage und zeigen Ihren Benutzern eine Meldung, die sich entschuldigt und sie bittet, später noch einmal wiederzukommen. Die Ausgabe einer kryptischen Fehlermeldung direkt aus PHP fördert nicht eben das Vertrauen in Ihre Website.

Zweitens stellt es ein Sicherheitsrisiko dar, wenn Sie Anwendern solche Fehlermeldungen zeigen. Abhängig von der Datenbank und der Art des Fehlers kann die Fehlermeldung Informationen darüber enthalten, wie man sich bei Ihrer Datenbank oder beim Server anmeldet und wie dieser strukturiert ist. Mithilfe solcher Informationen können böswillige Benutzer einen Angriff auf Ihre Website inszenieren.

Wenn beispielsweise Ihr Datenbankserver nicht läuft und Sie versuchen, sich mit ihm über `mysql_connect()` zu verbinden, generiert PHP die folgende Warnung:

```
<br>
<b>Warning</b>:  Can't connect to MySQL server on 'db.example.com' (111) in
<b>/www/docroot/beispiel.php</b> on line <b>3</b><br>
```

Wird diese Warnung an den Browser eines Anwenders gesendet, erfährt dieser, dass Ihr Datenbankserver *db.example.com* heißt, und kann versuchen, ihn anzugreifen.

Siehe auch

Rezept 9.4 dazu, wie man Fehler protokolliert; die Dokumentation zu den PHP-Konfigurationsdirektiven unter *http://www.php.net/configuration*.

9.2 Einstellungen zur Fehlerbehandlung vornehmen

Problem

Sie möchten die Empfindlichkeit der Fehlerprotokollierung für eine bestimmte Seite verändern. Dadurch können Sie steuern, welche Fehlerarten gemeldet werden.

Lösung

Um die Fehlerarten einzustellen, über die sich PHP beklagt, verwenden Sie `error_reporting()`:

```
error_reporting(E_ALL);               // alles
error_reporting(E_ERROR | E_PARSE);   // alles Wesentliche
error_reporting(E_ALL & ~E_NOTICE);   // alles außer Hinweisen
```

Diskussion

Mit jedem generierten Fehler ist eine Fehlerart verbunden. Wenn Sie beispielsweise versuchen, `array_pop()` mit einem String aufzurufen, beschwert sich PHP mit: »This argument needs to be an array« (dieses Argument muss ein Array sein), da ein Pop nur bei Arrays möglich ist. Der mit dieser Meldung verknüpfte Fehlertyp ist `E_NOTICE` (Hinweis), das heißt ein Laufzeitproblem, das nicht zum Abbruch führt.

Standardmäßig ist als Grad der Fehlerprotokollierung `E_ALL & ~E_NOTICE & ~E_STRICT` eingestellt, also alle Fehler außer Hinweisen und Verstößen gegen die reine Lehre der neuesten PHP-Version. Das `&` ist ein logisches `UND` und das `~` ein logisches `NICHT`. Die *php.ini-recommended*-Konfigurationsdatei setzt allerdings den Grad der Fehlerprotokollierung auf `E_ALL`, also auf alle Fehlerarten.

Seit PHP 5 gibt es den Fehlertyp `E_STRICT`, seit 5.2 den Typ `E_RECOVERABLE_ERROR` und seit 5.3 zusätzlich `E_DEPRECATED` und `E_USER_DEPRECATED`. Tabelle 9-1 liefert genaue Informationen zu den einzelnen Fehlertypen. Zu `E_STRICT` muss man wissen, dass es nicht in `E_ALL` enthalten ist. Wenn Sie also alle in PHP existierenden Fehlerlevel aktivieren wollen, müssen Sie den Grad der Fehlerprotokollierung auf `E_ALL | E_STRICT` setzen.

Bei Fehlermeldungen, die als Hinweis (`E_NOTICE`) gekennzeichnet sind, handelt es sich um Laufzeitprobleme, die weniger ernst sind als Warnungen. Sie sind nicht zwingend falsch, aber sie weisen auf ein potenzielles Problem hin. Ein Beispiel für `E_NOTICE` ist der Fehler »Undefined variable« (undefinierte Variable), der bei dem Versuch auftritt, eine Variable zu verwenden, ohne ihr zuvor einen Wert zuzuweisen:

```
// Führt zu einem E_NOTICE-Fehler.
foreach ($array as $value) {
    $html .= $value;
}

// Führt nicht zu einer Fehlermeldung.
$html = '';
foreach ($array as $value) {
    $html .= $value;
}
```

Im ersten Fall ist beim ersten Durchlauf von `foreach` die Variable `$html` undefiniert. Daher weist PHP Sie darauf hin, dass Sie etwas an eine undefinierte Variable hängen. Im zweiten Fall wird `$html` vor der Schleife ein leerer String zugewiesen, um den `E_NOTICE`-Fehler zu vermeiden. Die obigen beiden Programmabschnitte generieren identischen Code, da der Vorgabewert einer Variablen ein Leer-String ist. Die Meldung `E_NOTICE` kann aber hilfreich sein, wenn Sie zum Beispiel einen Variablennamen versehentlich falsch geschrieben haben:

```
foreach ($array as $value) {
    $hmtl .= $value; // Hoppla! Das sollte $html heißen.
}
```

```
$html = ''
foreach ($array as $value) {
    $hmtl .= $value; // Hoppla! Das sollte $html heißen.
}
```

Eine benutzerdefinierte Fehlerbehandlungsfunktion kann Fehlermeldungen anhand ihres Typs untersuchen und entsprechende Maßnahmen ergreifen. Tabelle 9-1 zeigt eine vollständige Liste der Fehlerarten.

Tabelle 9-1: Fehlerarten (Stand PHP 5.3)

Wert	Konstante	Beschreibung	Abfangbar
1	E_ERROR	Nicht reparabler Fehler.	Nein
2	E_WARNING	Reparabler Fehler.	Ja
4	E_PARSE	Parser-Fehler.	Nein
8	E_NOTICE	Möglicherweise ein Fehler.	Ja
16	E_CORE_ERROR	Wie E_ERROR, aber durch den PHP-Kern ausgelöst.	Nein
32	E_CORE_WARNING	Wie E_WARNING, aber durch den PHP-Kern ausgelöst.	Nein
64	E_COMPILE_ERROR	Wie E_ERROR, aber durch die Zend-Engine ausgelöst.	Nein
128	E_COMPILE_WARNING	Wie E_WARNING, aber durch die Zend-Engine ausgelöst.	Nein
256	E_USER_ERROR	Wie E_ERROR, aber durch den Aufruf von trigger_error() ausgelöst.	Ja
512	E_USER_WARNING	Wie E_WARNING, aber durch den Aufruf von trigger_error() ausgelöst.	Ja
1024	E_USER_NOTICE	Wie E_NOTICE, aber durch den Aufruf von trigger_error() ausgelöst.	Ja
2048	E_STRICT	Benachrichtigungen des Laufzeitsystems. Damit erhalten Sie von PHP Vorschläge für Änderungen des Programmcodes, die eine bestmögliche Interoperabilität und zukünftige Kompatibilität Ihres Codes gewährleisten (seit PHP 5.0).	–
4096	E_RECOVERABLE_ERROR	Abfangbarer fataler Fehler. Dies bedeutet, dass ein potenziell gefährlicher Fehler aufgetreten ist, die PHP-Engine aber nicht in einem instabilen Zustand hinterlassen hat. Wird der Fehler nicht durch eine benutzerdefinierte Fehlerbehandlungsroutine abgefangen, wird die Anwendung wie bei einem E_ERROR-Fehler abgebrochen (seit PHP 5.2).	Ja
8192	E_DEPRECATED	Notices zur Laufzeit des Programms. Aktivieren Sie diese Einstellung, um Warnungen über Codebestandteile zu erhalten, die in zukünftigen PHP-Versionen nicht mehr funktionieren werden (seit PHP 5.3).	–
16384	E_USER_DEPRECATED	Wie E_DEPRECATED, aber durch den Aufruf von trigger_error() ausgelöst (seit PHP 5.3).	–
30719	E_ALL	Alle Fehler und Warnungen, die unterstützt werden, mit Ausnahme von E_STRICT in PHP-Versionen < 6. Der Wert dieser Konstanten ändert sich immer wieder: 32767 in PHP 6, 30719 in PHP 5.3.x, 6143 in PHP 5.2.x, 2047 in früheren Versionen.	–

Als abfangbar markierte Fehler können durch die Funktion behandelt werden, die mit `set_error_handler()` registriert worden ist. Die anderen weisen auf ein derart ernsthaftes Problem hin, dass sie nicht von Anwendern behandelt werden können, sondern PHP sich selbst darum kümmern muss.

Siehe auch

Rezept 9.3 zeigt, wie man eine benutzerdefinierte Fehlerbehandlungsroutine aufsetzt; die Dokumentationen zu `error_reporting()` unter *http://www.php.net/error-reporting* und zu `set_error_handler()` unter *http://www.php.net/set-error-handler*; weitere Informationen zu Fehlern finden Sie unter *http://www.php.net/manual/ref.errorfunc.php*.

9.3 Eine benutzerdefinierte Funktion zur Fehlerbehandlung verwenden

Problem

Sie möchten einen benutzerdefinierten Error-Handler schreiben, mit dessen Hilfe Sie steuern können, wie PHP Fehler anzeigt.

Lösung

Um Ihre eigene Fehlerfunktion einzusetzen, verwenden Sie `set_error_handler()`:

```
set_error_handler('pc_error_handler');

function pc_error_handler($errno, $error, $file, $line) {
    $message = "[ERROR][$errno][$error][$file:$line]";
    error_log($message);
}
```

Diskussion

Eine benutzerdefinierte Funktion zur Fehlerbehandlung kann Fehler aufgrund ihres Typs analysieren und angemessene Maßnahmen ergreifen. Tabelle 9-1 in Rezept 9.2 zeigt eine Liste der Fehlerarten.

Übergeben Sie `set_error_handler()` den Namen einer Funktion, und PHP reicht alle Fehler an diese Funktion weiter. Die Fehlerbehandlungsfunktion kann bis zu fünf Parameter übernehmen. Der erste Parameter ist die Fehlerart, zum Beispiel `8` für `E_NOTICE`. Der zweite enthält die durch den Fehler ausgelöste Nachricht, zum Beispiel »Undefined variable: html«. Das dritte und vierte Argument sind der Dateiname und die Zeilennummer, in der PHP den Fehler gefunden hat. Der letzte Parameter ist ein Array mit allen Variablen, die im aktuellen Geltungsbereich definiert sind, und deren Werten.

Im folgenden Code wird beispielsweise etwas an `$html` angehängt, ohne dass der Variablen zuvor ein Anfangswert zugewiesen wurde:

```
error_reporting(E_ALL);
set_error_handler('pc_error_handler');

function pc_error_handler($errno, $error, $file, $line, $context) {
    $message = "[ERROR][$errno][$error][$file:$line]";
    print "$message";
    print_r($context);
}

$form = array('eins','zwei');

foreach ($form as $line) {
    $html .= "<b>$line</b>";
}
```

Nachdem der Fehler »Undefined variable« ausgelöst wurde, gibt der `pc_error_handler()` Folgendes aus:

```
[ERROR][8][Undefined variable:  html][err-all.php:16]
```

Nach der anfänglichen Fehlermeldung zeigt `pc_error_handler()` außerdem noch ein großes Array an, das alle globalen, Umgebungs-, Anfrage- und Session-Variablen enthält.

Sämtliche Fehler, die in Tabelle 9-1 als abfangbar gekennzeichnet sind, können durch die mithilfe von `set_error_handler()` angemeldete Funktion behandelt werden. Die anderen deuten auf ein derart schweres Problem hin, dass sie vom Benutzer nicht auf sichere Art behandelt werden können, sodass PHP sich um sie kümmern muss.

Siehe auch

Rezept 9.2 zeigt eine Liste der verschiedenen Fehlerarten; die Dokumentationen zu `set_error_handler()` unter *http://www.php.net/set-error-handler* und zu Exceptions unter *http://www.php.net/manual/en/language.exceptions.php*.

9.4 Fehler protokollieren

Problem

Sie möchten Programmfehler in ein Protokoll schreiben. Dazu kann alles, von Parser-Fehlern bis zu nicht gefundenen Dateien, fehlerhaften Datenbankabfragen und unterbrochenen Verbindungen, gehören

Lösung

Verwenden Sie `error_log()`, um in das Fehlerprotokoll zu schreiben:

```
// LDAP-Fehler
if (ldap_errno($ldap)) {
    error_log("LDAP-Fehler #" . ldap_errno($ldap) . ": " . ldap_error($ldap));
}
```

Diskussion

Die Protokollierung von Fehlern erleichtert das Debugging. Indem Sie Fehler geschickt protokollieren, erleichtern Sie deren Behebung. Protokollieren Sie immer Informationen über die Ursachen eines Fehlers:

```
$r = mysql_query($sql);
if (! $r) {
    $error = mysql_error();
    error_log('[DB: Abfrage @'.$_SERVER['REQUEST_URI']."][$sql]: $error");
} else {
    // Ergebnisse verarbeiten.
}
```

Wenn Sie nur einfach protokollieren, dass ein Fehler aufgetreten ist, und keine unterstützenden Informationen hinzufügen, bekommen Sie nicht die Hilfe beim Debuggen, die Sie erhalten könnten:

```
$r = mysql_query($sql);
if (! $r) {
    error_log("Fehlerhafte Abfrage");
} else {
    // Ergebnisse verarbeiten.
}
```

Eine andere nützliche Technik besteht darin, die Konstanten `__FILE__` und `__LINE__` in die Fehlermeldungen einzufügen:

```
error_log('['.__FILE__.']['.__LINE__."]: $error");
```

Die Konstante `__FILE__` bezeichnet den aktuellen Dateinamen, und `__LINE__` enthält die aktuelle Zeilennummer. Es gibt noch weitere vordefinierte Konstanten. Seit PHP 5.3 enthält `__NAMESPACE__` zum Beispiel den Namen des aktuellen Namensraums. Eine vollständige Übersicht finden Sie in der PHP-Dokumentation unter *http://www.php.net/manual/language.constants.predefined.php*.

Siehe auch

Rezept 9.1 zum Verbergen von Fehlermeldungen vor den Benutzern; die Dokumentation zu `error_log()` unter *http://www.php.net/error-log*.

9.5 Debug-Informationen protokollieren

Problem

Sie möchten das Debuggen erleichtern, indem Sie Anweisungen zur Ausgabe von Variablen einfügen. Aber Sie möchten auch auf einfache Weise zwischen dem Produktions- und dem Debug-Modus umschalten können.

Lösung

Schreiben Sie eine Funktion, die Mitteilungen in Abhängigkeit von der Definition einer Konstanten ausgibt, die mithilfe der Konfigurationseinstellung `auto_prepend_file` in der Seite eingefügt wird. Speichern Sie den folgenden Code in *debug.php*:

```
// Debugging einschalten.
define('DEBUG',true);

// Generische Debug-Funktion.
function pc_debug($message) {
    if (defined('DEBUG') && DEBUG) {
        error_log($message);
    }
}
```

Setzen Sie die Direktive `auto_prepend_file` in *php.ini*:

```
auto_prepend_file=debug.php
```

Rufen Sie nun `pc_debug()` aus Ihrem Programm auf, um die Debug-Informationen auszugeben:

```
$sql = 'SELECT farbe, form, geruch FROM gemuese';
pc_debug("[sql: $sql]"); // wird nur ausgegeben, wenn DEBUG wahr ist
$r = mysql_query($sql);
```

Diskussion

Die Fehlersuche ist eine notwendige Nebenerscheinung des Programmierens. Es gibt eine Vielzahl von Techniken, die Ihnen dabei helfen können, Ihre Wanzen schnell zu lokalisieren und zu zerquetschen. Bei vielen dieser Techniken müssen Sie zusätzliches »Gerüstmaterial« in das Programm einfügen, mit dessen Hilfe Sie die Richtigkeit Ihres Codes sicherstellen. Je komplizierter ein Programm ist, desto mehr Gerüste werden benötigt. Fred Brooks schätzt in seinem Buch *Vom Mythos des Mann-Monats* (Mitp-Verlag), dass »halb so viel Code im Gerüst steckt wie im Produkt«. Durch gute Vorausplanung können Sie die Gerüste in einer sauberen und effizienten Art und Weise in Ihre Programmlogik integrieren. Dazu müssen Sie aber im Voraus darüber nachdenken, was Sie messen und aufzeichnen möchten, und die Auswertung der mithilfe Ihrer Gerüste gesammelten Daten planen.

Als eine Technik zum Durchsieben der Informationen können Sie verschiedenen Arten von Debugging-Kommentaren verschiedene Prioritäten zuordnen. Dann gibt die Debug-Funktion nur Informationen aus, die höher bewertet sind als das aktuell eingestellte Debugging-Niveau.

```
define('DEBUG',2);

function pc_debug($message, $level = 0) {
    if (defined('DEBUG') && ($level > DEBUG) {
        error_log($message);
    }
}

$sql = 'SELECT farbe, form, geruch FROM gemuese';
pc_debug("[sql: $sql]", 1); // wird nicht ausgegeben, da 1 < 2
pc_debug("[sql: $sql]", 3); // wird ausgegeben, da 3 > 2
```

Eine andere Technik besteht darin, Wrapper-Funktionen zu schreiben und dadurch zusätzliche Informationen für das Performance-Tuning einzufügen, zum Beispiel die zur Ausführung einer Datenbankabfrage benötigte Zeit.

```
function db_query($sql) {
    if (defined(DEBUG) && DEBUG) {
         // Beginne Zeitabnahme, wenn DEBUG eingeschaltet ist.
         $DEBUG_STRING = "[sql: $sql]<br>\n";
         $starttime = microtime(true);
    }

    $r = mysql_query($sql);

    if (! $r) {
        $error = mysql_error();
        error_log('[DB: query @'.$_SERVER['REQUEST_URI']."][$sql]: $error");
    } elseif (defined('DEBUG') && DEBUG) {
        // Abfrage ist nicht gescheitert und DEBUG eingeschaltet,
        // also Zeitabnahme beenden.
        $endtime = microtime(true);
        $elapsedtime = $endtime - $starttime;
        $DEBUG_STRING .= "[time: $elapsedtime]<br>\n";
        error_log($DEBUG_STRING);
    }

    return $r;
}
```

Hier schreiben Sie nicht nur einfach die SQL-Anweisung in das Fehlerprotokoll, sondern zeichnen zusätzlich die Anzahl der Sekunden auf, die MySQL zur Ausführung der Abfrage benötigt. Daran können Sie erkennen, ob bestimmte Abfragen zu viel Zeit in Anspruch nehmen.

Informationen zur Funktion `microtime()` finden Sie in Rezept 3.15.

Siehe auch

Die Dokumentationen zu `define()` unter *http://www.php.net/define*, zu `defined()` unter *http://www.php.net/defined* und zu `error_log()` unter *http://www.php.net/error-log*; *Vom Mythos des Mann-Monats* von Frederick P. Brooks (Mitp-Verlag).

9.6 PHP-Fehler- und Warnmeldungen in Ausnahmen umwandeln

Problem

Sie wollen, dass bei allen Problemen, die in PHP Fehler- und/oder Warnmeldungen verursachen, stattdessen eine Exception geworfen wird. Zum Beispiel wollen Sie Code absichern, in dem schon andere Exceptions geworfen werden.

Lösung

Deklarieren Sie eine Error-Handler-Funktion, in der Sie eine `ErrorException` werfen, und registrieren Sie diese Funktion mit `set_error_handler()`:

```
function exception_error_handler($errno, $errstr, $errfile, $errline ) {
    throw new ErrorException($errstr, 0, $errno, $errfile, $errline);
}
set_error_handler("exception_error_handler", E_ALL - E_NOTICE);


try {
    $file = file_get_contents("/pfad/nach/nirgendwo");
}
catch (Exception $e) {
    echo "File not found!";
}
```

Diskussion

Wenn Sie von Grund auf neuen Code schreiben, werden Sie wahrscheinlich PHP-Funktionen nutzen, die konventionelle Warnmeldungen produzieren. Wollen Sie außerdem noch Erweiterungen verwenden, die auch Exceptions werfen, müssen Sie bei jedem Methodenaufruf überdenken, ob Sie es hier mit Warnmeldungen oder Exceptions zu tun bekommen. Zum Beispiel produziert

```
$dom = new DOMDocument;
$dom->load("someXML.xml");
```

nur eine Warnmeldung, wenn *someXML.xml* nicht zu öffnen ist oder kein wohlgeformtes XML enthält. Auf der anderen Seite bekommen Sie bei

```
$x = $dom->appendChild(new DOMElement($y));
```

eine Exception serviert, wenn `$y` keinen gültigen Elementnamen enthält. Wenn Sie sich also nicht ständig mit der Frage beschäftigen wollen, ob Ihre nächste Codezeile nun mit Warnungen oder Exceptions arbeitet, können Sie den obigen Code verwenden. Er sorgt dafür, dass jede Fehler- und Warnmeldung an die Funktion `exception_error_handler()` übergeben wird, die diese dann in eine `ErrorException` umwandelt.

Siehe auch

Die Dokumentation zu `set_error_handler()` unter *http://www.php.net/set-error-handler*; die Dokumentation zur Klasse `ErrorException` finden Sie unter *http://www.php.net/manual/en/class.errorexception.php*.

9.7 Ausnahmen abfangen

Problem

Sie haben ein Stück Code, in dem Exceptions auftreten können. Sie wollen diese Ausnahmen abfangen. Zum Beispiel wollen Sie einen `DirectoryIterator` erzeugen, können sich aber nicht darauf verlassen, dass der angegebene Pfad auch tatsächlich existiert. Tut er das nicht, wollen Sie stattdessen einen Default-Pfad angeben.

Lösung

Umgeben Sie den Code, der Ihnen möglicherweise eine Exception liefert, mit einem `try`-Block und fangen Sie die Exception in einem `catch`-Block ab:

```
try {
    $directory = new DirectoryIterator($path);
}
catch (Exception $e) {
    $directory = new DirectoryIterator($defaultPath);
}
foreach ($directory as $file) {
    ... // Verzeichnis durchlaufen.
```

Diskussion

Exceptions werden häufig nur zur Ausgabe von Fehlermeldungen verwendet. Oft ist eine Fehlermeldung aber eher lästig, weil das zugrunde liegende Problem nicht das Ende der Welt darstellt. Der `catch`-Block stellt Ihnen daher frei, was genau Sie im Fall einer Exception machen wollen. Das kann wie hier ersatzweise die Verwendung eines Default-Werts sein oder auch das Überspringen eines Schleifendurchlaufs:

```
foreach ($pathList as $path) {
    try {
        $directory = new DirectoryIterator($path);
```

```
        }
        catch (Exception $e) {
            continue; // Pfad überspringen.
        }
        foreach ($directory as $file) {
            ... // Verzeichnis durchlaufen.
    }
```

Im Gegensatz zu anderen Programmiersprachen brauchen Sie den Ausnahmezustand im `catch`-Block nicht explizit für beendet zu erklären, die Exception wird automatisch konsumiert.

Sie können die problematischen Statements auch in eine Funktion außerhalb des `try`-Blocks auslagern:

```
function getDir($path) {
    return new DirectoryIterator($path);
}

try {
    $directory = getDir($path);
}
catch (Exception $e) {
    $directory = getDir($defaultPath);
}
...
```

In diesem Fall wird die Exception nicht in der Funktion abgefangen. Nicht abgefangene Exceptions steigen auf, bis sie abgefangen werden – in diesem Fall auf der aufrufenden Ebene. Wird eine Ausnahme nirgendwo abgefangen, zeigt PHP sie als einen Fatal Error an:

```
Fatal error: Uncaught exception 'Exception' with message ...
```

Falls Sie die Exception zur Ausgabe einer Fehlermeldung oder zum Loggen des Fehlers verwenden wollen, können Sie sich die Tatsache zunutze machen, dass `$e` ein Objekt ist. Wollen Sie `$e` lediglich ausdrucken, bekommen Sie sämtliche Informationen über die `__toString()`-Methode des Objekts, die bei `echo $e;` automatisch aufgerufen wird.

Die einzelnen Detailinformationen zu `$e` lassen sich über die Methoden `getMessage()`, `getCode()`, `getLine()`, `getFile()`, `getTrace()` und `getTraceAsString()` abfragen. `getTrace()` liefert Ihnen ein Array mit den Funktionsaufrufen, die zur Exception geführt haben, während `getTraceAsString()` eine String-Version derselben Information zurückgibt.

Siehe auch

Die allgemeine Dokumentation zu Ausnahmen in PHP und zum Exception-Objekt unter *http://www.php.net/manual/en/language.exceptions.php*.

9.8 Eigene Ausnahmen werfen

Problem

Sie möchten in Ihren Funktionen oder Methoden Fehlerzustände durch eigene Ausnahmen aufzeigen.

Lösung

Werfen Sie Ihre eigenen Exceptions:

```
function setPrice($price) {
    if (!is_numeric($price)) throw new Exception("Price is not a number", 1);
    if ($price < 0) throw new Exception("Price is negative", 2);
    // Preis in Datenbank einfügen.
    ...
}

setPrice($price);
```

Diskussion

Wenn Sie in diesem Beispiel in Ihrem `catch`-Block wissen wollen, was nun genau passiert ist, müssen Sie sich den Code der Exception ansehen:

```
try {
    setPrice($price);
}
catch (Exception $e)
{
    switch ($e->getCode()) {
        case 1: echo "Bitte eine positive Zahl eingeben."; break;
        case 2: echo "Wie bitte? Negativer Preis???"; break;
    }
}
```

Wenn Sie jetzt bei Exceptions mit Fehlercode 2 einen negativen Preis in einen positiven umwandeln möchten, müssen Sie das so machen:

```
try {
    setPrice($price);
}
catch (Exception $e)
{
    switch ($e->getCode()) {
        case 1: echo "Bitte eine positive Zahl eingeben.";
        // Weitere Fehlerbehandlung, z.B. Neuladen eines Formulars.
        ...
        case 2: setPrice(-$price); break;
    }
}
```

Das funktioniert, ist aber etwas unschön. Warum? Ganz einfach: weil Sie in Ihrem `catch`-Block nur auf Daten zugreifen können, die auch außerhalb der Funktion `setPrice()` verfügbar sind. Benötigen Sie aber zur Ausnahmebehandlung Daten, die es nur innerhalb der Funktion gibt, haben Sie ein Problem. Das können Sie durch eine Erweiterung der `Exception`-Klasse lösen:

```
class EPriceException extends Exception {
    public $price; // Preis für den catch-Block erhalten.

    public function __construct($message, $code, $price) {
        parent::__construct($message, $code);
        $this->price = $price;
    }
}

function setPrice($price) {
    if (!is_numeric($price)) throw new EPriceException("Price is not a number", 1,
                                                       $price);
    if ($price < 0) throw new EPriceException("Price is negative", 2, $price);
    // Preis in Datenbank einfügen.
    ...
}

try {
    setPrice(-20);
}
catch (EPriceException $e)
{
    switch ($e->getCode()) {
        case 1: echo "Bitte eine positive Zahl eingeben.";
        // Weitere Fehlerbehandlung, z.B. Neuladen eines Formulars.
        ...
        case 2: setPrice(-$e->price); break;
    }
}
```

Beachten Sie, dass wir jetzt eine `EPriceException` abfangen. Das verhindert potenzielle Probleme beim Auftauchen anderer Exceptions, die zwar möglicherweise einen Code 2 haben, aber keine Eigenschaft `$price`.

Siehe auch

Rezept 9.9 zum Umgang mit Exceptions in Abhängigkeit von ihrer Klasse; die allgemeine Dokumentation zu Ausnahmen in PHP und zum Exception-Objekt unter *http://www.php.net/manual/en/language.exceptions.php*.

9.9 Klassenabhängiges Exception-Handling

Problem

Sie möchten einen Codeblock schützen, in dem verschiedene Klassen von Exceptions auftreten können. Dabei möchten Sie das Exception-Handling von der jeweiligen Klasse abhängig machen.

Lösung

Verwenden Sie mehrere catch-Blöcke:

```
class EPriceException extends Exception {
    public $price; // Preis für den catch-Block erhalten.

    public function __construct($message, $code, $price) {
        parent::__construct($message, $code);
        $this->price = $price;
    }
}

class EPriceNotANumberException extends EPriceException {
    public function __construct($price) {
        parent::__construct("Price is not a number", 1, $price);
    }
}

class EPriceNegativeException extends EPriceException {
    public function __construct($price) {
        parent::__construct("Price is negative", 1, $price);
    }
}

function setPrice($price) {
    if (!is_numeric($price)) throw new EPriceNotANumberException($price);
    if ($price < 0) throw new EPriceNegativeException($price);
}

try {
    setPrice(-20);
}
catch (EPriceNotANumberException $e)
{
    echo "Bitte eine positive Zahl eingeben.";
    // Weitere Fehlerbehandlung, z.B. Neuladen eines Formulars.
    ...
}
catch (EPriceNegativeException $e)
{
    setPrice(-$e->price);
}
```

Diskussion

Bei der Verwendung eigener Exception-Klassen für spezifische Fehlertypen können Sie Fehlertext und -code im Konstruktor der Klasse setzen. Beim Werfen der Exception an verschiedenen Stellen Ihres Codes brauchen Sie diese Informationen dann nicht immer wieder neu anzugeben.

Siehe auch

Rezept 9.8 zum Werfen eigener Exceptions; die allgemeine Dokumentation zu Ausnahmen in PHP und zum Exception-Objekt unter *http://www.php.net/manual/en/language.exceptions.php*.

9.10 Vordefinierte Exception-Klassen für alle Lebenslagen

Problem

Bei der Fehlerbehandlung sind bestimmte Fehlerkategorien in jedem Ihrer Projekte/Programme zu behandeln. Dazu erstellen Sie wahrscheinlich in jedem dieser Projekte immer wieder eigene Exception-Klassen im Userland-PHP-Code[1] oder verwenden solche aus vorherigen Projekten wieder. Im Vergleich zu den in PHP eingebauten Exception-Klassen bieten Ihre »Selbstbauten« jedoch eine schlechtere Performance.

Lösung

Die Standard PHP Library (SPL), die ab PHP 5.3 immer aktiviert ist, bietet eine Vielzahl von Exception-Klassen, die von der Erweiterung selbst verwendet werden. Diese können Sie jedoch auch in Ihren eigenen Programmen nutzen. Die bereitgestellten Klassen decken den grundlegenden Umfang von Fehlerarten ab. Es stehen sowohl Klassen zur Signalisierung von logischen Fehlern (`LogicException`) als auch Laufzeitfehlern (`RuntimeException`) zur Verfügung. Zu beiden Fehlerkategorien existieren viele weitere spezialisierte Exception-Klassen (siehe Tabelle 3-1).

1 Userland-Code ist vom Programmierer in normalem PHP geschriebener Code. Im Gegensatz dazu wird der Kern der Sprache PHP selbst sowie die PHP-Erweiterung (z.B. ext/mysqli) in C geschrieben. Der in C geschriebene Teil von PHP ist in der Regel performanter als Userland-Code.

Tabelle 9-2: Übersicht der vordefinierten Exception-Klassen der Standard PHP Library

Exception	Basisklasse	Beschreibung
BadFunctionCallException	LogicException	Ein Callback bezieht sich auf eine nicht definierte Funktion, oder es gibt ein Problem mit den Funktionsparametern.
BadMethodCallException	BadFunctionCallException	Wie BadFunctionCallException, allerdings im objektorientierten Kontext (Methode einer Klasse).
DomainException	LogicException	Signalisiert Verwendung eines Werts aus einem falschen Wertebereich im mathematischen Sinne.
InvalidArgumentException	LogicException	Übergabe von ungültigen Argumenten.
LengthException	LogicException	Ein Parameter überschreitet die zulässige Länge.
LogicException	Exception	Fehler in der Anwendungslogik.
OutOfBoundsException	RuntimeException	Verwendung eines ungültigen Index (zur Laufzeit des Skripts).
OutOfRangeException	LogicException	Verwendung eines ungültigen Index (zur Kompilierzeit des Skripts).
OverflowException	RuntimeException	Arithmetischer oder Buffer Overflow.
RangeException	RuntimeException	Weist auf einen Bereichsfehler während der Programmausführung hin. Wie DomainException, aber bei Laufzeitfehlern.
RuntimeException	Exception	Signalisiert einen Fehler, der nur während der Laufzeit eines Skripts feststellbar ist (z.B. Zugriff auf falschen Index eines Arrays).
UnderflowException	RuntimeException	Arithmetischer oder Buffer Overflow.
UnexpectedValueException	RuntimeException	Ein unerwarteter Wert wurde festgestellt. Nicht für arithmetik- oder bufferbezogene Fehler.

Der folgende Code zeigt ein Beispiel für die Verwendung einer Logik- und einer Laufzeit-Exception:

```
class SplExceptions {
    public function logicException(array $parameter) {
        if (3 != count($parameter)) {
            throw new InvalidArgumentException(
                "Array muss drei Werte enthalten");
        }

        // Parameter ist ok ... weiterarbeiten!
    }

    public function runtimeException($ar, $idx) {
        if (! array_key_exists($idx, $ar)) {
            throw new OutOfBoundsException("Array-Index fehlerhaft");
```

```
        }
        // ...
    }
}
```

Diskussion

Da die Exception-Klassen der SPL wie PHP selbst in der Programmiersprache C geschrieben werden, sind diese performanter als vom Programmierer im PHP-Userland erstellte Exception-Klassen. Sie stehen immer zur Verfügung, wenn die SPL aktiviert ist. Es ist also nicht notwendig, diese Klassen extra per `include` bzw. `require` zu laden.

Sollten Ihnen die bereitgestellten Exception-Klassen für irgendeinen Problemfall nicht ausreichen, besteht eine mögliche Lösung in der Verwendung der generischen Klassen `LogicException` bzw. `RuntimeException` in Verbindung mit Fehlermeldungen und speziellen Fehlercodes:

```
public function spezifischeException() {
    throw new RuntimeException(
        "Spezieller Laufzeitfehler", $fehlercode = 100);
}
```

Wenn Ihnen die Lösung mit den Fehlercodes jedoch nicht ausreicht, bleibt doch nur der Ausweg über die Erstellung eigener Exception-Klassen – Rezept 9.8 beschreibt diese Lösung detaillierter.

Siehe auch

Wie auch PHP entwickelt sich die SPL ständig weiter, und es kommen eventuell neue Exception-Klassen hinzu. Aktuelle Informationen finden Sie im PHP-Manual unter *http://php.net/manual/spl.exceptions.php*. Eine Übersicht über die Vererbungshierarchie der Exception-Klassen und vieles mehr wird unter der folgenden Adresse geboten: *http://www.php.net/~helly/php/ext/spl/*.

9.11 Ungefangene Exceptions zentral behandeln

Problem

Sie wollen an einer zentralen Stelle dafür sorgen, dass alle ungefangenen Ausnahmen (Exceptions) abgefangen werden. Dadurch könnten Sie beispielsweise eine geeignete Hinweismeldung ausgeben und Details zu diesem Fehler an Ihre private E-Mail-Adresse senden.

Lösung

Deklarieren Sie eine Exception-Handler-Funktion, in der Sie den Fehler behandeln, und registrieren Sie diese Funktion mit `set_exception_handler()`:

```
function exception_handler(Exception $exception) {
    echo "<h1>Uuups !</h1><p>Es ist ein Fehler aufgetreten.</p>";
    // Eine E-Mail an den Administrator senden.
    // ...
}
set_exception_handler('exception_handler');

throw new Exception ('unerwarteter Fehler');
```

Diskussion

Mit der Funktion `set_exception_handler()` registrieren Sie einen Handler, der bei jeder unbehandelten Exception aufgerufen wird. Normalerweise werden Ausnahmen mit einem try-catch-Block abgefangen und lokal behandelt. Eine lokale Behandlung versucht meist, einen Fehler unauffällig zu beheben oder nötige Aufräumarbeiten durchzuführen. Sollen allerdings Fehlermeldungen ausgegeben oder Log-Einträge gespeichert und E-Mails verschickt werden, bietet es sich an, dies an zentraler Stelle durch einen globalen Exception-Handler durchzuführen.

Dazu kann man sich hier noch vorstellen, ein Flag in der Anwendung zu definieren, das angibt, ob in einem Fehlerfall eine vertröstende Meldung oder eine detaillierte Fehlerbeschreibung ausgegeben werden soll. Für dieses Flag wird gern eine mit `define()` definierte Konstante `DEBUG` eingesetzt:

```
// Stell hier auf false, um eine vertröstende Meldung auszugeben.
define ('DEBUG', true);

function exception_handler(Exception $exception) {
       if (defined("DEBUG") && DEBUG)
    {
        // Details zum Fehler in $exception auf den Bildschirm ausgeben.
        echo "<h1>Exception</h1><b>".$exception->getMessage()."</b>";
        echo "<br/>Datei: ".$exception->getFile()."<br/>";
        echo "Zeile: ".$exception->getLine();

    } else {
        // Vertröstende Meldung.
        echo "Bitte versuchen Sie es sp&auml;ter noch einmal.<br/>";
        echo "Vielen Dank f&uuml;r Ihr Verst&auml;ndnis.";

        // Mail an Administrator mit Details.
        // ...
    }

}
```

```
set_exception_handler('exception_handler');

throw new Exception ('unerwarteter Fehler');
```

Siehe auch

Details zur Funktion set_exception_handler() unter *http://www.php.net/manual/en/function.set-exception-handler.php*; das Fangen von Ausnahmen wird in Rezept 9.7 beschrieben, das Definieren eines Fehler-Handlers und die Umwandlung eines Fehlers zu einer Ausnahme in Rezept 9.6.

9.12 Einen Stacktrace ausgeben

Problem

Sie möchten wissen, was an einem bestimmten Punkt Ihres Programms und auf den Stufen zu diesem Punkt passiert.

Lösung

Nutzen Sie debug_print_backtrace( ):

```
function stooges() {
  print "woo woo woo!\n";
  larry();
}
function larry() {
  curly();
}
function curly() {
  moe();
}
function moe() {
  debug_print_backtrace();
}
stooges();
```

Das erzeugt folgende Ausgabe:

```
woo woo woo!
#0  moe() called at [backtrace.php:14]
#1  curly() called at [backtrace.php:10]
#2  larry() called at [backtrace.php:6]
#3  stooges() called at [backtrace.php:21]
```

Diskussion

Die Funktion `debug_backtrace()` wurde in PHP 4.3.0 eingeführt, und in PHP 5.0.0 wurde die praktische Funktion `debug_print_backtrace()` ergänzt. Mithilfe dieser Funktionen können Sie sich schnell einen Überblick darüber verschaffen, was in Ihrer Anwendung geschah, unmittelbar bevor Sie eine bestimmte Funktion aufriefen.

Je komplizierter eine Anwendung ist, umso mehr Informationen können Sie als Rückgabe der Backtrace-Funktionen erwarten. Zum Debugging größerer Codegruppen sind Sie beim Debugging eventuell schneller erfolgreich, wenn Sie eine vollständige Debugging-Erweiterung wie Xdebug oder eine integrierte Entwicklungsumgebung (IDE) wie PHPEdit oder Zend Studio einsetzen, die es Ihnen ermöglicht, Breakpoints zu setzen, in Codeblöcke zu springen und aus ihnen heraus die Entwicklung von Variablen zu beobachten und vieles mehr.

Wenn Sie nur wenige weitere Informationen benötigen, als Ihnen hier und da in Ihren Code eingestreute `print 'Bin in Zeile ' . __LINE__;`-Anweisungen liefern können, sind `debug_backtrace()` und/oder `debug_print_backtrace()` Ihren Anforderungen vollkommen angemessen.

Nutzen Sie immer noch PHP 4 und wollen die nur in PHP 5 verfügbare Funktion `debug_print_backtrace()` nutzen, können Sie das PEAR-Kompatibilitätspaket PHP_Compat nutzen. `PHP_Compat` bietet eine Implementierung von `debug_print_backtrace()`, die mit der nativen PHP 5-Funktion identisch ist.

Siehe auch

Dokumentationen zu `debug_backtrace()` unter *http://www.php.net/debug-backtrace* und zu `debug_print_backtrace()` unter *http://www.php.net/debug-print-backtrace*; zum PEAR PHP_Compat-Paket unter *http://pear.php.net/package/PHP_Compat*, zur Zend Studio IDE unter *http://www.zend.com/products/zend_studio* und zur PHPEdit IDE unter *http://www.waterproof.fr/products/PHPEdit/*.

KAPITEL 10

Web-Grundlagen

10.0 Einführung

Wenn Sie dieses Buch lesen, geht es Ihnen wahrscheinlich um Webprogrammierung. Dafür wurde die erste PHP-Version geschrieben, und sie macht PHP heute so populär. Mit PHP kann man sehr einfach dynamische Webprogramme schreiben, die fast alles machen. Andere Kapitel behandeln diverse PHP-Fähigkeiten wie Grafik, reguläre Ausdrücke, Datenbankzugriffe und Datei-Ein-/Ausgabe. Dies sind alles Dinge, die Sie zur Webprogrammierung benötigen, aber dieses Kapitel konzentriert sich auf einige webspezifische Konzepte und organisatorische Themen, mit der Sie Ihre Webprogrammierung verstärken können.

Die Rezepte 10.1, 10.2 und 10.3 zeigen, wie man Cookies setzt, liest und löscht. Ein Cookie ist ein kurzes Stück Text, das der Browser auf Veranlassung des Servers mit seinen Anfragen zusammen versendet. Normalerweise sind HTTP-Anfragen nicht statusbehaftet; eine Anfrage kann nicht mit einer vorhergehenden in Verbindung gebracht werden. Ein Cookie kann aber verschiedene Anfragen desselben Benutzers miteinander verknüpfen. Auf diese Weise kann man leichter so etwas wie einen Einkaufswagen konstruieren oder die Such-Historie eines Benutzers verfolgen.

Das Rezept 10.4 zeigt, wie man Benutzer an eine andere als die angeforderte Webseite weiterverweist. Rezept 10.5 erklärt das Session-Modul, mit dem Sie auf einfache Weise persistente Daten mit einem Benutzer verknüpfen können, der sich durch Ihre Site bewegt. Rezept 10.6 führt vor, wie Sie Sitzungsinformationen in einer Datenbank speichern und damit die Skalierbarkeit und Flexibilität Ihrer Website erhöhen können. Wie man die Möglichkeiten eines Benutzer-Browsers erkundet, wird in Rezept 10.7 dargestellt. Rezept 10.8 zeigt detailliert die Zusammensetzung einer URL mit einem GET-Query-String einschließlich der korrekten Codierung von Sonderzeichen und der Behandlung von HTML-Entities.

Die darauf folgenden beiden Rezepte demonstrieren die Verwendung der Autorisierung, durch die Sie Ihre Webseiten mit Passworten schützen können. Die speziellen Möglichkeiten von PHP für den Umgang mit der HTTP-Basic-Authentifizierung werden in

Rezept 10.9 erklärt. Manchmal ist es jedoch besser, ein eigenes Autorisierungsverfahren mithilfe von Cookies zu betreiben, wie es in Rezept 10.10 zu sehen ist.

Die dann folgenden drei Rezepte behandeln die Ausgabekontrolle. Rezept 10.11 zeigt, wie man eine Ausgabe an den Browser versendet. Rezept 10.12 erklärt die Funktionen zur Zwischenspeicherung (Pufferung) der Ausgabe. Zwischenspeicher ermöglichen Ihnen, die Ausgaben abzufangen, die andernfalls im Browser dargestellt werden würden, oder die Ausgabe zu verzögern, bis eine Seite vollständig verarbeitet worden ist. Die automatische Komprimierung von Ausgaben wird in Rezept 10.13 dargestellt.

Rezept 10.14 enthält Strategien zur Vermeidung der verbreiteten Fehlermeldung »headers already sent«, darunter die in Rezept 10.12 behandelte Ausgabe-Zwischenspeicherung.

Die nächsten vier Rezepte erklären, wie man mit externen Variablen interagiert: Umgebungsvariablen und PHP-Konfigurationseinstellungen. Die Rezepte 10.15 und 10.16 behandeln Umgebungsvariablen, während die Rezepte 10.17 und 10.18 auf das Lesen und Verändern der PHP-Konfiguration eingehen. Wenn Sie Apache als Webserver verwenden, können Sie mithilfe der Techniken in Rezept 10.19 aus Ihren PHP-Programmen mit anderen Apache-Modulen kommunizieren.

Das Rezept 10.20 führt einige Methoden zum Profiling und Benchmarking Ihres Codes vor. Dadurch dass Sie feststellen, womit Ihre Programme den größten Teil ihrer Zeit verbringen, können Sie Ihre Entwicklungsbemühungen auf die Verbesserung jener Programmteile konzentrieren, die den spürbarsten Beschleunigungseffekt für Ihre Anwender haben. Rezept 10.21 zeigt, wie man einem Benutzer das mehrfache Herunterladen desselben dynamisch erzeugten Dokuments ersparen kann, ohne dass er dabei eventuelle Änderungen des Dokuments verpasst.

Dieses Kapitel enthält außerdem zwei Programme, die Sie bei der Wartung Ihrer Website unterstützen. Das Programm 10.22 überprüft Benutzerkonten, indem es an jeden neuen Benutzer eine E-Mail-Nachricht mit einem speziell angepassten Link versendet. Wenn der Benutzer den Link nicht innerhalb einer Woche nach dem Empfang der Nachricht aufsucht, wird das Konto gelöscht. Das Programm 10.23 überwacht in Realzeit Anfragen bezüglich der Benutzer und blockiert Anfragen von Benutzern, die zuvor Ihre Site mit Netzverkehr überflutet haben.

10.1 Cookies setzen

Problem

Sie möchten ein Cookie setzen.

Lösung

Verwenden Sie `setcookie()`:

```
setcookie('Geschmack','Chocolate-Chip');
```

Diskussion

Cookies werden durch HTTP-Header versandt, daher muss `setcookie()` aufgerufen werden, bevor irgendwelche Ausgaben generiert worden sind.

Sie können an `setcookie()` zusätzliche Argumente übergeben, um das Verhalten des Cookies zu steuern. Das dritte Argument für `setcookie()` ist ein Verfallsdatum, das als Epochen-Zeitstempel angegeben werden muss. Dieses Cookie beispielsweise läuft am Mittag des 3. Dezember 2004 (GMT) ab:

```
setcookie('Geschmack','Chocolate-Chip',1102075200);
```

Wenn das dritte Argument für `setcookie()` fehlt (oder leer ist), verfällt das Cookie beim Schließen des Browsers. Manche Systeme können mit keinem Cookie-Verfallsdatum umgehen, das größer als 2147483647 ist, denn dies ist der größte Epochen-Zeitstempel, der in eine 32-Bit-Integer-Zahl passt, wie in der Einführung zu Kapitel 3 bereits erörtert wurde.

Das vierte Argument für `setcookie()` ist der Pfad. Das Cookie wird nur dann an den Server zurückgesandt, wenn Seiten angefordert werden, deren Pfad mit dem angegebenen String übereinstimmt. Das folgende Cookie wird beispielsweise nur an Seiten zurückgesandt, deren Pfad mit */produkte/* beginnt:

```
setcookie('Geschmack','Chocolate-Chip','','/produkte/');
```

Der Pfad der Seite, die dieses Cookie setzt, muss nicht mit */produkte/* beginnen, aber das dabei entstehende Cookie wird nur an Seiten zurückgesandt, bei denen dies der Fall ist.

Das fünfte Argument für `setcookie()` ist ein Domain-Name. Das Cookie wird nur dann an den Server zurückgesandt, wenn Seiten angefordert werden, deren Host-Name mit der angegebenen Domain endet. Das erste Cookie im folgenden Code wird beispielsweise an alle Hosts in der Domain *example.com* zurückgesandt, aber das zweite Cookie wird nur bei Anfragen an den Host *jeannie.example.com* versandt:

```
setcookie('Geschmack','Chocolate-Chip','','','.example.com');
setcookie('Geschmack','Chocolate-Chip','','','jeannie.example.com');
```

Wenn die Domain des ersten Cookies einfach nur *example.com* anstelle von *.example.com* wäre, würde es nur an den einen Host *example.com* gesendet werden (und nicht an *www.example.com* oder *jeannie.example.com*).

Das letzte, optionale Argument für `setcookie()` ist ein Schalter, der, auf 1 gesetzt, den Browser dazu veranlasst, dieses Cookie nur über eine SSL-Verbindung zu senden. Dies kann hilfreich sein, wenn das Cookie sensible Informationen enthält; aber denken Sie daran, dass das Cookie dennoch in lesbarer Form auf dem Computer des Benutzers gespeichert wird.

Die verschiedenen Browser gehen mit Cookies in etwas unterschiedlicher Weise um, insbesondere in Hinsicht darauf, wie strikt sie die Übereinstimmung von Pfad- und Domain-Strings prüfen und wie sie die Prioritäten zwischen Cookies mit demselben

Namen bestimmen. Die setcookie()-Seite des Online-Handbuchs enthält einige hilfreiche Klärungen zu diesen Unterschieden.

Siehe auch

Rezept 10.2 zeigt das Lesen von Cookie-Inhalten; Rezept 10.3 zeigt das Löschen von Cookies; Rezept 10.12 erläutert das Puffern der Ausgabe; Rezept 10.14 zeigt, wie man die Fehlermeldung »headers already sent« vermeidet, die bisweilen beim Aufruf von setcookie() auftritt; die Dokumentation zu setcookie() unter *http://www.php.net/setcookie*; eine erweiterte Cookie-Spezifikation, detailliert dargestellt in RFC 2965 unter *http://www.faqs.org/rfcs/rfc2965.html*.

10.2 Cookie-Werte lesen

Problem

Sie möchten den Wert eines zuvor gesetzten Cookies lesen.

Lösung

Sehen Sie im superglobalen Array $_COOKIE nach:

```
if (isset($_COOKIE['Geschmack'])) {
    print "Sie haben einen $_COOKIE[Geschmack]-Keks gegessen.";
}
```

Diskussion

Der Wert eines Cookies ist innerhalb der Anfrage, in der das Cookie gesetzt wird, in $_COOKIE noch nicht verfügbar. Die Funktion setcookie() verändert den Inhalt von $_COOKIE also nicht. Bei nachfolgenden Anfragen werden dann jedoch alle Cookies in $_COOKIE gespeichert. Wenn register_globals eingeschaltet ist, werden außerdem die Cookie-Werte globalen Variablen zugewiesen.

Wenn ein Browser ein Cookie zurück an den Server sendet, übergibt er nur dessen Wert. Auf die Domain, den Pfad, das Verfallsdatum und die Sicherheitseinstellung können Sie über $_COOKIE nicht zugreifen, weil der Browser diese Informationen nicht an den Server sendet.

Um die Namen und Werte aller mit einer bestimmten Anfrage gesendeten Cookies auszugeben, können Sie das $_COOKIE-Array in einer Schleife durchlaufen:

```
foreach ($_COOKIE as $cookie_name => $cookie_value) {
    print "$cookie_name = $cookie_value<br>";
}
```

Siehe auch

Rezept 10.1 zeigt, wie Cookies gesetzt werden; Rezept 10.3 zeigt, wie man Cookies löscht; Rezept 10.12 erläutert die Ausgabe-Pufferung; Rezept 10.14 zeigt, wie man die Fehlermeldung »headers already sent« vermeiden kann, die bisweilen im Zusammenhang mit dem Aufruf von `setcookie()` auftritt; Rezept 11.7 mit Informationen zu `register_globals`.

10.3 Cookies löschen

Problem

Sie möchten ein Cookie löschen, sodass der Browser es nicht mehr zurück an den Server sendet. Beispiel: Sie nutzen Cookies, um festzustellen, ob ein User eingeloggt ist, und der User loggt sich aus.

Lösung

Rufen Sie `setcookie()` auf und geben Sie dabei keinen Inhalt an, aber ein in der Vergangenheit liegendes Verfallsdatum:

```
setcookie('Geschmack','',time()-86400);
```

Diskussion

Es ist eine gute Idee, das Verfallsdatum einige Stunden oder einen ganzen Tag in die Vergangenheit zu verlegen, falls die Uhren des Servers und des Benutzerrechners nicht synchron laufen. Wenn beispielsweise Ihr Server glaubt, es sei 15:06, aber der Computer eines Benutzers der Meinung ist, es sei 15:02 Uhr, wird ein Cookie mit dem Verfallsdatum 15:06 auf dem Rechner des Benutzers nicht gelöscht, obwohl die Zeit für den Server in der Vergangenheit liegt.

Der Aufruf von `setcookie()` zum Löschen eines Cookies muss die gleichen Argumente (mit Ausnahme des Werts und der Zeit) haben, wie der `setcookie()`-Aufruf zum Setzen des Cookie hatte. Geben Sie also auch den Pfad, die Domain und gegebenenfalls den Sicherheitsschalter an.

Siehe auch

Rezept 10.1 zeigt, wie man Cookies setzt; Rezept 10.2 zeigt, wie man Cookie-Werte liest; Rezept 10.12 erläutert die Ausgabe-Pufferung; Rezept 10.14 zeigt, wie man die Fehlermeldung »headers already sent« vermeidet, die bisweilen im Zusammenhang mit dem Aufruf von `setcookie()` auftritt; die Dokumentation zu `setcookie()` unter *http://www.php.net/setcookie*.

10.4 Zu einer anderen Adresse umleiten

Problem

Sie möchten einen Benutzer automatisch an eine andere URL verweisen. Beispielsweise können Sie nach dem erfolgreichen Speichern von Formulardaten den Benutzer zu einer Seite leiten, die den Empfang der Daten bestätigt.

Lösung

Übersenden Sie, bevor irgendetwas ausgegeben wird, mit `header()` einen `Location`-Header mit der neuen URL und rufen Sie danach `exit()` auf, sodass keine weiteren Ausgaben erfolgen:

```
header('Location: http://www.example.com/');
exit();
```

Diskussion

Wenn Sie der neuen Seite auch Variablen übergeben möchten, können Sie diese in den Query-String der URL einfügen:

```
header('Location: http://www.example.com/?affe=schildkroete');
```

Die URL, an die Sie einen Benutzer verweisen, wird mit GET abgerufen. Es ist nicht möglich, jemanden zum Abrufen einer URL über POST weiterzuleiten. Allerdings können Sie zusammen mit dem `Location`-Header noch weitere Header senden. Dies ist insbesondere dann sinnvoll, wenn der `Window-target`-Header gesetzt ist, der einen speziellen Frame oder ein Fenster benennt, in das die neue URL geladen werden soll:

```
header('Window-target: main');
header('Location: http://www.example.com/');
```

Die Umleitungs-URL muss den Protokoll- und den Host-Namen umfassen, der Pfadname allein genügt nicht:

```
// Funktionierende Umleitung
header('Location: http://www.example.com/katalog/essen/doerrfleisch.php');

// Nicht funktionierende Umleitung
header('Location: /katalog/essen/doerrfleisch.php');
```

Siehe auch

Die Dokumentation zu `header()` unter *http://www.php.net/header*.

10.5 Sitzungen verfolgen

Problem

Sie möchten Informationen über Benutzer erhalten und deren Bewegungen durch Ihre Site mitverfolgen.

Lösung

Setzen Sie das Session-Modul ein. Die Funktion `session_start()` initialisiert eine Session, und der Zugriff auf ein Element im globalen Array `$_SESSION` veranlasst PHP, die entsprechende Variable zu verfolgen.

```
session_start();
$_SESSION['visits']++;
print 'Sie haben uns '.$_SESSION['visits'].' mal besucht.';
```

Diskussion

Damit automatisch bei jeder Anfrage eine Session gestartet wird, setzen Sie in *php.ini* `session.auto_start` auf `1`. Wenn `session.auto_start` gesetzt ist, müssen Sie `session_start()` nicht aufrufen.

Die Session-Funktionen verfolgen die Aktivitäten der Benutzer, indem sie ihnen Cookies mit zufällig generierten Session-IDs übersenden. Wenn PHP feststellt, dass ein Benutzer das Cookie mit der Session-ID nicht annimmt, fügt es automatisch die Session-ID in die URLs und Formulare ein.[1] Sehen Sie sich beispielsweise den folgenden Code zur Ausgabe einer URL an:

```
print '<a href="train.php">Take the A Train</a>';
```

Wenn die Sitzungsverfolgung eingeschaltet ist, der Benutzer aber keine Cookies akzeptiert, sieht die an den Browser gesendete URL etwa folgendermaßen aus:

```
<a href="train.php?PHPSESSID=2eb89f3344520d11969a79aea6bd2fdd">Take the A Train</a>
```

In diesem Beispiel ist der Name der Session-ID `PHPSESSID`, und die Session-ID selbst hat den Wert `2eb89f3344520d11969a79aea6bd2fdd`. PHP fügt beide an die URL an, damit sie an die nächste Seite weitergegeben werden. Formulare werden so verändert, dass sie ein verborgenes Element zur Weitergabe der Session-ID enthalten. Umleitungen über den `Location`-Header werden nicht automatisch modifiziert, daher müssen Sie in diesem Fall die Session-ID mithilfe der Konstanten `SID` selbst hinzufügen:

```
$redirect_url = 'http://www.example.com/flugzeug.php';
if (defined('SID') && (! isset($_COOKIE[session_name()]))) {
```

1 Vor PHP 4.2.0 musste dieses Verhalten ausdrücklich aktiviert werden, indem der PHP-Build mit der Konfigurationseinstellung `--enable-trans-sid` ausgeführt wurde.

```
    $redirect_url .= '?' . SID;
}

header("Location: $redirect_url");
```

Aufgrund einer Vielzahl sicherheitsrelevanter Bedenken bezüglich der Einbettung der Session-ID in URLs ist `session.use_trans_sid` in der Grundeinstellung in der *php.ini* deaktiviert. Wenn Sie dieses Verhalten verändern möchten, müssen Sie entweder den entsprechenden Parameter in der *php.ini* aktivieren oder vor dem Start einer Sitzung `ini_set('session.use_trans_sid', true)` aufrufen.

Die Nutzung von `session.use_trans_sid` mag sehr bequem sein, ist aber vom Standpunkt der Sicherheit betrachtet nicht zu empfehlen. Da die URLs die Session-ID enthalten, kann sich jeder, der die URL sieht, als der Nutzer der Sitzung ausgeben. Bei einer Onlinebanking-Applikation kann das fatale Folgen haben. Ein Benutzer, der eine solche URL aus dem Browser kopiert und per E-Mail an Dritte versendet, gibt ungewollt seine Sitzungsdaten preis.

Die Funktion `session_name()` liefert den Namen des Cookies, in dem die Session-ID gespeichert ist, daher hängt dieser Code die `SID`-Konstante nur dann an `$redirect_url`, wenn die Konstante definiert ist und das Cookie nicht gesetzt wird.

Standardmäßig speichert PHP die Session-Daten auf dem Server im Verzeichnis */tmp*. Jede Session wird in ihrer eigenen Datei gespeichert. Um das Verzeichnis, in dem die Dateien abgelegt werden, zu ändern, setzen Sie in *php.ini* die Konfigurationsdirektive `session.save_path` auf das neue Verzeichnis. Sie können auch `session_save_path()` mit dem neuen Verzeichnis aufrufen, um das Verzeichnis zu ändern; allerdings müssen Sie dies vor dem Zugriff auf irgendwelche Session-Variablen tun.

Siehe auch

Die Dokumentationen zu `session_start()` unter *http://www.php.net/session-start* und zu `session_save_path()` unter *http://www.php.net/session-save-path*; das Session-Modul hat eine Reihe von Konfigurationsdirektiven, mit deren Hilfe Sie Dinge wie die maximale Länge einer Session und die Art ihrer Zwischenspeicherung verwalten können; Einzelheiten dazu finden Sie im Abschnitt »Session« des Online-Handbuchs unter *http://www.php.net/session*.

10.6 Sessions in einer Datenbank speichern

Problem

Sie möchten Sessions in einer Datenbank und nicht in Dateien speichern. Wenn mehrere Webserver auf dieselbe Datenbank zugreifen, werden auf diese Weise die Session-Daten über alle Webserver hinweg gespiegelt.

Lösung

Setzen Sie in *php.ini* den `session.save_handler` auf `user` und verwenden Sie die in Beispiel 10-1 dargestellte Klasse `pc_DB_Session`. Ein Beispiel:

```
$s = new pc_DB_Session('mysql://user:password@localhost/db');
ini_get('session.auto_start') or session_start();
```

Diskussion

Einer der leistungsfähigsten Aspekte des Session-Moduls ist seine Abstraktion der Art und Weise, wie die Session-Daten gespeichert werden. Die Funktion `session_set_save_handler()` veranlasst PHP dazu, andere Funktionen für diverse Session-Operationen wie das Speichern einer Session oder das Lesen von Session-Daten zu verwenden. Die Klasse `pc_DB_Session` speichert die Session-Daten in einer Datenbank. Wenn diese Datenbank von mehreren Webservern gemeinsam genutzt wird, sind die Benutzerinformationen zwischen allen diesen Webservern portabel. Wenn Sie also mehrere Webserver hinter einem Load-Balancer betreiben, müssen Sie keine raffinierten Tricks anwenden, um sicherzustellen, dass die Session-Daten der Benutzer unabhängig davon, auf welchen Server sie gerade geschickt werden, immer korrekt sind.

Um `pc_DB_Session` einzusetzen, übergeben Sie der Klasse beim Instantiieren einen Data Source Name (DSN). Die Session-Daten werden in einer Tabelle namens `php_session` gespeichert, deren Struktur folgendermaßen aussieht:

```
CREATE TABLE php_session (
  id CHAR(32) NOT NULL,
  data MEDIUMBLOB,
  last_access INT UNSIGNED NOT NULL,
  PRIMARY KEY(id)
)
```

Wenn der Tabellenname anders als `php_session` lauten soll, setzen Sie `session.save_path` in *php.ini* auf den neuen Tabellennamen. Beispiel 10-1 zeigt die Klasse `pc_DB_Session`.

Beispiel 10-1: Die Klasse pc_DB_Session

```
require 'PEAR.php';
require 'DB.php';

class pc_DB_Session extends PEAR {

    var $_dbh;
    var $_table;
    var $_connected = false;
    var $_gc_maxlifetime;
    var $_prh_read;
    var $error = null;

    /**
```

Beispiel 10-1: Die Klasse pc_DB_Session (Fortsetzung)

```
     * Constructor
     */
    function pc_DB_Session($dsn = null) {
        if (is_null($dsn)) {
            $this->error = PEAR::raiseError('Kein DSN angegeben');
            return;
        }

        $this->_gc_maxlifetime = ini_get('session.gc_maxlifetime');
        // Die Länge der Session beträgt einen Tag,
        // wenn nicht anders angegeben.
        if (! $this->_gc_maxlifetime) {
            $this->_gc_maxlifetime = 86400;
        }

        $this->_table = ini_get('session.save_path');
        if ((! $this->_table) || ('/tmp' == $this->_table)) {
            $this->_table = 'php_session';
        }

        $this->_dbh = DB::connect($dsn);
        if (DB::isError($this->_dbh)) {
            $this->error = $this->_dbh;
            return;
        }

        $this->_prh_read = $this->_dbh->prepare(
            "SELECT data FROM $this->_table WHERE id LIKE ? AND last_access >= ?");
        if (DB::isError($this->_prh_read)) {
            $this->error = $this->_prh_read;
            return;
        }

        if (! session_set_save_handler(array(&$this,'_open'),
                                       array(&$this,'_close'),
                                       array(&$this,'_read'),
                                       array(&$this,'_write'),
                                       array(&$this,'_destroy'),
                                       array(&$this,'_gc'))) {
            $this->error = PEAR::raiseError('session_set_save_handler() failed');
            return;
        }

        return $this->_connected = true;
    }

    function _open() {
        return $this->_connected;
    }

    function _close() {
```

Beispiel 10-1: Die Klasse pc_DB_Session (Fortsetzung)

```
        return $this->_connected;
    }

    function _read($id) {
        if (! $this->_connected) { return false; }
        $sth =
            $this->_dbh->execute($this->_prh_read,
                                 array($id,time() - $this->_gc_maxlifetime));
        if (DB::isError($sth)) {
            $this->error = $sth;
            return '';
        } else {
            if (($sth->numRows() == 1) &&
                ($ar = $sth->fetchRow(DB_FETCHMODE_ORDERED))) {
                return $ar[0];
            } else {
                return '';
            }
        }
    }

    function _write($id,$data) {
        $sth = $this->_dbh->query(
            "REPLACE INTO $this->_table (id,data,last_access) VALUES (?,?,?)",
            array($id,$data,time()));
        if (DB::isError($sth)) {
            $this->error = $sth;
            return false;
        } else {
            return true;
        }
    }

    function _destroy($id) {
        $sth = $this->_dbh->query("DELETE FROM $this->_table WHERE id LIKE ?",
                                  array($id));
        if (DB::isError($sth)) {
            $this->error = $sth;
            return false;
        } else {
            return true;
        }
    }

    function _gc($maxlifetime) {
        $sth = $this->_dbh->query("DELETE FROM $this->_table WHERE last_access < ?",
                                  array(time() - $maxlifetime));
        if (DB::isError($sth)) {
            $this->error = $sth;
            return false;
        } else {
```

Beispiel 10-1: Die Klasse pc_DB_Session (Fortsetzung)

```
            return true;
        }
    }
}
```

Die Methode `pc_DB_Session::_write()` verwendet den MySQL-spezifischen SQL-Befehl `REPLACE INTO`. Je nachdem, ob es in der Datenbank bereits einen Satz mit dem angegebenen `id`-Feld gibt, aktualisiert er den existierenden Datensatz oder fügt einen neuen ein. Wenn Sie eine andere Datenbank verwenden, müssen Sie die `_write()`-Funktion modifizieren, um dasselbe zu erreichen. So müssen Sie zum Beispiel die vorhandene Zeile (wenn es eine gibt) löschen und danach die neue einfügen, dies aber alles innerhalb einer Transaktion:

```
function _write($id,$data) {
    $sth = $this->_dbh->query('BEGIN WORK');
    if (DB::isError($sth)) {
        $this->error = $sth;
        return false;
    }
    $sth = $this->_dbh->query("DELETE FROM $this->_table WHERE id LIKE ?",
                              array($id));
    if (DB::isError($sth)) {
        $this->error = $sth;
        $this->_dbh->query('ROLLBACK');
        return false;
    }
    $sth = $this->_dbh->query(
        "INSERT INTO $this->_table (id,data,last_access) VALUES (?,?,?)",
        array($id,$data,time()));
    if (DB::isError($sth)) {
        $this->error = $sth;
        $this->_dbh->query('ROLLBACK');
        return false;
    }
    $sth = $this->_dbh->query('COMMIT');
    if (DB::isError($sth)) {
        $this->error = $sth;
        $this->_dbh->query('ROLLBACK');
        return false;
    }
         return true;
}
```

Siehe auch

Die Dokumentation zu `session_set_save_handler()` unter *http://www.php.net/session-set-save-handler*; ein Handler für PostgreSQL ist unter *http://www.zend.com/codex.php?id=456&single=1* verfügbar.

10.7 Verschiedene Browser erkennen

Problem

Sie möchten Inhalte generieren, die von den Fähigkeiten des vom Benutzer verwendeten Browsers abhängig sind.

Lösung

Verwenden Sie das von `get_browser()` zurückgegebene Objekt, um festzustellen, über welche Möglichkeiten ein Browser verfügt:

```
$browser = get_browser();

if ($browser->frames) {
    // Ein Layout mit Frames verwenden.
} elseif ($browser->tables) {
    // Ein Layout mit Tabellen verwenden.
} else {
    // Ein langweiliges Layout verwenden.
}
```

Diskussion

Die Funktion `get_browser()` untersucht die (vom Server gesetzte) Umgebungsvariable `$_ENV['HTTP_USER_AGENT']` und vergleicht sie mit den Browsern, die in einer externen Liste mit Browser-Fähigkeiten aufgeführt sind. Aufgrund von Lizenzproblemen wird PHP nicht mit einem solchen *Browser Capability File* ausgeliefert. Der Abschnitt »PHP beziehen« der PHP-FAQ (*http://www.php.net/faq.obtaining*) führt *http://browsers.garykeith.com/downloads.asp* als Quelle für solche Dateien auf.

Wenn Sie eine Browser-Capability-Datei heruntergeladen haben, müssen Sie PHP noch mitteilen, wo diese zu finden ist. Setzen Sie dazu die Konfigurationsdirektive `browscap` auf den Pfadnamen der Datei. Wenn Sie PHP als CGI verwenden, setzen Sie die Direktive in der *php.ini*-Datei:

```
browscap=/usr/local/lib/browscap.txt
```

Wenn Sie Apache verwenden, müssen Sie die Direktive in der Konfigurationsdatei Ihres Apache-Servers setzen:

```
php_value browscap "/usr/local/lib/browscap.txt"
```

Viele der Fähigkeiten, die `get_browser()` herausfindet, sind in Tabelle 10-1 dargestellt. Bei vom Benutzer konfigurierbaren Fähigkeiten wie `javascript` und `cookies` teilt Ihnen `get_browser()` nur mit, ob der Browser diese Funktionen prinzipiell unterstützt. Dass der Benutzer diese Funktionen abgeschaltet hat, erfahren Sie auf diese Weise nicht. Auch

wenn JavaScript bei einem JavaScript-fähigen Browser abgeschaltet wurde oder ein Benutzer die Annahme von Cookies verweigert, sobald der Browser ihn danach fragt, zeigt `get_browser()` trotzdem an, dass der Browser diese Funktionen unterstützt.

Tabelle 10-1: Eigenschaften des Browser-Capability-Objekts

Eigenschaft	Beschreibung
`platform`	Betriebssystem, unter dem der Browser läuft (z.B. Windows, Macintosh, UNIX, Win32, Linux, MacPPC)
`version`	vollständige Version des Browsers (z.B. 5.0, 3.5, 6.0b2)
`majorver`	Hauptversion des Browsers (z.B. 5, 3, 6)
`minorver`	Unterversion des Browsers (z.B. 0, 5, 02)
`frames`	1, wenn der Browser Frames unterstützt
`tables`	1, wenn der Browser Tabellen unterstützt
`cookies`	1, wenn der Browser Cookies unterstützt
`backgroundsounds`	1, wenn der Browser Hintergrundklänge mit `<embed>` oder `<bgsound>` unterstützt
`vbscript`	1, wenn der Browser VBScript unterstützt
`javascript`	1, wenn der Browser JavaScript unterstützt
`javaapplets`	1, wenn der Browser Java-Applets ausführen kann
`activexcontrols`	1, wenn der Browser ActiveX-Controls ausführen kann

Siehe auch

Die Dokumentation zu `get_browser()` unter *http://www.php.net/get-browser*.

10.8 Einen GET-Query-String bilden

Problem

Sie müssen einen Link aufbauen, der in einem Query-String Name/Wert-Paare enthält.

Lösung

Codieren Sie die Namen und Werte mit `urlencode()` und erzeugen Sie den Query-String mit `join()`:

```
$vars = array('name' => 'Oscar aus der Mülltonne',
              'farbe' => 'grün',
              'lieblingszeichen' => '#');
$querystring = http_build_query($vars);
$url = '/muppet/select.php?' . $querystring;
```

Diskussion

Die in der Lösung gebildete URL ist:

```
/muppet/select.php?name=Oscar+aus+der+M%FCllтonne&farbe=gr%FCn&lieblingszeichen=%23
```

Im Query-String sind die Leerzeichen als + codiert. Sonderzeichen sind hexadezimal codiert. Zum Beispiel ist das # als %23 codiert, da der ASCII-Wert von # 35 ist, und das ist als Hexadezimalzahl 23.

Obwohl `http_build_query()` verhindert, dass ein Sonderzeichen im Variablennamen oder -wert die erstellte URL stört, kann es Probleme mit Variablennamen geben, die mit den Namen von HTML-Entities beginnen. Betrachten Sie diesen Teil einer URL, mit dem man Informationen über eine Stereoanlage abrufen kann:

```
/stereo.php?speakers=12&cdplayer=52&amp=10
```

Die HTML-Entity für das kaufmännische Und (&) ist `&amp`, und das kann der Browser fehlinterpretieren als :

```
/stereo.php?speakers=12&cdplayer=52&=10
```

Um zu verhindern, dass eingebettete Entities Ihre URLs durcheinanderbringen, haben Sie drei Möglichkeiten. Die erste besteht darin, Variablennamen zu wählen, die nicht mit Entities verwechselt werden können, zum Beispiel `_amp` anstelle von `amp`. Als zweite Möglichkeit können Sie Zeichen, die HTML-Entity-Entsprechungen haben, zu diesen Entities konvertieren, bevor Sie die URL ausgeben. Verwenden Sie dazu `htmlentities()`:

```
$url = '/muppet/select.php?' . htmlentities($querystring);
```

Die resultierende URL ist:

```
/muppet/select.php?name=Oscar+aus+der+M%FCllтonne&farbe=gr%FCn&lieblingszeichen=%23
```

Ihre dritte Wahlmöglichkeit ist, dass Sie des Argument-Trennzeichens & durch ; ersetzen, indem Sie die Konfigurationsdirektive `arg_separator.input` auf ; setzen. Nun bilden Sie den Query-String, indem Sie die Name/Wert-Paare mit ; verknüpfen:

```
/muppet/select.php?name=Oscar+aus+der+M%FCllтonne;farbe=gr%FCn;lieblingszeichen=%23
```

Damit könnten Sie jedoch in Schwierigkeiten mit GET-Methoden-URLs geraten, die Sie nicht ausdrücklich mit Semikola zusammensetzen können. Dies ist bei Formularen der Fall, deren Methode auf GET gesetzt ist, da die Browser Ihrer Anwender das & als Trennzeichen für Argumente verwenden.

Weil viele Browser die Verwendung von ; als Argument-Trenner nicht unterstützen, vermeiden Sie die Probleme mit Entities in URLs am besten dadurch, dass Sie Variablennamen wählen, die sich nicht mit Entity-Namen überschneiden. Wenn Sie allerdings keine vollständige Kontrolle über die Variablennamen haben, verwenden Sie `htmlentities()`, um Ihre URLs vor der Dekodierung von Entities zu schützen.

Siehe auch

Die Dokumentationen zu `urlencode()` unter *http://www.php.net/urlencode* und zu `html-entities()` unter *http://www.php.net/htmlentities*.

10.9 HTTP-Basic- oder -Digest-Authentifizierung einsetzen

Problem

Sie möchten PHP einsetzen, um Teile Ihrer Website mit Passwörtern zu schützen. Anstatt die Passwörter in einer externen Datei zu speichern, um die Authentifizierung durch den Webserver abwickeln zu lassen, möchten Sie die Logik zur Prüfung des Passworts in einem PHP-Programm formulieren.

Lösung

Die globalen Variablen `$_SERVER['PHP_AUTH_USER']` und `$_SERVER['PHP_AUTH_PW']` enthalten den Benutzernamen und das Passwort, wenn der Benutzer sie angegeben hat. Um den Zugriff zu verweigern, versenden Sie einen `WWW-Authenticate`-Header mit dem Statuscode 401, der das Authentication Realm enthält, wie Sie es in Beispiel 10-2 sehen.

Beispiel 10-2: HTTP-Basic-Authentifizierung erzwingen

```
<?php
header('WWW-Authenticate: Basic realm="Meine Website"');
header('HTTP/1.0 401 Unauthorized');
echo "Sie müssen eine gültige Benutzername-Passwort-Kombination angeben.";
exit();
?>
```

Diskussion

Wenn ein Browser einen 401-Header erhält, blendet er ein Dialogfenster ein, das den Benutzer zur Eingabe eines Benutzernamens und eines Passworts auffordert. Die Authentifizierungsdaten (Benutzername und Passwort) werden, wenn sie vom Server akzeptiert werden, mit dem Realm (Reich) im `WWW-Authenticate`-Header verknüpft. Code, der die Authentifizierungsdaten prüft, muss ausgeführt werden, bevor Ausgaben an den Browser geschickt werden, da er Header versenden könnte. Beispielsweise können Sie eine Funktion wie `pc_validate()` verwenden, wie in Beispiel 10-3 gezeigt.

Beispiel 10-3: pc_validate()

```
<?php
function pc_validate($user,$pass) {
    /* Ersetzen Sie den folgenden Code durch eine geeignete Benutzername-
       Passwort-Prüfung, beispielsweise mit Rückgriff auf eine Datenbank. */
    $users = array('david' => 'fadj&32',
                   'adam'  => '8HEj838');

    if (isset($users[$user]) && ($users[$user] == $pass)) {
        return true;
    } else {
        return false;
    }
}
?>
```

Beispiel 10-4 zeigt, wie man `pc_validate( )` einsetzt.

Beispiel 10-4: Eine Validierungsfunktion einsetzen

```
<?php
if (! pc_validate($_SERVER['PHP_AUTH_USER'], $_SERVER['PHP_AUTH_PW'])) {
    header('WWW-Authenticate: Basic realm="Meine Website"');
    header('HTTP/1.0 401 Unauthorized');
    echo "Sie müssen eine gültige Benutzername-Passwort-Kombination angeben.";
    exit;
}
?>
```

Ersetzen Sie den Inhalt der Funktion `pc_validate( )` durch eine geeignete Logik zur Prüfung der Richtigkeit der gelieferten Benutzername-Passwort-Kombination. Sie können auch das Realm »Meine Website« und die Nachricht ändern, die ausgegeben wird, wenn der Benutzer im Authentifizierungsfenster seines Browser auf *Abbrechen* klickt, damit etwas anderes als »Sie müssen eine gültige Benutzername-Passwort-Kombination angeben.« ausgegeben wird.

PHP 5.1.0 und höher unterstützen neben HTTP Basic-Authentifizierung auch Digest-Authentifizierung. Bei der Basic-Authentifizierung werden Benutzername und Passwort im Klartext über das Netzwerk versendet, nur minimal verschleiert durch eine Base64-Kodierung. Bei der Digest-Authentifizierung wird das Passwort nie vom Browser an den Server geschickt. Stattdessen wird gemeinsam mit einigen anderen Werten ein Hash des Passworts verschickt. Das mindert das Risiko, dass der Netzwerkverkehr von einem Angreifer abgefangen und nachvollzogen wird. Die verbesserte Sicherheit der Digest-Authentifizierung bedeutet, dass die Implementierung der Authentifizierung komplizierter als ein einfacher Passwortvergleich ist. Beispiel 10-5 nutzt Funktionen, die die Digest-Authentifizierung umsetzen, wie in RFC 2617 beschrieben.

Beispiel 10-5: Digest-Authentifizierung einsetzen

```
<?php
/*  Ersetzen Sie Nachfolgendes durch eine geeignete Benutzername-Passwort-Prüfung,
    z.B. unter Rückgriff auf eine Datenbank. */
$users = array('david' => 'fadj&32',
               'adam'  => '8HEj838');
$realm = 'Meine Website';

$username = pc_validate_digest($realm, $users);

// Werden ungültige Daten angegeben, erreicht die Ausführung diesen Punkt nicht.
print "Hallo, " . htmlentities($username);

function pc_validate_digest($realm, $users) {
    // Abbrechen, wenn Client keinen Digest lieferte.
    if (! isset($_SERVER['PHP_AUTH_DIGEST'])) {
        pc_send_digest($realm);
    }
    // Abbrechen, wenn Digest nicht geparst werden kann.
    $username = pc_parse_digest($_SERVER['PHP_AUTH_DIGEST'], $realm, $users);
    if ($username === false) {
        pc_send_digest($realm);
    }
    // Digest enthielt gültigen Benutzernamen
    return $username;
}

function pc_send_digest($realm) {
    header('HTTP/1.0 401 Unauthorized');
    $nonce = md5(uniqid());
    $opaque = md5($realm);
    header("WWW-Authenticate: Digest realm=\"$realm\" qop=\"auth\" ".
           "nonce=\"$nonce\" opaque=\"$opaque\"");
    echo "Sie müssen eine gültige Benutzername-Passwort-Kombination angeben.";
    exit;
}

function pc_parse_digest($digest, $realm, $users) {
    // Wir müssen die Werte für folgende Elemente im Digest-Header finden:
    // username, uri, qop, cnonce, nc und response
    $digest_info = array();
    foreach (array('username','uri','nonce','cnonce','response') as $part) {
        // Der Begrenzer kann entweder ' oder " oder nichts sein (bei qop und nc).
        if (preg_match('/'.$part.'=([\'"]?)(.*?)\1/', $digest, $match)) {
            // Teil wurde gefunden und wird für Berechnungen gespeichert.
            $digest_info[$part] = $match[2];
        } else {
            // Der Teil fehlt, Digest kann also nicht validiert werden.
            return false;
        }
    }
```

Beispiel 10-5: Digest-Authentifizierung einsetzen (Fortsetzung)

```
    // Prüfen, ob der qop angegeben wurde.
    if (preg_match('/qop=auth(,|$)/', $digest)) {
        $digest_info['qop'] = 'auth';
    } else {
        return false;
    }
    // Prüfen, ob eine gültige nonce-Zahl angegeben wurde.
    if (preg_match('/nc=([0-9a-f]{8})(,|$)/', $digest, $match)) {
        $digest_info['nc'] = $match[1];
    } else {
        return false;
    }

    // Nachdem alle erforderlichen Werte aus dem Digest-Header ausgelesen wurden,
    // können die algorithmischen Berechnungen durchgeführt werden, mit denen
    // geprüft wird, ob die richtigen Informationen angegeben wurden.
    //
    // Diese Berechnungen werden in den Abschnitten 3.2.2, 3.2.2.1
    // und 3.2.2.2 von RFC 2617 beschrieben.
    // Der Algorithmus ist MD5.
     $A1 = $digest_info['username'] . ':' . $realm . ':' . $users[$digest_info['username']];
    // qop ist 'auth'
    $A2 = $_SERVER['REQUEST_METHOD'] . ':' . $digest_info['uri'];
     $request_digest = md5(implode(':', array(md5($A1), $digest_info['nonce'], $digest_
info['nc'],
    $digest_info['cnonce'], $digest_info['qop'], md5($A2))));

    // Entsprechen die übermittelten den berechneten Werten?
    if ($request_digest != $digest_info['response']) {
        return false;
    }

    // Alles in Ordnung, also Benutzername zurückliefern.
    return $digest_info['username'];
}
?>
```

Wenn Sie nicht mit PHP 5.1.0 oder höher arbeiten, PHP aber als Apache-Modul einsetzen, können Sie Digest-Authentifizierung mit Code wie der `HTTPDigest`-Klasse von Paul James implementieren, die Sie unter *http://www.peej.co.uk/projects/phphttpdigest.html* finden.

Weder HTTP Basic- noch Digest-Authentifizierung ist möglich, wenn Sie PHP als CGI-Programm ausführen. Können Sie PHP nicht als Servermodul ausführen, können Sie die Cookie-basierte Authentifizierung verwenden, die in Rezept 10.10 beschrieben wird.

Ein weiteres Problem bei der HTTP-Authentifizierung ist, dass sie dem Benutzer, mit Ausnahme der Beendigung der Browsersitzung, keine einfache Möglichkeit bietet, sich auszuloggen. Das PHP-Online-Manual schlägt einige Logout-Methoden vor, die je nach Server-Browser-Kombination unterschiedlich erfolgreich sind. Diese finden Sie unter *http://www.php.net/features.http-auth*.

Es gibt allerdings eine einfache Möglichkeit, das Ausloggen eines Benutzers nach einer bestimmten Zeit zu erzwingen: Schließen Sie eine Zeitberechnung in den Realm-String ein. Browser nutzen immer wieder die gleiche Benutzername-Passwort-Kombination, wenn sie aufgefordert werden, Zugangsberechtigungen für ein bestimmtes Realm bereitzustellen. Wird der Realm-Name geändert, muss der Browser den Benutzer erneut zur Eingabe der Berechtigungsdaten auffordern. Beispiel 10-6 nutzt eine Basic-Authentifizierung, die den Benutzer um Mitternacht automatisch ausloggt.

Beispiel 10-6: Logout bei Basic-Authentifizierung erzwingen

```
<?php
if (! pc_validate($_SERVER['PHP_AUTH_USER'],$_SERVER['PHP_AUTH_PW'])) {
    $realm = 'Meine Website: '.date('Y-m-d');
    header('WWW-Authenticate: Basic realm="'.$realm.'"');
    header('HTTP/1.0 401 Unauthorized');
    echo "Sie müssen eine gültige Benutzername-Passwort-Kombination angeben.";
    exit;
}
?>
```

Benutzerbezogene Verfallszeiten können Sie auch einrichten, ohne den Realm-Namen zu ändern, indem Sie die Zeit speichern, zu der ein Benutzer auf eine geschützte Seite zugreift. Die Funktion `pc_validate2( )` in Beispiel 10-7 speichert die Login-Zeit in einer Datenbank und erzwingt ein Logout, wenn mehr als 15 Minuten verstrichen sind, seit der Benutzer das letzte Mal auf eine geschützte Seite zugegriffen hat.

Beispiel 10-7: pc_validate2( )

```
<?php
function pc_validate2($user,$pass) {
    $safe_user = strtr(addslashes($user),array('_' => '\_', '%' => '\%'));
    $r = mysql_query("SELECT password,last_access
                      FROM users WHERE user LIKE '$safe_user'");

    if (mysql_numrows($r) == 1) {
        $ob = mysql_fetch_object($r);
        if ($ob->password == $pass) {
            $now = time();
            if (($now - $ob->last_access) > (15 * 60)) {
                return false;
            } else {
                // Die Zeit des letzten Zugriffs aktualisieren.
                mysql_query("UPDATE users SET last_access = NOW()
                             WHERE user LIKE '$safe_user'");
               return true;
            }
        }
    } else {
        return false;
    }
}
```

Siehe auch

Rezept 10.10; den Abschnitt zur HTTP-Authentifizierung im PHP-Online-Manual unter *http://www.php.net/features.http-auth*.

10.10 Cookie-Authentifizierung verwenden

Problem

Sie möchten mehr Kontrolle über die Benutzer-Anmeldeprozedur haben und dabei zum Beispiel ein eigenes Anmeldeformular anzeigen lassen.

Lösung

Speichern Sie den Authentifizierungsstatus als Teil einer Session in einem Cookie. Nachdem ein Benutzer sich erfolgreich angemeldet hat, legen Sie seinen Benutzernamen in einem Cookie ab. Außerdem fügen Sie den Hash-Code aus dem Benutzernamen und einem geheimen Wort hinzu, damit sich ein Benutzer nicht einfach selbst ein Authentifizierungs-Cookie mit einem darin enthaltenen Benutzernamen bauen kann:

```
$secret_word = 'if i ate spinach';
if (pc_validate($_REQUEST['username'],$_REQUEST['password'])) {
    setcookie('login',
              $_REQUEST['username'].','.md5($_REQUEST['username'].$secret_word));
}
```

Diskussion

Wenn Sie mit Cookie-Authentifizierung arbeiten möchten, müssen Sie ein eigenes Anmeldeformular ausgeben:

```
<form method="post" action="login.php">
Benutzername: <input type="text" name="username"> <br>
Passwort: <input type="password" name="password"> <br>
<input type="submit" value="Log In">
</form>
```

Sie können die Funktion `pc_validate()` aus Rezept 10.9 verwenden, um den Benutzernamen und das Passwort prüfen zu lassen. Der einzige Unterschied besteht darin, dass Sie Ihr `$_REQUEST['username']` und `$_REQUEST['password']` als Legitimation übergeben und nicht `$_SERVER['PHP_AUTH_USER']` und `$_SERVER['PHP_AUTH_PW']`. Wenn das Passwort korrekt ist, senden Sie ein Cookie zurück, das einen Benutzernamen und einen Hash aus dem Benutzernamen und einem geheimen Wort enthält. Auf diese Weise verhindern Sie, dass ein Benutzer die Anmeldung fälscht, indem er einfach ein Cookie mit einem darin enthaltenen Benutzernamen sendet.

Nachdem sich der Benutzer angemeldet hat, muss in den Seiten nur noch geprüft werden, ob ein gültiges Login-Cookie übersandt wurde, bevor spezielle Aufgaben für den angemeldeten Benutzer ausgeführt werden:

```
unset($username);
if ($_COOKIE['login']) {
    list($c_username,$cookie_hash) = split(',',$_COOKIE['login']);
    if (md5($c_username.$secret_word) == $cookie_hash) {
        $username = $c_username;
    } else {
        print "Sie haben ein ung&uuml;ltiges Cookie gesandt.";
    }
}

if ($username) {
    print "Guten Tag, $username.";
} else {
    print "Guten Tag, anonymer Benutzer.";
}
```

Wenn Sie die eingebaute Session-Unterstützung verwenden, können Sie den Benutzernamen und den Hash-Code auch den Session-Daten zufügen und dadurch das Versenden eines gesonderten Cookies vermeiden. Wenn sich jemand anmeldet, setzen Sie eine zusätzliche Variable in der Session, anstatt ein Cookie zu versenden:

```
if (pc_validate($_REQUEST['username'],$_REQUEST['password'])) {
    $_SESSION['login'] =
        $_REQUEST['username'].','.md5($_REQUEST['username'].$secret_word));
}
```

Der Code zur Überprüfung ist fast der gleiche, er verwendet lediglich `$_SESSION` anstelle von `$_COOKIE`:

```
unset($username);
if ($_SESSION['login']) {
    list($c_username,$cookie_hash) = explode(',',$_SESSION['login']);
    if (md5($c_username.$secret_word) == $cookie_hash) {
        $username = $c_username;
    } else {
        print "Sie haben an Ihrer Session herumgepfuscht.";
    }
}
```

Wenn Sie Cookie- oder Session-Authentifizierung anstelle der HTTP-Basic-Authentifizierung einsetzen, ist die Abmeldung von Benutzern viel einfacher: Sie löschen nur den Login-Cookie oder entfernen die Login-Variable aus der Session. Wenn Sie die Authentifizierungsinformation in der Session speichern, hat dies darüber hinaus den Vorteil, dass Sie die Webaktivitäten von angemeldeten Benutzern mit denen vor ihrer Anmeldung und nach ihrer Abmeldung verknüpfen können. Bei HTTP-Basic-Authentifizierung gibt es keine Möglichkeit, Anfragen nach Angabe eines Benutzernamens mit den Anfragen desselben Benutzers in Verbindung zu bringen, die er vor der Angabe seines Benutzernamens getätigt hat. Wenn Sie versuchen, nach Anfragen von derselben IP-Adresse zu

suchen, kann dies zu falschen Ergebnissen führen, insbesondere wenn sich Benutzer hinter einer Firewall oder einem Proxyserver befinden. Beim Einsatz von Sessions können Sie den Login-Vorgang so verändern, dass die Verbindung zwischen der Session-ID und dem Benutzernamen protokolliert wird:

```
if (pc_validate($_REQUEST['username'],$_REQUEST['password'])) {
    $_SESSION['login'] =
        $_REQUEST['username'].','.md5($_REQUEST['username'].$secret_word));
    error_log('Session id '.session_id().' log in as '.$_REQUEST['username']);
}
```

Dieses Beispiel schreibt eine Nachricht in ein Fehlerprotokoll, aber es könnte ebenso gut die Informationen in einer Datenbank aufzeichnen, sodass Sie diese verwenden können, wenn Sie die Seitenverwendung und den Netzverkehr analysieren.

Eine Gefahr bei der Verwendung von Session-IDs besteht darin, dass Sessions »entführt« werden können. Wenn Alice die Session-ID von Bob errät, kann sie sich dem Webserver gegenüber als Bob ausgeben. Das Session-Modul hat zwei optionale Konfigurationsdirektiven, mit denen Sie Session-IDs schwerer erratbar machen können. Die Direktive `session.entropy_file` enthält einen Pfad zu einem Device oder einer Datei, die zufällige Werte erzeugt, zum Beispiel */dev/random* oder */dev/urandom*. Die Direktive `session.entropy_length` enthält die Anzahl der Bytes, die aus der Entropie-Datei gelesen werden sollen, wenn eine Session-ID erzeugt wird.

Unabhängig davon, wie schwer Session-IDs zu erraten sind, können sie außerdem noch gestohlen werden, wenn sie im Klartext zwischen Ihrem Server und dem Browser des Benutzers übersandt werden. Bei der HTTP-Basic-Authentifizierung gibt es dieses Problem ebenfalls. Mithilfe von SSL können Sie sich gegen Netzwerkschnüffelei schützen, Rezept 17.10 sagt Ihnen, wie.

Siehe auch

Rezept 10.9; Rezept 17.3 behandelt das Verifizieren von Daten anhand von Hash-Codes; die Dokumentationen zu `setcookie()` unter *http://www.php.net/setcookie* und `md5()` unter *http://www.php.net/md5*.

10.11 Ausgaben vorzeitig an den Browser senden

Problem

Sie möchten erreichen, dass die Ausgabe an den Browser gesendet wird. Zum Beispiel sollen die Benutzer eine Statusmitteilung erhalten, bevor Sie eine langwierige Datenbankabfrage durchführen.

Lösung

Verwenden Sie `flush()`:

```
print 'Suche nach identischen Schneeflocken...';
flush();
$sth = $dbh->query(
    'SELECT form,COUNT(*) AS c FROM schnee GROUP BY form HAVING c > 1');
```

Diskussion

Die Funktion `flush()` sendet alle Ausgaben, die PHP intern zwischengespeichert hat, an den Webserver; allerdings kann der Webserver möglicherweise noch einen eigenen Zwischenspeicher haben, der die Auslieferung der Daten an den Browser verzögert. Außerdem zeigen nicht alle Browser die Daten unmittelbar dann an, wenn sie diese empfangen, und manche Versionen des Internet Explorer zeigen eine Seite erst an, wenn sie mindestens 256 Bytes empfangen haben. Mit Leerzeichen am Anfang der Seite können Sie den IE zur Ausgabe des Inhalts veranlassen:

```
print str_repeat(' ',300);
print 'Suche identische Schneeflocken...';
flush();
$sth = $dbh->query(
    'SELECT form,COUNT(*) AS c FROM schnee GROUP BY form HAVING c > 1');
```

Siehe auch

Rezept 21.17; die Dokumentation zu `flush()` unter *http://www.php.net/flush*.

10.12 Ausgaben an den Browser zwischenspeichern

Problem

Sie möchten mit dem Generieren von Ausgaben beginnen, bevor Sie mit dem Versenden von Headern oder Cookies fertig sind.

Lösung

Rufen Sie am Anfang Ihrer Seite `ob_start()` und am Ende Ihrer Seite `ob_end_flush()` auf. Dann können Sie Befehle zum Generieren von Ausgaben mit Befehlen zum Versenden von Headern vermischen. Die Ausgabe wird nicht versandt, bevor `ob_end_flush()` aufgerufen wird:

```
<?php ob_start(); ?>

Ich habe noch nicht entschieden, ob ich ein Cookie senden m&ouml;chte.
```

```
<?php setcookie('heron','great blue'); ?>

Ja, es war richtig, dieses Cookie zu senden.

<?php ob_end_flush(); ?>
```

Diskussion

Sie können `ob_start()` den Namen einer Callback-Funktion übergeben, damit der Ausgabepuffer mit dieser Funktion verarbeitet wird. Dies ist nützlich, wenn Sie den gesamten Inhalt einer Seite einer Nachverarbeitung unterziehen möchten, zum Beispiel um E-Mail-Adressen vor Robotern zu schützen, die Adressen sammeln:

```
<?php
function mangle_email($s) {
    return preg_replace('/([^@\s]+)@([-a-z0-9]+\.)+[a-z]{2,}/is',
                        '<$1@...>',
                        $s);
}

ob_start('mangle_email');
?>

Ich m&ouml;chte nicht, dass Spam an ronald@example.com gesendet wird!

<?php ob_end_flush(); ?>
```

Die Funktion `mangle_email()` transformiert die Ausgabe in:

```
Ich m&ouml;chte nicht, dass Spam an <ronald@...> gesendet wird!
```

Die Konfigurationsdirektive `output_buffering` schaltet die Ausgabe-Pufferung für alle Seiten ein:

```
output_buffering = On
```

Auf ähnliche Weise legt `output_handler` eine Callback-Funktion zur Verarbeitung des Ausgabepuffers für alle Seiten fest:

```
output_handler=mangle_email
```

Wenn Sie `output_handler` setzen, schaltet dies automatisch auch `output_buffering` auf `on`.

Siehe auch

Rezept Rezept 12.11 setzt die Ausgabe-Pufferung in einer Datenbank-Protokollfunktion ein; die Dokumentationen zu `ob_start()` unter *http://www.php.net/ob-start*, `ob_end_flush()` unter *http://www.php.net/ob-end-flush* und Ausgabe-Pufferung unter *http://www.php.net/outcontrol*.

10.13 Web-Ausgaben mit gzip komprimieren

Problem

Sie möchten Inhalte komprimiert an solche Browser senden, die automatische Dekompression unterstützen.

Lösung

Fügen Sie die folgende Einstellung in Ihre *php.ini*-Datei ein:

```
zlib.output_compression=1
```

Diskussion

Mit dem Header `Accept-Encoding` teilen Browser dem Server mit, dass sie komprimierte Antworten annehmen können. Wenn ein Browser `Accept-Encoding: gzip` oder `Accept-Encoding: deflate` sendet und ein PHP mit der *zlib*-Erweiterung gebaut worden ist, veranlasst die Konfigurationsdirektive `zlib.output_compression` PHP, die Ausgabe mit dem entsprechenden Algorithmus zu komprimieren, bevor sie an den Browser gesendet wird. Der Browser dekomprimiert die Daten, bevor er sie anzeigt.

Mit der Konfigurationsdirektive `zlib.output_compression_level` können Sie den Grad der Komprimierung einstellen:

```
; Minimale Komprimierung
zlib.output_compression_level=1
; Maximale Komprimierung
zlib.output_compression_level=9
```

Bei hohen Komprimierungsgraden müssen weniger Daten vom Server an den Browser gesendet werden, aber es wird mehr Server-CPU-Zeit benötigt, um die Daten zu komprimieren.

Siehe auch

Die Dokumentation zur *zlib*-Erweiterung unter *http://www.php.net/zlib*.

10.14 Den Fehler »headers already sent« vermeiden

Problem

Sie versuchen, mit `header()` einen HTTP-Header oder mit `setcookie()` ein Cookie zu senden, aber PHP meldet den Fehler »headers already sent«.

Lösung

Dieser Fehler tritt auf, wenn Sie Ausgaben senden, die keine Header sind, und danach `header()` oder `setcookie()` aufrufen.

Schreiben Sie Ihren Code so um, dass alle Ausgaben erst nach dem Senden der Header vorgenommen werden:

```
// Korrekt
setcookie("name", $name);
print "Hallo $name!";

// Falsch
print "Hallo $name!";
setcookie("name", $name);

// Korrekt
<?php setcookie("name",$name); ?>
<html><title>Hallo</title>
```

Diskussion

Eine HTTP-Nachricht besteht aus einem Header und einem Body, die in dieser Reihenfolge an den Client gesendet werden. Sobald Sie mit dem Versenden des Bodys beginnen, können Sie keinen Header mehr senden. Wenn Sie also `setcookie()` aufrufen, nachdem Sie bereits HTML ausgegeben haben, kann PHP die entsprechenden `Cookie`-Header nicht mehr senden.

Entfernen Sie auch Leerraum am Ende von Include-Dateien. Wenn Sie eine Datei mit leeren Zeilen außerhalb der `<?php ?>`-Tags einbeziehen, werden die Leerzeilen an den Browser gesendet. Mithilfe von `trim()` können Sie führende und nachfolgende Leerzeilen aus Dateien entfernen:

```
$file = '/pfad/zu/file.php';

// Sicherheitskopie
copy($file, "$file.bak") or die("Kann $file nicht kopieren: $php_errormsg);

// Datei lesen und beschneiden
$contents = trim(join('',file($file)));

// Schreiben
$fh = fopen($file, 'w')
       or die("Kann $file nicht zum Schreiben öffnen: $php_errormsg);
if (-1 == fwrite($fh, $contents))
       {die("Kann nicht in $file schreiben: $php_errormsg); }
fclose($fh)
       or die("Kann $file nicht schließen: $php_errormsg);
```

Es könnte bequemer sein, die Dateien nicht einzeln, sondern verzeichnisweise zu bearbeiten. Rezept 22.7 beschreibt, wie Sie alle Dateien in einem Verzeichnis verarbeiten können.

Wenn Sie sich keine Gedanken über Leerzeilen machen möchten, die das Versenden von Headern stören, sollten Sie die Zwischenspeicherung der Ausgaben einschalten. Das Puffern der Ausgabe hält PHP davon ab, sofort alle Ausgaben an den Client zu senden. Wenn Sie Ihre Ausgaben zwischenspeichern, können Sie Header und Body-Text beliebig mischen. Allerdings wirkt dies für die Anwender möglicherweise so, als würde Ihr Server mehr Zeit zur Abarbeitung ihrer Anfragen benötigen, da sie etwas länger warten müssen, bis der Browser eine Ausgabe anzeigt.

Siehe auch

Rezept 10.12 behandelt die Ausgabe-Pufferung; Rezept 22.7 zur Verarbeitung aller Dateien in einem Verzeichnis; die Dokumentation zu `header()` unter *http://www.php.net/header*.

10.15 Umgebungsvariablen lesen

Problem

Sie benötigen den Inhalt einer Umgebungsvariablen.

Lösung

Lesen Sie den Wert aus dem superglobalen Array `$_ENV`:

```
$name = $_ENV['USER'];
```

Diskussion

Umgebungsvariablen sind benannte, einem Prozess zugeordnete Werte. Unter Unix können Sie zum Beispiel den Inhalt von `$_ENV['HOME']` überprüfen, um das Home-Verzeichnis eines Benutzers herauszubekommen:

```
print $_ENV['HOME']; // Home-Verzeichnis des Benutzers
/home/adam
```

Frühere PHP-Versionen erzeugten standardmäßig PHP-Variablen automatisch für alle Umgebungsvariablen. Seit 4.1.0 schaltet *php.ini-recommended* dies aus Geschwindigkeitsgründen ab; *php.ini-dist* erlaubt aber weiterhin das Lesen der Umgebungsvariablen, um abwärtskompatibel zu bleiben.

Das Array `$_ENV` wird nur angelegt, wenn der Wert der Konfigurationsdirektive `variables_order` ein `E` enthält. Wenn `$_ENV` nicht verfügbar ist, können Sie eine Umgebungsvariable auch mit `getenv()` auslesen:

```
$path = getenv('PATH');
```

Sie können die Funktion `getenv()` jedoch nicht verwenden, wenn Sie PHP als ISAPI-Modul laufen lassen.

Siehe auch

Rezept 10.16 zum Einstellen von Umgebungsvariablen; die Dokumentation zu `getenv()` unter *http://www.php.net/getenv*; Informationen zu Umgebungsvariablen in PHP unter *http://www.php.net/manual/reserved.variables.environment.php*.

10.16 Umgebungsvariablen setzen

Problem

Sie möchten in einem Skript oder in Ihrer Serverkonfiguration eine Umgebungsvariable setzen. Wenn Sie Umgebungsvariablen in der Serverkonfiguration für die Hosts einzeln setzen, können Sie virtuelle Hosts unterschiedlich konfigurieren.

Lösung

Um eine Umgebungsvariable in einem Skript zu setzen, verwenden Sie `putenv()`:

```
putenv('ORACLE_SID=ORACLE'); // OCI-Erweiterung konfigurieren
```

Um eine Umgebungsvariable in Ihrer Apache-Konfigurationsdatei *httpd.conf* zu setzen, verwenden Sie `SetEnv` (Variablen, die auf diese Weise gesetzt werden, tauchen im superglobalen Array `$_SERVER` und nicht in `$_ENV` auf):

```
SetEnv DATABASE_PASSWORD password
```

Diskussion

Variablen in *httpd.conf* zu setzen hat den Vorteil, dass Sie die Lese-Erlaubnis restriktiver handhaben können als bei Ihren PHP-Skripten. PHP-Dateien müssen für den Webserver-Prozess lesbar sein, und daher können sie im Allgemeinen auch von anderen Anwendern gesehen werden. Dadurch dass Sie Passwörter in *httpd.conf* speichern, vermeiden Sie es, dass die Passworte in öffentlich zugänglichen Dateien liegen. Außerdem können Sie, wenn bei Ihnen mehrere Host-Namen auf dieselbe Dokumentwurzel abgebildet werden, Ihre Skripten so konfigurieren, dass sich die Serverprozesse abhängig vom Host-Namen unterschiedlich verhalten.

Angenommen, Sie haben die Host-Namen *mitglied.example.com* und *gast.example.com*. Die Mitglied-Version erfordert eine Authentifizierung und ermöglicht zusätzliche Zugriffsmöglichkeiten. Die Gast-Version bietet nur beschränkte Möglichkeiten, benötigt aber keine Authentifizierung:

```
$version = $_ENV['SITE_VERSION'];

// Zu http://gast.example.com umleiten, wenn sich der Benutzer nicht korrekt anmeldet.
if ('mitglied' == $version) {
    if (!authenticate_user($_REQUEST['username'], $_REQUEST['password'])) {
        header('Location: http://gast.example.com/');
        exit;
    }
}

include_once "${version}_header"; // Lädt benutzerspezifischen Header.
```

Siehe auch

Rezept 10.15 zum Auslesen von Umgebungsvariablen; die Dokumentation zu `putenv()` unter *http://www.php.net/putenv*; Informationen zum Setzen von Umgebungsvariablen in Apache unter *http://httpd.apache.org/docs/mod/mod_env.html*.

10.17 Konfigurationsvariablen lesen

Problem

Sie möchten den Inhalt einer PHP-Konfigurationseinstellung auslesen.

Lösung

Verwenden Sie `ini_get()`:

```
// Den Include-Pfad herausfinden:
$include_path = ini_get('include_path');
```

Diskussion

Alle Konfigurationsvariablen auf einmal erhalten Sie durch den Aufruf von `ini_get_all()`. Diese Funktion liefert die Variablen in einem assoziativen Array, wobei jedes Array-Element selbst wieder ein assoziatives Array ist. Das zweite Array hat drei Elemente: einen globalen Wert für die Einstellung, einen lokalen Wert und einen Zugriffs-Code:

```
// Alle Konfigurationsvariablen in einem assoziativen Array ablegen.
$vars = ini_get_all();
print_r($vars['include_path']);
Array
(
    [global_value] => .:/usr/local/lib/php/
    [local_value] => .:/usr/local/lib/php/
    [access] => 7
)
```

Der global_value ist der in *php.ini* gesetzte Wert; der local_value berücksichtigt Änderungen, die möglicherweise in der Konfigurationsdatei des Webservers, allen relevanten *.htaccess*-Dateien sowie im aktuellen Skript vorgenommen wurden. Der Inhalt von access ist eine numerische Konstante, die die Stellen repräsentiert, an denen dieser Wert geändert werden kann. Tabelle 10-2 erläutert die access-Werte. Beachten Sie, dass der Name access (Zugriff) insofern etwa irreführend ist, als der Wert einer Einstellung zwar jederzeit geprüft, nur nicht immer verändert werden kann.

Tabelle 10-2: Access-Werte

Wert	PHP-Konstante	Bedeutung
1	PHP_INI_USER	Jedes Skript mithilfe von ini_set().
2	PHP_INI_PERDIR	Auf Verzeichnisebene mithilfe von *.htaccess*.
4	PHP_INI_SYSTEM	Auf Systemebene mithilfe von *php.ini* oder *httpd.conf*.
7	PHP_INI_ALL	Überall: Skripten, Verzeichnisse und System.

Ein Wert von 6 bedeutet, dass die Einstellung sowohl auf der Verzeichnis- als auch auf der Systemebene verändert werden kann, denn 2 + 4 = 6. In der Praxis gibt es keine Variablen, die nur in PHP_INI_USER oder PHP_INI_PERDIR geändert werden können, und alle Variablen sind in PHP_INI_SYSTEM änderbar. Es kommen also nur die Werte 4, 6 und 7 vor.

Sie können auch Variablen auslesen, die zu einer bestimmten Erweiterung gehören, indem Sie den Erweiterungsnamen an ini_get_all() übergeben:

```
// Nur die für das Session-Modul spezifischen Variablen zurückgeben.
$session = ini_get_all('session');
```

Laut Konvention werden die Variablen zu einer Erweiterung mit dem Namen der Erweiterung und einem Punkt als Präfix versehen. Dementsprechend beginnen all Session-Variablen mit session., so wie alle Java-Variablen mit java. beginnen.

Da ini_get() den aktuellen Wert zu einer Konfigurationsdirektive liefert, verwenden Sie get_cfg_var() zur Prüfung des Originalwerts aus der *php.ini*-Datei:

```
$original = get_cfg_var('sendmail_from'); // Wurde Adresse geändert?
```

Der von get_cfg_var() zurückgegebene Wert ist der gleiche, der in dem global_value-Element des von ini_get_all() gelieferten Arrays zu finden ist.

Siehe auch

Rezept 10.18 zum Setzen von Konfigurationsvariablen; die Dokumentationen zu ini_get() unter *http://www.php.net/ini-get*, ini_get_all() unter *http://www.php.net/ini-get-all* und get_cfg_var() unter *http://www.php.net/get-cfg-var*; eine vollständige Liste der Konfigurationsvariablen einschließlich der Information, wo sie verändert werden können, unter *http://www.php.net/manual/function.ini-set.php*.

10.18 Konfigurationsvariablen setzen

Problem

Sie möchten den Wert einer PHP-Konfigurationseinstellung ändern.

Lösung

Verwenden Sie `ini_set()`:

```
// Include-Pfad um ein Verzeichnis erweitern.
ini_set('include_path', ini_get('include_path') . ':/home/fezzik/php');
```

Diskussion

Die Konfigurationsvariablen werden durch `ini_set()` nicht permanent verändert. Der neue Wert bleibt nur für die Dauer der Anfrage gültig, in der `ini_set()` aufgerufen worden ist. Um eine persistente Änderung durchzuführen, müssen Sie den in der *php.ini*-Datei gespeicherten Wert ändern.

Die Änderung bestimmter Variablen hat keine Auswirkung, dies ist zum Beispiel bei `asp_tags` und `register_globals` der Fall. Zu dem Zeitpunkt, an dem Sie `ini_set()` zum Ändern der Einstellung aufrufen, ist es für die Änderung des durch sie beeinflussten Verhaltens zu spät. Wenn eine Variable nicht geändert werden kann, gibt `ini_set()` den Wert `false` zurück.

In manchen Seiten ist das Ändern von Konfigurationsvariablen durchaus sinnvoll: Wenn Sie beispielsweise ein Skript von der Befehlszeile aus starten, sollten Sie `html_errors` auf `off` setzen.

Um eine Variable auf ihre ursprüngliche Einstellung zurückzusetzen, verwenden Sie `ini_restore()`:

```
ini_restore('sendmail_from'); // zurück zum Vorgabewert
```

Siehe auch

Rezept 10.17 zum Auslesen der Inhalte von Konfigurationsvariablen; die Dokumentationen zu `ini_set()` unter *http://www.php.net/ini-set* und `ini_restore()` unter *http://www.php.net/ini-restore*.

10.19 Innerhalb von Apache kommunizieren

Problem

Sie möchten von PHP aus mit anderen Teilen des Apache-Anfragevorgangs kommunizieren. Dazu gehört auch, Variablen im *access_log* zu setzen.

Lösung

Verwenden Sie apache_note():

```
// Wert lesen
$session = apache_note('session');

// Wert setzen
apache_note('session', $session);
```

Diskussion

Wenn der Apache-Server eine Client-Anfrage verarbeitet, durchläuft er diverse Schritte, wobei PHP nur ein Teil in dieser gesamten Kette darstellt. Apache formt daneben auch URLs um, authentifiziert Benutzer, protokolliert Anfragen und mehr. Während der Abarbeitung einer Anfrage hat jeder Handler Zugriff auf eine Reihe von Schlüssel/Wert-Paaren, die als *Notes Table* (Notiztabelle) bezeichnet wird. Die Funktion apache_note() ermöglicht den Zugriff auf die Notes Table, um von einem anderen Handler zuvor gesetzte Informationen auszulesen oder Informationen für einen späteren Handler zu hinterlassen.

Wenn Sie beispielsweise mithilfe des Session-Moduls Benutzer verfolgen und Variablen über Anfragen hinweg bewahren möchten, können Sie dies mit Ihrer Logfile-Analyse verbinden, sodass Sie gleichzeitig auch die durchschnittliche Anzahl von Page-Views pro Benutzer ermitteln können. Verwenden Sie apache_note() in Kombination mit dem Protokoll-Modul, um die Session-ID bei jeder Anfrage in das *access_log* zu schreiben:

```
// Die Session-ID ermitteln und in die Apache Notes Table eintragen.
apache_note('session_id', session_id());
```

Modifizieren Sie dann Ihre *httpd.conf*-Datei, indem Sie die folgende Zeichenkette in Ihr LogFormat einfügen:

```
%{session_id}n
```

Aufgrund des n am Ende verwendet Apache eine Variable, die von einem anderen Modul in seiner Notes Table abgelegt worden ist.

Sobald der PHP-Build mit der Konfigurationsoption --enable-memory-limit durchgeführt worden ist, speichert PHP den höchsten Speicherbedarf jedes Requests in einer Note mit dem Namen mod_php_memory_usage. Diese Informationen über die Speichernutzung können Sie folgendermaßen in ein LogFormat einfügen:

```
%{mod_php_memory_usage}n
```

Siehe auch

Die Dokumentation zu apache_note() unter *http://www.php.net/apache-note*; Informationen zur Protokollierung in Apache unter *http://httpd.apache.org/docs/mod/mod_log_config.html*.

10.20 Code-Profile generieren

Problem

Sie haben einen Code-Block, für den Sie ein Profil erstellen möchten. Daran können Sie erkennen, wie viel Zeit die Ausführung einzelner Anweisungen benötigt.

Lösung

Verwenden Sie das PEAR-Benchmark-Modul:

```
require 'Benchmark/Timer.php';

$timer =& new Benchmark_Timer(true);

$timer->start();
// Hier Setup-Code einfügen.
$timer->setMarker('setup');
// Hier wird etwas Code ausgeführt.
$timer->setMarker('middle');
// Hier folgt weiterer Code.
$timer->setmarker('done');
// Und hier das letzte Stück des Codes.
$timer->stop();

$timer->display();
```

Diskussion

Durch den Aufruf von `setMarker()` wird die Zeit aufgezeichnet. Die Methode `display()` gibt eine Liste von Markern mit der Zeit aus, zu der sie gesetzt worden sind, und der seit dem letzten Marker verstrichenen Zeit:

```
------------------------------------------------------------
marker    time index            ex time               perct
------------------------------------------------------------
Start     1029433375.42507400   -                      0.00%
------------------------------------------------------------
setup     1029433375.42554800   0.00047397613525391    29.77%
------------------------------------------------------------
middle    1029433375.42568700   0.00013899803161621     8.73%
------------------------------------------------------------
done      1029433375.42582000   0.00013303756713867     8.36%
------------------------------------------------------------
Stop      1029433375.42666600   0.00084602832794189    53.14%
------------------------------------------------------------
total     -                     0.0015920400619507    100.00%
------------------------------------------------------------
```

Das Benchmark-Modul enthält außerdem die Klasse `Benchmark_Iterate`, mit der die Zeit für die mehrfache Ausführung einer einzelnen Funktion gestoppt werden kann:

```
require 'Benchmark/Iterate.php';

$timer =& new Benchmark_Iterate;

// Eine einfache zu messende Funktion
function use_preg($ar) {
    for ($i = 0, $j = count($ar); $i < $j; $i++) {
        if (preg_match('/Gouda/',$ar[$i])) {
            // Es ist Gouda.
        }
    }
}

// Eine weitere zu messendende Funktion
function use_equals($ar) {
    for ($i = 0, $j = count($ar); $i < $j; $i++) {
        if ('Gouda' == $ar[$i]) {
            // Es ist Gouda.
        }
    }
}

// use_preg() 1000-mal laufen lassen
$timer->run(1000,'use_preg',
                array('Gouda','Schweizer','Gruyere','Münster','Parmesan'));
$results = $timer->get();
print "Mittlere Ausführungszeit für use_preg(): $results[mean]\n";

// use_equals() 1000-mal laufen lassen
$timer->run(1000,'use_equals',
                array('Gouda','Schweizer','Gruyere','Münster','Parmesan'));
$results = $timer->get();
print "Mittlere Ausführungszeit für use_equals(): $results[mean]\n";
```

Die Methode `Benchmark_Iterate::get()` liefert ein assoziatives Array. Das Element `mean` dieses Arrays enthält die mittlere Ausführungszeit aller Durchgänge dieser Funktion. Das Element `iterations` enthält die Anzahl der Durchgänge. Die Ausführungszeit für jeden einzelnen Durchgang der Funktion wird in einem Array-Element mit einem Integer-Schlüssel gespeichert. Die Zeit für den ersten Durchgang befindet sich in `$results[1]` und die des 37. Durchgangs in `$results[37]`.

Um automatisch die nach jeder Zeile PHP-Code vergangene Ausführungszeit zu messen, verwenden Sie das `declare`-Konstrukt und die Direktive `ticks`:

```
function profile($display = false) {
    static $times;

    switch ($display) {
    case false:
        // Aktuelle Zeit in Liste der aufgezeichneten Zeiten eintragen.
        $times[] = microtime();
```

```
            break;
        case true:
            // Abgelaufene Zeit in Mikrosekunden zurückgeben.
            $start = array_shift($times);

            $start_mt = explode(' ', $start);
            $start_total = doubleval($start_mt[0]) + $start_mt[1];

            foreach ($times as $stop) {
                $stop_mt = explode(' ', $stop);
                $stop_total = doubleval($stop_mt[0]) + $stop_mt[1];
                $elapsed[] = $stop_total - $start_total;
            }

            unset($times);
            return $elapsed;
            break;
        }
    }

    // Tick-Handler registrieren.
    register_tick_function('profile');

    // Startzeit abnehmen.
    profile();

    // Code ausführen und dabei die Zeit für jede
    // ausgeführte Anweisung aufzeichnen.
    declare (ticks = 1) {
        foreach ($_SERVER['argv'] as $arg) {
            print strlen($arg);
        }
    }

    // Gemessene Zeiten ausgeben.
    $i = 0;
    foreach (profile(true) as $time) {
        $i++;
        print "Zeile $i: $time\n";
    }
```

Mithilfe der Direktive `ticks` können Sie eine Funktion für einen Code-Block wiederholt ausführen. Die `ticks` zugewiesene Zahl gibt an, wie viele Anweisungen weitergelaufen werden soll, bevor die mit `register_tick_function()` angemeldeten Funktionen ausgeführt werden.

Im vorstehenden Beispiel wird eine einzige Funktion registriert und die `profile()`-Funktion für jede einzelne Anweisung innerhalb des `declare`-Blocks ausgeführt. Wenn es in `$_SERVER['argv']` zwei Elemente gibt, wird `profile()` viermal ausgeführt: einmal für jeden Durchgang der `foreach`-Schleife und einmal für jede Ausführung der Zeile `print strlen($arg)`.

Sie können das Ganze auch so einrichten, dass jeweils nach drei Anweisungen zwei Funktionen aufgerufen werden:

```
register_tick_function('profile');
register_tick_function('backup');

declare (ticks = 3) {
    // Code...
}
```

Zusätzlich können Sie den registrierten Funktionen auch Parameter übergeben, und es kann sich um Objekt-Methoden anstelle von normalen Funktionen handeln:

```
// "parameter" an profile() übergeben.
register_tick_function('profile', 'parameter');

// $car->drive() aufrufen.
$car = new Vehicle;
register_tick_function(array($car, 'drive'));
```

Wenn Sie eine Objekt-Methode ausführen möchten, übergeben Sie das Objekt und den Namen der Methode, beides gekapselt in einem Array. Dann weiß `register_tick_function()`, dass Sie ein Objekt meinen und keine Funktion.

Rufen Sie `unregister_tick_function()`, um eine Funktion aus der Liste der Tick-Funktionen zu entfernen:

```
unregister_tick_function('profile');
```

Siehe auch

http://pear.php.net/package-info.php?package=Benchmark zu Informationen über die PEAR-Benchmark-Klasse; die Dokumentationen zu `register_tick_function()` unter *http://www.php.net/register-tick-function*, `unregister_tick_function()` unter *http://www.php.net/unregister-tick-function* und `declare` unter *http://www.php.net/declare*.

10.21 Geänderte Dateien herunterladen und unveränderte vom Browser cachen lassen

Problem

Sie haben eine Webseite mit Dokumenten, die Sie durch ein Skript herunterladen lassen wollen. Die Dokumente verändern sich oft, aber nicht so häufig, dass der Browser sie jedes Mal neu herunterladen sollte. Zum Beispiel haben Sie eine Seite mit Bildern, die gelegentlich ausgetauscht werden müssen, die Sie aber nur eingeloggten Benutzern zugänglich machen wollen.

Lösung

Verwenden Sie die Funktion `pc_sendFile()`, um die gewünschte Datei vor dem Download auf Aktualität zu prüfen:

```
function pc_sendFile($fileName) {
    // HTTP-Header lesen.
    $headers = getallheaders();
    // Wenn die Kopie im Cache noch aktuell ist, keine neue Datei senden.
    if ($headers["If-Modified-Since"] != "") {
        $ifModifiedSince = strtotime($headers["If-Modified-Since"]);
        if ($ifModifiedSince >= filemtime($fileName)) { // Kopie ist aktuell.
            header("HTTP/1.1 304 Not Modified"); // Browser darauf hinweisen.
            exit();
        }
    }
    // Aktuelle Datei öffnen ...
    $fileHandle = @fopen($fileName,"rb");
    if (!$fileHandle) {
        die("Datei nicht gefunden!");
    }
    // ... und auslesen.
    $fileContent = fread($fileHandle, filesize($fileName));
    // Header an den Browser ausgeben.
    header("Content-Type: ".mime_content_type($fileName)); // Dateityp
    // Date-Header: momentanes Datum und Zeit am Server in GMT
    header("Date: ".gmdate("D, d M Y H:i:s") . " GMT");
    // Last-Modified-Header auf GMT-Modifikationsdatum der Datei setzen.
    header("Last-Modified: ".gmdate("D, d M Y H:i:s",filemtime($fileName)). " GMT");
    header("Cache-Control: must-revalidate"); // Browser soll verifizieren.
    echo $fileContent; // Datei an Browser ausgeben.
}
```

Diskussion

Wenn Sie ein normales Dokument, z.B. eine Bilddatei, in Ihrer Website haben und diese direkt durch Aufruf der URL mit dem Dateinamen herunterladen, teilt der Server dem Browser mit, wann die Datei das letzte Mal geändert wurde. Sollte der Browser die Datei zwischenspeichern (*cachen*) und sie später noch einmal benötigen, kann er das Dokument beim Server durch Senden des `If-Modified-Since`-Headers bedingt anfordern: Hat sich die Datei seit dem ersten Herunterladen nicht geändert, antwortet der Server mit dem Antwortcode `304 - Not Modified`. Damit weiß der Browser, dass er die Kopie des Dokuments im Cache weiter verwenden kann. Hat sich die Datei auf dem Server seither geändert, schickt der Server stattdessen die neue Version des Dokuments zurück.

Bei großen Bilddateien beispielsweise kann das zu sehr signifikanten Einsparungen in der Download-Zeit führen: Sie können in einem Bilderbuch blättern, bekommen jeweils die aktuellsten Versionen der Bilder angezeigt, müssen aber nicht doppelt herunterladen, um sich ein Bild zweimal ansehen zu können.

Wenn Sie Ihr Dokument aber nicht direkt herunterladen, sondern dynamisch durch ein Skript öffnen (oder erstellen) und ausgeben lassen, haben Sie ein Problem: Der Server kann jetzt nicht mehr sehen, ob Ihr Dokument aktuell ist oder nicht. Vorsichtshalber geht der Server daher davon aus, dass Ihr Dokument immer taufrisch ist und gerade eben erst geändert wurde. Wenn Sie mit Ihrem Skript Bilder aus einem ansonsten versteckten Verzeichnis ausgeben, müssen dieselben Bilder immer wieder neu heruntergeladen werden, egal ob sie sich geändert haben oder nicht.

Hier bietet `pc_sendFile()` eine Lösung: Die Funktion wird mit dem Dateinamen und dem MIME-Dateityp der angeforderten Datei aufgerufen, z.B. so:

```
$fileName = get_filename_by_id($_GET["ID"]); // Lookup-Funktion zum Auffinden des
Dateinamens.
$mimeType = get_mime_content_type($fileName); // MIME-Type der Datei finden.
pc_sendFile($fileName, $mimeType);
```

Den Namen Ihrer Datei sollten Sie *immer* über einen Lookup ermitteln und *nie* durch vom Browser vorgegebene Variablen setzen lassen – das würde alle Dateien auf Ihrem Server der Öffentlichkeit zugänglich machen. Den MIME-Typ Ihrer Datei können Sie auch über die Funktion `mime_content_type()` herausfinden, sofern Sie Ihre PHP-Installation richtig konfiguriert haben.[2] Wenn Sie schon im Voraus wissen, welchen Typ Ihre Datei hat (z.B. `image/jpeg` für JPEG-Fotos), können Sie sich die Prozedur natürlich auch sparen.

`pc_sendFile()` sieht zunächst einmal in den Headern des HTTP-Requests nach, ob der Browser einen `If-Modified-Since`-Header mitgeschickt hat. Wenn ja, vergleicht die Funktion das Datum des Headers mit dem Modifikationsdatum der gewünschten Datei. Ist die Datei neuer als das angegebene Datum, wird sie ausgeben.

Dabei werden vier Header gesetzt: `Content-Type`, um dem Browser anzuzeigen, welche Art Dokument er zu erwarten hat, `Last-Modified` für den letzten Modifikationszeitpunkt der Datei, `Date` als aktuelle Referenzzeit am Server und schließlich `Cache-Control: must-revalidate`, damit der Browser weiß, dass er bei wiederholtem Herunterladen der Datei einen `If-Modified-Since`-Header mitschicken soll.

Ist die Datei noch aktuell, reicht ein `HTTP/1.1 304 - Not Modified` Header aus.

Sie können die Funktion auch so abändern, dass sie statt einer echten Datei z.B. einen Datenbank-Eintrag als HTML ausgibt. Ebenso können Sie die Modifikationszeit selbst bestimmen, z.B. könnten Sie entscheiden, einem Browser nur dann ein neues Bild zukommen zu lassen, wenn das alte mindestens einen Tag alt ist.

2 Unter Windows müssen Sie dazu in *php.ini* `extension=php_mime_magic.dll` aktivieren und einen Parameter `mime_magic.magicfile = "c:/Pfad/zu/magic.mime"` definieren. Die Datei *magic.mime* finden Sie im Ordner *extras/* Ihrer PHP-Installation.

Siehe auch

Die Dokumentation zu `header()` unter *http://www.php.net/manual/en/function.header.php*; die Dokumentation zu HTTP-Response-Codes und Headern unter *http://www.faqs.org/rfcs/rfc2616*.

10.22 Programm: (De-)Aktivator für Website-Konten

Wenn sich Benutzer bei Ihrer Website anmelden, ist es gut zu wissen, ob Sie von diesen eine korrekte E-Mail-Adresse erhalten haben. Um die angegebene Adresse zu überprüfen, senden Sie eine E-Mail an die vom Benutzer bei der Anmeldung angegebene Adresse. Wenn der Benutzer dann eine bestimmte, in der E-Mail angegebene URL nicht innerhalb von ein paar Tagen aufsucht, deaktivieren Sie das Konto.

Das System hat drei Teile. Der erste besteht aus dem in Beispiel 10-8 dargestellten Programm *notify-user.php*, das E-Mails an neue Benutzer sendet und sie bittet, eine URL zur Verifikation aufzusuchen. Der zweite Teil ist die Seite *verify-user.php* in Beispiel 10-9; sie ist die Verifizierungs-URL und kennzeichnet Benutzer als gültig. Der dritte Teil ist das Programm *delete-user.php*, das Konten von Benutzern deaktiviert, die nicht innerhalb einer bestimmten Zeit die Verifizierungs-URL aufgesucht haben. Dieses Programm ist in Beispiel 10-10 zu sehen.

Mit der folgenden SQL-Anweisung wird die Tabelle angelegt, in der die Benutzerinformationen gespeichert sind:

```
CREATE TABLE users (
 email VARCHAR(255) NOT NULL,
 created_on DATETIME NOT NULL,
 verify_string VARCHAR(16) NOT NULL,
 verified TINYINT UNSIGNED
);
```

Sie werden vermutlich mehr als diese Informationen über Ihre Benutzer speichern wollen, aber zum Verifizieren werden nur diese wenigen Daten benötigt. Wenn Sie ein Benutzerkonto anlegen, speichern Sie die Informationen in der Tabelle `users` und senden dem Benutzer eine E-Mail, die ihm mitteilt, wie das Konto verifiziert wird. Der in Beispiel 10-8 dargestellte Code geht davon aus, dass die E-Mail-Adresse des Benutzers in der Variablen `$email` gespeichert ist.

Beispiel 10-8: notify-user.php

```
// Verifizierungs-String generieren.
$verify_string = '';
for ($i = 0; $i < 16; $i++) {
    $verify_string .= chr(mt_rand(32,126));
}

// Benutzer in Datenbank eintragen.
```

Beispiel 10-8: notify-user.php (Fortsetzung)

```
if (! mysql_query("INSERT INTO users (email,created_on,verify_string,verified)
    VALUES ('".addslashes($email)."',NOW(),'".addslashes($verify_string)."',0)")) {
    error_log("Kann Benutzer nicht einfügen: ".mysql_error());
    exit;
}

$verify_string = urlencode($verify_string);
$safe_email = urlencode($email);

$verify_url = "http://www.example.com/verify.php";

$mail_body=<<<_MAIL_
An $email:

Bitte klicken Sie auf den folgenden Link,
um Ihre Anmeldung zu bestaetigen:

$verify_url?email=$safe_email&verify_string=$verify_string

Wenn Sie Ihr Konto nicht innerhalb von sieben Tagen best&auml;tigen,
wird es gel&ouml;scht.
_MAIL_;

mail($email,"Benutzer-Bestaetigung",$mail_body);
```

Zur Verifizierungsseite kommen die Benutzer, wenn sie dem Link in der E-Mail folgen. Sie aktualisiert die Benutzertabelle, sofern die korrekten Informationen übergeben werden, wie Beispiel 10-9 zeigt.

Beispiel 10-9: verify-user.php

```
$safe_email = addslashes($_REQUEST['email']);
$safe_verify_string = addslashes($_REQUEST['verify_string']);

if ($r = mysql_query("UPDATE users SET verified = 1 WHERE email
    LIKE '$safe_email' AND
    verify_string = '$safe_verify_string' AND verified = 0")) {
    if (mysql_affected_rows() == 1) {
        print "Vielen Dank, Ihr Konto ist best&auml;tigt.";
    } else {
        print "Es tut uns leid, aber wir konnten Sie wegen eines ";
        print "Datenbankfehlers nicht best&auml;tigen.";
    }
} else {
    print "Bitte versuchen Sie es sp&auml;ter noch einmal.";
}
```

Der Verifizierungsstatus des Benutzers wird nur aktualisiert, wenn die E-Mail-Adresse und der übergebene Verifizierungs-String mit einer Zeile in der Datenbank übereinstimmen, die noch nicht verifiziert worden ist. Der letzte Schritt besteht in einem kur-

zen Programm, das nicht verifizierte Benutzer nach einer angemessenen Zeit löscht, in Beispiel 10-10 zu sehen.

Beispiel 10-10: delete-user.php

```
$window = 7; // Tage

if ($r = mysql_query("DELETE FROM users WHERE verified = 0 AND
    created_on < DATE_SUB(NOW(),INTERVAL $window DAY)")) {
    if ($deleted_users = mysql_affected_rows()) {
        print "$deleted_users Benutzer deaktiviert.\n";
    }
} else {
    print "Kann keine Benutzer l&ouml;schen: ".mysql_error();
}
```

Lassen Sie dieses Programm einmal am Tag laufen, um die `users`-Tabelle von nicht-verifizierten Benutzern zu säubern. Wenn Sie den Zeitraum ändern möchten, in dem die Benutzer sich verifizieren können, passen Sie den Wert von `$window` an und ändern den Text der an die Benutzer gesandten E-Mail so an, dass er mit dem neuen Wert übereinstimmt.

10.23 Programm: Störungsprüfer

Wegen der Geschwindigkeit des gemeinsam benutzten Arbeitsspeichers (Shared-Memory) bietet dieser eine ideale Möglichkeit zur Ablage von Daten, auf die mehrere Webserver-Prozesse häufig zugreifen müssen und für die eine Datei oder eine Datenbank zu langsam wäre. Beispiel 10-11 zeigt die Klasse `pc_Web_Abuse_Check`, die mithilfe von gemeinsamem Speicher Zugriffe auf Webseiten verfolgt, um Benutzer zu isolieren, die Ihre Site missbrauchen, indem sie diese mit Anfragen bombardieren.

Beispiel 10-11: pc_Web_Abuse_Check class

```
class pc_Web_Abuse_Check {
  var $sem_key;
  var $shm_key;
  var $shm_size;
  var $recalc_seconds;
  var $pageview_threshold;
  var $sem;
  var $shm;
  var $data;
  var $exclude;
  var $block_message;

  function pc_Web_Abuse_Check() {
    $this->sem_key = 5000;
    $this->shm_key = 5001;
    $this->shm_size = 16000;
```

Beispiel 10-11: pc_Web_Abuse_Check class (Fortsetzung)

```
    $this->recalc_seconds = 60;
    $this->pageview_threshold = 30;

    $this->exclude['/ok-to-bombard.html'] = 1;
    $this->block_message =<<<END
<html>
<head><title>403 Forbidden</title></head>
<body>
<h1>Verboten</h1>
Der Aufruf dieser Seiten ist f&uuml;r Sie gesperrt aufgrund
wiederholter missbr&auml;uchlicher Aktivit&auml;ten unter Ihrem
Konto. Wenn Sie annehmen, dass dies ein Fehler ist, wenden Sie sich bitte an
<a href="mailto:webmaster@example.com?subject=Site+Abuse">webmaster@example.com</a>.
</body>
</html>
END;
   }

  function get_lock() {
    $this->sem = sem_get($this->sem_key,1,0600);
    if (sem_acquire($this->sem)) {
      $this->shm = shm_attach($this->shm_key,$this->shm_size,0600);
      $this->data = shm_get_var($this->shm,'data');
    } else {
      error_log("Kann Semaphore nicht erhalten: $this->sem_key");
    }
  }

  function release_lock() {
    if (isset($this->data)) {
      shm_put_var($this->shm,'data',$this->data);
    }
    shm_detach($this->shm);
    sem_release($this->sem);
  }

  function check_abuse($user) {
    $this->get_lock();
    if ($this->data['abusive_users'][$user]) {
      // Wenn Benutzer in der Liste ist, Semaphore und Speicher freigeben.
      $this->release_lock();
      // Sperr-Seite ausgeben.
      header('HTTP/1.0 403 Forbidden');
      print $this->block_message;
      return true;
    } else {
     // Festhalten, dass dieser Benutzer eben eine Seite betrachtet.
     $now = time();
     if (! $this->exclude[$_SERVER['PHP_SELF']]) {
       $this->data['user_traffic'][$user]++;
     }
```

Beispiel 10-11: pc_Web_Abuse_Check class (Fortsetzung)

```
    // Liste (gelegentlich) aktualisieren und Störer hinzufügen.
    if (! $this->data['traffic_start']) {
      $this->data['traffic_start'] = $now;
    } else {
      if (($now - $this->data['traffic_start']) > $this->recalc_seconds) {
        while (list($k,$v) = each($this->data['user_traffic'])) {
          if ($v > $this->pageview_threshold) {
            $this->data['abusive_users'][$k] = $v;
            // Protokollieren, dass Benutzer in Störerliste eingetragen wurde.
            error_log("Abuse: [$k] (from ".$_SERVER['REMOTE_ADDR'].')');
          }
        }
        $this->data['traffic_start'] = $now;
        $this->data['user_traffic'] = array();
      }
    }
    $this->release_lock();
   }
   return false;
  }
}
```

Sie verwenden diese Klasse, indem Sie deren Methode `check_abuse()` am Anfang der Seite aufrufen und dabei den Benutzernamen eines angemeldeten Anwenders übergeben:

```
// get_logged_in_user_name() ist eine Funktion, die feststellt,
// ob ein Benutzer angemeldet ist.
if ($user = get_logged_in_user_name()) {
    $abuse = new pc_Web_Abuse_Check();
    if ($abuse->check_abuse($user)) {
        exit;
    }
}
```

Die Methode `check_abuse()` sichert den exklusiven Zugriff auf das Shared-Memory-Segment, in dem die Informationen über Benutzer und Seitenabrufe gespeichert sind, mit der Methode `get_lock()` ab. Wenn der aktuelle Benutzer sich bereits in der Liste der Störer befindet, gibt sie ihr Sperre für das Shared-Memory frei, gibt eine Fehlerseite für den Benutzer aus und liefert `true` zurück. Die Fehlerseite wird im Konstruktor der Klasse definiert.

Wenn der Benutzer nicht in der Liste steht und sich die aktuelle Seite (gespeichert in `$_SERVER['PHP_SELF']`) nicht in einer Liste von Seiten befindet, die von der Störungsprüfung ausgenommen werden sollen, wird der Zähler für die vom Benutzer betrachteten Seiten inkrementiert. Die Liste der auszuschließenden Seite wird ebenfalls im Konstruktor definiert. Rufen Sie `check_abuse()` am Anfang jeder Seite auf und tragen Sie alle Seiten, die als nicht störungsgefährdet gelten, in das `$exclude`-Array ein; damit stellen Sie sicher, dass ein Störer die Fehlerseite auch dann zu sehen bekommt, wenn er eine Seite

abruft, die beim Ermitteln der Störung nicht mitgezählt worden ist. Somit verhält sich Ihre Site konsistenter.

Der nächste Abschnitt von `check_abuse()` ist dafür zuständig, Benutzer in die Störerliste einzutragen. Wenn seit dem letzen Mal, als Benutzer in die Störerliste eingetragen wurden, mehr als `$this->recalc_seconds` vergangen sind, sieht die Methode bei jedem Benutzer im Zähler für abgerufene Seiten nach. Wenn ein Zähler höher als `$this->pageview_threshold` ist, trägt die Methode diesen Benutzer in die Störerliste ein und schreibt eine Mitteilung in das Fehlerprotokoll. Der Code zum Setzen von `$this->data['traffic_start']` wird, wenn dies nicht bereits geschehen ist, nur beim allerersten Aufruf von `check_abuse()` ausgeführt. Immer wenn ein Störer auftaucht, setzt `check_abuse()` den Zähler für Benutzer und Seitenaufrufe zurück und startet ein neues Intervall, bis die Störerliste zum nächsten Mal aktualisiert wird. Nach der Freigabe der Sperre auf dem Shared-Memory-Segment gibt die Methode `false` zurück.

Alle Informationen, die `check_abuse()` für seine Berechnungen benötigt, werden in einem einzigen assoziativen Array namens `$data` gespeichert, darunter die Störerliste, die Seitenabrufzähler für die Benutzer und den letzten Zeitpunkt, zu dem Störer ermittelt wurden. Dadurch ist das Lesen und Schreiben von Werten im Shared-Memory einfacher, als wenn die Informationen in getrennten Variablen untergebracht wären, da nur ein Aufruf von `shm_get_var()` und `shm_put_var()` erforderlich ist.

Die Klasse `pc_Web_Abuse_Check` schließt Störer aus, bietet aber keine Reporting-Fähigkeiten und keine Möglichkeit, bestimmte Benutzer zu der Liste hinzuzufügen oder aus ihr zu entfernen. Beispiel 10-12 zeigt das Programm *abuse-manage.php*, mit dem Sie die Daten zu den Störern verwalten können.

Beispiel 10-12: abuse-manage.php

```
// Die Klasse pc_Web_Abuse_Check ist in abuse-check.php definiert.
require 'abuse-check.php';

$abuse = new pc_Web_Abuse_Check();
$now = time();

// Befehle ausführen, wenn angegeben.
$abuse->get_lock();
switch ($_REQUEST['cmd']) {
    case 'clear':
      $abuse->data['traffic_start'] = 0;
      $abuse->data['abusive_users'] = array();
      $abuse->data['user_traffic'] = array();
      break;
    case 'add':
      $abuse->data['abusive_users'][$_REQUEST['user']] = 'web @ '.strftime('%c',$now);
      break;
    case 'remove':
      $abuse->data['abusive_users'][$_REQUEST['user']] = 0;
      break;
}
```

Beispiel 10-12: abuse-manage.php (Fortsetzung)

```
$abuse->release_lock();

// Die relevanten Daten sind jetzt in $abuse->data.

print 'Es ist jetzt <b>'.strftime('%c',$now).'</b><br>';
print 'Beginn des aktuellen Intervalls: <b>'.strftime('%c',$abuse->data['traffic_start']);
print '</b> (vor '.($now - $abuse->data['traffic_start']).' Sekunden).<p>';

print 'Verkehr im aktuellen Intervall:<br>';
if (count($abuse->data['user_traffic'])) {
  print '<table border="1"><tr><th>Benutzer</th><th>Seiten</th></tr>';
  while (list($user,$pages) = each($abuse->data['user_traffic'])) {
    print "<tr><td>$user</td><td>$pages</td></tr>";
  }
  print "</table>";
} else {
  print "<i>Kein Verkehr.</i>";
}
print '<p>St&ouml;rer:';

if ($abuse->data['abusive_users']) {
  print '<table border="1"><tr><th>Benutzer</th><th>Seiten</th></tr>';
  while (list($user,$pages) = each($abuse->data['abusive_users'])) {
    if (0 === $pages) {
      $pages = 'Gel&ouml;scht';
      $remove_command = '';
    } else {
      $remove_command =
         "<a href=\"$_SERVER[PHP_SELF]?cmd=remove&user=".urlencode($user)."\">
             l&ouml;schen</a>";
    }
    print "<tr><td>$user</td><td>$pages</td><td>$remove_command</td></tr>";
  }
  print '</table>';
} else {
  print "<i>Keine St&ouml;rer.</i>";
}

print<<<END
<form method="post" action="$_SERVER[PHP_SELF]">
<input type="hidden" name="cmd" value="add">
Folgenden Benutzer in die Liste der St&ouml;rer eintragen:
<input type="text" name="user" value="">
<br>
<input type="submit" value="Add User">
</form>
<hr>
<form method="post" action="$_SERVER[PHP_SELF]">
<input type="hidden" name="cmd" value="clear">
<input type="submit" value="Liste der St&ouml;rer leeren">
END;
```

Beispiel 10-12 gibt Informationen über die aktuellen Seitenabrufzähler der Benutzer und die aktuelle Liste der Störer aus, wie sie in Abbildung 10-1 zu sehen ist. Sie ermöglicht auch, bestimmte Benutzer in die Liste einzutragen oder aus ihr zu löschen sowie den Inhalt der gesamten Liste zu löschen.

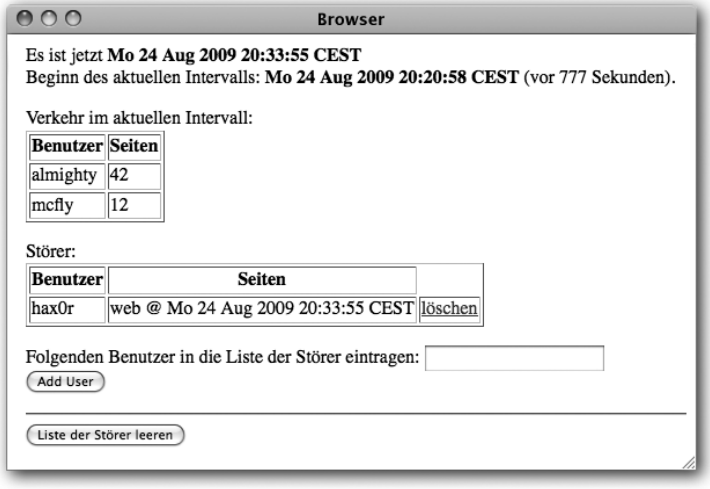


Abbildung 10-1: Seitenabrufzähler der Benutzer und Liste der Störer

Wenn das Programm Namen aus der Liste der Störer löscht, tut es nicht dies:

```
unset($abuse->data['abusive_users'][$_REQUEST['user']])
```

sondern setzt das Folgende auf 0:

```
$abuse->data['abusive_users'][$_REQUEST['user']]
```

Auch dadurch wird `check_abuse()` veranlasst, `false` zurückzugeben; trotzdem können in der Seite Benutzer explizit aufgeführt werden, die sich in der Störerliste befanden, aber aus ihr entfernt worden sind. Dies ist hilfreich in dem Fall, dass ein Benutzer, der aus der Liste entfernt wurde, erneut Probleme verursacht.

Beim Eintragen eines Namens in die Liste der Störer zeichnet das Skript nicht den Seitenabrufzähler auf, sondern die Zeit, zu der die Eintragung erfolgt ist. Dadurch kann leichter nachvollzogen werden, von wem und warum ein Benutzer manuell in die Liste eingetragen wurde.

Wenn Sie `pc_Web_Abuse_Check` und diese Wartungsseite auf Ihrem Server in Betrieb nehmen, sollten Sie sicherstellen, dass die Wartungsseite durch ein Passwort geschützt oder auf andere Weise für die allgemeine Öffentlichkeit unzugänglich gemacht wird. Es ist klar, dass dieses Programm nichts nützt, wenn sich die Störer selbst aus der Störerliste austragen können.

KAPITEL 11

Formulare

11.0 Einführung

Das Geniale an PHP ist die nahtlose Integration der Formularvariablen in die Programme. Dadurch wird Webprogrammierung elegant und einfach, und zwar vom Webformular über den PHP-Code bis zur HTML-Ausgabe.

HTTP kennt keinen eingebauten Mechanismus, mit dem Sie Informationen aus einer Seite so sichern können, dass Sie von anderen Seiten aus darauf zugreifen können. Das liegt daran, dass HTTP ein zustandsloses Protokoll ist. Die Rezepte 11.1, 11.3, 11.4 und 11.5 zeigen Ihnen verschiedene Möglichkeiten, wie Sie dieses fundamentale Problem umgehen können, um herauszufinden, welcher Benutzer Ihrer Website welche Anfragen tätigt.

Die Verarbeitung der vom Benutzer kommenden Daten ist das andere Hauptthema dieses Kapitels. Sie können den von einem Browser gelieferten Daten niemals vertrauen, daher ist es zwingend notwendig, stets alle Felder zu überprüfen – sogar verborgene Formularelemente. Die Validierung kann viele Formen haben, zum Beispiel behandelt Rezept 11.2, wie Sie sicherstellen, dass die Daten mit bestimmten Kriterien übereinstimmen, und Rezept 11.8, wie Sie Daten in HTML-Entities umwandeln, mit denen Sie Benutzereingaben sicher darstellen können. Des Weiteren zeigt Ihnen Rezept 11.7, wie Sie die Sicherheit Ihres Webservers schützen können, und Rezept 11.6 behandelt den Umgang mit von Benutzern hochgeladenen Dateien.

Immer wenn PHP eine Seite verarbeitet, wird diese auf GET- und POST-Formularvariablen, hochgeladene Dateien, anwendbare Cookies sowie Webserver- und Umgebungsvariablen untersucht. Auf diese kann dann in den folgenden Arrays direkt zugegriffen werden: `$_GET`, `$_POST`, `$_FILES`, `$_COOKIE`, `$_SERVER` und `$_ENV`. Sie enthalten alle Variablen, die durch GET-Anfragen, POST-Anfragen, hochgeladene Dateien, Cookies, den Webserver bzw. die Umgebung gesetzt worden sind. Außerdem gibt es noch `$_REQUEST`, ein einziges riesiges Array, das die Werte der anderen sechs Arrays enthält.

Wenn PHP die Elemente in $_REQUEST einträgt und dabei auf Arrays stößt, die einen Schlüssel mit dem gleichen Namen haben, greift es auf die Konfigurationsdirektive variables_order zurück. Standardmäßig ist variables_order auf EGPCS gesetzt (oder GPCS, wenn Sie die Konfigurationsdatei *php.ini-recommended* verwenden). Dementsprechend kopiert PHP erst die Umgebungsvariablen nach $_REQUEST und fügt danach die GET-, POST-, Cookie- und Webserver-Variablen (in dieser Reihenfolge) hinzu. Da beispielsweise in der Standardreihenfolge C nach P kommt, überschreibt ein Cookie namens username eine POST-Variable, die username heißt.

Wenn Sie keinen Zugang zu den PHP-Konfigurationsdateien haben, können Sie eine Einstellung mit ini_get() prüfen:

```
print ini_get('variables_order');
EGPCS
```

Dies müssen Sie möglicherweise tun, weil Ihr ISP Ihnen nicht erlaubt, die Konfigurationseinstellungen anzusehen, oder weil Ihr Skript auf einem fremden Server läuft. Sie können die Einstellungen auch mit phpinfo() ansehen. Wenn Sie sich aber nicht darauf verlassen können, dass variables_order einen bestimmten Wert hat, sollten Sie direkt auf $_GET und $_POST anstelle von $_REQUEST zugreifen.

Die Arrays mit externen Variablen, wie etwa $_REQUEST, sind superglobal. Als solche müssen sie nicht innerhalb einer Funktion oder Klasse als global deklariert werden. Außerdem bedeutet dies, dass Sie diesen Variablen normalerweise nichts zuweisen sollten, da Sie sonst die in ihnen gespeicherten Daten überschreiben.

Vor PHP 4.1 existierten diese superglobalen Variablen noch nicht. Stattdessen gab es normale Arrays mit den Namen $HTTP_COOKIE_VARS, $HTTP_ENV_VARS, $HTTP_GET_VARS, $HTTP_POST_VARS, $HTTP_POST_FILES und $HTTP_SERVER_VARS. Diese Arrays sind aus Kompatibilitätsgründen immer noch verfügbar, aber mit den neuen Arrays kann man leichter arbeiten. Die alten Arrays werden nur gefüllt, wenn die Konfigurationsdirektive track_vars auf on geschaltet ist, allerdings ist seit PHP 4.0.3 diese Möglichkeit immer aktiviert.

Schließlich sind, sofern die Konfigurationsdirektive register_globals auf on steht, alle Variablen auch im globalen Namensraum verfügbar. Somit wird aus $_GET['password'] einfach $password. Das ist zwar bequem, führt aber zu schweren Sicherheitsproblemen, da ein böswilliger Benutzer auf einfache Weise Variablen von außen setzen und damit vermeintlich sichere interne Variablen überschreiben kann. Seit PHP 4.2 ist register_globals daher standardmäßig off.

Wenn man dies alles weiß, kann man mit dem folgenden grundlegenden Skript alles zusammenbringen. Das Formular bittet den Benutzer, seinen Vornamen einzugeben, und antwortet mit einem Willkommensgruß. Der HTML-Code für das Formular sieht folgendermaßen aus:

```
<form action="/hello.php" method="post">
Wie lautet Ihr Vorname?
<input type="text" name="vorname">
<input type="submit" value="Sag Hallo">
</form>
```

Der Name des Texteingabeelements im Formular ist `first_name`. Außerdem ist `post` die Methode (`method`) für das Formular. Dies bedeutet, dass nach dem Absenden des Formulars `$_POST['first_name']` die Eingabe des Benutzers enthält. (Natürlich kann es auch leer sein, wenn nichts hineingeschrieben wurde.)

Der Einfachheit halber wollen wir aber annehmen, dass der Inhalt der Variablen gültig ist. (Was unter »gültig« zu verstehen ist, ist eine Definitionsfrage und kann von verschiedenen Kriterien abhängen, zum Beispiel davon, dass der Wert nicht leer ist, keinen Einbruchsversuch in das System darstellt usw.) Somit wird hier die Fehlerprüfung außen vor gelassen, die zwar wichtig ist, bei diesem einfachen Beispiel aber eher stört. Hier ist als das einfache Skript *hello.php* zur Verarbeitung des Formulars:

```
echo 'Hallo ' . $_POST['vorname'] . '!';
```

Wenn der Benutzer den Vornamen Joe hat, gibt PHP aus:

```
Hallo Joe!
```

11.1 Formulareingaben verarbeiten

Problem

Sie möchten mit derselben HTML-Seite ein Formular ausgeben und danach die in diesem Formular eingegebenen Daten verarbeiten. Mit anderen Worten: Sie möchten die ungebremste Vermehrung von Seiten vermeiden, die jeweils einzelne Schritte einer Transaktion abarbeiten.

Lösung

Verwenden Sie ein verborgenes Feld im Formular, mit dem Sie Ihrem Programm mitteilen, dass es das Formular verarbeiten soll. In diesem Fall hat das verborgene Feld den Namen `stage` und den Inhalt `process`:

```
if (isset($_POST['stage']) && ('process' == $_POST['stage'])) {
    process_form();
} else {
    print_form();
}
```

Diskussion

Wenn Menschen in den frühen Tagen des Webs Formulare schufen, erstellten sie immer zwei Seiten: eine statische HTML-Seite mit dem Formular und ein Skript zur Verarbeitung des Formulars, das eine dynamisch generierte Antwort an den Benutzer lieferte. Dies war etwas unhandlich, denn *form.html* verwies auf *form.cgi*, und wenn die Menschen eine Seite änderten, mussten sie daran denken, auch die andere zu bearbeiten, da das Skript sonst vielleicht nicht mehr funktionsfähig war.

Formulare sind einfacher zu pflegen, wenn sich alle Teile in derselben Datei befinden und der Kontext bestimmt, welcher Teil gerade dargestellt werden soll. Verwenden Sie ein verborgenes Formularfeld mit dem Namen `stage` (Verarbeitungsstufe), mit dem Sie die Position im Verarbeitungsvorgang des Formulars verfolgen; es dient als Auslöser für die Schritte, die den jeweils passenden HTML-Code an den Benutzer zurückgeben. Manchmal ist es allerdings nicht möglich, das Programm auf diese Weise zu entwerfen, z.B. wenn Ihr Formular von einem Skript auf einem fremden Server verarbeitet wird.

Wenn Sie den HTML-Code für Ihr Formular schreiben, sollten Sie den Pfad zu Ihrer Seite in `action` nicht direkt fest einprogrammieren. Sonst können Sie Ihre Seite nicht umbenennen oder verlegen, ohne sie zu bearbeiten. PHP bietet eine hilfreiche Variable, die Sie stattdessen verwenden können:

```
$_SERVER['PHP_SELF']
```

Diese Variable ist ein Pseudonym für die URL der aktuellen Seite. Setzen Sie also den Wert des `action`-Attributs auf diese Variable, und Ihr Formular ruft sich immer selbst auf, auch wenn Sie die Datei an eine andere Stelle auf Ihrem Server verschoben haben.

Das Beispiel aus der Einführung zu diesem Kapitel sieht jetzt also folgendermaßen aus:

```
if (isset($_POST['stage']) && ('process' == $_POST['stage'])) {
    process_form();
} else {
    print_form();
}

function print_form() {
    echo <<<END
        <form action="$_SERVER[PHP_SELF]" method="post">
        Wie lautet Ihr Vorname?
        <input type="text" name="vorname">
        <input type="hidden" name="stage" value="process">
        <input type="submit" value="Sag Hallo">
        </form>
END;
}

function process_form() {
    echo 'Hallo ' . $_POST['vorname'] . '!';
}
```

Wenn Ihr Formular mehr als einen Schritt erfordert, setzen Sie `stage` einfach für jeden Schritt auf einen neuen Wert.

Siehe auch

Rezept 11.3 zur Behandlung von Formularen mit mehreren Seiten.

11.2 Formulareingaben prüfen

Problem

Sie möchten sicherstellen, dass die durch ein Formular eingegebenen Daten bestimmten Kriterien genügen.

Lösung

Legen Sie eine Funktion an, die einen String zur Überprüfung annimmt und `true` zurückgibt, wenn der String der Prüfung standhält, und `false`, wenn dies nicht der Fall ist. In der Funktion verwenden Sie reguläre Ausdrücke und Vergleiche, um die Daten zu prüfen. Beispiel 11-1 zeigt als Beispiel die Funktion `pc_validate_zipcode()` zur Überprüfung einer amerikanischen Postleitzahl.

Beispiel 11-1: pc_validate_zipcode()

```
function pc_validate_zipcode($zipcode) {
    return preg_match('/^[0-9]{5}([- ]?[0-9]{4})?$/', $zipcode);
}
```

Und so wird sie verwendet:

```
if (pc_validate_zipcode($_REQUEST['zipcode'])) {
    // Postleitzahl ist in Ordnung, Weiterverarbeitung möglich.
    process_data();
} else {
    // Diese Postleitzahl ist fehlerhaft; Meldung ausgeben.
    print "Eine US-Postleitzahl muss 5 Stellen haben ";
    print "(oder 9 Stellen, wenn Sie ZIP+4 verwenden).";
    print_form();
}
```

Diskussion

Die Entscheidung darüber, was gültige und ungültige Daten ausmacht, ist fast eher eine philosophische Aufgabe als eine geradlinige Angelegenheit, bei der man einer Reihe festgelegter Schritte folgen kann. In vielen Fällen ist eine Lösung, die in einer Situation perfekt in Ordnung ist, in einer anderen Situation nicht korrekt.

Die einfachste Prüfung besteht darin sicherzustellen, dass ein Feld nicht leer ist. Diese Aufgabe kann man bestens mit der Funktion `empty()` erledigen:

```
if (empty($_POST['vorname'])) {
    // Meldung: Feld VORNAME darf nicht leer sein
}
```

Oftmals möchten Sie auch einfach nur sicherstellen, dass ein gesendeter Wert vom richtigen Datentyp ist und sich im richtigen Wertebereich befindet. Die Validierung einer

Altersangabe, bei der geprüft wird, ob der Wert größer oder gleich null ist, könnte wie folgt aussehen:

```
if (! ctype_digit($_POST['alter'])) {
    print 'Altersangabe: Der Wert muss größer oder gleich null sein.';
}
```

Die Validierung von Zahlen scheint auf den ersten Blick einfach zu sein, ist bei genauerer Betrachtung allerdings nicht ganz so leicht. Oftmals tendiert man dazu, `is_numeric()` zur Validierung von Zahlen zu verwenden. Allerdings ist das, was `is_numeric()` als Zahl ansieht, eher das, was ein Computer darunter versteht. Zum Beispiel werden hexadezimale Werte wie 0xCAFE und Zahlen in exponentieller Schreibweise wie 10e40 als »numerisch« erkannt. Die ctype-Erweiterung bietet neben `ctype_digit()` noch viele weitere hilfreiche Funktionen, die unter *http://php.net/manual/ref.ctype.php* beschrieben werden.

Danach kommen relativ einfache Prüfungen wie bei der US-Postleitzahl. Solche Probleme werden gewöhnlich mit ein oder zwei regulären Ausdrücken gelöst. Zum Beispiel findet:

```
/^[0-9]{5}([- ]?[0-9]{4})?$/
```

alle gültigen amerikanischen Postleitzahlen.

Manchmal ist es allerdings schwierig, den richtigen regulären Ausdruck herauszufinden. Um sicherzustellen, dass jemand nur zwei Namen eingegeben hat, zum Beispiel »Alfred Aho«, können Sie mit diesem Ausdruck prüfen:

```
/^[A-Za-z]+ +[A-Za-z]+$/
```

Tim O'Reilly besteht diesen Test jedoch nicht. Eine Alternative wäre `/^\S+\s+\S+$/`;, aber dann würde Donald E. Knuth zurückgewiesen. Denken Sie also sorgfältig über die ganze Breite der gültigen Eingaben nach, bevor Sie Ihren regulären Ausdruck schreiben.

In manchen Fällen wird es auch mit regulären Ausdrücken schwierig nachzuprüfen, ob ein Feldinhalt zulässig ist. Eine besonders trickreiche Aufgabe ist die Prüfung einer E-Mail-Adresse, wie sie in Rezept 16.5 behandelt wird, und ebenso die Feststellung, ob jemand den Namen seines US-Bundesstaates korrekt angegeben hat. Sie können hier anhand einer Liste von Staaten prüfen, aber was soll geschehen, wenn die für die Post übliche Abkürzung angegeben wurde? Wird »MA« anstelle von Massachusetts funktionieren? Und was ist mit »Mass.«?

Als eine Möglichkeit, dieses Problem zu vermeiden, kann man dem Benutzer eine Dropdown-Liste von zuvor generierten Auswahlwerten präsentieren. Das `select`-Element zwingt die Benutzer durch das Formulardesign, den Staat immer in einem funktionierenden Format auszuwählen, und vermeidet damit Fehler. Dies bringt allerdings eine weitere Reihe von Schwierigkeiten mit sich. Was ist, wenn der Benutzer an einem Ort lebt, der keiner der angebotenen Wahlmöglichkeiten entspricht? Was tun, wenn die Auswahlliste so lang ist, dass dies keine realistische Lösung darstellt?

Es gibt verschiedene Möglichkeiten, derartige Problemen zu lösen. Erstens können Sie eine Option »Sonstige« in der Liste anbieten, sodass auch ein Nicht-US-Benutzer das Formular erfolgreich ausfüllen kann. (Andernfalls würde vermutlich ein zufälliger Wert ausgewählt, um die Site weiter verwenden zu können.) Als Nächstes können Sie den Anmeldevorgang in eine zweiteilige Folge aufteilen. Bei einer langen Liste von Optionen beginnt der Benutzer zuerst damit, einen Buchstaben aus dem Alphabet auszuwählen, mit dem seine Auswahl beginnt, und eine neue Seite präsentiert ihm dann eine Liste mit den Auswahlmöglichkeiten, die mit diesem Buchstaben beginnen.

Wie können Sie sicherstellen, dass bei Verwendung eines Drop-downs auch tatsächlich ein Wert ausgewählt wird, der im Drop-down angeboten wird? Der Nutzer kann einen Request ja auch manipulieren und einen Wert senden, der gar nicht zur Auswahl stand. Die Lösung dafür ist die Verwendung der Funktion `in_array()`:

```
$auswahl = array('eier' => 'Spiegelei',
                 'toast' => 'Buttertoast mit Marmelade',
                 'kaffee' => 'Heißer Kaffee');
echo "<select name='essensauswahl'>\n";
foreach ($auswahl as $key => $name) {
   echo "<option value='$key'>$name</option>\n";
}
echo "</select>";
// Später die Validierung
if (! array_key_exists($_POST['essensauswahl'], $choices)) {
    echo "Sie müssen eine valide Auswahl aus der Liste treffen.";
}
```

Die Auswahlmöglichkeiten müssen in einem Array vorliegen, aus dem auch gleich die Auswahlliste für das Drop-down generiert werden kann. Wenn der vom Benutzer gewählte Wert vom Browser gesendet wird, können Sie mit `in_array()` sicherstellen, dass es sich um eine valide Eingabe handelt, die vom Drop-down angeboten wurde.

Schließlich gibt es sogar noch diffizilere Probleme. Was tun Sie, wenn Sie zwar sicherstellen möchten, dass ein Benutzer eine Information korrekt eingegeben hat, ihm aber nicht mitteilen wollen, dass Sie dies überprüft haben? Eine Situation, bei der dies wichtig ist, ist ein Preisausschreiben; hier gibt es häufig eine spezielle Code-Box auf dem Eingabeformular, in der die Benutzer eine Zeichenkette – `AD78DQ` – aus einer zuvor erhaltenen E-Mail oder einem Werbezettel eintragen sollen. Sie möchten sicherstellen, dass es keine Tippfehler gibt, oder den jeweiligen Benutzer andernfalls nicht als gültigen Teilnehmer zählen. Außerdem möchten Sie ihm nicht ermöglichen, Codes einfach zu erraten, da man dann die Codes ausprobieren und auf diese Weise das System knacken könnte.

Die Lösung besteht darin, zwei Eingabefelder anzubieten. Der Benutzer gibt seinen Code zweimal ein; wenn die beiden Felder übereinstimmen, akzeptieren Sie die Daten als legal und überprüfen sie dann (im Stillen). Wenn die Felder nicht übereinstimmen, weisen Sie die Eingabe zurück und lassen sie vom Benutzer korrigieren. Dieses Vorgehen vermeidet Tippfehler, legt aber nicht offen, wie die Code-Überprüfung funktioniert; so können auch falsch geschriebene E-Mail-Adressen vermieden werden.

Letztendlich ermöglicht PHP nur die Validierung auf dem Server. Serverseitige Validierung erfordert, dass eine Anfrage an den Server gestellt wird und eine Seite als Antwort zurückgesandt wird; im Endeffekt kann dies langsam sein. Daneben gibt es die Möglichkeit der clientseitigen Validierung mithilfe von JavaScript. Während clientseitige Eingabeüberprüfung schneller ist, legt sie aber Ihren Code den Anwendern gegenüber offen und funktioniert möglicherweise nicht, wenn ein Client JavaScript nicht unterstützt. Deshalb sollten Sie clientseitigen Validierungs-Code immer auf dem Server duplizieren.

Siehe auch

Rezept 16.5 zu einem regulären Ausdruck zur Prüfung von E-Mail-Adressen; Kapitel 9, »*Validierung mit PHP und JavaScript*« in *Webdatenbank-Applikationen mit PHP & MySQL*, 2. Auflage (Hugh Williams und David Lane, O'Reilly Verlag).

11.3 Mit mehrseitigen Formularen arbeiten

Problem

Sie möchten ein Formular verwenden, das mehr als eine Seite anzeigt und die Daten von einer zur nächsten Seite aufhebt.

Lösung

Machen Sie vom Session-Tracking Gebrauch:

```
session_start();
$_SESSION['username'] = $_GET['username'];
```

Sie können auch Variablen aus den vorhergehenden Seiten eines Formulars als verborgene Eingabefelder in späteren Seiten einfügen:

```
<input type="hidden" name="username"
       value="<?php echo htmlentities($_GET['username']); ?>">
```

Diskussion

Benutzen Sie möglichst immer das Session-Tracking. Dies ist sicherer, da Benutzer die Session-Variablen nicht verändern können. Um eine Sitzung zu beginnen, rufen Sie `session_start()` auf; damit wird eine neue Session angelegt oder eine vorhandene fortgeführt. Beachten Sie, dass dieser Schritt nicht erforderlich ist, wenn Sie in Ihrer *php.ini*-Datei `session.auto_start` eingeschaltet haben. Variablen, die `$_SESSION` zugewiesen sind, werden automatisch propagiert. Im Lösungsbeispiel wird die Formularvariable `username` dadurch bewahrt, dass sie mit `$_GET['username']` mit `$_SESSION['username']` der Session zugewiesen wird.

Um während einer nachfolgenden Anfrage auf diesen Wert zuzugreifen, rufen Sie `session_start()` auf und prüfen dann `$_SESSION['username']`:

```
session_start();
$username = htmlentities($_SESSION['username']);
print "Hallo $username.";
```

Wenn Sie in diesem Fall `session_start()` nicht aufrufen, ist `$_SESSION` nicht gesetzt.

Achten Sie darauf, dass der Server und der Speicherort Ihrer Session-Dateien (Dateisystem, Datenbank usw.) gesichert sind; andernfalls ist Ihr System verwundbar, wenn falsche Identitäten vorgetäuscht werden.

Wenn in Ihrer PHP-Installation das Session-Tracking nicht aktiviert ist, können Sie ersatzweise auch verborgene Formularvariablen verwenden. Die Weitergabe von Daten mithilfe verborgener Formularelemente ist aber nicht sicher, da jedermann diese Felder editieren und somit Anfragen verfälschen kann. Allerdings können Sie mit geringem Aufwand die Sicherheit auf ein verlässliches Niveau anheben.

Am einfachsten benutzen Sie verborgene Felder, indem Sie diese in Ihr Formular einfügen.

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>"
      method="get">

<input type="hidden" name="username"
       value="<?php echo htmlentities($_GET['username']); ?>">
```

Wenn dieses Formular erneut abgesendet wird, enthält `$_GET['username']` seinen vorherigen Wert, sofern nicht jemand ihn verändert hat.

Eine kompliziertere, aber auch sicherere Lösung besteht darin, dass Sie Ihre Variablen mithilfe von `serialize()` in einen String konvertieren, einen geheimen Hash von den Daten berechnen und diese beiden Teile der Information in das Formular einfügen. Bei der nächsten Anfrage validieren Sie dann die Daten und deserialisieren sie. Wenn sie den Verifikationstest nicht bestehen, wissen Sie, dass irgendjemand die Informationen verändert hat.

Die in Beispiel 11-2 dargestellte Codierungsfunktion `pc_encode()` übernimmt die zu codierenden Daten in Form eines Arrays.

Beispiel 11-2: pc_encode()

```
$secret = 'Foo25bAr52baZ';

function pc_encode($data) {
  $data = serialize($data);
  $hash = md5($GLOBALS['secret'] . $data);
  return array($data, $hash);
}
```

Die Funktion pc_encode() serialisiert die Daten zu einem String, berechnet dann einen Validierungs-Hash und gibt diese beiden Variablen zurück.

Die Funktion pc_decode() in Beispiel 11-3 macht die Arbeit ihres Gegenstücks wieder rückgängig.

Beispiel 11-3: pc_decode()

```
function pc_decode($data, $hash) {
  if (!empty($data) && !empty($hash)) {
    if (md5($GLOBALS['secret'] . $data) == $hash) {
      return unserialize($data);
    } else {
      error_log("Validierungsfehler: Daten wurden verändert");
      return false;
    }
  }
  return false;
}
```

Die Funktion pc_decode() erzeugt den Hash mit dem geheimen Wort erneut und vergleicht ihn mit dem Hash-Wert aus dem Formular. Wenn sie gleich sind, ist $data gültig und kann deserialisiert werden. Wenn der Test fehlschlägt, schreibt die Funktion eine Mitteilung in das Fehlerprotokoll und gibt false zurück.

Diese Funktionen arbeiten folgendermaßen zusammen:

```
<?php
$secret = 'Foo25bAr52baZ';

// Alte Daten laden und validieren.
if (! $data = pc_decode($_GET['data'], $_GET['hash'])) {
  // Einbruchsversuch
}

// Formular verarbeiten (neue Formulardaten sind in $_GET).

// $data aktualisieren.
$data['username'] = $_GET['username'];
$data['stage']++;
unset($data['password']);

// Ergebnis codieren.
list ($data, $hash) = pc_encode($data);

// Daten und Hash in das Formular einfügen.
?>
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="get">
...
<input type="hidden" name="data"
       value="<?php echo htmlentities($data); ?>">
<input type="hidden" name="hash"
       value="<?php echo htmlentities($hash); ?>">
</form>
```

Am Anfang des Skripts werden die Variablen aus dem Formular zur Decodierung an `pc_decode()` übergeben. Nachdem die Informationen in `$data` geladen sind, kann die Formularverarbeitung fortgesetzt werden, indem `$_GET` auf neue Variablen und `$data` auf vorhandene Daten geprüft wird. Wenn Sie damit fertig sind, aktualisieren Sie `$data`, sodass es auch die neuen Variablen enthält, codieren den Inhalt und berechnen dabei einen neuen Hash. Schließlich geben Sie das neue Formular aus, dabei fügen Sie `$data` und `$hash` als verborgene Variablen ein.

Siehe auch

Rezepte 10.5 und 10.6 mit Informationen zur Verwendung des Session-Moduls; Rezept 11.8 mit Einzelheiten zur Verwendung von `htmlentities()` zum Auszeichnen von Kontrollzeichen in der HTML-Ausgabe; Rezept 17.3 mit Informationen zum Verifizieren von Daten mit Hash-Codes; die Dokumentation zum Session-Tracking unter *http://www.php.net/session* und in Rezept 10.5; die Dokumentationen zu `serialize()` unter *http://www.php.net/serialize* und `unserialize()` unter *http://www.php.net/unserialize*.

11.4 Formulare mit erhaltenen Informationen und Fehlermeldungen erneut anzeigen

Problem

Wenn es ein Problem mit den in einem Formular eingegebenen Daten gibt, möchten Sie anstelle einer generischen Fehlermeldung am Anfang der Seite die Fehlermeldungen in der Nähe der Problemfelder anzeigen. Außerdem möchten Sie, dass die Informationen erhalten bleiben, die der Benutzer beim ersten Mal in das Formular eingegeben hat.

Lösung

Verwenden Sie ein Array `$errors` und speichern Sie Ihre Meldungen in dem Array, wobei Sie den Feldnamen als Index benutzen.

```
if (! pc_validate_zipcode($_REQUEST['zipcode'])) {
    $errors['zipcode'] = "Dies ist eine fehlerhafte Postleitzahl. "
                       . "Postleitzahlen m&uuml;ssen aus 5 Ziffern "
                       . "bestehen und d&uuml;rfen keine Buchstaben "
                       . "enthalten.";
}
```

Wenn Sie dann das Formular erneut ausgeben, können Sie jeden Fehler bei dem zugehörigen Feld anzeigen und den ursprünglichen Wert in das Feld einfügen:

```
echo $errors['zipcode'];
$value = isset($_REQUEST['zipcode']) ?
               htmlentities($_REQUEST['zipcode']) : '';
echo "<input type=\"text\" name=\"zipcode\" value=\"$value\">";
```

Diskussion

Wenn Ihre Benutzer beim Ausfüllen eines langen Formulars auf Fehler stoßen, können Sie die Bedienbarkeit Ihres Formulars insgesamt dadurch verbessern, dass Sie auf die Stelle hinweisen, an der genau die Fehler korrigiert werden müssen.

Alle Fehler in einem einzigen Array zusammenzuführen hat viele Vorteile. Erstens können Sie einfach prüfen, ob bei Ihrem Prüfungsvorgang Fehler gefunden wurden, die korrigiert werden müssen; verwenden Sie dazu einfach `count($errors)`. Diese Methode ist einfacher, als dafür eine gesonderte Variable zu benutzen, insbesondere wenn der Ablauf komplex oder über mehrere Funktionen verteilt ist. Beispiel 11-4 zeigt die Validierungsfunktion `pc_validate_form()`, die ein `$errors`-Array verwendet.

Beispiel 11-4: pc_validate_form()

```
function pc_validate_form() {
  if (! pc_validate_zipcode($_POST['zipcode'])) {
     $errors['zipcode'] = "Eine Postleitzahl hat 5 Ziffern";
  }

  if (! pc_validate_email($_POST['email'])) {
     $errors['email'] = "E-Mail-Adressen sehen aus wie user@example.com";
  }

  return $errors;
}
```

Dies ist sauberer Code, denn alle Fehler werden in einer Variablen gespeichert. Diese Variable können Sie einfach herumreichen, wenn Sie sie nicht im globalen Geltungsbereich unterbringen möchten.

Dadurch dass Sie den Variablennamen als Schlüssel verwenden, erhalten Sie die Verbindung zwischen dem Feld, das den Fehler verursacht hat, und der eigentlichen Fehlermeldung. Diese Verknüpfungen erleichtern es auch, die Elemente in einer Schleife zu durchlaufen, wenn Sie die Fehler anzeigen.

Die wiederholte Darstellung des Formulars können Sie automatisieren; die Funktion `pc_print_form()` in Beispiel 11-5 zeigt Ihnen, wie das geht.

Beispiel 11-5: pc_print_form()

```
function pc_print_form($errors) {
    $fields = array('name'   => 'Name',
                    'rank'   => 'Rank',
                    'serial' => 'Serial');

    if (count($errors)) {
        echo 'Bitte korrigieren Sie die Fehler im folgenden Formular.';
    }
```

Beispiel 11-5: pc_print_form() (Fortsetzung)

```
        echo '<table>';

        // Fehlermeldungen und Formularvariablen ausgeben.
        foreach ($fields as $field => $field_name) {
            // Zeile beginnen.
            echo '<tr><td>';

            // Fehler ausgeben.
            if (!empty($errors[$field])) {
                echo $errors[$field];
            } else {
                echo ' '; // vermeidet unschöne Tabellen
            }

            echo "</td><td>";

            // Name und Eingabefeld darstellen.
            $value = isset($_REQUEST[$field]) ?
                           htmlentities($_REQUEST[$field]) : '';

            echo "$field_name: ";
            echo "<input type=\"text\" name=\"$field\" value=\"$value\">";
            echo '</td></tr>';
        }

        echo '</table>';
    }
```

Der komplexe Teil in `pc_print_form()` resultiert aus dem `$fields`-Array, dessen Schlüssel die Variablennamen und dessen Werte der anzuzeigenden Feldnamen sind. Wenn Sie das Array zu Beginn der Funktion definieren, können Sie eine Schleife erzeugen und mit `foreach` die Werte durchlaufen; andernfalls würden Sie drei getrennte Zeilen identischen Codes benötigen. Dies passt zu dem Variablennamen als Schlüssel in `$errors`, denn Sie können die Fehlermeldung in der Schleife einfach herausfinden, indem Sie `$errors[$field]` überprüfen.

Wenn Sie dieses Beispiel über `input`-Felder vom Typ `text` hinaus erweitern möchten, modifizieren Sie `$fields` so, dass es weitere Metainformationen über Ihre Formularfelder enthält:

```
$fields = array('name' => array('name' => 'Name', 'type' => 'text'),
                'rank' => array('name' => 'Rank', 'type' => 'password'),
                'serial' => array('name' => 'Serial', 'type' => 'hidden')
               );
```

Siehe auch

Rezept 11.2 zur einfachen Formularvalidierung.

11.5 Mehrfaches Absenden desselben Formulars verhindern

Problem

Sie möchten dafür sorgen, dass Anwender nicht mehrmals dasselbe Formular übersenden.

Lösung

Sie generieren einen eindeutigen Identifikator und speichern das Kennzeichen im Formular als verborgenes Feld. Bevor Sie das Formular verarbeiten, prüfen Sie, ob das Kennzeichen bereits übersandt wurde. Wenn dies nicht der Fall ist, fahren Sie fort; wurde es bereits übersandt, sollten Sie einen Fehler generieren.

Beim Erzeugen des Formulars erhalten Sie mithilfe von `uniqid()` einen eindeutigen Identifikator:

```
<?php
$unique_id = uniqid(microtime(),1);
...
?>
<input type="hidden" name="unique_id" value="<?php echo $unique_id; ?>">
</form>
```

Beim Verarbeiten achten Sie dann auf diese ID:

```
$unique_id  = $dbh->quote($_GET['unique_id']);
$sth = $dbh->query("SELECT * FROM database WHERE unique_id = $unique_id");

if ($sth->numRows()) {
    // Bereits erhalten; Fehler auslösen.
} else {
   // Mit den Daten weiterarbeiten.
}
```

Diskussion

Aus verschiedenen Gründen senden Benutzer ein Formular häufig mehrmals ab. Meistens ist es ein Maus-Lapsus: ein Doppelklick auf den Absenden-Button. Benutzer drücken auch schon mal die Zurück-Schaltfläche des Browsers, um die Informationen noch einmal zu sehen oder sie zu ändern, klicken dann aber erneut auf »Absenden« anstelle von »Vorwärts«. Es kann auch absichtlich sein, beispielsweise beim Versuch, die Wahlurne einer Online-Befragung oder den Briefkasten eines Preisausschreibens vollzustopfen. Unsere Lösung verhindert versehentliche Attacken und kann böswillige Anwender zumindest behindern. Sie kann allerdings nicht jede betrügerische Benutzung ausschalten – dazu sind kompliziertere Maßnahmen erforderlich.

Die Lösung verhindert, dass Ihre Datenbank mit allzu vielen Kopien desselben Datensatzes überhäuft wird. Indem Sie ein Kennzeichen generieren und in das Formular einfügen, können Sie dieses spezielle Exemplar eindeutig identifizieren, sogar wenn Cookies deaktiviert sind. Wenn Sie die Daten des Formulars speichern, legen Sie das Kennzeichen mit ihnen ab. Auf diese Weise können Sie einfach prüfen, ob Sie dieses Formular und den zugehörigen Satz in der Datenbank bereits gesehen haben.

Beginnen Sie, indem Sie Ihrer Datenbanktabelle eine zusätzliche Spalte namens `unique_id` hinzufügen, die den Identifikator aufnehmen soll. Wenn Sie die Daten für einen Satz einfügen, fügen Sie auch die ID hinzu, zum Beispiel:

```
$username  = $dbh->quote($_GET['username']);
$unique_id = $dbh->quote($_GET['unique_id']);

$sth = $dbh->query("INSERT INTO members ( username,  unique_id)
                                  VALUES ($username, $unique_id)");
```

Wenn Sie die genaue Datenbankzeile mit dem Formular verknüpfen, können Sie mit einer erneuten Übersendung besser umgehen. Auf die Frage, was dann zu tun ist, kann es hier keine korrekte Antwort geben; es hängt von der jeweiligen Situation ab. In manchen Fällen können Sie die zweite Zusendung insgesamt ignorieren. In anderen werden Sie prüfen wollen, ob sich der Datensatz geändert hat, und wenn dies zutrifft, dem Benutzer eine Dialogbox mit der Frage präsentieren, ob die Daten mit der neuen Information überschrieben werden oder die alten Daten unverändert bleiben sollen. Schließlich können Sie auf die zweite Übersendung des Formulars auch so reagieren, dass Sie den Datensatz stillschweigend überschreiben, und der Benutzer erfährt niemals etwas von dem Problem.

Alle diese Möglichkeiten sollten unter Berücksichtigung der Besonderheiten der jeweiligen Interaktion abgewogen werden. Unserer Auffassung nach gibt es keinen Grund, das Benutzererlebnis durch die Defizite des HTTP-Protokolls einschränken zu lassen. Daher ist die dritte Option, die stillschweigende Aktualisierung, zwar nicht das, was normalerweise geschieht, aber in vieler Hinsicht die natürlichste Wahl. Anwendungen, die mit diesem Vorgehen entwickelt werden, sind benutzerfreundlicher, während die beiden anderen Methoden die meisten Anwender frustrieren oder verwirren.

Vielleicht neigen Sie dazu, die Generierung eines zufälligen Kennzeichens zu vermeiden und stattdessen die um eins erhöhte Anzahl der bereits in der Datenbank vorhandenen Datensätze zu verwenden? Dann ist das Kennzeichen identisch mit dem Primärschlüssel, und Sie benötigen keine zusätzliche Spalte. Bei diesem Vorgehen gibt es aber (mindestens) zwei Probleme. Erstens führt dies zu einer Race Condition. Was geschieht, wenn eine zweite Person das Formular aufruft, bevor die erste Person es abgeschlossen hat? Dann erhält das zweite Formular dasselbe Kennzeichen wie das erste, und es kommt zu einem Konflikt. Diesen können Sie umgehen, indem Sie beim Abruf des Formulars eine leere Zeile in die Datenbank einfügen, sodass die zweite Personen eine um eins höhere

Zahl als die erste erhält. Dies kann jedoch zu leeren Zeilen in der Datenbank führen, wenn Benutzer das Formular nicht abschließen.

Außerdem spricht dagegen, dass es dann trivial ist, einen anderen Satz in der Datenbank zu ändern, weil man einfach die ID auf eine andere Zahl setzen kann. Abhängig von Ihren Sicherheitseinstellungen können mit einer gefälschten GET- oder POST-Sendung die Daten problemlos geändert werden. Ein langes, zufällig gebildetes Kennzeichen kann dagegen nicht einfach dadurch erraten werden, dass man eine andere Ganzzahl aussucht.

Siehe auch

Rezept 17.3 mit weiteren Einzelheiten zur Verifikation von Daten mit Hash-Codes; die Dokumentation zu `uniqid()` unter *http://www.php.net/uniqid*.

11.6 Hochgeladene Dateien verarbeiten

Problem

Sie möchten eine vom Benutzer hochgeladene Datei verarbeiten.

Lösung

Verwenden Sie das Array `$_FILES`:

```
// Aus <input name="event" type="file">
if (is_uploaded_file($_FILES['event']['tmp_name'])) {
    // Datei auf dem Bildschirm ausgeben.
    readfile($_FILES['event']['tmp_name']);
}
```

Diskussion

Seit PHP 4.1 erscheinen alle hochgeladenen Dateien in dem superglobalen Array `$_FILES`. Zu jeder Datei gibt es fünf Informationen:

`name`
: Der dem Eingabeelement im Formular zugewiesene Name.

`type`
: Der MIME-Typ der Datei.

`size`
: Die Größe der Datei in Bytes.

`tmp_name`
: Der Ort, an dem die Datei temporär auf dem Server gespeichert ist.

`error`
: Ein Fehlercode, der bei fehlgeschlagenem Upload genauere Problemhinweise gibt (verfügbar in PHP-Versionen >= 4.2).

Wenn Sie eine frühere PHP-Version einsetzen, müssen Sie stattdessen `$HTTP_POST_FILES` verwenden.

Mögliche Werte des `error`-Elements sind:

`UPLOAD_ERR_OK (0)`
: Upload erfolgreich (keine Fehler).

`UPLOAD_ERR_INI_SIZE (1)`
: Die Größe der hochgeladenen Datei ist größer als der Wert der `upload_max_filesize`-*php.ini*-Direktive.

`UPLOAD_ERR_FORM_SIZE (2)`
: Die Größe der hochgeladenen Datei ist größer als der Wert des Formularelements. `MAX_FILE_SIZE`.

`UPLOAD_ERR_PARTIAL (3)`
: Die Datei wurde nur teilweise hochgeladen.

`UPLOAD_ERR_NO_FILE (4)`
: Es wurde keine Datei hochgeladen.

`UPLOAD_ERR_NO_TMP_DIR (6)`
: Der Upload ist fehlgeschlagen, weil kein temporäres Verzeichnis zum Abspeichern vorhanden war (verfügbar in PHP 4.3.10, 5.0.3 und späteren Versionen).

`UPLOAD_ERR_CANT_WRITE (7)`
: PHP konnte die Datei nicht auf den Datenträger schreiben (verfügbar in PHP >= 5.1.0).

Die aufgeführten Konstanten sind ab PHP 4.3 oder später verfügbar. In früheren Versionen benutzen Sie bitte die in Klammern aufgeführte Zahl zum Auswerten des Werts für den Fehlercode.

Nachdem Sie eine Datei aus dem Array ausgewählt haben, prüfen Sie mit `is_uploaded_file()`, ob es sich bei der Datei, die Sie gerade verarbeiten möchten, um eine legitime, von einem Benutzer hochgeladene Datei handelt. Dann verarbeiten Sie die Datei, wie Sie jede andere Datei auf Ihrem System verarbeiten. Führen Sie die Prüfung in jedem Fall durch; wenn Sie dem vom Benutzer übergebenen Dateinamen blind vertrauen, kann jemand die Anfrage verändern und solche Namen wie */etc/passwd* in die Liste der zu verarbeitenden Dateien einfügen.

Sie können auch die Datei an einen permanenten Speicherort verschieben; verwenden Sie für den sicheren Transfer der Datei die Funktion `move_uploaded_file()`:

```
// Die Datei verschieben: move_uploaded_file() prüft zugleich die
// Legitimität der Datei, daher brauchen Sie nicht is_uploaded_file()
// aufzurufen.
move_uploaded_file($_FILES['event']['tmp_name'], '/path/to/file.txt');
```

Beachten Sie, dass der in tmp_name gespeicherte Name der komplette Dateipfad ist und nicht nur der Basisname. Mithilfe von basename() können Sie bei Bedarf die davor stehenden Verzeichnisnamen abschneiden.

Stellen Sie sicher, dass PHP das Lese- und Schreibrecht sowohl in dem Verzeichnis hat, in dem die temporären Dateien gespeichert werden (in der Konfigurationsdirektive upload_tmp_dir können Sie nachsehen, wo dies ist), als auch an dem Ort, an den Sie die Datei hinkopieren möchten. Dies kann häufig der Benutzer nobody oder apache anstelle Ihres persönlichen Benutzernamens sein. Aus diesem Grund werden Sie, wenn Sie im safe_mode laufen, nach dem Kopieren der Datei an einen anderen Ort wahrscheinlich nicht noch einmal darauf zugreifen können.

Die Verarbeitung von Dateien ist häufig eine subtile Aufgabe, da nicht alle Browser die gleichen Informationen übersenden. Es ist aber wichtig, dass es auf korrekte Weise geschieht, da Sie sonst ein mögliches Sicherheitsloch öffnen. Letztendlich ermöglichen Sie jedem Fremden, eine beliebige von ihm gewählte Datei auf Ihren Rechner zu laden; böswillige Menschen können darin eine Gelegenheit sehen, in den Rechner einzudringen oder ihn zum Absturz zu bringen.

Aus diesem Grund bietet PHP eine Anzahl von Möglichkeiten, mit denen Sie Restriktionen auf hochgeladene Dateien legen können; zum Beispiel kann das Dateihochladen insgesamt abgeschaltet werden. Wenn Sie bei der Verarbeitung von hochgeladenen Dateien auf Schwierigkeiten stoßen, sollten Sie nachsehen, ob Ihre Datei nicht vielleicht abgewiesen wird, weil sie ein Sicherheitsrisiko darzustellen scheint.

Zu diesem Zweck stellen Sie zuerst sicher, dass file_uploads in der Konfigurationsdatei auf On gestellt ist. Danach prüfen Sie, ob die Datei nicht größer als upload_max_filesize ist; die Vorgabe von 2 MByte soll verhindern, dass jemand durch Auffüllen der Festplatte mit einer gigantischen Datei den Rechner zum Absturz bringen kann. Zusätzlich gibt es noch eine Direktive post_max_size, mit der die Maximalgröße aller POST-Daten innerhalb einer einzelnen Anfrage festgelegt wird; der Anfangswert hierfür beträgt 8 MByte.

Wenn es Ihnen nicht gelingt, $_FILES mit Informationen zu füllen, sollten Sie auch in Bezug auf Browser-Unterschiede und Benutzerfehler sicherstellen, dass Sie in das öffnende Tag des Formulars enctype="multipart/form-data" eintragen. PHP benötigt dies, um die Verarbeitung auszulösen. Wenn Sie dies nicht tun können, müssen Sie $HTTP_RAW_POST_DATA manuell parsen. (Siehe die RFCs 1521 und 1522 bezüglich der MIME-Spezifikation unter *http://www.faqs.org/rfcs/rfc1521.html* und *http://www.faqs.org/rfcs/rfc1522.html*.)

Außerdem setzen PHP-Versionen vor 4.1 tmp_name auf none, wenn keine Datei zum Hochladen ausgewählt worden ist; neuere Versionen setzen tmp_name dann auf einen Leer-String. PHP 4.2.1 lässt Dateien mit der Länge 0 zu. Um sicher zu sein, dass eine Datei hochgeladen wurde und nicht leer ist (obwohl unter Umständen leere Dateien das sind, was Sie benötigen), müssen Sie sicherstellen, dass tmp_name gesetzt und size größer als 0 ist. Und schließlich senden nicht alle Browser denselben MIME-Typ für eine Datei; was sie senden, hängt von ihrer jeweiligen Kenntnis verschiedener Dateiarten ab.

Siehe auch

Die Dokumentation zum Umgang mit hochgeladenen Dateien unter *http://www.php.net/features.file-upload* und zu `basename()` unter *http://www.php.net/basename*.

11.7 Die Formularverarbeitung durch PHP absichern

Problem

Sie möchten Eingabevariablen aus Formularen sicher verarbeiten und niemandem die böswillige Veränderung Ihres Codes ermöglichen.

Lösung

Deaktivieren Sie die Konfigurationsdirektive `register_globals` und greifen Sie nur über das `$_REQUEST`-Array auf Variablen zu. Noch weiter erhöhen Sie die Sicherheit, indem Sie `$_GET`, `$_POST` und `$_COOKIE` verwenden und somit genau wissen, wo Ihre Variablen herkommen.

Stellen Sie zu diesem Zweck sicher, dass in Ihrer *php.ini*-Datei die folgende Zeile auftaucht:

```
register_globals = Off
```

Seit PHP 4.2 ist dies die standardmäßige Konfiguration.

Diskussion

Wenn `register_globals` eingeschaltet ist, werden externe Variablen, darunter auch die aus Formularen und Cookies, direkt in den globalen Namensraum importiert. Dies ist zwar sehr bequem, kann aber auch einige Sicherheitslücken mit sich bringen, wenn Sie nicht konsequent die Inhalte und die Herkunft Ihrer Variablen prüfen. Warum ist das so? Es kann vorkommen, dass eine intern verwendete Variable, die nicht für den Zugriff von außen vorgesehen ist, ohne Ihr Wissen von außen überschrieben werden kann.

Dazu ein einfaches Beispiel: Sie haben eine Seite, in der Anwender ihren Benutzernamen und ihr Passwort eingeben. Wenn beide validiert sind, geben Sie eine Identifikationsnummer zurück; anhand dieses numerischen Identifikators suchen Sie die persönlichen Informationen des Benutzers und geben sie aus:

```
// Voraussetzung: magic_quotes_gpc ist auf Off gesetzt.
$username = $dbh->quote($_GET['username']);
$password = $dbh->quote($_GET['password']);

$sth = $dbh->query("SELECT id FROM users WHERE username = $username AND
                    password = $password");
```

```
if (1 == $sth->numRows()) {
    $row = $sth->fetchRow(DB_FETCHMODE_OBJECT);
    $id = $row->id;
} else {
    "Benutzername und Passwort ung&uuml;ltig";
}

if (!empty($id)) {
    $sth = $dbh->query("SELECT * FROM profile WHERE id = $id");
}
```

Normalerweise wird `$id` nur als Ergebnis der Datenbanksuche zur Verifikation durch Ihr Programm gesetzt. Wenn allerdings jemand den GET-String verändert und einen Wert für `$id` übergibt, führt Ihr Skript, sofern `register_globals` aktiviert ist, auch nach einer erfolglosen Suche nach Benutzernamen und Passwort die zweite Datenbankabfrage durch und gibt die Ergebnisse zurück. Ohne `register_globals` bleibt `$id` leer, da nur `$_REQUEST['id']` (und `$_GET['id']`) gesetzt werden.

Es gibt natürlich noch andere Möglichkeiten, dieses Problem auch mit `register_globals` zu lösen. Sie können etwa Ihren Code so umstrukturieren, dass diese Lücke nicht mehr entsteht:

```
$sth = $dbh->query("SELECT id FROM users WHERE username = $username AND
                    password = $password");

if (1 == $sth->numRows()) {
    $row = $sth->fetchRow(DB_FETCHMODE_OBJECT);
    $id = $row->id;
    if (!empty($id)) {
        $sth = $dbh->query("SELECT * FROM profile WHERE id = $id");
    }
} else {
    "Benutzername und Passwort ung&uuml;ltig";
}
```

Nun verwenden Sie `$id` nur noch, wenn der Wert explizit durch einen Datenbankaufruf gesetzt worden ist. Manchmal ist so etwas aber schwierig, wenn das Programm nicht entsprechend ausgelegt ist. Alternativ können Sie auch am Anfang des Skripts alle Variablen mit `unset()` löschen oder sie initialisieren:

```
unset($id);
```

Dadurch wird ein fehlerhafter `$id`-Wert entfernt, bevor er Ihr Programm beeinflussen kann. Da PHP aber keine Variablen-Initialisierung verlangt, kann man dies unter Umständen einmal vergessen, und so kann sich der Fehler einschleichen, ohne dass es eine Warnung von PHP gibt.

Siehe auch

Die Dokumentation zu `register_globals` unter *http://www.php.net/manual/en/security.globals.php*.

11.8 Steuerzeichen in Benutzerdaten durch Escape-Sequenzen ersetzen

Problem

Sie möchten Daten, die von Benutzern eingegeben worden sind, auf einer HTML-Seite sicher ausgeben, um z.B. Cross-Site Scripting (XSS) Angriffen vorzubeugen.

Lösung

Für HTML-Code, den Sie als einfachen Text ausgeben möchten und der Links und andere Tags enthält, verwenden Sie `htmlentities()`:

```
echo htmlentities('<p>O'Reilly & Associates</p>');
&lt;p&gt;O'Reilly & Associates&lt;/p&gt;
```

Diskussion

PHP verfügt über einige Funktionen, mit denen Sie Zeichen durch HTML-Entities ersetzen können. Die elementarste von ihnen ist `htmlspecialchars()`, die folgende vier Zeichen ersetzt: < > " und &. Abhängig von optionalen Parametern kann sie auch ' anstelle von oder zusätzlich zu " übersetzen. Für komplexere Codierungen verwenden Sie `htmlentities()`; diese Funktion erweitert `htmlspecialchars()`, sodass alle Zeichen ersetzt werden, zu denen es HTML-Entities gibt.

```
$html = "<a href='fletch.html'>Hans' Lieblingsfilm.</a>\n";
print htmlspecialchars($html);              // doppelte Anführungszeichen
print htmlspecialchars($html, ENT_QUOTES);  // einfache und doppelte Anführungszeichen
print htmlspecialchars($html, ENT_NOQUOTES); // weder noch
&lt;a href=&quot;fletch.html&quot;&gt;Hans' Lieblingsfilm.&lt;/a&gt;
&lt;a href=&quot;fletch.html&quot;&gt;Hans&#039; Lieblingsfilm.&lt;/a&gt;
&lt;a href="fletch.html"&gt;Hans' Lieblingsfilm.&lt;/a&gt;
```

Beiden Funktionen können Sie eine Codierungstabelle übergeben, die festlegt, welche Zeichen auf welche Entities abgebildet werden sollen. Die in den obigen Funktionen verwendeten Tabellen erhalten Sie mit der Funktion `get_html_translation_table()`, der Sie `HTML_ENTITIES` oder `HTML_SPECIALCHARS` übergeben. Es wird dann ein Array zurückgegeben, das Zeichen auf Entities abbildet; dieses können Sie dann als Grundlage für eine eigene Tabelle verwenden.

```
$copyright = "Copyright © 2003 O'Reilly & Associates\n";
$table = get_html_translation_table(); // enthält <, >, " und &
$table[©] = '&copy;'                   // fügt © hinzu
print strtr($copyright, $table);
Copyright &copy; 2003 O'Reilly &amp; Associates
```

Siehe auch

Rezepte 16.8, 17.12 und 17.16; die Dokumentationen zu `htmlentities()` unter *http://www.php.net/htmlentities* und `htmlspecialchars()` unter *http://www.php.net/htmlspecialchars*.

11.9 Mit Formularvariablen arbeiten, deren Name einen Punkt enthält

Problem

Sie möchten eine Variable verarbeiten, in deren Name sich ein Punkt befindet. Wenn das Formular abgesendet wird, können Sie aber die Variable nicht finden.

Lösung

Ersetzen Sie den Punkt im Variablennamen durch einen Unterstrich. Wenn Sie beispielsweise ein Formular-Eingabeelement namens `foo.bar` haben, greifen Sie darauf innerhalb von PHP unter dem Variablennamen `$_REQUEST['foo_bar']` zu.

Diskussion

Da PHP den Punkt als Operator zur String-Verkettung verwendet, wird eine Variable mit dem Namen `tier.groesse` automatisch in `tier_groesse` konvertiert; dadurch wird eine Mehrdeutigkeit für den Parser vermieden. Bei `$_REQUEST['tier.groesse']` kann diese Mehrdeutigkeit zwar nicht mehr entstehen, aber aus Kompatibilitäts- und Konsistenzgründen geschieht dies unabhängig davon, wie Sie `register_globals` eingestellt haben.

Normalerweise haben Sie mit dieser automatischen Namenskonvertierung zu tun, wenn Sie ein Bild zum Absenden eines Formulars verwenden. Ein Beispiel: Sie haben eine Straßenkarte mit den Standorten Ihrer Geschäfte, und Sie möchten, dass die Anwender auf einen der Standorte klicken, um weitere Informationen zu erhalten. Hier ist ein Beispiel:

```
<input type="image" name="locations" src="locations.gif">
```

Wenn ein Benutzer auf das Bild klickt, werden die x- und y-Koordinaten in der Form von `locations.x` und `locations.y` übertragen. Um herauszufinden, wo ein Benutzer geklickt hat, müssen Sie daher in PHP die Variablen `$_REQUEST['locations_x']` und `$_REQUEST['locations_y']` prüfen.

Mithilfe einer Reihe von Manipulationen kann man auch in PHP eine Variable mit einem Punkt im Namen erzeugen:

```
${"a.b"} = 123; // durch {} erzwungen

$var = "c.d";   // indirekte Benennung einer Variablen
```

```
$$var = 456;

print ${"a.b"} . "\n";
print $$var . "\n";
123
456
```

Aufgrund der umständlichen Syntax ist so etwas jedoch eher verpönt.

Siehe auch

Die Dokumentation zu externen PHP-Variablen unter *http://www.php.net/manual/language. variables.external.php*.

11.10 Formularelemente mit Mehrfachoptionen verwenden

Problem

Sie haben ein Formularelement mit mehreren Werten, zum Beispiel ein `checkbox`- oder `select`-Element, PHP sieht aber nur einen Wert.

Lösung

Schreiben Sie eckige Klammern ([ ]) hinter den Variablennamen:

```
<input type="checkbox" name="boroughs[]" value="bronx"> die Bronx
<input type="checkbox" name="boroughs[]" value="brooklyn"> Brooklyn
<input type="checkbox" name="boroughs[]" value="manhattan"> Manhattan
<input type="checkbox" name="boroughs[]" value="queens"> Queens
<input type="checkbox" name="boroughs[]" value="statenisland"> Staten Island
```

Innerhalb Ihres Programms behandeln Sie die Variable als Array:

```
print 'Ich mag ' . join(' und ', $boroughs) . '!';
```

Diskussion

Wenn Sie [ ] hinter einem Variablennamen einfügen, behandelt PHP die Variable als Array und nicht als Skalar. Wird einer solchen Variablen ein weiterer Wert zugewiesen, vergrößert PHP automatisch die Größe des Arrays und fügt den neuen Wert am Ende hinzu. Wenn also die ersten drei Felder in der Lösung angekreuzt sind, hat das die gleiche Wirkung, als hätten Sie den folgenden Code am Anfang des Skripts geschrieben:

```
$boroughs[ ] = "bronx";
$boroughs[ ] = "brooklyn";
$boroughs[ ] = "manhattan";
```

Auf diese Weise können Sie auch Informationen aus einer Datenbank zurückgeben, wenn mehrere Datensätze gefunden werden:

```
foreach ($_GET['boroughs'] as $b) {
  $boroughs[] = strtr($dbh->quote($b),array('_' => '\_', '%' => '\%'));
}
$locations = join(',', $boroughs);

$dbh->query("SELECT address FROM locations WHERE borough IN ($locations)");
```

Diese Syntax funktioniert auch bei multidimensionalen Arrays:

```
<input type="checkbox" name="population[NY][NYC]" value="8008278">New York...
```

Wenn dieses Formularelement angekreuzt ist, wird der Wert von `$population['NY']['NYC']` auf `8008278` gesetzt.

Ein Variablenname mit angefügten [ ] kann zu Problemen führen, wenn Sie versuchen, das Element in JavaScript anzusprechen. Anstatt das Element mit seinem Namen zu adressieren, verwenden Sie hier die numerische ID. Sie können den Elementnamen auch in einfache Anführungszeichen platzieren. Eine weitere Möglichkeit besteht darin, dem Element eine ID zuzuweisen, etwa den Namen ohne die [ ], und stattdessen diese ID zu verwenden. Bei dem folgenden Formular:

```
<form>
<input type="checkbox" name="myName[]" value="myValue" id="myName">
</form>
```

referenzieren die folgenden drei Ausdrücke dasselbe Formularelement:

```
document.forms[0].elements[0];             // numerische ID
document.forms[0].elements['myName[]'];    // Name in Anführungszeichen
document.forms[0].elements['myName'];      // zugewiesene ID
```

Siehe auch

Die Einführung zu Kapitel 4 mit weiteren Informationen zu Arrays.

11.11 Drop-down-Menüs auf Basis des aktuellen Datums erzeugen

Problem

Sie möchten eine Reihe von Drop-down-Menüs erstellen, die automatisch auf dem aktuellen Datum stehen.

Lösung

Finden Sie mit `date()` das aktuelle Datum in der Zeitzone des Webservers heraus und durchlaufen Sie die Tage mit `mktime()`.

Der folgende Code generiert option-Werte für den heutigen Tag sowie die sechs folgenden Tage. Im diesem Fall ist »heute« der 1. Januar 2002.

```
list($hour, $minute, $second, $month, $day, $year) =
                                  split(':', date('h:i:s:m:d:Y'));

// Eine Woche in einzelnen Tagen
for ($i = 0; $i < 7; ++$i) {
    $timestamp = mktime($hour, $minute, $second, $month, $day + $i, $year);
    $date = date("D, F j, Y", $timestamp);

    print "<option value=\"$timestamp\">$date</option>\n";
}
<option value="946746000">Tue, January 1, 2002</option>
<option value="946832400">Wed, January 2, 2002</option>
<option value="946918800">Thu, January 3, 2002</option>
<option value="947005200">Fri, January 4, 2002</option>
<option value="947091600">Sat, January 5, 2002</option>
<option value="947178000">Sun, January 6, 2002</option>
<option value="947264400">Mon, January 7, 2002</option>
```

Diskussion

In der Lösung setzen wir den value für jedes Datum auf die Unix-Zeitstempel-Repräsentation, da wir damit innerhalb unseres Programms einfacher umgehen können. Natürlich können Sie aber ein anderes Format verwenden, das Ihnen am nützlichsten und am besten geeignet erscheint.

Geben Sie nicht der Versuchung nach, die mktime()-Aufrufe wegzulassen. Datums- und Zeitwerte sind nicht so konsistent, wie Sie vielleicht hoffen. Je nachdem, was Sie tun, erhalten Sie möglicherweise nicht das erwartete Ergebnis. Ein Beispiel:[1]

```
$timestamp = mktime(0, 0, 0, 10, 24, 2002); // 24. Oktober 2002
$one_day = 60 * 60 * 24; // Anzahl der Sekunden eines Tages

// Die Woche als einzelne Tage ausgeben.
for ($i = 0; $i < 7; ++$i) {
    $date = date("D, F j, Y", $timestamp);

    print "<option value=\"$timestamp\">$date</option>";

    $timestamp += $one_day;
}
<option value="972619200">Fri, October 25, 2002</option>
<option value="972705600">Sat, October 26, 2002</option>
<option value="972792000">Sun, October 27, 2002</option>
<option value="972878400">Sun, October 27, 2002</option>
```

1 Die Funktion date() formatiert das Datum in englischer Sprache. Mehr über die Lokalisierung von Datumsangaben finden Sie in Kapitel 19.

```
<option value="972964800">Mon, October 28, 2002</option>
<option value="973051200">Tue, October 29, 2002</option>
<option value="973137600">Wed, October 30, 2002</option>
```

Dieses Skript soll eigentlich den Tag, den Monat und das Jahr für eine Periode von sieben Tagen ausgeben, die am 24. Oktober 2002 beginnt. Sie funktionert aber nicht wie erwartet.

Warum gibt es zwei »Sun, October 27, 2002«? Die Antwort liegt in der Sommerzeit. Es trifft nicht zu, dass die Anzahl der Sekunden eines Tages konstant bleibt; tatsächlich ändert sich diese Zahl fast mit Sicherheit. Besonders problematisch ist dabei, dass Sie, wenn Sie nicht gerade in der Nähe eines der Übergangstage sind, den Fehler beim Testen kaum finden werden.

Siehe auch

Kapitel 3, zum Teil Rezept 3.14, aber auch die Rezepte 3.1, 3.2, 3.4, 3.11 und 3.15; die Dokumentationen zu `date()` unter *http://www.php.net/date* und `mktime()` unter *http://www.php.net/mktime*.

KAPITEL 12

Datenbankzugriff

12.0 Einführung

Datenbanken sind für viele Webanwendungen zentral. . Sie enthalten Sammlungen von Daten, die man durchsuchen und aktualisieren kann, wie beispielsweise Benutzerlisten, einen Warenkatalog oder brandheiße Schlagzeilen. Einer der Gründe, aus denen PHP so ungemein gut für die Webprogrammierung geeignet ist, ist seine umfassende Datenbankunterstützung. PHP konnte (bei der letzten Zählung) mit mehr als 20 verschiedenen Datenbanksystemen interagieren, von denen einige relational sind, andere nicht. Die relationalen Datenbanken, mit denen es kommunizieren kann, sind Apache Derby, DB++, FrontBase, IBM Cloudscape, IBM DB2, Informix, Interbase, Ingres II, Microsoft SQL Server, mSQL, MySQL, MySQL MaxDB, Oracle, Ovrimos SQL Server, PostgreSQL, SQLite und Sybase, die nicht relationalen Datenbanken sind dBase, filePro, HyperWave, Paradox und die DBM-Familie einfacher Dateidatenbanken. Außerdem bietet es ODBC-Unterstützung – selbst wenn Ihre Lieblingsdatenbank nicht auf der Liste ist, können Sie diese also mit PHP nutzen, wenn sie ODBC unterstützt.

DBM-Datenbanken, die wir uns in Rezept 12.2 ansehen werden, sind einfache, robuste und effiziente, nur minimal strukturierte Dateien, sie können Daten allerdings nur in Form von Schlüssel/Wert-Paare speichern. Wenn Ihre Daten als Schlüssel/Wert-Abbildung organisiert werden können, sind DBM-Datenbanken eine ausgezeichnete Wahl.

Von seiner besten Seite zeigt sich PHP aber vor allem, wenn es mit einer SQL-Datenbank arbeiten darf. Diese Kombination wird in den meisten Rezepten dieses Kapitels verwendet. SQL-Datenbanken können kompliziert sein, sind gleichzeitig aber äußerst mächtig. Wenn Sie eine bestimmte SQL-Datenbank unter PHP einsetzen wollen, muss PHP bereits bei der Kompilierung so eingestellt werden, dass es eine Unterstützung für diese spezielle Datenbank bietet. Wurde PHP mit Unterstützung für das dynamische Laden von Modulen kompiliert, kann die Datenbankunterstützung auch als dynamisches Modul kompiliert sein.

Ab PHP 5.3 steht Ihnen der MySQL Native Driver (*mysqlnd*) zur Verfügung. Dabei handelt es sich um eine Erweiterung, die ein Bindeglied zwischen PHP und einer MySQL-Datenbank darstellt. In Versionen vor PHP 5.3 hat man hierfür in der Regel die MySQL-

Client-Library (*libmysql*) eingesetzt. Der Vorteil der *mysqlnd*-Erweiterung ist, dass sie Teil von PHP selbst ist. Sie ist unter der PHP License lizenziert, und Quellcodes sowie Binärdateien können direkt von *http://php.net* bezogen werden. Die *mysqlnd*-Erweiterung kann mit den folgenden Datenbankerweiterungen genutzt werden: *ext/mysql*, *ext/mysqli* und *PDO_MYSQL*. Der MySQL Native Driver kann mit dem MySQL-Server Version 4.1 oder neuer zusammenarbeiten.

Als Programmierer müssen Sie sich beim Einsatz von *mysqlnd* an kein neues Programmier-Interface (API) gewöhnen, da es kein neues gibt. Die Änderungen finden alle auf einer tieferen Ebene statt – »unter der Haube« sozusagen.

Die Vorteile beim Einsatz von *mysqlnd* sind sehr zahlreich. Zunächst ist der Code absolut auf PHP zugeschnitten, und es werden keine externen Bibliotheken benötigt. Das Copyright liegt allein bei der PHP Group. Durch die hohe Integration mit PHP selbst beachtet *mysqlnd* außerdem Speicherplatzbeschränkungen, die in der *php.ini*-Datei festgelegt werden können. Weiterhin erlaubt *mysqlnd* verbesserte persistente Verbindungen und die Möglichkeit, Performancestatistiken des MySQL-Servers auszulesen.

Gibt es Nachteile? Ein paar wenige. Der MySQL Native Driver implementiert (noch) nicht alle Möglichkeiten, die die MySQL Client Library bietet. Derzeit gibt es keine Möglichkeit zum Einsatz einer Datenkompression und auch keinen SSL-Support. Wenn Sie diese Features nicht benötigen, ist der Einsatz von *mysqlnd* unbedingt zu empfehlen. Weitere Informationen zu *mysqlnd* finden Sie z.B. im PHP-Handbuch unter *http://php.net/mysqli.mysqlnd*.

Die Beispiele zu SQL-Datenbanken in diesem Kapitel nutzen die Datenbankzugriffsschicht PDO von PHP 5. Bei PDO nutzen Sie immer die gleichen PHP-Funktionen, egal mit welchem Datenbanksystem Sie arbeiten. Auch wenn die SQL-Syntax bei den einzelnen Datenbanksystemen anders aussehen kann, bleibt der PHP-Code jedoch immer der gleiche. PDO bietet in dieser Hinsicht eine Abstraktion für den Datenbankzugriff, keine absolute Datenbankabstraktion. Andere PHP-Bibliotheken wie PEAR DB (*http://pear.php.net/package/db*), ADODb (*http://adodb.sourceforge.net/*) und MDB2 (*http://pear.php.net/package/MDB2*) versuchen, die Datenbankseite vollständig zu abstrahieren – sie verbergen die Implementierungsdetails der unterschiedlichen Datenbanken wie den Umgang mit Datumswerten und Spaltentypen hinter einer Codeschicht. Eine Abstraktion dieser Art kann Ihnen zwar etwas Arbeit sparen, wenn Sie Software schreiben, die mit vielen verschiedenen Typen von Datenbanken verwendet werden soll, kann aber andere Probleme verursachen. Wenn Sie SQL schreiben, das für einen bestimmten Typ Datenbank gedacht ist, können Sie die Funktionalitäten dieser Datenbank nutzen, um höchste Leistung zu erhalten.

In PHP 5 integriert ist SQLite, ein mächtiges Datenbanksystem, das keinen separaten Server erfordert. SQLite ist eine ausgezeichnete Wahl, wenn Sie eine moderate Zahl von Zugriffen haben und sich nicht damit herumschlagen wollen, einen Datenbankserver laufen zu lassen. Rezept 12.3 betrachtet einige Vor- und Nachteile von SQLite.

Unter PHP 4 können Sie SQLite über die PECL SQLite-Erweiterung nutzen (*http://pecl.php.net/package/SQLite*).

Viele SQL-Beispiele in diesem Kapitel nutzen eine Tabelle mit Daten über Sternzeichen. Die Struktur dieser Tabelle sehen Sie in Beispiel 12-1, die Daten in der Tabelle in Beispiel 12-2.

Beispiel 12-1: Tabellenstruktur für die Beispiele

```
CREATE TABLE zodiac (
  id INT UNSIGNED NOT NULL,
  sign CHAR(11),
  symbol CHAR(13),
  planet CHAR(7),
  element CHAR(5),
  start_month TINYINT,
  start_day TINYINT,
  end_month TINYINT,
  end_day TINYINT,
  PRIMARY KEY(id)
);
```

Beispiel 12-2: Die Daten der Beispieltabelle

```
INSERT INTO zodiac VALUES (1,'Aries','Ram','Mars','fire',3,21,4,19);
INSERT INTO zodiac VALUES (2,'Taurus','Bull','Venus','earth',4,20,5,20);
INSERT INTO zodiac VALUES (3,'Gemini','Twins','Mercury','air',5,21,6,21);
INSERT INTO zodiac VALUES (4,'Cancer','Crab','Moon','water',6,22,7,22);
INSERT INTO zodiac VALUES (5,'Leo','Lion','Sun','fire',7,23,8,22);
INSERT INTO zodiac VALUES (6,'Virgo','Virgin','Mercury','earth',8,23,9,22);
INSERT INTO zodiac VALUES (7,'Libra','Scales','Venus','air',9,23,10,23);
INSERT INTO zodiac VALUES (8,'Scorpio','Scorpion','Mars','water',10,24,11,21);
INSERT INTO zodiac VALUES (9,'Sagittarius','Archer','Jupiter','fire',11,22,12,21);
INSERT INTO zodiac VALUES (10,'Capricorn','Goat','Saturn','earth',12,22,1,19);
INSERT INTO zodiac VALUES (11,'Aquarius','Water Carrier','Uranus','air',1,20,2,18);
INSERT INTO zodiac VALUES (12,'Pisces','Fishes','Neptune','water',2,19,3,20);
```

Die Rezepte 12.4 bis 12.9 behandeln die Grundlagen der Kommunikation mit einem Datenbankserver: Verbindungsherstellung, Absenden von Abfragen und Abrufen von Ergebnissen sowie Abfragen, die Daten in der Datenbank ändern.

Ein typisches PHP-Programm sammelt Daten aus HTML-Formularfeldern und speichert sie in der Datenbank. Einige Zeichen wie das Apostroph und der Backslash haben in SQL eine besondere Bedeutung. Sie müssen folglich sicherstellen, dass Ihre Formulardaten diese Zeichen nicht enthalten. PHP bietet eine Funktionalität namens »magic quotes« (magisches Maskieren), die das vereinfachen soll. Wenn die Konfigurationseinstellung `magic_quotes_gpc` auf `on` gesetzt ist, werden in Variablen, die aus `get`- oder `post`-Anfragen und Cookies stammen, einfache Anführungszeichen, doppelte Anführungszeichen, Backslashs und NULs mit einem Backslash maskiert. Sie können auch `magic_quotes_runtime` einschalten, damit Anführungszeichen, Backslashs und NULs aus externen Quellen

wie Datenbankabfragen oder Textdateien ebenfalls automatisch maskiert werden. Ist `magic_quotes_runtime` eingeschaltet, wenn Sie mit `file()` Text in ein Array einlesen, werden die Sonderzeichen in diesem Array mit einem Backslash maskiert.

Unglücklicherweise erweisen sich die erträumten magischen Maskierungen meist eher als »nervende Maskierungen«. Wenn Sie übermittelte Formulardaten in anderem Kontext als SQL-Abfragen nutzen möchten (beispielsweise um sie in einer Seite anzuzeigen), müssen Sie die Maskierungen rückgängig machen, damit die Seite richtig aussieht. Sollte Ihnen schon einmal eine PHP-Website über den Weg gelaufen sein, in der sich vor einfachen Anführungszeichen in Textfeldern die Backslashs zu ballen scheinen, haben das meist die magischen Maskierungen zu verantworten. Rezept 12.8 erläutert PDOs Unterstützung für *gebundene Parameter*, die eine bessere Möglichkeit bieten, Sonderzeichen in Benutzereingaben zu maskieren, die in SQL-Abfragen eingebaut sind. Rezept 12.10 befasst sich ausführlicher mit der Maskierung von Sonderzeichen in Abfragen. Allgemeine Debugging-Techniken, die Sie einsetzen können, um aus Datenbankabfragen resultierende Fehler zu klären, werden in Rezept 12.11 behandelt.

Die restlichen Rezepte befassen sich mit Datenbankaufgaben, die komplexer als einfache Abfragen sind. Rezept 12.12 zeigt, wie man automatisch eindeutige Identifikationsnummern generiert, die man als Datensatz-Kennnummern verwenden kann. Rezept 12.13 erläutert, wie man zur Laufzeit Abfragen aus einer Feldliste aufbaut. Dieses Rezept vereinfacht den Umgang mit `INSERT`- und `UPDATE`-Abfragen, die viele Spalten betreffen. Rezept 12.14 führt vor, wie man Links anzeigt, über die man durch über mehrere Seiten verteilte Ergebnisse navigieren kann, auf denen jeweils nur eine beschränkte Zahl von Datensätzen angezeigt wird. Wenn Sie den Datenbankzugriff beschleunigen wollen, können Sie Abfragen und ihre Ergebnisse zwischenspeichern, wie Rezept 12.15 erläutert.

Rezept 12.16 zeigt einige Techniken für die Verwaltung des Zugriffs auf eine einzige Datenbankverbindung von unterschiedlichen Stellen eines umfangreichen Programms. Zum Abschluss bindet Rezept 12.17 einige der in diesem Kapitel behandelten Themen zu einem vollständigen Programm zusammen, das ein Thread-basiertes Forum in einer Datenbank speichert.

12.1 Textdateien als Datenbanken verwenden

Problem

Sie benötigen eine Möglichkeit, mit einfachen Mitteln Informationen zwischen Anfragen aufzubewahren.

Lösung

Verwenden Sie eine Textdatei mit freiwilligen Sperren (advisory locking) zur Vermeidung von Konflikten. Sie können die Daten in der Textdatei in einem beliebigen Format spei-

chern (CSV, Pipe-begrenzt usw.). Als eine Möglichkeit können Sie alle Daten, die Sie speichern möchten, in einer Variablen ablegen (einem großen assoziativen Array), mit dieser Variablen `serialize( )` aufrufen und das daraus resultierende Ergebnis in die Datei schreiben:

```
$data_file = '/tmp/data';

// Die Datei zum Lesen und Schreiben öffnen.
$fh = fopen($data_file,'a+') or die($php_errormsg);
rewind($fh)                  or die($php_errormsg);

// Die Datei exklusiv sperren.
flock($fh,LOCK_EX)           or die($php_errormsg);

// Daten einlesen und deserialisieren.
$serialized_data = fread($fh,filesize($data_file)) or die($php_errormsg);
$data = unserialize($serialized_data);

/*
 * Mit den Daten tun, was Sie tun müssen ...
 */
// Die Daten wieder serialisieren.
$serialized_data = serialize($data);

// Die Datei leeren.
rewind($fh)                  or die($php_errormsg);
ftruncate($fh,0)             or die($php_errormsg);

// Daten in die Datei zurückschreiben und Sperre freigeben.
if (-1 == (fwrite($fh,$serialized_data))) { die($php_errormsg); }
fflush($fh)                  or die($php_errormsg);
flock($fh,LOCK_UN)           or die($php_errormsg);
fclose($fh)                  or die($php_errormsg);
```

Diskussion

Wenn Sie Ihre Daten in einer Textdatei speichern, brauchen Sie keine zusätzliche Datenbanksoftware zu installieren, aber das ist auch so ziemlich der einzige Vorteil. Die größten Nachteile dieser Methode sind Umständlichkeit und Ineffizienz. Zu Beginn einer Anfrage müssen Sie Ihre Textdatei sperren und alle Daten aus ihr herauslesen, sogar wenn Sie nur einen kleinen Teil der Daten benötigen. Bis Sie die Datei am Ende der Anfrage wieder freigeben, müssen sich alle anderen Prozesse hinten anstellen und warten, was bedeutet, dass deren Benutzer ebenfalls warten müssen. Einer der wesentlichen Vorteile einer Datenbank besteht darin, dass sie einen strukturierten Zugriff auf die Daten ermöglicht, daher müssen Sie nur diejenigen Daten sperren (und in den Arbeitsspeicher laden), die Sie tatsächlich interessieren. Mit der Textdateilösung ist so etwas nicht möglich.

Noch schlimmer ist, dass die Sperren bei einer Textdatei nicht annähernd so robust sind wie die einer Datenbank. Da `flock()` nur eine Art freiwillige Sperre ermöglicht, können mehrere Prozesse lediglich durch Höflichkeit und sorgfältige Programmierung davon abgehalten werden, sich gegenseitig in die Quere zu kommen und Ihre Daten zu zerstören. Es gibt keine Garantie für die Sicherheit Ihrer Daten vor unschuldig-inkompetenten oder absichtlich bösartigen Programmen.

Siehe auch

Rezept 5.7 behandelt das Serialisieren von Daten; Rezept 21.22 geht auf die Einzelheiten von Dateisperren ein; die Dokumentationen zu `flock()` unter *http://www.php.net/flock*, zu `serialize()` unter *http://www.php.net/serialize* und zu `unserialize()` unter *http://www.php.net/unserialize*.

12.2 DBM-Datenbanken verwenden

Problem

Ihre Daten lassen sich recht leicht als Schlüssel/Wert-Paare darstellen. Sie möchten sie sicher speichern und schnell über die Schlüssel nachschlagen können.

Lösung

Nutzen Sie die DBA-Abstraktionsschicht, um, wie in Beispiel 12-3 gezeigt, auf eine Datenbank nach DBM-Art zuzugreifen.

Beispiel 12-3: Mit einer DBM-Datenbank arbeiten

```
<?php
$dbh = dba_open('fish.db','c','gdbm') or die($php_errormsg);
// Werte abfragen und ändern
if (dba_exists('flounder',$dbh)) {
  $flounder_count = dba_fetch('flounder',$dbh);
  $flounder_count++;
  dba_replace('flounder',$flounder_count, $dbh);
  print "Updated the flounder count.";
} else {
  dba_insert('flounder',1, $dbh);
  print "Started the flounder count.";
}

// Nicht noch mehr Tilapia-Barsch
dba_delete('tilapia',$dbh);

// Was für Fisch haben wir?
for ($key = dba_firstkey($dbh);  $key !== false; $key = dba_nextkey($dbh)) {
```

Beispiel 12-3: Mit einer DBM-Datenbank arbeiten (Fortsetzung)

```
    $value = dba_fetch($key, $dbh);
    print "$key: $value\n";
}

dba_close($dbh);
?>
```

Diskussion

PHP bietet Unterstützung für unterschiedliche Arten von DBM-Backends: GDBM, NDBM, DB2, DB3, DBM und CDB. Die DBA-Abstraktionsschicht ermöglicht Ihnen, die gleichen Funktionen für alle DBM-Backends zu nutzen. Alle diese Backends speichern Schlüssel/Wert-Paare. Sie können alle Schlüssel in einer Datenbank durchlaufen, den mit einem bestimmten Schlüssel verknüpften Wert abrufen und prüfen, ob es einen bestimmten Schlüssel gibt. Schlüssel und Werte sind jeweils Strings.

Das Programm in Beispiel 12-4 verwaltet eine Liste mit Benutzernamen und Passwörtern in einer DBM-Datenbank. Der Benutzername ist das erste Kommandozeilenargument, das Passwort das zweite. Wenn es den angegebenen Benutzernamen in der Datenbank bereits gibt, wird das Passwort in das angegebene Passwort geändert; andernfalls wird die angegebene Benutzername/Passwort-Kombination der Datenbank hinzugefügt.

Beispiel 12-4: Benutzer und Passwörter in einer DBM-Datenbank festhalten

```
<?php
$user = $_SERVER['argv'][1];
$password = $_SERVER['argv'][2];

$data_file = '/tmp/users.db';

$dbh = dba_open($data_file,'c','gdbm') or die("Can't open db $data_file");

if (dba_exists($user,$dbh)) {
    print "User $user exists. Changing password.";
} else {
    print "Adding user $user.";
}

dba_replace($user,$password,$dbh) or die("Can't write to database $data_file");

dba_close($dbh);
?>
```

Die Funktion `dba_open( )` liefert ein Handle auf eine DBM-Datei (oder `false` bei einem Fehler). Sie erwartet drei Argumente. Das erste ist der Dateiname der DBM-Datei, das zweite der Modus für das Öffnen der Datei. Der Modus `r` öffnet eine bestehende Datenbank für den Lesezugriff, der Modus `c` öffnet eine Datenbank für den Lese-/Schreibzugriff und erstellt die Datenbank, wenn sie noch nicht besteht. Der Modus `n` macht das

Gleiche wie c, löscht aber die Datenbank, wenn sie bereits besteht. Das dritte Argument für dba_open( ) gibt den zu verwendenden DBM-Handler an; dieses Beispiel nutzt 'gdbm'. Welche DBM-Handler in Ihre PHP-Installation einkompiliert sind, können Sie herausfinden, indem Sie einen Blick in den *DBA*-Abschnitt Ihrer phpinfo( )-Ausgabe werfen. Die Zeile *Supported handlers* zeigt Ihnen, welche Optionen Ihnen zur Verfügung stehen.

Ob ein bestimmter Schlüssel in einer DBM-Datenbank gesetzt ist, prüfen Sie mit dba_exists( ). Diese Funktion erwartet zwei Argumente: einen String-Schlüssel und ein DBM-Datei-Handle. Sie sucht nach dem Schlüssel in der DBM-Datei und liefert true, wenn sie den Schlüssel findet (oder andernfalls false). Die Funktion dba_replace( ) erwartet drei Argumente: einen String-Schlüssel, einen String-Wert und ein DBM-Datei-Handle. Sie steckt das Schlüssel/Wert-Paar in die DBM-Datei. Wenn es bereits einen Eintrag mit diesem Schlüssel gibt, wird dieser Eintrag mit dem neuen Wert überschrieben.

Mit dba_close( ) schließen Sie eine Datenbank. Eine mit dba_open( ) geöffnete DBM-Datei wird nach Abschluss der Anfrage automatisch geschlossen, aber Sie müssen dba_close( ) explizit aufrufen, um eine mit dba_open( ) erstellte persistente Verbindung zu schließen.

Mit dba_firstkey( ) und dba_nextkey( ) können Sie alle Schlüssel in einer DBM-Datei durchlaufen und mit dba_fetch( ) die mit dem Schlüssel verknüpften Werte abrufen. Das Programm in Beispiel 12-5 berechnet die Länge aller Passwörter in einer DBM-Datei.

Beispiel 12-5: Mit DBM die Länge von Passwörtern berechnen

```
<?php
$data_file = '/tmp/users.db';
$total_length = 0;
if (! ($dbh = dba_open($data_file,'r','gdbm'))) {
    die("Kann Datenbank $data_file nicht öffnen");
}

$k = dba_firstkey($dbh);
while ($k) {
    $total_length += strlen(dba_fetch($k,$dbh));
    $k = dba_nextkey($dbh);
}

print "Die Gesamtlänge aller Passwörter ist $total_length Zeichen.";

dba_close($dbh);
```

Die Funktion dba_firstkey( ) initialisiert $k auf den ersten Schlüssel in der DBM-Datei. Bei jedem Durchlauf der while-Schleife ruft dba_fetch( ) den mit dem Schlüssel $k verbundenen Wert ab. $total_length wird dann um die (mit strlen( ) berechnete) Länge dieses Werts erhöht. Mit dba_nextkey( ) wird $k auf den nächsten Schlüssel in der Datei gesetzt.

Eine Möglichkeit, komplexe Daten in einer DBM-Datenbank zu speichern, bietet `serialize( )`. Beispiel 12-6 speichert die strukturierten Benutzerinformationen in einer DBM-Datenbank, indem die Struktur vor dem Speichern serialisiert und beim Abruf deserialisiert wird.

Beispiel 12-6: Strukturierte Daten in einer DBM-Datenbank speichern

```
<?php
$dbh = dba_open('users.db','c','gdbm') or die($php_errormsg);

// Daten einlesen und deserialisieren.
if ($exists = dba_exists($_POST['username'], $dbh)) {
    $serialized_data = dba_fetch($_POST['username'], $dbh) or die($php_errormsg);
    $data = unserialize($serialized_data);
} else {
    $data = array();
}

// Werte aktualisieren.
if ($_POST['new_password']) {
    $data['password'] = $_POST['new_password'];
}
$data['last_access'] = time();

// Daten wieder in Datei speichern.
if ($exists) {
    dba_replace($_POST['username'],serialize($data), $dbh);
} else {
    dba_insert($_POST['username'],serialize($data), $dbh);
}

dba_close($dbh);
?>
```

Mit Beispiel 12-6 können Sie zwar die Daten mehrerer Benutzer auf einmal in einer Datei speichern, aber Sie können nicht nach Dingen wie der letzten Zugriffszeit eines Benutzers suchen, ohne die Schlüssel in der Datei zu durchlaufen. Wenn Sie derartige Suchvorgänge benötigen, sollten Sie Ihre Daten in eine SQL-Datenbank stecken.

Bei manchen Dingen verhalten sich die verschiedenen DBM-Handler jeweils unterschiedlich: GDBM bietet beispielsweise interne Sperren. Hat ein Prozess eine GDBM-Datei im Lese-/Schreibmodus geöffnet, schlagen weitere `dba_open( )`-Aufrufe fehl, mit denen die gleiche Datei im Lese-/Schreibmodus geöffnet werden soll. Bei anderen DBM-Handlern müssen Sie an den Modus, den Sie an `dba_open( )` übergeben, entweder `l` anhängen, um die Datenbank mithilfe einer separaten *.lck*-Datei zu sperren, oder ein `d`, um die Datenbankdatei selbst zu sperren. Außerdem gibt es zwei datenbankspezifische DBA-Funktionen: `dba_optimize( )` und `dba_sync( )`. Die Funktion `dba_optimize( )` ruft eine Handlerspezifische Funktion zur Optimierung von DBM-Dateien auf. Aktuell ist diese Funktion nur für GDBM definiert und leitet dazu an die Funktion `gdbm_reorganize( )` weiter. Die

Funktion dba_sync( ) ruft eine Handler-spezifische Synchronisierungsfunktion für DBM-Dateien auf. Bei DB2 und DB3 wird die jeweilige sync( )-Funktion aufgerufen, bei GDBM die Funktion gdbm_sync( ). Bei den anderen DBM-Handlern passiert nichts.

DBM-Datenbanken sind ein erster Schritt über einfache Textdateien hinaus. Ihnen fehlen aber viele der Funktionalitäten von SQL-Datenbanken. Die Datenstruktur ist auf Schlüssel/Wert-Paare beschränkt, und die Zuverlässigkeit des Sperrmechanismus ist stark vom DBM-Handler abhängig. Für Daten, die nur gelesen werden, auf die aber sehr häufig zugegriffen wird, können DBM-Handler eine gute Wahl sein.

Siehe auch

Rezept 5.7 befasst sich mit der Serialisierung von Daten; die Dokumentation zu den DBA-Funktionen unter *http://www.php.net/dba*; mehr Informationen zu den DB2- und DB3-DBM-Handlern finden Sie unter *http://www.sleepycat.com/products/bdb.html* (beachten Sie, dass diese Handler auch für eine nicht kommerzielle Nutzung nicht grundsätzlich frei sind); für GDBM sollten Sie sich *http://www.gnu.org/directory/gdbm.html* oder *http://www.mit.edu:8001/afs/athena.mit.edu/project/gnu/doc/html/gdbm_toc.html* ansehen.

12.3 Eine SQLite-Datenbank verwenden

Problem

Sie möchten eine relationale Datenbank nutzen, für die kein separater Serverprozess erforderlich ist.

Lösung

Nutzen Sie SQLite. Dieses robuste und mächtige Datenbankprogramm wird mit PHP 5 ausgeliefert und benötigt keinen separaten Server. Eine SQLite-Datenbank ist einfach eine Datei. Beispiel 12-7 erzeugt eine SQLite-Datenbank, füllt sie mit einer Tabelle, wenn sie noch nicht besteht, und steckt dann einige Daten in die Tabelle.

Beispiel 12-7: Eine SQLite-Datenbank erstellen

```
<?php
$db = new PDO('sqlite:/usr/local/zodiac');

// Die Tabelle erstellen und atomar füllen!
$db->beginTransaction();
// Versuchen, einen Tabelle namens 'zodiac' zu finden.
$q = $db->query("SELECT name FROM sqlite_master WHERE type = 'table'" .
                " AND name = 'zodiac'");
// Wenn die Abfrage keine Zeile lieferte, die Tabelle erstellen
```

Beispiel 12-7: Eine SQLite-Datenbank erstellen (Fortsetzung)

```
// und die Daten einfügen.
if ($q->fetch() === false) {
    $db->exec(<<<_SQL_
CREATE TABLE zodiac (
  id INT UNSIGNED NOT NULL,
  sign CHAR(11),
  symbol CHAR(13),
  planet CHAR(7),
  element CHAR(5),
  start_month TINYINT,
  start_day TINYINT,
  end_month TINYINT,
  end_day TINYINT,
  PRIMARY KEY(id)
)
_SQL_
);

    // Die einzelnen SQL-Anweisungen ...
    $sql=<<<_SQL_
INSERT INTO zodiac VALUES (1,'Aries','Ram','Mars','fire',3,21,4,19);
INSERT INTO zodiac VALUES (2,'Taurus','Bull','Venus','earth',4,20,5,20);
INSERT INTO zodiac VALUES (3,'Gemini','Twins','Mercury','air',5,21,6,21);
INSERT INTO zodiac VALUES (4,'Cancer','Crab','Moon','water',6,22,7,22);
INSERT INTO zodiac VALUES (5,'Leo','Lion','Sun','fire',7,23,8,22);
INSERT INTO zodiac VALUES (6,'Virgo','Virgin','Mercury','earth',8,23,9,22);
INSERT INTO zodiac VALUES (7,'Libra','Scales','Venus','air',9,23,10,23);
INSERT INTO zodiac VALUES (8,'Scorpio','Scorpion','Mars','water',10,24,11,21);
INSERT INTO zodiac VALUES (9,'Sagittarius','Archer','Jupiter','fire',11,22,12,21);
INSERT INTO zodiac VALUES (10,'Capricorn','Goat','Saturn','earth',12,22,1,19);
INSERT INTO zodiac VALUES (11,'Aquarius','Water Carrier','Uranus','air',1,20,2,18);
INSERT INTO zodiac VALUES (12,'Pisces','Fishes','Neptune','water',2,19,3,20);
_SQL_;

    // Die SQL-Zeilen zerlegen und ausführen.
    foreach (explode("\n",trim($sql)) as $q) {
        $db->exec(trim($q));
    }
    $db->commit();
} else {
    // Wenn nichts passiert, als Transaktion zurückrollen.
    $db->rollback();
}
?>
```

Diskussion

Da SQLite-Datenbanken gewöhnliche Textdateien sind, gelten alle Vorsichtsmaßnahmen und Haken beim Dateizugriff in PHP auch für SQLite-Datenbanken. Es ist ausgesprochen ratsam, sie an einem Ort zu speichern, der sich außerhalb der Dokumentwurzel

des Webservers befindet. Kann die Datenbankdatei direkt vom Webserver gelesen werden, kann sie auch von einem Benutzer abrufen werden, der ihren Speicherort errät und damit die Einschränkungen umgeht, die Sie in Ihren PHP-Programmen in Ihre Abfragen eingebaut haben.

In PHP bietet die *sqlite*-Erweiterung gleichzeitig gewöhnlichen SQLite-Zugriff wie auch einen PDO-Treiber für SQLite Version 2. Die Erweiterung *pdo_sqlite* bietet einen PDO-Treiber für SQLite Version 3. Wenn Sie eine neue Datenbank aufsetzen, sollten Sie den PDO-Treiber für SQLite 3 einsetzen, da er schneller ist und mehr Funktionalitäten bietet. Wenn Sie bereits eine SQLite 2-Datenbank haben, sollten Sie erwägen, mithilfe der PDO-Treiber zu SQLite 3 zu migrieren.

Die in Beispiel 12-7 referenzierte Tabelle `sqlite_master` ist eine besondere Systemtabelle, die Informationen zu anderen Tabellen enthält – sie kann also eingesetzt werden, um zu prüfen, ob es eine bestimmte Tabelle bereits gibt. Andere Datenbanksysteme stellen ein System derartiger System-Metadaten auf andere Weisen zur Verfügung.

Siehe auch

Die Dokumentationen zu SQLite unter *http://www.sqlite.org/docs.html* und zu `sqlite_master` unter *http://www.sqlite.org/faq.html#q9*.

12.4 Mit einer SQL-Datenbank verbinden

Problem

Sie möchten auf eine SQL-Datenbank zugreifen, um Daten zu speichern oder abzurufen. Ohne Datenbanken wären dynamische Websites nicht sonderlich dynamisch.

Lösung

Erstellen Sie ein neues `PDO`-Objekt mit dem entsprechenden Verbindungsstring. Beispiel 12-8 zeigt, wie man `PDO`-Objekte für verschiedene Datenbanksysteme erstellt.

Beispiel 12-8: Mit PDO verbinden

```
<?php
// MySQL erwartet die Parameter in einem String.
$mysql = new PDO('mysql:host=db.example.com', $user, $password);
// Mehrere Parameter mit einem ; trennen.
$mysql = new PDO('mysql:host=db.example.com;port=31075', $user, $password);
$mysql = new PDO('mysql:host=db.example.com;port=31075;dbname=food', $user, $password);
// Mit einem lokalen MySQL-Server verbinden.
$mysql = new PDO('mysql:unix_socket=/tmp/mysql.sock', $user, $password);

// Auch PostgreSQL erwartet die Parameter in einem String.
```

Beispiel 12-8: Mit PDO verbinden (Fortsetzung)

```
$pgsql = new PDO('pgsql:host=db.example.com', $user, $password);
// Aber mehrere Parameter werden mit  ' ' getrennt.
$pgsql = new PDO('pgsql:host=db.example.com port=31075', $user, $password);
$pgsql = new PDO('pgsql:host=db.example.com port=31075 dbname=food', $user, $password);
// Benutzer und Passwort können Sie in den DSN stecken.
$pgsql = new PDO("pgsql:host=db.example.com port=31075 dbname=food user=$user password
=$password");

// Oracle
// Wenn der Datenbankname in tnsnames.ora definiert ist, kommt dieser in den DSN.
$oci = new PDO('oci:food', $user, $password);
// Andernfalls ein Instant Client URI.
$oci = new PDO('oci:dbname=//db.example.com:1521/food', $user, $password);

// Sybase (wenn PDO FreeTDS nutzt)
$sybase = new PDO('sybase:host=db.example.com;dbname=food', $user, $password);
// Microsoft SQL Server (wenn PDO die MS SQL Server-Bibliotheken nutzt)
$mssql = new PDO('mssql:host=db.example.com;dbname=food', $user, $password);
// DBLib (für andere Versionen von DB-lib)
$dblib = new PDO('dblib:host=db.example.com;dbname=food', $user, $password);

// ODBC - eine vordefinierte Verbindung
$odbc = new PDO('odbc:DSN=food');
// ODBC - eine Ad-hoc-Verbindung. Liefern Sie, was der entsprechende Treiber benötigt.
$odbc = new PDO('odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=
C:\\data\\food.mdb;Uid=Chef');

// SQLite benötigt nur einen Dateinamen - keinen Benutzer, kein Passwort.
$sqlite = new PDO('sqlite:/usr/local/zodiac.db');
$sqlite = new PDO('sqlite:c:/data/zodiac.db');
// SQLite kann auch mit temporären, im Speicher gehaltenen Datenbanken umgehen.
$sqlite = new PDO('sqlite::memory:');
// Ein SQLite v2-DSN ist einem v3-DNS ähnlich.
$sqlite2 = new PDO('sqlite2:/usr/local/old-zodiac.db');
?>
```

Diskussion

Wenn alles glattgeht, liefert der `PDO`-Konstruktor ein neues Objekt, das eingesetzt werden kann, um die Datenbank abzufragen. Gibt es ein Problem, wird eine `PDOException` ausgelöst.

Wie Sie in Beispiel 12-8 sehen können, ist das Format des DSN stark von dem Datenbanksystem abhängig, mit dem Sie sich verbinden wollen. In der Regel ist das erste Argument für den PDO-Konstruktor allerdings ein String, der den Ort und den Namen der Datenbank angibt, die Sie benötigen, das zweite und das dritte sind der Benutzername und das Passwort, über die die Verbindung hergestellt werden soll. Beachten Sie, dass PHP mit Unterstützung für das jeweilige PDO-Backend kompiliert sein muss, damit Sie

es verwenden können. Nutzen Sie die Ausgabe von `phpinfo()`, um zu ermitteln, welche PDO-Backends Ihre PHP-Einrichtung unterstützt.

Siehe auch

Rezept 12.5 zur Abfrage einer Datenbank; Rezept 12.7 zur Veränderung einer SQL-Datenbank; die Dokumentation zu PDO unter *http://www.php.net/PDO*.

12.5 Eine SQL-Datenbank abfragen

Problem

Sie möchten Daten aus Ihrer Datenbank abfragen.

Lösung

Nutzen Sie `PDO::query()`, um die SQL-Abfrage an das Datenbanksystem zu senden, und durchlaufen Sie dann mit einer `foreach`-Schleife die Ergebnisse, um sie zeilenweise abzurufen, wie Sie es in Beispiel 12-9 sehen.

Beispiel 12-9: Eine SQL-Abfrage an das Datenbanksystem senden

```
<?php
$st = $db->query('SELECT symbol,planet FROM zodiac');
foreach ($st->fetchAll() as $row) {
    print "{$row['symbol']} ist verbunden mit {$row['planet']} <br/>\n";
}
?>
```

Diskussion

Die `query()`-Methode liefert ein `PDOStatement`-Objekt. Dessen `fetchAll()`-Methode bietet eine kompakte Möglichkeit, etwas mit allen Zeilen zu machen, die von einer Abfrage zurückgeliefert werden.

Die `fetch()`-Methode liefert die Ergebnisse zeilenweise, wie Beispiel 12-10 zeigt.

Beispiel 12-10: Einzelne Zeilen abrufen

```
<?php
$rows = $db->query('SELECT symbol,planet FROM zodiac');
$firstRow = $rows->fetch();
print "Das erste Ergebnis sagt: {$row['symbol']} ist verbunden {$row['planet']}";
?>
```

Jeder `fetch()`-Aufruf liefert jeweils die nächste Zeile in der Ergebnismenge. Sind keine Zeilen mehr verfügbar, liefert `fetch()` `false`.

Standardmäßig liefert fetch( ) ein Array, das alle Spalten in der Ergebniszeile doppelt enthält – einmal mit einem Index, der dem Spaltennamen entspricht, und einmal mit einem numerischen Index. Das bedeutet, dass das $firstRow zugewiesene Array in Beispiel 12-10 vier Elemente hat: $firstRow[0] ist Ram, $firstRow[1] ist Mars, $firstRow['symbol'] ist Ram und $firstRow['planet'] ist Mars.

Soll fetch( ) Zeilen in einem anderen Format liefern, müssen Sie query( ) als zweites Argument eine der PDO::FETCH_*-Konstanten übergeben. Sie können die jeweilige Konstante auch als erstes Argument an fetch( ) übergeben. Die möglichen Konstanten und das, was sie fetch( ) zurückliefern lassen, werden in Tabelle 12-1 aufgeführt.

Tabelle 12-1: PDO::FETCH_*-Konstanten

Konstante	Zeilenformat
PDO::FETCH_BOTH	Ein Array mit numerischen Schlüsseln und String-Schlüsseln (Spaltennamen). Das Default-Format.
PDO::FETCH_NUM	Ein Array mit numerischen Schlüsseln.
PDO::FETCH_ASSOC	Ein Array mit String-Schlüsseln (Spaltennamen).
PDO::FETCH_OBJ	Ein Objekt der Klasse stdClass mit Eigenschaften, die den Spaltennamen entsprechen.
PDO::FETCH_LAZY	Ein Objekt der Klasse PDORow mit Eigenschaften, die den Spaltennamen entsprechen. Die Eigenschaften werden erst gefüllt, wenn auf sie zugegriffen wird. Beachten Sie, dass ein gespeichertes Objekt, das auf diese Weise erhalten wurde, mit den Werten aus der neuen Zeile aktualisiert wird, wenn eine neue Zeile abgerufen wird.

Neben den in Tabelle 12-1 aufgeführten Optionen gibt es noch weitere Möglichkeiten, Zeilen zu strukturieren. Diese erfordern allerdings mehr, als einfach nur eine Konstante an query( ) oder fetch( ) zu übergeben.

In Verbindung mit bindColumn( ) ermöglicht Ihnen der Modus PDO::FETCH_BOUND, Variablen einzurichten, deren Werte bei jedem fetch( )-Aufruf aktualisiert werden. Beispiel 12-11 zeigt Ihnen, wie das funktioniert.

Beispiel 12-11: Ergebnisspalten binden

```
<?php
$row = $db->query('SELECT symbol,planet FROM zodiac',PDO::FETCH_BOUND);
// Den Wert der Spalte 'symbol' in $symbol stecken.
$row->bindColumn('symbol', $symbol);
// Den Wert der zweiten Spalte ('planet') in $planet stecken.
$row->bindColumn(2, $planet);
while ($row->fetch()) {
    print "$symbol ist verbunden mit $planet. <br/>\n";
}
?>
```

In Beispiel 12-11 werden $symbol und $planet bei jedem fetch( )-Aufruf neue Werte zugewiesen. Beachten Sie, dass Sie mit bindColumn( ) Spaltennamen oder Spaltennummern verwenden können. Die Spalten werden von 1 ab nummeriert.

Werden die Konstanten PDO::FETCH_INTO und PDO::FETCH_CLASS mit query( ) genutzt, werden die Ergebniszeilen in spezielle Objekte spezieller Klassen gesteckt. Wenn Sie diese Modi einsetzen möchten, müssen Sie zunächst eine Klasse erstellen, die die eingebaute Klasse PDOStatement erweitert. Beispiel 12-12 erweitert PDOStatement mit einer Methode, die die mittlere Länge aller Spaltenwerte ermittelt und dann eine Abfrage einrichtet, die diese nutzt.

Beispiel 12-12: PDOStatement erweitern

```
<?php
class AvgStatement extends PDOStatement {
    public function avg() {
        $sum = 0;
        $vars = get_object_vars($this);
        // PDOStatements eingebaute Variable 'queryString' entfernen.
        unset($vars['queryString']);
        foreach ($vars as $var => $value) {
            $sum += strlen($value);
        }
        return $sum / count($vars);
    }
}
$row = new AvgStatement;
$results = $db->query('SELECT symbol,planet FROM zodiac',PDO::FETCH_INTO, $row);
// $row wird jetzt bei jedem fetch()-Aufruf neu gefüllt.
while ($results->fetch()) {
    print "$row->symbol ist verbunden mit $row->planet (Durchschnitt: {$row->avg()}) <br/>\
n";
}
?>
```

In Beispiel 12-12 sagen das zweite und das dritte Argument für query( ), dass bei jedem Abruf einer neuen Zeile die Werte in die Eigenschaften der Variablen $row gesteckt werden sollen. In der while( )-Schleife ist neben den Eigenschaften von $row dann auch die neue definierte Methode avg( ) verfügbar.

PDO::FETCH_INTO ist nützlich, wenn Sie Ihre Daten über alle fetch()-Aufrufe immer mit dem gleichen Objekt festhalten wollen, beispielsweise wenn Sie ermitteln, ob Sie eine Zeile mit gerader oder ungerader Nummer anzeigen müssen. Aber wenn Sie für jede Zeile ein neues Objekt haben wollen, sollten Sie PDO::FETCH_CLASS nutzen. Übergeben Sie diese Konstante wie PDO::FETCH_INTO an query( ), aber geben Sie im dritten Argument für query( ) einen Klassennamen, keine Objektinstanz an. Die Klasse, deren Namen Sie mit PDO::FETCH_CLASS angeben, muss PDOStatement erweitern.

Siehe auch

Rezept 12.6 zu anderen Möglichkeiten, Daten abzurufen; Rezept 12.7 zur Modifikation einer SQL-Datenbank; Rezept 12.8 zur effizienten Wiederholung von Abfragen; die Dokumentation zu PDO unter *http://www.php.net/PDO*.

12.6 Zeilen ohne eine Schleife abrufen

Problem

Sie suchen nach einem kompakten Mittel, eine Abfrage auszuführen und die von ihr gelieferten Daten abzurufen.

Lösung

Nutzen Sie `fetchAll()`, um alle Ergebnisse einer Abfrage in einem Rutsch abzurufen, wie Sie es in Beispiel 12-13 sehen.

Beispiel 12-13: Alle Ergebnisse auf einmal abrufen

```
<?php
$st = $db->query('SELECT planet, element FROM zodiac');
$results = $st->fetchAll();
foreach ($results as $i => $result) {
   print "Planet $i ist {$result['planet']} <br/>\n";
}
?>
```

Diskussion

Die Methode `fetchAll()` ist hilfreich, wenn Sie etwas ausführen müssen, wozu Sie alle Zeilen benötigen, die eine Abfrage liefert, beispielsweise wenn gezählt werden muss, wie viele Zeilen es gibt, oder wenn Sie die Zeilen in einer anderen Reihenfolge verarbeiten müssen. Wie `fetch()` liefert `fetchAll()` die einzelnen Zeilen standardmäßig als Array mit numerischen Schlüsseln und String-Schlüsseln und akzeptiert verschiedene `PDO::FETCH_*`-Konstanten zur Änderung dieses Verhaltens.

`fetchAll()` akzeptiert außerdem einige weitere Konstanten, die sich auf die zurückgelieferten Ergebnisse auswirken. Brauchen Sie nur eine einzige String-Spalte aus den Ergebnissen, übergeben Sie `PDO::FETCH_COLUMN` und als zweites Argument den Index der gewünschten Spalte. Die erste Spalte ist 0, nicht 1.

Wenn Sie den MySQL Native Driver nutzen, steht Ihnen ab PHP 5.3 die Funktion `mysqli_fetch_all()` bzw. die Methode `MySQLi_Result::fetch_all()` zur Verfügung. Damit ist es auch mit der MySQLi-Erweiterung möglich, alle Zeilen, die als Ergebnis einer MySQL-Datenbankabfrage geliefert werden, direkt in ein Array zu lesen:

```
$mysqli = new mysqli($host, $user, $passwd, $schema);
$query = 'SELECT * FROM help_category h;';

if (! $result = $mysqli->query($query)) {
    die("Ausführen der Abfrage nicht möglich: " . $mysqli->error);
}
```

```
$data = $result->fetch_all(MYSQLI_ASSOC);
print_r($data);
$result->close();
Array
(
    [0] => Array
        (
            [help_category_id] => 1
            [name] => Geographic
            [parent_category_id] => 0
            [url] =>
        )

    [1] => Array
        (
            [help_category_id] => 2
            [name] => Polygon properties
            [parent_category_id] => 32
            [url] =>
        )

    ...

    [36] => Array
        (
            [help_category_id] => 37
            [name] => Data Definition
            [parent_category_id] => 33
            [url] =>
        )

)
```

Die Funktionen akzeptieren einen optionalen Parameter `$resulttype`, der angibt, ob ein assoziatives (`MYSQLI_ASSOC`) oder ein numerisches (`MYSQLI_NUM`) Array zurückgeliefert werden soll. Zusätzlich ist eine Kombination aus beiden Arten möglich (`MYSQLI_BOTH`). Wenn Sie nichts angeben, wird ein numerisches Array zurückgegeben.

In PHP-Versionen vor 5.3 war es mit *ext/mysqli* nicht möglich, alle Zeilen einer Datenbankabfrage direkt in ein Array zu lesen. Die übliche Vorgehensweise sah so aus, dass eine Datenbankabfrage gesendet und danach über das Resultset iteriert wurde, um jede Zeile einzeln als Array auszulesen:

```
if ($result = $mysqli->query($query)) {
    while ($row = $result->fetch_array(MYSQLI_ASSOC)) {
        print_r($row);
    }
    $result->close();
}
Array
(
    [help_category_id] => 1
    [name] => Geographic
```

```
    [parent_category_id] => 0
    [url] =>
)
Array
(
    [help_category_id] => 2
    [name] => Polygon properties
    [parent_category_id] => 32
    [url] =>
)
...
Array
(
    [help_category_id] => 37
    [name] => Data Definition
    [parent_category_id] => 33
    [url] =>
)
```

Siehe auch

Rezept 12.5 zum Abfragen von SQL-Datenbanken und weiteren Informationen zu den Abfrufmodi; Rezept 12.7 zur Modifikation einer SQL-Datenbank; Rezept 12.8 zur effizienten Wiederholung von Abfragen; die Dokumentation zu PDO unter *http://www.php.net/PDO*. Informationen zu `mysqli_fetch_all()` und `mysqli_fetch_array()` finden Sie auf den entsprechenden Seiten des PHP-Manuals unter *http://php.net/manual/mysqli-result.fetch-all.php* bzw. *http://php.net/manual/mysqli-result.fetch-array.php*. Details zum MySQL Native Driver können Sie auf der Seite *http://php.net/manual/mysqli.overview.php#mysqli.overview.mysqlnd* nachlesen.

12.7 Daten in einer SQL-Datenbank modifizieren

Problem

Sie möchten Daten in eine SQL-Datenbank einfügen, aus einer SQL-Datenbank entfernen oder in einer SQL-Datenbank ändern.

Lösung

Nutzen Sie `PDO::exec()`, um, wie in Beispiel 12-14 zu sehen, eine `INSERT`-, `DELETE`- oder `UPDATE`-Anweisung zu senden.

Beispiel 12-14: PDO::exec() einsetzen

```
<?php
$db->exec("INSERT INTO family (id,name) VALUES (1,'Vito')");
```

Beispiel 12-14: PDO::exec() einsetzen (Fortsetzung)

```
$db->exec("DELETE FROM family WHERE name LIKE 'Fredo'");

$db->exec("UPDATE family SET is_naive = 1 WHERE name LIKE 'Kay'");
?>
```

Sie können auch mit `PDO::prepare()` eine Abfrage vorbereiten und mit `PDOStatement::execute()` ausführen, wie Sie es in Beispiel 12-15 sehen.

Beispiel 12-15: Eine Abfrage vorbereiten und ausführen

```
<?php
$st = $db->prepare('INSERT INTO family (id,name) VALUES (?,?)');
$st->execute(array(1,'Vito'));

$st = $db->prepare('DELETE FROM family WHERE name LIKE ?');
$st->execute(array('Fredo'));

$st = $db->prepare('UPDATE family SET is_naive = ? WHERE name LIKE ?');
$st->execute(array(1,'Kay'));
?>
```

Diskussion

Die `exec()`-Methode sendet ihr Argument an die Datenbank. Bei `INSERT`-, `UPDATE`- und `DELETE`-Abfragen liefert sie die Anzahl der von der Abfrage betroffenen Zeilen.

`prepare()` und `execute()` sind insbesondere für Abfragen geeignet, die Sie mehrfach ausführen wollen. Haben Sie eine Abfrage einmal vorbereitet, können Sie sie beliebig oft mit neuen Werten ausführen, ohne sie erneut vorzubereiten. Beispiel 12-16 nutzt dreimal die gleiche Abfrage.

Beispiel 12-16: Eine vorbereitete Abfrage mehrfach verwenden

```
<?php
$st = $db->prepare('DELETE FROM family WHERE name LIKE ?');
$st->execute(array('Fredo'));
$st->execute(array('Sonny'));
$st->execute(array('Luca Brasi'));
?>
```

Siehe auch

Rezept 12.8 für mehr Informationen zur effizienten Wiederholung von Abfragen; die Dokumentationen zu `PDO::exec()` unter *http://www.php.net/PDO.exec*, zu `PDO::prepare()` unter *http://www.php.net/PDO.prepare* und zu `PDOStatement::execute()` unter *http://www.php.net/PDOStatement.execute*.

12.8 Abfragen effizient wiederholen

Problem

Sie möchten eine bestimmte Abfrage mehrfach wiederholen und dabei jedes Mal andere Werte einsetzen.

Lösung

Richten Sie die Abfrage mit `PDO::prepare()` ein und führen Sie sie dann aus, indem Sie auf der von `prepare()` zurückgelieferten vorbereiteten Anweisung `execute()` aufrufen. Die Platzhalter in der an `prepare()` übergebenen Abfrage werden von `execute()` durch die Daten ersetzt, wie Beispiel 12-17 zeigt.

Beispiel 12-17: Eine vorbereitete Abfrage ausführen

```
<?php
// Vorbereiten.
$st = $db->prepare("SELECT sign FROM zodiac WHERE element LIKE ?");
// Einmal ausführen.
$st->execute(array('fire'));
while ($row = $st->fetch()) {
    print $row[0] . "<br/>\n";
}
// Erneut ausführen.
$st->execute(array('water'));
while ($row = $st->fetch()) {
    print $row[0] . "<br/>\n";
}
?>
```

Diskussion

Die an `execute()` übergebenen Werte werden als *gebundene Parameter* bezeichnet – jeder Wert ist mit einem Platzhalter in der Abfrage verknüpft (oder an einen Platzhalter »gebunden«). Zwei wunderbare Dinge an gebundenen Parametern sind Sicherheit und Geschwindigkeit. Wenn Sie gebundene Parameter einsetzen, müssen Sie sich nicht mehr vor SQL-Injection-Angriffen fürchten. PDO maskiert alle Parameter auf die erforderliche Weise, die dafür sorgt, dass Sonderzeichen neutralisiert werden. Außerdem parsen viele Datenbank-Backends bei `prepare()` die Abfrage und optimieren sie. Das sorgt dafür, dass die einzelnen `execute()`-Aufrufe schneller sind, als wenn Sie `exec()` oder `query()` mit einer vollständigen Abfrage in einem String aufrufen, den Sie selbst aufgebaut haben.

Das erste `execute()` in Beispiel 12-17 führt die Abfrage `SELECT sign FROM zodiac WHERE element LIKE 'fire'` aus, das zweite die Abfrage `SELECT sign FROM zodiac WHERE element LIKE 'water'`.

Jedes Mal setzt execute( ) den Wert im zweiten Argument für den Platzhalter ? ein. Gibt es mehrere Platzhalter, stecken Sie die Argumente in der Reihenfolge in ein Array, in der sie in der Abfrage auftauchen sollen. Beispiel 12-18 zeigt prepare( ) und execute( ) mit zwei Platzhaltern.

Beispiel 12-18: Mehrere Platzhalter

```
<?php
$st = $db->prepare(
    "SELECT sign FROM zodiac WHERE element LIKE ? OR planet LIKE ?");
// SELECT sign FROM zodiac WHERE element LIKE 'earth' OR planet LIKE 'Mars'
$st->execute(array('earth','Mars'));
?>
```

Neben den ?-Platzhaltern unterstützt PDO außerdem noch benannte Platzhalter. Wenn eine Abfrage sehr viele Platzhalter enthält, können benannte Platzhalter die Lesbarkeit der Abfrage verbessern. Statt ? geben Sie in der Abfrage Platzhalternamen (die mit einem Doppelpunkt beginnen müssen) ein und nutzen dann diese Platzhalternamen (ohne den Doppelpunkt) als Schlüssel in dem Parameter-Array, das Sie execute( ) übergeben. Beispiel 12-19 zeigt, wie man benannte Platzhalter einsetzt.

Beispiel 12-19: Benannte Platzhalter verwenden

```
<?php
$st = $db->prepare(
    "SELECT sign FROM zodiac WHERE element LIKE :element OR planet LIKE :planet");

// SELECT sign FROM zodiac WHERE element LIKE 'earth' OR planet LIKE 'Mars'
$st->execute(array('planet' => 'Mars', 'element' => 'earth'));
$row = $st->fetch();
```

Benannte Platzhalter verbessern die Lesbarkeit Ihrer Abfragen und ermöglichen Ihnen, die Werte für execute( ) in beliebiger Reihenfolge anzugeben. Beachten Sie allerdings, dass ein Platzhaltername in einer Abfrage nur einmal auftauchen darf. Wenn Sie in einer Abfrage den gleichen Wert zweimal verwenden wollen, müssen Sie zwei verschiedene Platzhalternamen verwenden und den Wert zweimal in das an execute( ) übergebene Array einschließen.

Neben ? und benannten Platzhaltern bietet prepare( ) noch eine dritte Möglichkeit: Packen Sie Werte in Abfragen mit bindParam( ). Diese Methode verbindet automatisch den Wert einer Variablen mit einem bestimmten Platzhalter. Beispiel 12-20 zeigt, wie man bindParam( ) einsetzt.

Beispiel 12-20: bindParam() verwenden

```
<?php
$pairs = array('Mars' => 'water',
               'Moon' => 'water',
               'Sun' => 'fire');
```

Beispiel 12-20: bindParam() verwenden (Fortsetzung)

```
$st = $db->prepare(
    "SELECT sign FROM zodiac WHERE element LIKE :element AND planet LIKE :planet");
$st->bindParam(':element', $element);
$st->bindparam(':planet', $planet);
foreach ($pairs as $planet => $element) {
    // Jetzt muss execute() nichts mehr übergeben werden -
    // diese Werte kommen aus $element und $planet.
    $st->execute();
    var_dump($st->fetch());
}
?>
```

In Beispiel 12-20 besteht kein Bedarf, Werte an `execute()` zu übergeben. Die beiden Aufrufe von `bindParam()` sagen PDO, dass bei jeder Ausführung von `$st` der jeweils aktuelle Wert von `$element` für den Platzhalter `:element` und der jeweils aktuelle Wert von `$planet` für den Platzhalter `:planet` verwendet werden soll. Welche Werte diesen Variablen beim Aufruf von `bindParam()` zugewiesen sind, spielt keine Rolle – nur die Werte beim Aufruf von `execute()` zählen. Da die `foreach`-Anweisung die Array-Schlüssel in `$planet` und die Array-Werte in `$element` packt, werden die Schlüssel und Werte aus `$pairs` in die Abfrage eingesetzt.

Wenn Sie `?`-Platzhalter mit `prepare()` verwenden, geben Sie als erstes Argument für `bindParam()` eine Platzhalterposition statt eines Platzhalternamens an. Platzhalter werden von 1 ausgehend nummeriert, nicht von 0.

Wie `bindParam()` mit dem übergebenen Wert umgeht, wird durch den PHP-Typ des Werts bestimmt. Dass `bindParam()` den Wert als einen ganz bestimmten Typ behandelt, könnten Sie erzwingen, indem Sie eine der unten stehenden Konstanten als drittes Argument übergeben. Die Typkonstanten, die `bindParam()` kennt, werden in Tabelle 12-2 aufgeführt.

Tabelle 12-2: PDO::PARAM_*-Konstanten

Konstante	Typ
`PDO::PARAM_NULL`	`NULL`
`PDO::PARAM_BOOL`	Boolean
`PDO::PARAM_INT`	Integer
`PDO::PARAM_STR`	String
`PDO::PARAM_LOB`	Large Object

Der Typ `PDO::PARAM_LOB` ist besonders praktisch, weil er dafür sorgt, dass der Parameter als Stream behandelt wird. Er bietet eine effiziente Möglichkeit, den Inhalt von Dateien (oder allem anderen, das als Stream repräsentiert werden kann wie eine entfernte URL) in eine Datenbanktabelle einzufügen. Beispiel 12-21 nutzt `glob()`, um den Inhalt aller Dateien in einem Verzeichnis in eine Datenbanktabelle einzupflegen.

Beispiel 12-21: Dateiinhalte mit PDO::PARAM_LOB in eine Datenbank packen

```
<?php
$st = $db->prepare('INSERT INTO files (path,contents) VALUES (:path,:contents)');
$st->bindParam(':path',$path);
$st->bindParam(':contents',$fp,PDO::PARAM_LOB);
foreach (glob('c:/documents/*.*') as $path) {
    // Ein Datei-Handle erzeugen, das PDO::PARAM_LOB nutzen kann.
    $fp = fopen($path,'r');
    $st->execute();
}
?>
```

Wie man `PDO::PARAM_LOB` effizient einsetzt, ist von der zugrunde liegenden Datenbank abhängig. Beispielsweise muss Ihre Abfrage bei Oracle ein leeres LOB-Handle erzeugen und in einer Transaktion sein. Das »Inserting an image into a database: Oracle«-Beispiel in der PDO-Dokumentation *http://www.php.net/class.pdo* zeigt, wie Sie das machen.

Siehe auch

Die Dokumentationen zu `PDO::prepare()` unter *http://www.php.net/PDO.prepare*, zu `PDOStatement::execute()` unter *http://www.php.net/PDOStatement.execute*, zu `PDOStatement::bindParam()` unter *http://www.php.net/PDOStatement.bindParam* und zu `PDO::PARAM_LOB` unter *http://www.php.net/pdo.lobs*.

12.9 Ermitteln, wie viele Zeilen eine Abfrage geliefert hat

Problem

Sie möchten wissen, wie viele Zeilen eine `SELECT`-Abfrage lieferte oder wie viele Zeilen von einer `INSERT`-, `UPDATE`- oder `DELETE`-Abfrage geändert wurden.

Lösung

Wenn Sie eine `INSERT`-, `UPDATE`- oder `DELETE`-Abfrage mit `PDO::exec()` ausführen, ist der Rückgabewert von `exec()` die Anzahl der geänderten Zeilen.

Führen Sie ein `INSERT`, `UPDATE` oder `DELETE` mit `PDO::prepare()` und `PDOStatement::execute()` aus, können Sie `PDOStatement::rowCount()` aufrufen, um die Anzahl der geänderten Zeilen zu ermitteln, wie Sie es in Beispiel 12-22 sehen.

Beispiel 12-22: Zeilen mit rowCount() zählen

```
<?php
$st = $db->prepare('DELETE FROM family WHERE name LIKE ?');
$st->execute(array('Fredo'));
```

Beispiel 12-22: Zeilen mit rowCount() zählen (Fortsetzung)

```
print "Gelöschte Zeilen: " . $st->rowCount();
$st->execute(array('Sonny'));
print "Gelöschte Zeilen: " . $st->rowCount();
$st->execute(array('Luca Brasi'));
print "Gelöschte Zeilen: " . $st->rowCount();
?>
```

Beim Ausführen einer SELECT-Anweisung gibt es nur ein absolut sicheres Verfahren, um zu ermitteln, wie viele Zeilen zurückgeliefert wurden: Rufen Sie alle Zeilen mit `fetchAll()` ab und zählen Sie dann alle Zeilen, wie Sie es in Beispiel 12-23 sehen.

Beispiel 12-23: Die Zeilen aus einem SELECT zählen

```
<?php
$st = $db->query('SELECT symbol,planet FROM zodiac');
$all= $st->fetchAll(PDO::FETCH_COLUMN, 1);
print "Abgerufene ". count($all) . " Zeilen";
?>
```

Diskussion

Einige Datenbank-Backends liefern PDO Informationen zur Anzahl der von einem SELECT abgerufenen Zeilen, und unter diesen Umständen kann `rowCount()` auch dann funktionieren. Da das aber nicht alle tun, ist es nicht ratsam, sich auf dieses Verhalten zu verlassen.

Es kann allerdings ineffizient sein, alles in einer großen Ergebnismenge abzurufen. Eine Alternative ist, die Datenbank mit der Funktion `COUNT(*)` die Zeilen in einer Ergebnismenge zählen zu lassen. Nutzen Sie die WHERE-Klausel, die Sie auch in der vollständigen Abfrage nutzen, aber fordern Sie SELECT auf, statt einer Feldliste `COUNT(*)` zu liefern.

Siehe auch

Die Dokumentationen zu `PDOStatement::rowCount` unter *http://www.php.net/PDOStatement.rowCount* und zu `PDO::exec()` unter *http://www.php.net/pdo.exec*.

12.10 Anführungszeichen maskieren

Problem

Sie müssen Text- oder Binärdaten für eine Abfrage sicher machen.

Lösung

Schreiben Sie alle Ihre Abfragen mit Platzhaltern, damit `prepare()` und `execute()` die Strings für Sie maskieren können. Rezept 12.8 zeigte die verschiedenen Arten, Platzhalter zu nutzen.

Müssen Sie das Maskieren selbst vornehmen, müssen Sie die Methode PDO::quote( ) nutzen. Eine der seltenen Gelegenheiten, bei denen Sie dazu gezwungen sein könnten, ist das Maskieren von SQL-Jokerzeichen, die aus Benutzereingaben stammen, wie Sie es in Beispiel 12-24 sehen.

Beispiel 12-24: Manuelles Maskieren

```
<?php
$safe = $db->quote($_GET['searchTerm']);
$safe = strtr($safe,array('_' => '\_', '%' => '\%'));
$st = $db->query("SELECT * FROM zodiac WHERE planet LIKE $safe");
?>
```

Diskussion

Die Methode PDO::quote( ) sichert, dass Text- und Binärdaten angemessen maskiert werden. Sie müssen aber ebenfalls dafür sorgen, dass die SQL-Jokerzeichen % und _ maskiert werden, damit SELECT-Anweisungen, die den LIKE-Operator nutzen, die richtigen Ergebnisse liefern. Wenn $_GET['searchTerm'] auf Melm% gesetzt wäre und Beispiel 12-24 strtr( ) nicht aufgerufen hätte, dann lieferte die Abfrage Zeilen, in denen planet gleich Melmac, Melmacko, Melmacedonia oder irgendetwas anderes wäre, das mit Melm beginnt.

Da % das SQL-Jokerzeichen ist, das »finde eine beliebige Anzahl von Zeichen« (wie * beim Shell-Globbing) sagt, und _ das SQL-Jokerzeichen ist, das »finde ein Zeichen« (wie ? beim Shell-Globbing) sagt, müssen auch diese Zeichen mit einem Backslash maskiert werden.

strtr( ) muss nach PDO::quote( ) aufgerufen werden, da PDO::quote( ) ansonsten die Backslashs maskieren würde, die strtr( ) einfügt. Wird erst PDO::quote( ) aufgerufen, wird Melm_ zu Melm_, und das interpretiert die Datenbank als »die Zeichenfolge M e l m und dann ein Unterstrich«. Wird PDO::quote( ) nach strtr( ) aufgerufen, wird Melm_ zu Melm_. Das wird von der Datenbank als »die Zeichenfolge Melm, dann ein Backslash und dann ein beliebiges anderes Zeichen« interpretiert. Und genau das würde auch passieren, wenn wir SQL-Jokerzeichen maskierten und die resultierenden Werte dann als gebundene Parameter verwendeten.

Platzhalterwerte werden auch maskiert, wenn magic_quotes_gpc oder magic_quotes_runtime eingeschaltet ist. Gleiches passiert, wenn Sie bei eingeschalteten magischen Maskierungen PDO::quote( ) auf einem Wert aufrufen, da der Wert bereits maskiert wurde. Maximale Portabilität erreichen Sie, indem Sie die durch magic_quotes_* eingefügten Backslashs entfernen, wenn Sie eine Abfrage mit Platzhalten verwenden oder PDO::quote( ) aufrufen. Beispiel 12-25 zeigt, wie man das macht.

Beispiel 12-25: Prüfen, ob magisch maskiert wird

```
<?php
// Auch das Verhalten von magic_quotes_sybase kann sich darauf auswirken.
if (get_magic_quotes_gpc() && (! ini_get('magic_quotes_sybase'))) {
```

Beispiel 12-25: Prüfen, ob magisch maskiert wird (Fortsetzung)

```
    $fruit = stripslashes($_GET['fruit']);
} else {
    $fruit = $_GET['fruit'];
}
$st = $db->prepare('UPDATE orchard SET trees = trees - 1 WHERE fruit = ?');
$st->execute(array($fruit));
?>
```

Wenn Sie den Server selbst verwalten, sollten Sie magische Maskierungen ausschalten, da Sie sich das Leben damit erheblich vereinfachen. Aber wollen Sie versuchen, so portablen Code wie möglich zu schreiben, der auch in einer Umgebung läuft, die Sie nicht steuern können, müssen Sie dieses Problem im Griff haben.

Siehe auch

Die Dokumentationen zu `PDO::quote()` unter *http://www.php.net/PDO.quote* und zu Magic-Quotes unter *http://www.php.net/manual/en/ref.info.php#ini.magic-quotes-gpc*.

12.11 Debugging-Informationen und Fehler protokollieren

Problem

Sie möchten auf Informationen zugreifen, die Ihnen beim Debugging von Datenbankfehlern helfen. Sie möchten beispielsweise die Fehlermeldung sehen, die die Datenbank zu einer fehlgeschlagenen Abfrage liefert.

Lösung

Nutzen Sie nach einem Vorgang `PDO::errorCode()` oder `PDOStatement::errorCode()`, um einen Fehlercode zu erhalten, wenn der Vorgang fehlschlägt. Die verwandte Methode `errorInfo()` liefert ausführlichere Informationen zum Fehler. Beispiel 12-26 verarbeitet den Fehler, der daraus entsteht, dass auf eine nicht bestehende Tabelle zugegriffen wird.

Beispiel 12-26: Fehlerinformationen ausgeben

```
<?php
$st = $db->prepare('SELECT * FROM imaginary_table');
if (! $st) {
    $error = $db->errorInfo();
    print "Problem ({$error[2]})";
}
?>
```

Diskussion

Die `errorCode()`-Methode liefert einen fünfstelligen Fehlercode. PDO nutzt die SQL 92 SQLSTATE-Fehlercodes. Gemäß diesem Standard heißt `00000` »kein Fehler«. Ein Aufruf von `errorCode()`, der `00000` liefert, zeigt also Erfolg an.

Die Methode `errorInfo()` liefert ein Array mit drei Elementen. Das erste Element enthält den fünfstelligen SQLSTATE-Code (genau das, was auch `errorCode()` liefert). Das zweite Element ist ein datenbankspezifischer Fehlercode. Das dritte Element ist eine datenbankspezifische Fehlermeldung.

Achten Sie darauf, dass Sie `errorCode()` oder `errorInfo()` auf dem Objekt aufrufen, auf dem Sie auch die Methode aufriefen, die Sie auf einen Fehler prüfen wollen. In Beispiel 12-26 wird `prepare()` auf dem `PDO`-Objekt aufgerufen, `errorInfo()` also ebenfalls. Wenn Sie prüfen wollen, ob ein `fetch()`-Aufruf auf einem `PDOStatement`-Objekt erfolgreich war, müssen Sie `errorCode()` oder `errorInfo()` auf dem `PDOStatement`-Objekt aufrufen.

Eine Ausnahme ist die Erstellung eines neuen `PDO`-Objekts. Schlägt diese fehl, löst PDO eine Exception aus, weil es dann kein Objekt gibt, auf dem man `errorCode()` oder `errorInfo()` aufrufen könnte. Die Meldung in der Exception führt aus, warum der Verbindungsversuch fehlschlug.

Wenn PDO bei jedem Fehler eine Exception auslösen soll, können Sie auf Ihrem `PDO`-Objekt, nachdem Sie es erstellt haben, `setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION)` aufrufen. Auf diese Weise können Sie Datenbankprobleme einheitlich behandeln, statt Ihren Code mit wiederholten Aufrufen von `errorCode()` und `errorInfo()` zu überfrachten. Beispiel 12-27 führt eine Folge von Datenbankoperationen aus, die in einem `try/catch`-Block geschachtelt sind.

Beispiel 12-27: Datenbank-Exceptions abfangen

```
<?php
try {
    $db = new PDO('sqlite:/usr/local/zodiac.db');
    // Alle DB-Fehler sollen Exceptions auslösen.
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $st = $db->prepare('SELECT * FROM zodiac');
    $st->execute();
    while ($row = $st->fetch(PDO::FETCH_NUM)) {
        print implode(',',$row). "<br/>\n";
    }
} catch (Exception $e) {
    print "Databankfehler: " . $e->getMessage();
}
?>
```

Auch in Transaktionen ist es hilfreich, PDO-Fehler als Exceptions zu behandeln. Taucht, nachdem eine Transaktion gestartet wurde, ein Problem mit einer Abfrage auf, können Sie leicht die Transaktion zurückrollen, wenn Sie die Exception verarbeiten.

Ähnlich dem Exception-Fehlermodus gibt es einen »Warnung«-Fehlermodus. `setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING)` sagt PDO, dass Warnungen abgesetzt werden sollen, wenn ein Datenbankfehler auftritt. Ziehen Sie es vor, statt mit Exceptions mit gewöhnlichen PHP-Fehlern zu arbeiten, ist das der Fehlermodus für Sie. Richten Sie mit `set_error_handler()` einen benutzerdefinierten Fehler-Handler ein, der Ereignisse der Stufe `E_WARNING` verarbeitet, können Sie Ihre Datenbankprobleme in diesem Fehler-Handler verarbeiten.

Unabhängig davon, mit welchem Fehlermodus Sie arbeiten, löst PDO eine Exception aus, wenn die Erstellung des `PDO`-Objekts fehlschlägt. Wenn man mit PDO arbeitet, ist es äußerst ratsam, mit `set_exception_handler()` einen Default-Exception-Handler einzurichten. Gibt es diesen nicht, führt eine nicht abgefangene Exception dazu, dass ein vollständiger Stacktrace ausgegeben wird, falls `display_errors` eingeschaltet wird. Wird beim Versuch, eine Verbindung mit der Datenbank herzustellen, eine Exception ausgelöst, kann dieser Stacktrace schützenswerte Informationen einschließlich der Berechtigungen für den Datenbankzugriff enthalten.

Siehe auch

Die Dokumentationen zu `PDO::errorCode()` unter *http://www.php.net/PDO.errorCode*, zu `PDO::errorInfo()` unter *http://www.php.net/PDO.errorInfo*, zu `PDOStatement::errorCode()` unter *http://www.php.net/PDOStatement.errorCode*, zu `PDOStatement::errorInfo()` unter *http://www.php.net/PDOStatement.errorInfo*, zu `set_exception_handler()` unter *http://www.php.net/manual/en/function.set-exception-handler* und zu `set_error_handler()` unter *http://www.php.net/set-error-handler*. Eine Liste der SQL 92 SQLSTATE-Fehlercodes, die PDO kennt, finden Sie unter *http://cvs.php.net/viewvc.cgi/php-src/ext/pdo/pdo_sqlstate.c?view=markup*, einige Datenbank-Backends können aber auch andere Fehler melden als die dort aufgeführten.

12.12 Eindeutige Identifikationsnummern erstellen

Problem

Sie möchten Benutzern, Waren oder anderen Objekten eindeutige Identifikationsnummern verleihen, wenn sie in die Datenbank eingefügt werden.

Lösung

Nutzen Sie die Funktion `uniqid()`, um eine Identifikationsnummer zu erstellen. Die Zeichen, die in dieser ID verwendet werden, können Sie einschränken, indem Sie die ID mit der Funktion `md5()` verarbeiten, die einen String liefert, der nur Ziffern und die Buchstaben `a` bis `f` enthält. Beispiel 12-28 erstellt IDs unter Verwendung beider Techniken.

Beispiel 12-28: Eindeutige IDs erstellen

```
<?php
$st = $db->prepare('INSERT INTO users (id, name) VALUES (?,?)');
$st->execute(array(uniqid(), 'Jacob'));
$st->execute(array(md5(uniqid()), 'Ruby'));
?>
```

Sie können auch eine datenbankspezifische Methode erstellen, um die Datenbank die ID generieren zu lassen. Beispielsweise unterstützen SQLite 3 und MySQL `AUTOINCREMENT`-Spalten, die einer Spalte automatisch wachsende Integer-Werte zuweisen, wenn Zeilen eingefügt werden.

Diskussion

`uniqid()` nutzt die aktuelle Zeit (in Mikrosekunden) und eine Zufallszahl, um einen String zu generieren, der äußerst schwer zu erraten wäre. `md5()` berechnet einen Hash-Wert für das übergebene Argument. Die Funktion verleiht der ID keine Zufälligkeit, schränkt aber die Zeichen ein, die in ihr erscheinen. Die Ergebnisse, die `md5()` liefert, enthalten keine Interpunktionszeichen – Sie müssen sich also keine Gedanken über Maskierungsfragen machen. Außerdem kann man allein mit den ersten sechs Buchstaben des Alphabets keine hässlichen oder unanständigen Wörter buchstabieren.

Wenn Sie die Verantwortung für die Erstellung der IDs lieber der Datenbank geben möchten, müssen Sie bei der Erstellung der Tabelle geeignete Syntaxformen verwenden. Beispiel 12-29 zeigt, wie unter SQLite eine Tabelle mit einer Spalte erstellt wird, die einen automatisch inkrementierten Integer-Wert erhält, wenn eine neue Zeile eingefügt wird.

Beispiel 12-29: Unter SQLite eine Autoinkrement-Spalte erstellen

```
<?php
// Der Typ INTEGER PRIMARY KEY AUTOINCREMENT sagt SQLite,
// dass aufsteigende IDs zugewiesen werden sollen.
$db->exec(<<<_SQL_
  CREATE TABLE users (
    id  INTEGER PRIMARY KEY AUTOINCREMENT,
    name VARCHAR(255)
  )
_SQL_
);

// In 'id' muss kein Wert eingefügt werden - SQLite weist ihn zu.
$st = $db->prepare('INSERT INTO users (name) VALUES (?)');

// Diesen Zeilen werden 'id'-Werte zugewiesen.
foreach (array('Jacob','Ruby') as $name) {
    $st->execute(array($name));
}
?>
```

Beispiel 12-30 zeigt das Gleiche für MySQL.

Beispiel 12-30: Unter MySQL eine Autoinkrement-Spalte erstellen

```
<?php
// AUTO_INCREMENT sagt MySQL, dass der Spalte aufsteigende IDs zugewiesen werden sollen.
// Die Spalte muss der PRIMARY KEY sein.
$db->exec(<<<_SQL_
  CREATE TABLE users (
    id  INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(255),
    PRIMARY KEY(id)
  )
_SQL_
);

// In 'id' muss kein Wert eingefügt werden - MySQL weist ihn zu.
$st = $db->prepare('INSERT INTO users (name) VALUES (?)');

// Diesen Zeilen werden 'id'-Werte zugewiesen.
foreach (array('Jacob','Ruby') as $name) {
    $st->execute(array($name));
}
?>
```

Wenn die Datenbank automatisch ID-Werte erzeugt, können sie mit der Methode `PDO::lastInsertId()` abgerufen werden. Rufen Sie auf Ihrem `PDO`-Objekt `lastInsertId()` auf, um für die zuletzt eingefügte Zeile automatisch generierte IDs zu erhalten. Bei einigen Datenbank-Backends können Sie `lastInsertId()` auch einen Sequenznamen übergeben, um den letzten Wert für diese Sequenz zu erhalten.

Siehe auch

Die Dokumentationen zu `uniqid()` unter *http://www.php.net/manual/en/function.uniqid*, zu `md5()` unter *http://www.php.net/manual/en/function.md5*, zu `PDO::lastInsertId()` unter *http://www.php.net/PDO.lastInsertId*, zu SQLite und `AUTOINCREMENT` unter *http://www.sqlite.org/faq.html#q1* sowie zu MySQL und `AUTO_INCREMENT` unter *http://dev.mysql.com/doc/mysql/en/example-auto-increment.html*.

12.13 Abfragen dynamisch aufbauen

Problem

Sie möchten eine `INSERT`- oder `UPDATE`-Anweisung auf Basis eines Arrays mit Feldnamen aufbauen. Nehmen wir beispielsweise an, Sie wollen einen neuen Benutzer in eine Datenbank einfügen. Statt die Felder mit den Benutzerinformationen (wie Benutzername, E-Mail-Adresse, Anschrift, Geburtstag usw.) festzuschreiben, stecken Sie die Feldnamen in

ein Array und nutzen dieses Array dann, um die Abfrage aufzubauen. Das vereinfacht die Wartung, insbesondere wenn Sie mit der gleichen Gruppe von Feldern bedingte INSERT- oder UPDATE-Anweisungen ausführen müssen.

Lösung

Um eine UPDATE-Abfrage zu bilden, bauen Sie ein Array mit Feld/Wert-Paaren auf und nutzen dann implode( ), um die Elemente, wie in Beispiel 12-31 gezeigt, zusammenzufügen.

Beispiel 12-31: Eine UPDATE-Anweisung erstellen

```
<?php
// Eine Liste mit Feldnamen.
$fields = array('symbol','planet','element');

$update_fields = array();
$update_values = array();
foreach ($fields as $field) {
    $update_fields[] = "$field = ?";
    // Gehen wir davon aus, die Daten kämen aus einem Formular.
    $update_values[] = $_POST[$field];
}

$st = $db->prepare("UPDATE zodiac SET " .
                   implode(',', $update_fields) .
                   'WHERE sign = ?');

// Dem Array mit den Werten 'sign' hinzufügen.
$update_values[] = $_GET['sign'];

// Die Abfrage ausführen.
$st->execute($update_values);
?>
```

Bei einer INSERT-Abfrage machen Sie das Gleiche, nur die SQL-Syntax sieht etwas anders aus, wie Beispiel 12-32 zeigt.

Beispiel 12-32: Eine INSERT-Abfrage aufbauen

```
<?php
// Eine Liste mit Feldnamen.
$fields = array('symbol','planet','element');
$placeholders = array();
$values = array();
foreach ($fields as $field) {
    // Ein Platzhalter pro Feld.
    $placeholders[] = '?';
    // Gehen wir davon aus, die Daten kämen aus einem Formular.
    $values[] = $_POST[$field];
}
```

Beispiel 12-32: Eine INSERT-Abfrage aufbauen (Fortsetzung)

```
$st = $db->prepare('INSERT INTO zodiac (' .
                   implode(',',$fields) .
                   ') VALUES (' .
                   implode(',', $placeholders) .
                   ')');
// Die Abfrage ausführen.
$st->execute($values);
?>
```

Diskussion

Platzhalter machen solche Sachen zu einem Kinderspiel. Weil sie sich darum kümmern, dass die übergebenen Daten maskiert werden, können Sie vom Benutzer eingesandte Daten ganz leicht in dynamisch aufgebaute Abfragen packen.

Wenn Sie Sequenz-generierte Integer als Primärschlüssel nutzen, können Sie die beiden Techniken zum Abfrageaufbau zu einer Funktion kombinieren. Diese Funktion ermittelt, ob ein Datensatz besteht, und generiert dann die richtige Abfrage, einschließlich einer ID. Die Funktion `pc_build_query( )` in Beispiel 12-33 zeigt, wie das geht.

Beispiel 12-33: pc_build_query()

```
<?php
function pc_build_query($db,$key_field,$fields,$table) {
    $values = array();
    if (! empty($_POST[$key_field])) {
        $update_fields = array();
        foreach ($fields as $field) {
            $update_fields[] = "$field = ?";
            // Gehen wir davon aus, die Daten kämen aus einem Formular.
            $values[] = $_POST[$field];
        }
        // Den Wert des Schlüsselfelds dem Array $values hinzufügen.
        $values[] = $_POST[$key_field];
        $st = $db->prepare("UPDATE $table SET " .
                   implode(',', $update_fields) .
                   "WHERE $key_field = ?");
    } else {
        // Beginnen Sie bei den Werten mit einer eindeutigen ID
        // oder nutzen Sie NULL, wenn Ihre DB diesen Wert generiert.
        $values[] = md5(uniqid());
        $placeholders = array('?');
        foreach ($fields as $field) {
            // Ein Platzhalter pro Feld.
            $placeholders[] = '?';
            // Gehen wir davon aus, die Daten kämen aus einem Formular.
            $values[] = $_POST[$field];
        }
        $st = $db->prepare("INSERT INTO $table ($key_field," .
                           implode(',',$fields) . ') VALUES ('.
                           implode(',',$placeholders) .')');
```

Beispiel 12-33: pc_build_query() (Fortsetzung)

```
    }
    $st->execute($values);
    return $st;
}
?>
```

Mithilfe dieser Funktion erstellen Sie eine einfache Seite, über die alle Daten in der Tabelle zodiac bearbeitet werden können. Eine solche sehen Sie in Beispiel 12-34.

Beispiel 12-34: Eine einfache Datensatz hinzufügen/bearbeiten-Seite

```
<?php
$db = new PDO('sqlite:/usr/local/data/zodiac.db');
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$fields = array('sign','symbol','planet','element',
                'start_month','start_day','end_month','end_day');

$cmd = isset($_REQUEST['cmd']) ? $_REQUEST['cmd'] : 'show';

switch ($cmd) {
 case 'edit':
 try {
    $st = $db->prepare('SELECT ' . implode(',',$fields) .
                       ' FROM zodiac WHERE id = ?');
    $st->execute(array($_REQUEST['id']));
    $row = $st->fetch(PDO::FETCH_ASSOC);
 } catch (Exception $e) {
     $row = array();
 }
 case 'add':
     print '<form method="post" action="' .
           htmlentities($_SERVER['PHP_SELF']) . '">';
     print '<input type="hidden" name="cmd" value="save">';
     print '<table>';
     if ('edit' == $_REQUEST['cmd']) {
         printf('<input type="hidden" name="id" value="%d">',
                $_REQUEST['id']);
     }
     foreach ($fields as $field) {
         if ('edit' == $_REQUEST['cmd']) {
             $value = htmlentities($row[$field]);
         } else {
             $value = '';
         }
         printf('<tr><td>%s: </td><td><input type="text" name="%s" value="%s">',
                $field,$field,$value);
         printf('</td></tr>');
     }
     print '<tr><td></td><td><input type="submit" value="Speichern"></td></tr>';
     print '</table></form>';
     break;
 case 'save':
```

Beispiel 12-34: Eine einfache Datensatz hinzufügen/bearbeiten-Seite (Fortsetzung)

```
    try {
      $st = pc_build_query($db,'id',$fields,'zodiac');
      print 'Daten eingefügt.';
    } catch (Exception $e) {
      print "Konnte Daten nicht einfügen: " . htmlentities($e->getMessage());
    }
    print '<hr>';
case 'show':
default:
    $self = htmlentities($_SERVER['PHP_SELF']);
    print '<ul>';
    foreach ($db->query('SELECT id,sign FROM zodiac') as $row) {
        printf('<li> <a href="%s?cmd=edit&id=%s">%s</a>',
               $self,$row['id'],$row['sign']);
    }
    print '<hr><li> <a href="'.$self.'?cmd=add">Neue Daten einfügen</a>';
    print '</ul>';
    break;
}?>
```

Die `switch`-Anweisung steuert auf Basis des Werts von `$_REQUEST['cmd']`, welche Aktion das Programm durchführt. Ist `$_REQUEST['cmd']` `add` oder `edit`, zeigt das Programm ein Formular mit Textfeldern für alle Felder im Array `$fields` an, das dem entspricht, was Abbildung 12-1 zeigt. Ist `$_REQUEST['cmd']` `edit`, werden die Werte für die Zeile mit der übergebenen `$id` aus der Datenbank geladen und als Defaults angezeigt. Ist `$_REQUEST['cmd']` `save`, nutzt das Programm `pc_build_query()`, um eine geeignete Abfrage zu erzeugen, über die entweder neue Daten in die Datenbank eingefügt oder Daten in der Datenbank aktualisiert werden. Nach dem Speichern (oder wenn `$_REQUEST['cmd']` nicht angegeben ist), zeigt das Programm die Liste aller Sternzeichen an, die Sie in Abbildung 12-2 sehen.

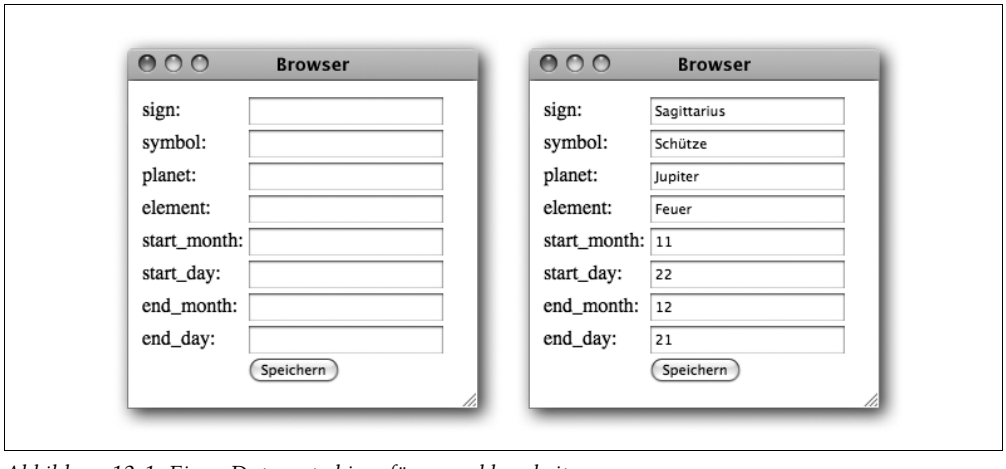


Abbildung 12-1: Einen Datensatz hinzufügen und bearbeiten

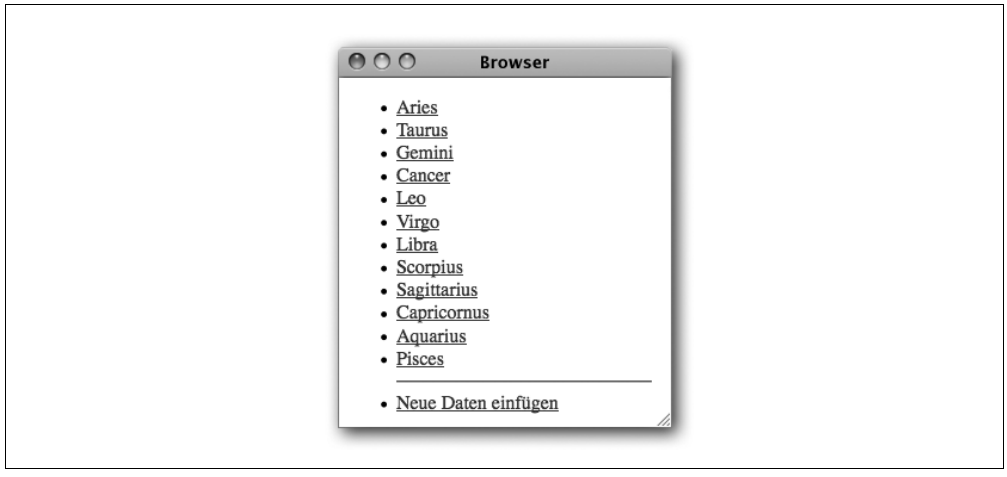


Abbildung 12-2: Eine Datensatzaufstellung

Ob `pc_build_query()` eine `INSERT`- oder eine `UPDATE`-Anweisung aufbaut, hängt davon ab, ob die Abfragevariable `$_REQUEST['id']` vorhanden ist (da `id` in `$key_field` übergeben wird). Ist `$_REQUEST['id']` nicht leer, baut die Funktion eine `UPDATE`-Anweisung auf, die die Zeile mit dieser ID ändert. Ist `$_REQUEST['id']` leer (oder überhaupt nicht gesetzt), generiert die Funktion eine neue ID und nutzt diese in einer `INSERT`-Abfrage, die der Tabelle eine Zeile hinzufügt. Wenn `pc_build_query()` die `AUTOINCREMENT`-Einstellung einer Datenbank respektieren soll, müssen Sie `$values` auf `null` statt auf `md5(uniqid())` setzen.

Siehe auch

Rezept 12.8 mit Informationen zu Platzhaltern, `prepare()` und `execute()`; die Dokumentationen zu `PDO::prepare()` unter *http://www.php.net/PDO.prepare* und zu `PDOStatement::execute()` unter *http://www.php.net/PDOStatement.execute()*.

12.14 Paginierte Links für eine Gruppe von Datensätzen anzeigen

Problem

Sie möchten eine große Datenmenge seitenweise anzeigen und Links für die Navigation durch die Datenmenge bereitstellen.

Lösung

Nutzen Sie die datenbankspezifische Syntax, um nur einen Teil der Zeilen abzurufen, die von Ihrer Abfrage gefunden werden. Beispiel 12-35 zeigt, wie das bei SQLite funktioniert.

Beispiel 12-35: Ergebnismengen auf mehreren Seiten anzeigen mit SQLite

```
<?php
// Fünf Zeilen auswählen und dabei nach den ersten drei beginnen.
foreach ($db->query('SELECT * FROM zodiac ' .
                    'ORDER BY sign LIMIT 5 ' .
                    'OFFSET 3') as $row) {
     // Mit jeder Zeile etwas machen.
}
?>
```

Die Funktionen `pc_indexed_links( )` und `pc_print_link( )` in diesem Rezept helfen Ihnen bei der Ausgabe der Daten für die Seitennavigation. In Beispiel 12-36 sehen Sie, wie man sie einsetzt.

Beispiel 12-36: Ergebnisse seitenweise anzeigen

```
<?php
$offset = isset($_GET['offset']) ? intval($_GET['offset']) : 1;
if (! $offset) { $offset = 1; }
$per_page = 5;
$total = $db->query('SELECT COUNT(*) FROM zodiac')->fetchColumn(0);

$limitedSQL = 'SELECT * FROM zodiac ORDER BY id ' .
              "LIMIT $per_page OFFSET " . ($offset-1);
$lastRowNumber = $offset - 1;

foreach ($db->query($limitedSQL) as $row) {
    $lastRowNumber++;
    print "{$row['sign']}, {$row['symbol']} ({$row['id']}) <br/>\n";
}

pc_indexed_links($total,$offset,$per_page);
print "<br/>";
print "(Zeige $offset - $lastRowNumber von $total)";
?>
```

Diskussion

`pc_print_link( )` wird in Beispiel 12-37 und `pc_indexed_links( )` in Beispiel 12-38 demonstriert.

Beispiel 12-37: pc_print_link( )

```
<?php
function pc_print_link($inactive,$text,$offset='') {
    if ($inactive) {
        print "<span class='inactive'>$text</span>";
    } else {
        print "<span class='active'>".
              "<a href='" . htmlentities($_SERVER['PHP_SELF']) .
              "?offset=$offset'>$text</a></span>";
    }
```

Beispiel 12-37: pc_print_link() (Fortsetzung)

```
}
?>
```

Beispiel 12-38: pc_indexed_links()

```
<?php
function pc_indexed_links($total,$offset,$per_page) {
    $separator = ' | ';

    // "Zurück"-Link ausgeben.
    pc_print_link($offset == 1, '&lt;&lt; Zur&uuml;ck', $offset - $per_page);

    // Alle Gruppen außer der letzten anzeigen.
    for ($start = 1, $end = $per_page;
         $end < $total;
         $start += $per_page, $end += $per_page) {
             print $separator;
             pc_print_link($offset == $start, "$start-$end", $start);
    }

    /* Die letzte Gruppe anzeigen -
     * hier zeigt $start auf das Element am Anfang
     * der letzten Gruppe.
     */

    /* Der Text sollte nur dann einen Bereich enthalten, wenn auf der letzten Seite
     * mehrere Elemente sind. Die letzte Gruppe sollte bei 11 Elementen, die jeweils zu
     * fünf Elementen pro Seite angezeigt werden, also nur "11", nicht "11-11" sagen.
     */
    $end = ($total > $start) ? "-$total" : '';

    print $separator;
    pc_print_link($offset == $start, "$start$end", $start);

    // "Weiter>>"-Link ausgeben.
    print $separator;
    pc_print_link($offset == $start, 'Weiter &gt;&gt;',$offset + $per_page);
}
?>
```

Wenn Sie diese Funktionen einsetzen wollen, müssen Sie die erforderliche Teilmenge der Daten abrufen und dann ausgeben. Verwenden Sie dann `pc_indexed_links()`, um die indizierten Links aufzurufen.

Nachdem die Verbindung zur Datenbank hergestellt wurde, müssen Sie prüfen, ob `$offset` einen geeigneten Wert hat. `$offset` ist der erste Datensatz in der Ergebnismenge, der angezeigt werden soll. Wenn Sie am Anfang der Ergebnismenge beginnen wollen, sollte `$offset` gleich `1` sein. Die Variable `$per_page` ist auf die Anzahl der auf einer Seite anzuzeigenden Datensätze gesetzt, und `$total` ist die Gesamtzahl an Datensätzen. In diesem Beispiel sollen alle Sternzeichen-Datensätze angezeigt werden, `$total` entspricht also der Gesamtzahl aller Zeilen in der gesamten Tabelle.

Die SQL-Abfrage, die die Daten in der erforderlichen Reihenfolge abruft, ist:

```
<?php
$limitedSQL = 'SELECT * FROM zodiac ORDER BY id ' .
"LIMIT $per_page OFFSET " . ($offset-1);
?>
```

Über die Schlüsselwörter `LIMIT` und `OFFSET` sagen Sie SQLite, dass nur eine Teilmenge der gefundenen Zeilen geliefert werden soll.

Die relevanten Zeilen werden mit `$db->query($limitedSQL)` abgerufen, und dann werden die Daten aus jeder Zeile angezeigt. Nach den Zeilen liefert `pc_indexed_links( )` die Navigationslinks. Die Ausgabe, die Sie erhalten, wenn `$offset` nicht (oder auf `1`) gesetzt ist, sehen Sie in Abbildung 12-3.

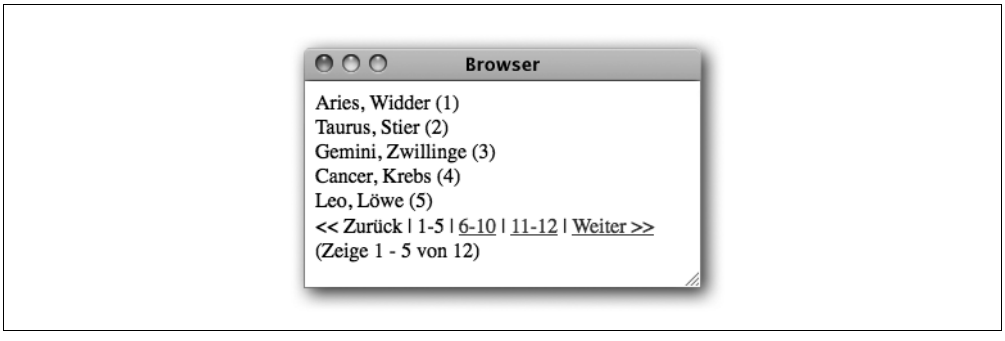


Abbildung 12-3: Ergebnisse mit pc_indexed_links( ) auf mehrere Seiten verteilen

In Abbildung 12-3 sind »6-10«-, »11-12«- und »Weiter«-Links auf die gleiche Seite gesetzt, aber mit jeweils anderem `$offset`-Argument, während »Zurück« und »1-5« deaktiviert sind, da das Links auf den Inhalt wären, der aktuell angezeigt wird.

Siehe auch

Eine Diskussion über Seiteneinteilung im Solar-Framework unter *http://paul-m-jones.com/blog/?p=185* und Informationen zur Seiteneinteilungssyntax bei den unterschiedlichen Datenbanksystemen unter *http://troels.arvin.dk/db/rdbms/#select-limit-offset*.

12.15 Ergebnisse und Abfragen cachen

Problem

Sie möchten möglicherweise ressourcenaufwendige Datenbankabfragen nicht erneut ausführen, wenn sich die Ergebnisse nicht geändert haben.

Lösung

Nutzen Sie das PEAR-Paket *Cache_Lite*, das das Cachen beliebiger Daten zu einem Kinderspiel macht. Cachen Sie hier beispielsweise die Ergebnisse einer SELECT-Abfrage und nutzen Sie den Text der Abfrage als Cache-Schlüssel. Beispiel 12-39 zeigt, wie man mit *Cache_Lite* Abfrageergebnisse zwischenspeichert.

Beispiel 12-39: Abfrageergebnisse cachen

```
<?php
require_once 'Cache/Lite.php';
$opts = array(
   // Wo packen wir die gecachten Daten hin?
   'cacheDir' => 'c:/tmp',
   // Speichern wir im Cache Arrays?
  'automaticSerialization' => true,
  // Wie lange der Kram aufgehoben werden soll, der sich im Cache befindet.
  'lifeTime' => 600 /* ten minutes */);

// Den Cache erstellen.
$cache = new Cache_Lite($opts);

// Mit der Datenbank verbinden.
$db = new PDO('sqlite:c:/data/zodiac.db');

// Die Abfrage und ihre Parameter definieren.
$sql = 'SELECT * FROM zodiac WHERE planet = ?';
$params = array($_GET['planet']);

// Den eindeutigen Cache-Schlüssel abrufen.
$key = cache_key($sql, $params);

// Versuchen, die Ergebnisse aus dem Cache zu lesen.
$results = $cache->get($key);

if ($results === false) {
    // Keine Ergebnisse gefunden, also Abfrage ausführen und Ergebnisse im Cache speichern.
    $st = $db->prepare($sql);
    $st->execute($params);
    $results = $st->fetchAll();
    $cache->save($results);
}

// Ob die Daten aus dem Cache stammen oder nicht, $results enthält sie jetzt.
foreach ($results as $result) {
    print "$result[id]: $result[planet], $result[sign] <br/>\n";
}

function cache_key($sql, $params) {
    return md5($sql .
               implode('|',array_keys($params)) .
               implode('|',$params));
}
?>
```

Diskussion

`Cache_Lite` ist ein allgemeiner, leichtgewichtiger Mechanismus zum Cachen beliebiger Daten. Er nutzt Dateien, um die gecachten Daten zu speichern. Der `Cache_Lite`-Konstruktor erwartet ein Array mit Optionen, die sein Verhalten steuern. Die beiden wichtigsten Optionen in Beispiel 12-39 sind `automaticSerialization` und `cacheDir`. `automaticSerialization` vereinfacht die Speicherung von Arrays im Cache, und `cacheDir` definiert den Ort, an dem die Cache-Dateien gespeichert werden. Achten Sie darauf, dass `cacheDir` mit einem `/` endet.

Der Cache ist einfach eine Schlüssel/Wert-Zuordnung. Dass ein Cache-Schlüssel angegeben wird, der die zu speichernden Daten eindeutig identifiziert, ist unsere Aufgabe – hier nutzen wir dazu die SQL-Abfrage und die an sie gebundenen Parameter. Die Funktion `cache_key` berechnet einen geeigneten Schlüssel. Anschließend prüft Beispiel 12-39, ob sich die Ergebnisse bereits im Cache befinden. Ist das nicht der Fall, wird die Abfrage auf der Datenbank ausgeführt. Dann werden die Ergebnisse für den nächsten Zugriff im Cache gespeichert.

Beachten Sie, dass Sie in den Cache keine `PDO`- oder `PDOStatement`-Objekte stecken können – Sie müssen die Ergebnisse abrufen und diese dann im Cache speichern.

Standardmäßig verbleiben die Einträge für eine Stunde im Cache. Das können Sie anpassen, indem Sie einen anderen Wert (in Sekunden) für die Option `lifeTime` übergeben, wenn Sie ein neues `Cache_Lite`-Objekt erstellen. Übergeben Sie `null`, wenn die Daten nicht automatisch verfallen sollen.

Der Cache wird nicht geändert, wenn die Datenbank durch eine `INSERT`-, `UPDATE`- oder `DELETE`-Abfrage geändert wird. Gibt es gecachte `SELECT`-Anweisungen, die auf Daten verweisen, die in der Datenbank nicht mehr vorhanden sind, müssen Sie explizit alles aus dem Cache entfernen, indem Sie die Methode `Cache_Lite::clean( )` aufrufen. Ein einzelnes Element können Sie aus dem Cache löschen, indem Sie einen Cache-Schlüssel an `Cache_Lite::remove( )` übergeben.

Die Funktion `cache_key( )` aus Beispiel 12-39 berücksichtigt Groß-/Kleinschreibung. Das bedeutet, dass keine Ergebnisse im Cache gefunden werden, wenn sich die Ergebnisse von `SELECT * FROM zodiac` im Cache befinden, Sie aber die Abfrage `SELECT * from zodiac` ausführen. Die Abfrage wird dann also erneut ausgeführt. Ihr Cache wird effizienter genutzt, wenn Sie beim Aufbau Ihrer SQL-Abfragen bei der Groß-/Kleinschreibung, den Leerzeichen und der Feldabfolge konsistent bleiben.

Siehe auch

Die Dokumentation zu `Cache_Lite` unter *http://pear.php.net/manual/en/package.caching.cache-lite.php*.

12.16 An beliebiger Stelle eines Programms auf eine Datenbankverbindung zugreifen

Problem

Sie haben ein Programm mit vielen Funktionen und Klassen und möchten nur eine einzige Datenbankverbindung vorhalten, auf die von allen Stellen des Programms zugegriffen werden kann.

Lösung

Nutzen Sie eine statische Klassenmethode, die die Verbindung herstellt, wenn sie noch nicht existiert, und die die Verbindung zurückliefert (siehe Beispiel 12-40).

Beispiel 12-40: Eine Datenbankverbindung in einer statischen Klassenmethode erstellen

```
<?php
class DBCxn {
    // Mit welchem DSN verbinden?
    public static $dsn = 'sqlite:c:/data/zodiac.db';
    public static $user = null;
    public static $pass = null;
    public static $driverOpts = null;

    // Die interne Variable, die die Verbindung festhält.
    private static $db;
    // Kein Klonen oder Instantiieren erlaubt.
    final private function __construct() { }
    final private function __clone() { }

    public static function get() {
        // Verbinden, wenn noch keine Verbindung besteht.
        if (is_null(self::$db)) {
            self::$db = new PDO(self::$dsn, self::$user, self::$pass,
                                self::$driverOpts);
        }
        // Die Verbindung zurückliefern.
        return self::$db;
    }
}
?>
```

Diskussion

Die in Beispiel 12-40 definierte Methode `DBCxn::get()` macht zwei Dinge: Sie können sie von einer beliebigen Stelle des Programms aufrufen, ohne dass Sie sich über den Geltungsbereich von Variablen den Kopf zerbrechen müssen, und sie verhindert, dass in einem Programm mehrere Verbindungen erstellt werden.

Möchten Sie die Art der Verbindung ändern, die DBCxn::get( ) liefert, müssen Sie nur die Eigenschaften $dsn, $user, $pass und $driverOpts anpassen. Wenn Sie während der Ausführung eines Skripts mehrere Datenbankverbindungen verwalten müssen, können Sie $dsn und $db in Arrays umwandeln und get( ) ein Argument akzeptieren lassen, das die zu verwendende Verbindung identifiziert. Beispiel 12-41 zeigt eine Version von DBCxn, die Zugriff auf drei unterschiedliche Datenbanken bietet.

Beispiel 12-41: Verbindungen zu mehreren Datenbanken verwalten

```
<?php
class DBCxn {
    // Mit welchem DSN verbinden?
    public static $dsn =
       array('zodiac' => 'sqlite:c:/data/zodiac.db',
             'users' => array('mysql:host=db.example.com','monty','7f2iuh'),
             'stats' => array('oci:statistics', 'statsuser','statspass'));

    // Die interne Variable, die die Verbindungen festhält.
    private static $db = array();
    // Klonen und Instantiieren nicht gestattet
    final private function __construct() { }
    final private function __clone() { }

    public static function get($key) {
        if (! isset(self::$dsn[$key])) {
            throw new Exception("Unknown DSN: $key");
        }
        // Verbinden, wenn noch keine Verbindung besteht.
        if (! isset(self::$db[$key])) {
            if (is_array(self::$dsn[$key])) {
                // Die beiden folgenden Zeilen funktionieren nur unter PHP 5.1.3 und höher.
                $c = new ReflectionClass('PDO');
                self::$db[$key] = $c->newInstanceArgs(self::$dsn[$key]);
            } else {
                self::$db[$key] = new PDO(self::$dsn[$key]);
            }
        }
        // Die Verbindung zurückliefern.
        return self::$db[$key];
    }
}
?>
```

In Beispiel 12-41 müssen Sie DBCxn::get( ) einen Schlüssel übergeben, der den Eintrag in $dsn identifiziert, der verwendet werden soll. Der Code in get( ) ist ebenfalls etwas komplizierter, da er mit einer variablen Anzahl von Argumenten für den PDO-Konstruktor umgehen können muss. Einige Datenbanken wie SQLite benötigen nur ein Argument. Andere brauchen eventuell zwei, drei oder vier Argumente. Beispiel 12-41 nutzt deswegen die in PHP 5.1.3 eingeführte Methode ReflectionClass::newInstanceArgs( ), um auf

kompakte Weise einen Konstruktor aufzurufen und die Argumente in einem Array zu übergeben. Wenn Sie eine ältere Version von PHP nutzen, müssen Sie die Aufrufe von `new ReflectionClass('PDO')` und `newInstanceArgs()` mit dem Code in Beispiel 12-42 ersetzen.

Beispiel 12-42: Den PDO-Konstruktor unter älteren PHP-Versionen aufrufen

```
<?php
$args = self::$dsn[$key];
$argCount = count($args);
if ($argCount == 1) {
    self::$db[$key] = new PDO($args[0]);
} else if ($argCount == 2) {
    self::$db[$key] = new PDO($args[0],$args[1]);
} else if ($argCount == 3) {
    self::$db[$key] = new PDO($args[0],$args[1],$args[2]);
} else if ($argCount == 4) {
    self::$db[$key] = new PDO($args[0],$args[1],$args[2],$args[3]);
}
?>
```

Beispiel 12-42 prüft die möglichen Argumentzahlen für den PDO-Konstruktor und ruft den entsprechenden Konstruktor auf.

Siehe auch

Die Dokumentationen zu `PDO::__construct()` unter *http://www.php.net/PDO.__construct* und zu `ReflectionClass::newInstanceArgs()` unter *http://www.php.net/language.oop5.reflection*.

12.17 Programm: Ein Thread-basiertes Forum

Beim Speichern und Abrufen Thread-basierter Nachrichten ist besondere Sorgfalt erforderlich, damit diese in der richtigen Reihenfolge angezeigt werden. Das Suchen der Kinder der einzelnen Nachrichten und der Aufbau eines Baums mit Nachrichtenbeziehungen kann sehr leicht zu einem rekursiven Netz von Abfragen führen. In der Regel betrachten Leser deutlich häufiger eine Liste der Nachrichten und lesen einzelne Nachrichten, als dass sie eigene Nachrichten schreiben. Wenn beim Speichern neuer Nachrichten etwas zusätzliche Arbeit geleistet wird, wird die Abfrage, die eine Liste anzuzeigender Nachrichten abruft, einfacher und erheblich effizienter.

Speichern Sie Nachrichten in einer folgendermaßen strukturierten Tabelle:

```
CREATE TABLE pc_message (
  id INT UNSIGNED NOT NULL,
  posted_on DATETIME NOT NULL,
  author CHAR(255),
```

```
    subject CHAR(255),
    body MEDIUMTEXT,
    thread_id INT UNSIGNED NOT NULL,
    parent_id INT UNSIGNED NOT NULL,
    level INT UNSIGNED NOT NULL,
    thread_pos INT UNSIGNED NOT NULL,
    PRIMARY KEY(id)
);
```

Der Primärschlüssel, `id`, ist ein eindeutiger Integer, der eine bestimmte Nachricht identifiziert. Datum und Uhrzeit der Nachricht werden in `posted_on` gespeichert. `author`, `subject` und `body` sind (Überraschung!) Autor, Thema und Inhalt einer Nachricht. Die verbleibenden vier Felder halten die Thread-Beziehungen unter den Nachrichten fest. Der Integer `thread_id` identifiziert die einzelnen Threads. Alle Nachrichten in einem bestimmten Thread haben die gleiche `thread_id`. Ist eine Nachricht eine Antwort auf eine andere Nachricht, ist `parent_id` die `id` der Nachricht, auf die geantwortet wurde. `level` hält fest, auf der wievielten Antwortstufe eines Threads eine Nachricht steht. Die erste Nachricht in einem Thread hat die Stufe 0. Eine Antwort auf die erste Nachricht hat Stufe 1, eine Antwort auf eine Nachricht der Stufe 1 hat Stufe 2. Mehrere Nachrichten in einem Thread können den gleichen `level` und die gleiche `parent_id` haben. Beginnt beispielsweise jemand einen Thread zu den Vorteilen von BeOS im Vergleich zu CP/M, haben die wütenden Einwürfe der CP/M-Fanschaaren alle die gleiche Stufe und eine `parent_id`, die der `id` der ursprünglichen Nachricht entspricht.

Das letzte Feld, `thread_pos`, ist das Feld, das die einfache Anzeige der Nachrichten erst möglich macht. Bei der Anzeige werden alle Nachrichten in einem Thread anhand ihres `thread_pos`-Werts sortiert.

Hier sind die Regeln für die Berechnung von `thread_pos`:

- Die erste Nachricht in einem Thread hat `thread_pos = 0`.
- Gibt es für eine neue Nachricht `N` in dem Thread noch keine weiteren Nachrichten mit der gleichen Elternnachricht wie `N`, ist der `thread_pos`-Wert von `N` um 1 größer als der `thread_pos`-Wert der Elternnachricht.
- Gibt es für eine neue Nachricht `N` im Thread bereits Nachrichten mit der gleichen Elternnachricht wie `N`, ist der `thread_pos`-Wert von `N` um 1 größer als der größte `thread_pos`-Wert aller Nachrichten mit der gleichen Elternnachricht wie `N`.
- Nachdem der `thread_pos`-Wert der neuen Nachricht `N` ermittelt wurde, wird der `thread_pos`-Wert aller Nachrichten im gleichen Thread, deren `thread_pos` größer oder gleich dem von `N` ist, um 1 erhöht (um für `N` Platz zu schaffen).

Das Forumprogramm *message.php* in Beispiel 12-43 speichert Nachrichten und berechnet den passenden `thread_pos`-Wert. Eine Beispielausgabe sehen Sie in Abbildung 12-4.

Abbildung 12-4: Ein Thread-basiertes Forum

Beispiel 12-43: message.php

```
<?php

$board = new MessageBoard();
$board->go();

class MessageBoard {
    protected $db;
    protected $form_errors = array();
    protected $inTransaction = false;

    public function __construct() {
        set_exception_handler(array($this,'logAndDie'));
        $this->db = new PDO('sqlite:/usr/local/data/message.db');
        $this->db->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
    }

    public function go() {
        // Der Wert von $_REQUEST['cmd'] sagt uns, was wir tun müssen.
        $cmd = isset($_REQUEST['cmd']) ? $_REQUEST['cmd'] : 'show';
        switch ($cmd) {
            case 'read':          // Eine einzelne Nachricht lesen.
              $this->read();
              break;
            case 'post':          // Das Formular zum Schreiben einer Nachricht anzeigen.
              $this->post();
              break;
            case 'save':          // Eine abgeschickte Nachricht speichern.
              if ($this->valid()) { // Wenn die Nachricht gültig ist,
                  $this->save();    // Nachricht speichern
                  $this->show();    // und Nachrichtenliste anzeigen,
              } else {
                  $this->post();    // andernfalls erneut Formular anzeigen.
              }
```

Beispiel 12-43: message.php (Fortsetzung)

```
            break;
        case 'show':          // Standardmäßig eine Nachrichtenliste anzeigen.
        default:
            $this->show();
            break;
    }
}

// save() speichert die Nachrichten in der Datenbank.
protected function save() {

    $parent_id = isset($_REQUEST['parent_id']) ?
                 intval($_REQUEST['parent_id']) : 0;

    // Sichern, dass sich pc_message nicht ändert, während wir damit arbeiten.
    $this->db->beginTransaction();
    $this->inTransaction = true;

    // Ist die Nachricht eine Antwort?
    if ($parent_id) {
        // thread, level und thread_pos der Elternnachricht abrufen.
        $st = $this->db->prepare("SELECT thread_id,level,thread_pos
                            FROM pc_message WHERE id = ?");
        $st->execute(array($parent_id));
        $parent = $st->fetch();

        // Die Stufe einer Antwort ist um 1 höher als die der Elternnachricht.
        $level = $parent['level'] + 1;

        /* Welcher thread_pos ist der höchste in diesem Thread
        unter den Nachrichten mit der gleichen Elternnachricht? */
        $st = $this->db->prepare('SELECT MAX(thread_pos) FROM pc_message
                WHERE thread_id = ? AND parent_id = ?');
        $st->execute(array($parent['thread_id'], $parent_id));
        $thread_pos = $st->fetchColumn(0);

        // Gibt es bereits Antworten zu dieser Elternnachricht?
        if ($thread_pos) {
            // Diese thread_pos kommt nach der größten vorhandenen.
            $thread_pos++;
        } else {
           // Das ist die erste Antwort, sie kommt also gleich hinter der Elternnachricht.
            $thread_pos = $parent['thread_pos'] + 1;
        }

        /* Die thread_pos aller Nachrichten im Thread, die nach dieser kommen,
         um 1 erhöhen. */
        $st = $this->db->prepare('UPDATE pc_message SET thread_pos = thread_pos + 1
                     WHERE thread_id = ? AND thread_pos >= ?');
        $st->execute(array($parent['thread_id'], $thread_pos));
```

Beispiel 12-43: message.php (Fortsetzung)

```
            // Die neue Nachricht soll mit der thread_id der Elternnachricht gespeichert werden.
            $thread_id = $parent['thread_id'];
        } else {
            // Die Nachricht ist keine Antwort, beginnt also einen neuen Thread.
            $thread_id = $this->db->query('SELECT MAX(thread_id) + 1 FROM pc_message')
                          ->fetchColumn(0);
            $level = 0;
            $thread_pos = 0;
        }

        /* Nachricht in Datenbank einfügen. Der Einsatz von prepare() und execute()
        sichert, dass alle Felder ordentlich maskiert werden. */
        $st = $this->db->prepare("INSERT INTO pc_message (id,thread_id,parent_id,
                      thread_pos,posted_on,level,author,subject,body)
                      VALUES (?,?,?,?,?,?,?,?,?)");

        $st->execute(array(null,$thread_id,$parent_id,$thread_pos,
                           date('c'),$level,$_REQUEST['author'],
                           $_REQUEST['subject'],$_REQUEST['body']));

        // Alle Operationen festschreiben.
        $this->db->commit();
        $this->inTransaction = false;
    }

    // show() zeigt eine Liste aller Nachrichten an.
    protected function show() {
        print '<h2>Liste der Nachrichten</h2><p>';

        /* Die Nachrichten nach Thread (thread_id) und ihrer Position
        im Thread (thread_pos) sortieren. */
        $st = $this->db->query("SELECT id,author,subject,LENGTH(body) AS body_length,
                      posted_on,level FROM pc_message
                      ORDER BY thread_id,thread_pos");
        while ($row = $st->fetch()) {
            // Nachrichten mit level > 0 einrücken.
            print str_repeat(' ',4 * $row['level']);
            // Informationen zur Nachricht mit Link zum Lesen ausgeben.
            print "<a href='" . htmlentities($_SERVER['PHP_SELF']) .
            "?cmd=read&amp;id={$row['id']}'>" .
            htmlentities($row['subject']) . '</a> von ' .
            htmlentities($row['author']) . ' @ ' .
            htmlentities($row['posted_on']) .
            " ({$row['body_length']} Bytes) <br/>";
        }

        // Ein Mittel bieten, Nicht-HTML-Nachrichten einzusenden.
        print "<hr/><a href='" .
              htmlentities($_SERVER['PHP_SELF']) .
              "?cmd=post'>Neue Diskussion beginnen</a>";
    }
```

Beispiel 12-43: message.php (Fortsetzung)

```
    // read() zeigt eine einzelne Nachricht an.
    public function read() {

        /* Prüfen, ob die übergebene Nachrichten-ID tatsächlich ein Integer ist und
        eine Nachricht repräsentiert. */
        if (! isset($_REQUEST['id'])) {
            throw new Exception('Keine Nachrichten-ID angegeben');
        }
        $id = intval($_REQUEST['id']);
        $st = $this->db->prepare("SELECT author,subject,body,posted_on
                                  FROM pc_message WHERE id = ?");
        $st->execute(array($id));
        $msg = $st->fetch();
        if (! $msg) {
            throw new Exception('Ungültige Nachrichten-ID');
        }

        /* Vom Benutzer eingegebenes HTML nicht anzeigen, aber Zeilenumbrüche als
        HTML-Zeilenumbrüche wiedergeben. */
        $body = nl2br(htmlentities($msg['body']));

        // Die Nachricht mit Links auf Antwort und Nachrichtenliste anzeigen.
        $self = htmlentities($_SERVER['PHP_SELF']);
        $subject = htmlentities($msg['subject']);
        $author = htmlentities($msg['author']);
        print<<<_HTML_
<h2>$subject</h2>
<h3>von $author</h3>
<p>$body</p>
<hr/>
<a href="$self?cmd=post&parent_id=$id">Reply</a>
<br/>
<a href="$self?cmd=list">Nachrichtenliste</a>
_HTML_;
    }

    // post() zeigt das Formular zum Schreiben von Nachrichten an.
    public function post() {
        $safe =  array();
        foreach (array('author','subject','body') as $field) {
            // Sonderzeichen in den Default-Werten der Felder maskieren.
            if (isset($_POST[$field])) {
                $safe[$field] = htmlentities($_POST[$field]);
            } else {
                $safe[$field] = '';
            }
            // Die Fehlermeldungen in Rot anzeigen.
            if (isset($this->form_errors[$field])) {
                $this->form_errors[$field] = '<span style="color: red">' .
                   $this->form_errors[$field] . '</span><br/>';
            } else {
```

Beispiel 12-43: message.php (Fortsetzung)

```
                $this->form_errors[$field] = '';
            }
        }

        // Ist die Nachricht eine Antwort?
        if (isset($_REQUEST['parent_id']) &&
        $parent_id = intval($_REQUEST['parent_id'])) {

        // Beim Übermitteln des Formulars parent_id mitsenden.
        $parent_field =
            sprintf('<input type="hidden" name="parent_id" value="%d" />',
                    $parent_id);

        // Wurde kein Thema übergeben, Thema der Elternnachricht verwenden.
        if (! strlen($safe['subject'])) {
             $st = $this->db->prepare('SELECT subject FROM pc_message WHERE id = ?');
             $st->execute(array($parent_id));
             $parent_subject = $st->fetchColumn(0);

            /* Dem Thema der Elternnachricht 'Re: ' voranstellen, wenn es eins gibt
               und ihm noch nicht 'Re:' vorangestellt wurde. */
            $safe['subject'] = htmlentities($parent_subject);
            if ($parent_subject && (! preg_match('/^re:/i',$parent_subject))) {
                $safe['subject'] = "Re: {$safe['subject']}";
            }
          }
        } else {
            $parent_field = '';
        }

    // Das Formular mit Fehlern und Default-Werten anzeigen.
    $self = htmlentities($_SERVER['PHP_SELF']);
    print<<<_HTML_
<form method="post" action="$self">
<table>
<tr>
 <td>Ihr Name:</td>
 <td>{$this->form_errors['author']}
     <input type="text" name="author" value="{$safe['author']}" />
</td>
<tr>
 <td>Thema:</td>
 <td>{$this->form_errors['subject']}
     <input type="text" name="subject" value="{$safe['subject']}" />
</td>
<tr>
 <td>Nachricht:</td>
 <td>{$this->form_errors['body']}
     <textarea rows="4" cols="30" wrap="physical"
               name="body">{$safe['body']}</textarea>
</td>
```

Beispiel 12-43: message.php (Fortsetzung)

```
<tr><td colspan="2"><input type="submit" value="Absenden" /></td></tr>
</table>
$parent_field
<input type="hidden" name="cmd" value="save" />
</form>
_HTML_;
    }

    // validate() prüft, ob alle Felder ausgefüllt wurden.
    public function valid() {
        $this->form_errors = array();
        if (! (isset($_POST['author']) && strlen(trim($_POST['author'])))) {
            $this->form_errors['author'] = 'Bitte geben Sie Ihren Namen ein.';
        }
        if (! (isset($_POST['subject']) && strlen(trim($_POST['subject'])))) {
            $this->form_errors['subject'] = 'Bitte geben Sie ein Nachrichtenthema ein.';
        }
        if (! (isset($_POST['body']) && strlen(trim($_POST['body'])))) {
            $this->form_errors['body'] = 'Bitte geben Sie einen Nachrichteninhalt ein.';
        }

        return (count($this->form_errors) == 0);
    }

    public function logAndDie(Exception $e) {
        print 'ERROR: ' . htmlentities($e->getMessage());
        if ($this->db && $this->db->inTransaction) {
            $this->db->rollback();
        }
        exit();
    }
}
?>
```

Damit gleichzeitiger Zugriff kein Problem wird, muss `save()` in der Zeit zwischen dem Anfang der Berechnung der `thread_pos` der neuen Nachricht und dem tatsächlichen Einfügen der Nachricht in die Datenbank ausschließlichen Zugriff auf die `msg`-Tabelle haben. Wir haben die PDO-Methoden `beginTransaction()` und `commit()` eingesetzt, um das zu erreichen. Beachten Sie, dass `logAndDie()`, der Exception-Handler, gegebenenfalls die Transaktion zurückrollt, wenn in der Transaktion ein Fehler auftrat. PDO ruft am Ende eines Skripts immer `rollback()` auf, wenn eine Transaktion gestartet wurde, aber dadurch, dass wir den Aufruf explizit in `logAndDie()` einschließen, wird es für diejenigen, die den Code lesen, klarer, was passiert.

Das Feld `level` kann bei der Anzeige der Nachrichten genutzt werden, um einzuschränken, was von der Datenbank abgerufen wird. Wenn die Threads sehr tief werden, kann damit verhindert werden, dass Seiten zu groß werden. Beispiel 12-44 zeigt, wie man nur die erste Nachricht eines Threads und die Antworten auf diese erste Nachricht anzeigt.

Beispiel 12-44: Beschränkte Thread-Tiefe

```
<?php
$st = $this->db->query(
    "SELECT * FROM pc_message WHERE level <= 1 ORDER BY thread_id,thread_pos");
while ($row = $st->fetch()) {
    // Die einzelnen Nachrichten anzeigen.
}
?>
```

Wollen Sie in Ihre Website ein Forum einbauen, sollten Sie überlegen, eins der existierenden PHP-Forenpakete zu verwenden. Ein beliebtes ist FUDForum (*http://fudforum.org/forum/*), und unter *http://www.zend.com/apps.php?CID=261* wird noch eine Reihe weiterer aufgeführt.

KAPITEL 13

Web-Automatisierung

13.0 Einführung

Normalerweise läuft PHP als Teil eines Webservers und sendet Inhalte an die Browser der Anwender. Auch wenn Sie PHP von der Befehlszeile aus aufrufen, wickelt es im Allgemeinen eine Aufgabe ab und gibt irgendwelche Ergebnisse aus. PHP kann aber auch nützlich sein, wenn es selbst die Rolle des Webbrowsers übernimmt – dabei liest es URLs aus und arbeitet mit deren Inhalten. Die meisten Rezepte in diesem Kapitel handeln davon, wie man URLs abfragt und die Ergebnisse verarbeitet. Allerdings kommen noch einige andere Aufgaben vor wie die Verwendung von Vorlagen (*Templates*) und die Verarbeitung von Serverprotokollen.

Es gibt vier Möglichkeiten, mit PHP eine entfernte URL zu lesen. Welche davon Sie auswählen, hängt davon ab, welche Anforderungen Sie bezüglich Einfachheit, Kontrollierbarkeit und Portabilität stellen. Die vier Möglichkeiten sind: `fopen( )`, `fsockopen( )`, die cURL-Erweiterung und die PEAR-Klasse `HTTP_Request`.

Die Arbeit mit `fopen( )` ist einfach und bequem. Diesen Ansatz behandeln wir in Rezept 13.1. Die Funktion `fopen( )` folgt automatisch Umleitungen (*Redirects*); wenn Sie also mit dieser Funktion das Verzeichnis *http://www.example.com/people* auslesen möchten und der Server Sie an *http://www.example.com/people/* verweist, erhalten Sie den Inhalt der Index-Seite des Verzeichnisses, aber keine Nachricht darüber, dass die URL verlegt worden ist. `fopen( )` funktioniert sowohl mit HTTP als auch mit FTP. Zu den Nachteilen von `fopen( )` gehört: Diese Funktion kann nur mit HTTP-GET-Anfragen umgehen (nicht mit HEAD oder POST), Sie können mit der Anfrage keine zusätzlichen Header oder Cookies versenden, und Sie können mit ihr nur den Response-Body auslesen und keine Response-Header. Zudem erfordert diese Lösung, dass die Konfigurationsdirektive `allow_url_fopen` aktiviert ist.

Wenn Sie `fsockopen( )` verwenden, erfordert dies mehr Arbeit, bietet dafür aber mehr Flexibilität. In Rezept 13.2 setzen wir `fsockopen( )` ein. Nachdem Sie mit `fsockopen( )` ein

Socket geöffnet haben, müssen Sie den passenden HTTP-Request in das Socket schreiben und dann die Antwort lesen und verarbeiten. Auf diese Weise haben Sie die Möglichkeit, der Anfrage Header hinzuzufügen und auf alle Response-Header zuzugreifen. Allerdings benötigen Sie zusätzlichen Code, um die Antwort korrekt zu parsen und die daraus resultierenden Aktionen auszuführen, zum Beispiel Umleitungen zu folgen.

Wenn Sie Zugang zur cURL-Erweiterung oder zur PEAR-Klasse `HTTP_Request` haben, sollten Sie statt `fsockopen( )` jedoch lieber diese verwenden. cURL unterstützt eine Reihe unterschiedlicher Protokolle (darunter HTTPS, das wir in Rezept 13.7 behandeln) und ermöglicht Ihnen den Zugriff auf die Response-Header. In den meisten Rezepten dieses Kapitels setzen wir cURL ein. Damit Sie cURL verwenden können, muss die cURL-Bibliothek installiert sein; sie ist unter *http://curl.haxx.se* erhältlich. Außerdem muss beim PHP-Build die Konfigurationsoption `--with-curl` angegeben worden sein.

Die PEAR-Klasse `HTTP_Request`, die wir in den Rezepten 13.2, 13.3 und 13.4 verwenden, unterstützt zwar kein HTTPS, ermöglicht Ihnen aber auch den Zugriff auf die Header und kann mit jeder HTTP-Methode arbeiten. Wenn dieses PEAR-Modul auf Ihrem System nicht installiert ist, können Sie es von *http://pear.php.net/get/HTTP_Request* herunterladen. Möchten Sie das Modul verwenden, müssen sich nur seine Dateien im `include_path` befinden; daher handelt es sich um eine sehr portable Lösung. Die Rezepte 13.5 und 13.6 zeigen, wie Sie Requests mit einer beliebigen Request-Methode (nicht nur GET oder POST) senden und wie Sie bei gesendeten HTTP-Requests die Möglichkeit eines auftretenden Timeout berücksichtigen.

Rezept 13.8 zeigt Ihnen, wie Sie hinter die Kulissen von HTTP-Anfragen schauen und die Header von Requests und Responses untersuchen. Wenn eine Anfrage, die Sie aus einem Programm heraus starten, nicht die gewünschten Ergebnisse liefert, können Sie die Header überprüfen und erhalten dadurch häufig Hinweise darauf, was nicht funktioniert.

Nachdem Sie den Inhalt einer Webseite in einem Programm eingelesen haben, können Sie mithilfe der Rezepte 13.9 bis 13.13 die Seiteninhalte manipulieren. Rezept 13.9 demonstriert, wie Sie einzelne Wörter in einer Seite mit farbigen Blöcken markieren. Diese Technik ist beispielsweise hilfreich, wenn Sie Suchbegriffe hervorheben möchten. Rezept 13.10 enthält eine Funktion, mit der man alle Links in einer Seite herausfinden kann. Dies ist ein grundlegender Baustein für einen Web-Spider oder einen Link-Checker. Die Konvertierung zwischen einfachen ASCII-Texten und HTML wird in den Rezepten 13.11 und 13.12 behandelt. Rezept 13.13 zeigt Ihnen, wie Sie alle HTML- und PHP-Tags aus einer Webseite entfernen.

Rezept 13.14 behandelt eine gängige Aufgabe der Serververwaltung – die Aufbereitung von Server-Zugriffsprotokollen. Die Rezepte 13.15 und 13.16 befassen sich mit dem Zusammenspiel von PHP und JavaScript. In Rezept 13.15 geht es darum, wie PHP auf Requests antworten kann, die im Browser aus JavaScript-Code heraus gesendet wurden. Hier muss man z.B. ein großes Augenmerk auf Caching und alternative Content-Types legen. Rezept 13.16 liefert ein vollständiges Beispiel für die Integration von PHP und JavaScript unter Verwendung des populären Dojo-Toolkit.

Zwei Beispielprogramme setzen den Link-Extraktor aus Rezept 13.10 ein. Das Programm in Rezept 13.17 durchsucht die Links in einer Seite und meldet, welche von ihnen noch gültig sind, welche umgezogen sind und welche nicht mehr funktionieren. Das Programm in 13.18 generiert einen Bericht über die Aktualität der Links. Es sagt Ihnen, wann eine Seite, auf die ein Link verweist, zuletzt modifiziert oder ob sie an einen anderen Ort verschoben wurde.

13.1 Eine URL mit der GET-Methode abrufen

Problem

Sie möchten den Inhalt einer URL lesen. Beispielsweise möchten Sie Teile einer Webseite in den Inhalt einer anderen Seite einfügen.

Lösung

Lesen Sie die URL, indem Sie sie an `file_get_contents( )` übergeben:

```
$page = file_get_contents('http://www.example.com/robots.txt','r');
```

Oder Sie verwenden die cURL-Erweiterung:

```
$c = curl_init('http://www.example.com/robots.txt');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
$page = curl_exec($c);
curl_close($c);
```

Sie können auch die PEAR-Klasse `HTTP_Request` benutzen:

```
require 'HTTP/Request.php';

$r = new HTTP_Request('http://www.example.com/robots.txt');
$r->sendRequest();
$page = $r->getResponseBody();
```

Diskussion

Wenn Sie eine geschützte Seite lesen wollen, können Sie auch den Benutzernamen und das Passwort in die URL einfügen. Bei dem folgenden Beispiel ist der Benutzername `david` und das Passwort `haxOr`. Hier sehen Sie, wie dies mit `file_get_contents( )` gemacht wird:

```
$page = file_get_contents('http://david:haxOr@www.example.com/secrets.html','r');
```

So funktioniert es mit cURL:

```
$c = curl_init('http://www.example.com/secrets.html');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($c, CURLOPT_USERPWD, 'david:haxOr');
$page = curl_exec($c);
curl_close($c);
```

Und so mit HTTP_Request:

```
$r = new HTTP_Request('http://www.example.com/secrets.html');
$r->setBasicAuth('david','haxOr');
$r->sendRequest();
$page = $r->getResponseBody();
```

Während file_get_contents( ) Umleitungen folgt, die im Response-Header Location angegeben sind, tut HTTP_Request dies nicht. cURL folgt ihnen nur, wenn die Option CURLOPT_FOLLOWLOCATION gesetzt ist:

```
$c = curl_init('http://www.example.com/directory');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($c, CURLOPT_FOLLOWLOCATION, 1);
$page = curl_exec($c);
curl_close($c);
```

cURL kann noch etwas mehr mit der gelesenen Seite machen. Wenn die Option CURLOPT_RETURNTRANSFER gesetzt ist, gibt curl_exec( ) den Inhalt der Seite in einem String zurück:

```
$c = curl_init('http://www.example.com/files.html');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
$page = curl_exec($c);
curl_close($c);
```

Um die Seite in einer Datei zu speichern, öffnen Sie mit fopen( ) eine Datei zum Schreiben und setzen die Option CURLOPT_FILE auf das Handle der Datei:

```
$fh = fopen('lokale-kopie-der-dateien.html','w') or die($php_errormsg);
$c = curl_init('http://www.example.com/files.html');
curl_setopt($c, CURLOPT_FILE, $fh);
curl_exec($c);
curl_close($c);
```

Wollen Sie die cURL-Ressource und den Inhalt der gelesenen Seite an eine Funktion übergeben, setzen Sie die Option CURLOPT_WRITEFUNCTION auf den Namen der Funktion:

```
// URL und Seiteninhalt in einer Datenbank speichern.
function save_page($c,$page) {
    $info = curl_getinfo($c);
    mysql_query("INSERT INTO pages (url,page) VALUES ('" .
                mysql_escape_string($info['url']) . "', '" .
                mysql_escape_string($page) . "')");
}

$c = curl_init('http://www.example.com/files.html');
curl_setopt($c, CURLOPT_WRITEFUNCTION, 'save_page');
curl_exec($c);
curl_close($c);
```

Wenn keine der Optionen CURLOPT_RETURNTRANSFER, CURLOPT_FILE oder CURLOPT_WRITEFUNCTION gesetzt ist, gibt cURL den Inhalt der erhaltenen Seite aus.

Die file_get_contents()-Funktion kann wie auch die Direktiven include und require entfernte Dateien nur dann lesen, wenn die URL-fopen-Wrapper aktiviert sind. Diese sind standardmäßig aktiviert und werden durch die Konfigurationsdirektive allow_url_

fopen gesteuert. Unter Windows ist es bei PHP-Versionen vor 4.4 allerdings selbst dann nicht möglich, entfernte Dateien mit include und require zu lesen, wenn allow_url_fopen auf on steht.

Siehe auch

Rezept 13.2 zum Abrufen einer URL mit der POST-Methode; Rezept 21.3 behandelt das Öffnen entfernter Dateien mit fopen(); die Dokumentationen zu file_get_contents() unter *http://www.php.net/file-get-contents*, zu fopen( ) unter *http://www.php.net/fopen*, include unter *http://www.php.net/include*, curl_init( ) unter *http://www.php.net/curl-init*, curl_setopt( ) unter *http://www.php.net/curl-setopt*, curl_exec( ) unter *http://www.php.net/curl-exec* und curl_close( ) unter *http://www.php.net/curl-close*; die PEAR-Klasse HTTP_Request unter *http://pear.php.net/package/HTTP_Request*.

13.2 Eine URL mit der POST-Methode abrufen

Problem

Sie möchten eine URL mit der POST-Methode auslesen und nicht mit der standardmäßigen GET-Methode. Beispielsweise möchten Sie ein HTML-Formular abschicken.

Lösung

Verwenden Sie file_get_contents() mit einem Stream-Kontext:

```
$body = http_build_query(array("affe" => "onkel", "nashorn" => "tante"));
$contextOptions = array(
    "http" =>
        array (
            "method" => "POST",
            "header" =>
                "Content-type: application/x-www-form-urlencoded\r\n"
                . "Content-length: " . strlen($body) . "\r\n",
            "content" => $body
            )
);
$context = stream_context_create($contextOptions);
$page = file_get_contents("http://www.example.com/submit.php", false, $context);
```

Etwas kürzer: Verwenden Sie die cURL-Erweiterung und setzen Sie die Option CURLOPT_POST:

```
$c = curl_init('http://www.example.com/submit.php');
curl_setopt($c, CURLOPT_POST, 1);
curl_setopt($c, CURLOPT_POSTFIELDS, 'affe=onkel&nashorn=tante');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
$page = curl_exec($c);
curl_close($c);
```

Wenn die cURL-Erweiterung nicht verfügbar ist, können Sie auch die PEAR-Klasse `HTTP_Request` benutzen:

```
require 'HTTP/Request.php';

$r = new HTTP_Request('http://www.example.com/submit.php');
$r->setMethod(HTTP_REQUEST_METHOD_POST);
$r->addPostData('affe','onkel');
$r->addPostData('nashorn','tante');
$r->sendRequest();
$page = $r->getResponseBody();
```

Diskussion

Das Versenden eines POST-Requests erfordert eine besondere Behandlung jedes einzelnen Arguments. Bei einem GET-Request befinden sich diese Argumente im Query-String; bei einem POST-Request kommen sie dagegen in den Body der Anfrage. Außerdem benötigt der Request einen `Content-Length`-Header, der dem Server die Länge des im Request-Body zu erwartenden Inhalts mitteilt.

Es ist insbesondere dann sinnvoll, einen URL mit POST anstelle von GET abzurufen, wenn die URL andernfalls länger als etwa 200 Zeichen werden würde. Die HTTP-Spezifikation 1.1 in RFC 2616 definiert keine Maximallänge für URLs, daher variiert das Verhalten zwischen verschiedenen Web- und Proxyservern. Wenn Sie eine URL mit GET lesen und dabei unerwartete Ergebnisse oder den Status-Code 414 (»Request-URI Too Long«) erhalten, sollten Sie die Anfrage in einen POST-Request umwandeln.

Siehe auch

Rezept 13.1 zum Abrufen einer URL mit der GET-Methode; die Dokumentationen zu `file_get_contents()` unter *http://www.php.net/file-get-contents*, zu `http_build_query()` unter *http://www.php.net/http-build-query*, zu `stream_context_create()` unter *http://www.php.net/stream-context-create*, zu `curl_setopt()` unter *http://www.php.net/curl-setopt* und `fsockopen()` unter *http://www.php.net/fsockopen*; die PEAR-Klasse `HTTP_Request` unter *http://pear.php.net/package-info.php?package=HTTP_Request*; RFC 2616 ist unter *http://www.faqs.org/rfcs/rfc2616.html* verfügbar.

13.3 Eine URL mit Cookies abrufen

Problem

Sie möchten eine Seite abrufen, bei der ein Cookie zusammen mit der Seitenanfrage übersandt werden muss.

Lösung

Verwenden Sie `file_get_contents()` mit einem Stream-Kontext:

```
$contextOptions = array(
    "http" =>
        array ("header"
            => "Cookie: user=ellen; activity=schwimmen\r\n"));
$context = stream_context_create($contextOptions);
$page = file_get_contents("http://www.example.com/braucht-cookies.php", false, $context);
```

Alternativ können Sie die cURL-Erweiterung und die Option `CURLOPT_COOKIE` verwenden:

```
$c = curl_init('http://www.example.com/braucht-cookies.php');
curl_setopt($c, CURLOPT_VERBOSE, 1);
curl_setopt($c, CURLOPT_COOKIE, 'user=ellen; activity=schwimmen');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
$page = curl_exec($c);
curl_close($c);
```

Wenn cURL nicht verfügbar ist, können Sie die Methode `addHeader()` der PEAR-Klasse `HTTP_Request` benutzen:

```
require 'HTTP/Request.php';

$r = new HTTP_Request('http://www.example.com/braucht-cookies.php');
$r->addHeader('Cookie','user=ellen; activity=schwimmen');
$r->sendRequest();
$page = $r->getResponseBody();
```

Diskussion

Ein Cookie wird im Request-Header `Cookie` an den Server übersandt. Bei `file_get_contents()` geht das über eine HTTP-Kontextoption. Die cURL-Erweiterung bietet eine Cookie-spezifische Option, während Sie bei `HTTP_Request` den `Cookie`-Header genau wie jeden anderen Request-Header hinzufügen müssen. Mehrere Cookie-Werte werden in einer durch Semikola getrennten Liste übersandt. Die Beispiele in der Lösung senden zwei Cookies: eines namens `user` mit dem Inhalt `ellen` und eines mit dem Namen `activity` und dem Inhalt `schwimmen`.

Mithilfe des zur cURL-Erweiterung gehörenden Cookie-Jar (»Keksdose«) können Sie eine Seite anfordern, die Cookies setzt, und nachfolgend Anfragen starten, die diese neu gesetzten Cookies enthalten. Bei der ersten Anfrage setzen Sie `CURLOPT_COOKIEJAR` auf den Namen einer Datei, in der die Cookies gespeichert werden sollen. Bei den nachfolgenden Anfragen setzen Sie `CURLOPT_COOKIEFILE` auf denselben Dateinamen; cURL liest dann die Cookies aus der Datei und sendet sie zusammen mit der Anfrage. Dies ist besonders hilfreich bei einer Abfolge von Anfragen, in der sich die erste Anfrage bei einer Site anmeldet, die Session- oder Authentifizierungs-Cookies setzt, und die übrigen Anfragen diese Cookies enthalten müssen, um gültig zu sein:

```
$cookie_jar = tempnam('/tmp','cookie');

// Anmelden.
$c = curl_init('https://bank.example.com/login.php?user=donald&password=b1gmoney$');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($c, CURLOPT_COOKIEJAR, $cookie_jar);
$page = curl_exec($c);
curl_close($c);

// Kontostand abfragen.
$c = curl_init('http://bank.example.com/balance.php?account=checking');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($c, CURLOPT_COOKIEFILE, $cookie_jar);
$page = curl_exec($c);
curl_close($c);

// Einzahlung vornehmen.
$c = curl_init('http://bank.example.com/deposit.php');
curl_setopt($c, CURLOPT_POST, 1);
curl_setopt($c, CURLOPT_POSTFIELDS, 'account=checking&amount=122.44');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($c, CURLOPT_COOKIEFILE, $cookie_jar);
$page = curl_exec($c);
curl_close($c);

// Cookie-Jar entfernen.
unlink($cookie_jar) or die("Löschen von $cookie_jar nicht möglich");
```

Überlegen Sie sich gut, wo Sie das Cookie-Jar speichern. Die Datei muss sich an einem Ort befinden, auf den der Webserver schreibend zugreifen kann. Wenn aber andere Benutzer die Datei lesen können, können sie möglicherweise an die in den Cookies gespeicherten Authentifizierungsangaben gelangen.

Siehe auch

Rezept 13.2 zum Abrufen einer Seite mit POST und Kontext. Die Dokumentationen zu `file_get_contents()` unter *http://www.php.net/file-get-contents,* zu `stream_context_create()` unter *http://www.php.net/stream-context-create*, zu `curl_setopt()` unter *http://www.php.net/curl-setopt*; die PEAR-Klasse `HTTP_Request` unter *http://pear.php.net/package-info.php?package=HTTP_Request*.

13.4 Eine URL mit Headern abrufen

Problem

Sie möchten eine URL auslesen, bei der bestimmte Header zusammen mit der Seitenanfrage gesendet werden müssen.

Lösung

Verwenden Sie `file_get_contents()` mit einem Stream-Kontext:

```
$contextOptions = array(
    "http" =>
        array ("header"
            => "('X-Faktor: 12\r\nMein-Header: Bob\r\n"));
$context = stream_context_create($contextOptions);
$page = file_get_contents("http://www.example.com/spezieller-header.php", false,
$context);
```

Als Alternative können Sie die cURL-Erweiterung und die Option `CURLOPT_HTTPHEADER` verwenden:

```
$c = curl_init('http://www.example.com/spezieller-header.php');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($c, CURLOPT_HTTPHEADER, array('X-Faktor: 12', 'Mein-Header: Bob'));
$page = curl_exec($c);
curl_close($c);
```

Wenn cURL nicht verfügbar ist, verwenden Sie die Methode `addHeader()` in `HTTP_Request`:

```
require 'HTTP/Request.php';

$r = new HTTP_Request('http://www.example.com/spezieller-header.php');
$r->addHeader('X-Faktor',12);
$r->addHeader('Mein-Header','Bob');
$r->sendRequest();
$page = $r->getResponseBody();
```

Diskussion

cURL verfügt über spezielle Optionen, mit denen Sie die Reqest-Header `Referer` und `User-Agent` setzen können, und zwar `CURLOPT_REFERER` und `CURLOPT_USERAGENT`:

```
$c = curl_init('http://www.example.com/submit.php');
curl_setopt($c, CURLOPT_VERBOSE, 1);
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($c, CURLOPT_REFERER, 'http://www.example.com/form.php');
curl_setopt($c, CURLOPT_USERAGENT, 'CURL via PHP');
$page = curl_exec($c);
curl_close($c);
```

Siehe auch

Rezept 13.2 zum Abrufen einer Seite mit POST und Kontext. Rezept 13.14 mit einer Erklärung dafür, warum »referrer« im Zusammenhang mit der Web-Programmierung häufig fälschlich »referer« geschrieben wird; die Dokumentationen zu `file_get_contents()` unter *http://www.php.net/file-get-contents*, zu `stream_context_create()` unter

http://www.php.net/stream-context-create, und zu `curl_setopt()` unter *http://www.php.net/curl-setopt*; die PEAR-Klasse `HTTP_Request` unter *http://pear.php.net/package-info.php?package=HTTP_Request*.

13.5 Eine URL über eine beliebige HTTP-Methode abrufen

Problem

Sie möchten eine URL über eine exotischere Methode als `get` oder `post` anfordern, `put` oder `delete` beispielsweise.

Lösung

Setzen Sie, wie beim Einsatz von `post`, die Stream-Kontextoptionen `method` und `content`, wenn Sie den `http`-Stream nutzen, wie Sie es in Beispiel 13-1 sehen.

Beispiel 13-1: put mit dem http-Stream verwenden

```
<?php
$url = 'http://www.example.com/put.php';
// Der Request-Body in beliebigem Format
$body = '<menu>
 <dish type="appetizer">Hühnersuppe</dish>
 <dish type="main course">Frittiertes Affenhirn</dish>
</menu>';
$options = array('method' => 'PUT', 'content' => $body);
// Den Stream-Kontext erzeugen
$context = stream_context_create(array('http' => $options));
// Den Kontext an file_get_contents() übergeben
print file_get_contents($url, false, $context);
?>
```

Setzen Sie bei cURL die Option `CURLOPT_CUSTOMREQUEST` auf den Methodennamen. Setzen Sie `CURLOPT_POSTFIELDS` auf den Body, um einen Request-Body einzuschließen, wie Sie es in Beispiel 13-2 sehen.

Beispiel 13-2: put mit cURL verwenden

```
<?php
// Der Request-Body in beliebigem Format
$body = '<menu>
 <dish type="appetizer">Hühnersuppe</dish>
 <dish type="main course">Frittiertes Affenhirn</dish>
</menu>';
$c = curl_init($url);
curl_setopt($c, CURLOPT_CUSTOMREQUEST, 'PUT');
```

Beispiel 13-2: put mit cURL verwenden (Fortsetzung)

```
curl_setopt($c, CURLOPT_POSTFIELDS, $body);
curl_setopt($c, CURLOPT_RETURNTRANSFER, true);
$page = curl_exec($c);
curl_close($c);
?>
```

Beispiel 13-3 zeigt, wie Sie put mit einem HTTP_Request einsetzen: Übergeben Sie HTTP_REQUEST_METHOD_PUT an den Konstruktor und rufen Sie setBody( ) mit dem Inhalt des Request-Bodys auf.

Beispiel 13-3: put mit HTTP_Request verwenden

```
<?php
require 'HTTP/Request.php';

$url = 'http://www.example.com/put.php';
$body = '<menu>
 <dish type="appetizer">Chicken Soup</dish>
 <dish type="main course">Fried Monkey Brains</dish>
</menu>';
$r = new HTTP_Request($url);
$r->setMethod(HTTP_REQUEST_METHOD_PUT);
$r->setBody($body);
$page = $r->getResponseBody();
?>
```

Diskussion

In gleichem Maße, wie sich REST-Webservices-API verbreiten, so nimmt auch der Einsatz von HTTP-Requests zu, die die weniger prominenten Götter des Request-Methoden-Pantheons wie put und delete einsetzen.

Die put-Methode wird häufig eingesetzt, um Inhalte von Dateien hochzuladen. cURL bietet drei spezielle Optionen, die Sie dabei unterstützen: CURLOPT_PUT, CURLOPT_INFILE und CURLOPT_INFILESIZE. Wenn Sie mit put und cURL eine Datei hochladen wollen, setzen Sie CURLOPT_PUT auf true, CURLOPT_INFILE auf ein für die hochzuladende Datei geöffnetes Datei-Handle und CURLOPT_INFILESIZE auf die Größe der Datei. Beispiel 13-4 lädt eine Datei mit put hoch.

Beispiel 13-4: Mit cURL und put eine Datei hochladen

```
<?php
$url = 'http://www.example.com/upload.php';
$filename = '/usr/local/data/pictures/piggy.jpg';
$fp = fopen($filename,'r');
$c = curl_init($url);
curl_setopt($c, CURLOPT_PUT, true);
curl_setopt($c, CURLOPT_INFILE, $fp);
curl_setopt($c, CURLOPT_INFILESIZE, filesize($filename));
curl_setopt($c, CURLOPT_RETURNTRANSFER, true);
```

Beispiel 13-4: Mit cURL und put eine Datei hochladen (Fortsetzung)

```
$page = curl_exec($c);
print $page;
curl_close($c);
?>
```

Siehe auch

Dokumentation zu `curl_setopt()` unter *http://www.php.net/curl-setopt* und zu den Stream-Optionen unter *http://www.php.net/wrappers.http*; zur PEAR-Klasse `HTTP_Request` unter *http://pear.php.net/package/HTTP_Request*; Abschnitt 5.1.1 von RFC 2616, der die Request-Methoden behandelt, ist unter *http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5.1.1* verfügbar.

13.6 URL-Anforderung mit Timeout

Problem

Sie möchten eine entfernte URL anfordern, möchten aber nicht zu lange warten, wenn der entfernte Server überlastet oder zu langsam ist.

Lösung

Setzen Sie die Konfigurationsoption `default_socket_timeout`, wenn Sie mit `http`-Stream arbeiten. Beispiel 13-5 wartet nicht länger als 15 Sekunden, dass eine Verbindung mit dem entfernten Server hergestellt wird.

Beispiel 13-5: Mit http-Stream einen Timeout festlegen

```
<?php
// Timeout von 15 Sekunden
ini_set('default_socket_timeout', 15);
$page = file_get_contents('http://slow.example.com/');
```

Beachten Sie, dass sich die Änderung von `default_socket_timeout` auf alle neuen Sockets oder entfernten Verbindungen auswirkt, die während der weiteren Skriptausführung erzeugt werden.

Setzen Sie bei der Arbeit mit cURL die Option `CURLOPT_CONNECTTIMEOUT`, wie in Beispiel 13-6 gezeigt.

Beispiel 13-6: Bei cURL einen Timeout einrichten

```
<?php
$c = curl_init('http://slow.example.com/');
curl_setopt($c, CURLOPT_RETURNTRANSFER, true);
```

Beispiel 13-6: Bei cURL einen Timeout einrichten (Fortsetzung)

```
curl_setopt($c, CURLOPT_CONNECTTIMEOUT, 15);
$page = curl_exec($c);
curl_close($c);
?>
```

Bei `HTTP_Request` setzen Sie das `timeout`-Element in einem Parameter-Array, das an den `HTTP_Request`-Konstruktor übergeben wird, wie Sie es in Beispiel 13-7 sehen.

Beispiel 13-7: Bei HTTP_Request einen Timeout einrichten

```
<?php
require_once 'HTTP/Request.php';
$opts = array('timeout' => 15);
$req = new HTTP_Request('http://slow.example.com/', $opts);
$req->sendRequest();
?>
```

Diskussion

Entfernte Server sind unberechenbare Gesellen. Selbst der robusteste missionkritische Enterprise-Server kann mal außer Dienst sein. Oder ein entfernter Dienst, auf den Sie angewiesen sind, läuft, ist aber aufgrund von Netzwerkproblemen zwischen Ihrem Server und dem entfernten Server nicht dazu in der Lage, Ihre Anfrage zu bearbeiten. Die Zeit zu beschränken, die PHP wartet, bis die Verbindung zu einem entfernten Server hergestellt ist, ist deswegen eine gute Idee, wenn Daten aus den entfernten Quellen beim Aufbau Ihrer Seiten relevant sind.

Alle Techniken, die wir im Lösungsabschnitt skizziert haben, begrenzen die Zeit, die PHP darauf wartet, bis die Verbindung zum entfernten Server steht. Ist die Verbindung einmal hergestellt, helfen Sie Ihnen in Bezug auf die Reaktionszeit nicht mehr. Wenn schnelle Antworten für Sie von entscheidender Bedeutung sind, sollten Sie zusätzlich eine Obergrenze dafür einrichten, wie lange PHP auf den Empfang von Daten von einem bereits verbundenen Socket wartet. Bei einer Stream-Verbindung nutzen Sie dazu die Funktion `stream_set_timeout()`. Dieser Funktion muss eine Stream-Ressource übergeben werden. Sie müssen also mit `fopen()` einen Stream öffnen, was auch bedeutet, das Sie `file_get_contents()` in diesem Fall nicht einsetzen können. Beispiel 13-8 beschränkt den Timeout für Lesevorgänge auf 20 Sekunden.

Beispiel 13-8: Den Lese-Timeout für den http-Stream setzen

```
<?php
$url = 'http://slow.example.com';
$stream = fopen($url, 'r');
stream_set_timeout($stream, 20);
$response_body = stream_get_contents($stream);
?>
```

Bei cURL setzen Sie die Option CURLOPT_TIMEOUT auf die maximale Zeitdauer, die curl_exec() arbeiten soll. Das schließt sowohl den Verbindungs-Timeout, als auch die Zeit zum Lesen des gesamten Antwortinhalts.

Bei HTTP_Request fügen Sie dem an den Konstruktor übergebenen Parameter-Array einen readTimeout-Wert hinzu. Dieser Wert muss ein Array mit zwei Elementen sein, die die Sekunden bzw. Mikrosekunden angeben. Beispiel 13-9 setzt den Lese-Timeout auf 20 Sekunden.

Beispiel 13-9: Einen Lese-Timeout für HTTP_Request setzen

```
<?php
require_once 'HTTP/Request.php';
$opts = array('readTimeout' => array(20,0));
$req = new HTTP_Request('http://slow.example.com/', $opts);
$req->sendRequest();
?>
```

Obwohl das Setzen von Timeouts für Verbindungsherstellung und Lesevorgänge die Leistung verbessern kann, kann es auch zu ruinierten Ausgaben führen. Ihr Skript könnte eine fragmentarische Antwort lesen, bevor der Timeout verstreicht. Wenn Sie Timeouts setzen, sollten Sie die Antwort, die Sie erhalten haben, auf Vollständigkeit prüfen. Alternativ können Sie in Situationen, in denen eine schnelle Seitengenerierung entscheidend ist, externe Daten in einem separaten Prozess abrufen und in einen lokalen Cache schreiben. Ihre Seiten können dann den Cache nutzen und brauchen sich um Timeouts oder unvollständige Antworten nicht mehr zu kümmern.

Siehe auch

Dokumentation zu curl_setopt() unter *http://www.php.net/curl-setopt*, zu stream_set_timeout() unter *http://www.php.net/stream_set_timeout*, zu default_socket_timeout unter *http://www.php.net/filesystem* und zur PEAR-Klasse HTTP_Request unter *http://pear.php.net/package/HTTP_Request*.

13.7 Eine HTTPS-URL abrufen

Problem

Sie möchten eine sichere URL auslesen.

Lösung

Verwenden Sie file_get_contents() mit einer HTTPS-URL:

```
$page = file_get_contents("https://sicher.example.com/kontostand.php");
```

Falls der `https://`-Wrapper nicht zur Verfügung steht, verwenden Sie die cURL-Erweiterung mit einer HTTPS-URL:

```
$c = curl_init('https://sicher.example.com/kontostand.php');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
$page = curl_exec($c);
curl_close($c);
```

Diskussion

Um sichere URLs mit `file_get_contents()` abrufen zu können, muss PHP mit `--with-openssl` kompiliert worden sein. Die cURL-Erweiterung auf eine SSL-Bibliothek wie zum Beispiel OpenSSL zugreifen können. Diese Bibliothek muss beim Build von PHP und der cURL-Erweiterung verfügbar sein. Abgesehen davon, dass diese zusätzliche Bibliothek erforderlich wird, behandelt cURL sichere URLs genau so wie normale. Sie können bei sicheren Anfragen auch die gleichen cURL-Optionen anwenden, zum Beispiel die Request-Methode ändern oder POST-Daten hinzufügen.

Siehe auch

Das OpenSSL-Projekt unter *http://www.openssl.org/*.

13.8 Den Datenaustausch auf HTTP-Ebene debuggen

Problem

Sie möchten einen von einem Browser auf Ihrem Server ausgelösten HTTP-Request sowie die zugehörige HTTP-Response analysieren, weil beispielsweise Ihr Server nicht die erwartete Antwort auf eine bestimmte Anfrage liefert und Sie sich deshalb die Bestandteile des Requests genau ansehen möchten.

Lösung

Für einfache Anfragen können Sie sich über *telnet* mit dem Server verbinden und die Request-Header manuell eingeben:

```
% telnet www.example.com 80
Trying 208.77.188.166...
Connected to www.example.com.
Escape character is '^]'.
GET / HTTP/1.0
Host: www.example.com

HTTP/1.1 200 OK
Date: Sat, 17 Aug 2002 06:10:19 GMT
```

```
Server: Apache/1.3.26 (Unix) PHP/4.2.2 mod_ssl/2.8.9 OpenSSL/0.9.6d
X-Powered-By: PHP/4.2.2
Connection: close
Content-Type: text/html

// ... Body der Seite ...
```

Diskussion

Wenn Sie die Request-Header manuell eingeben, merkt der Server nicht, dass sie von Ihnen geschrieben und nicht von einem Browser übersendet worden sind. Allerdings haben manche Webserver einen Timeout für die Zeit, die sie auf eine Anfrage warten. Es kann daher ganz sinnvoll sein, den Request vorab zu schreiben und in *telnet* einzufügen. Die erste Zeile des Requests enthält die Request-Methode (`GET`), ein Leerzeichen, den von Ihnen gewünschten Pfad (`/`), dann ein weiteres Leerzeichen und schließlich das verwendete Protokoll (`HTTP/1.0`). Die nächste Zeile, der `Host`-Header, nennt dem Server den virtuellen Host für den Fall, dass sich mehrere Hosts dieselbe IP-Adresse teilen. Eine leere Zeile zeigt dem Server das Ende der Anfrage an; er schickt Ihnen daraufhin die Antwort zurück: erst die Header, dann eine Leerzeile und schließlich den Response-Body.

Es kann mühsam sein, den Text in *telnet* einzufügen, und es ist noch schwieriger, auf diese Weise eine Anfrage mit der POST-Methode zu starten. Wenn Sie eine Anfrage mit `HTTP_Request` vornehmen, können Sie die Response-Header und den Response-Body mit den Methoden `getResponseHeader( )` und `getResponseBody( )` ermitteln:

```
require 'HTTP/Request.php';

$r = new HTTP_Request('http://www.example.com/submit.php');
$r->setMethod(HTTP_REQUEST_METHOD_POST);
$r->addPostData('affe','onkel');
$r->sendRequest();

$response_headers = $r->getResponseHeader();
$response_body    = $r->getResponseBody();
```

Um einen bestimmten Response-Header auszulesen, übergeben Sie der Funktion `getResponseHeader( )` den Header-Namen. Wird die Funktion ohne Argumente aufgerufen, gibt `getResponseHeader( )` ein Array zurück, das alle Response-Header enthält. `HTTP_Request` speichert den ausgehenden Request nicht in einer Variablen, aber Sie können ihn rekonstruieren, indem Sie die private Methode `_buildRequest( )` aufrufen:

```
require 'HTTP/Request.php';

$r = new HTTP_Request('http://www.example.com/submit.php');
$r->setMethod(HTTP_REQUEST_METHOD_POST);
$r->addPostData('affe','onkel');

print $r->_buildRequest();
```

Dann wird folgender Request ausgegeben:

```
POST /submit.php HTTP/1.1
User-Agent: PEAR HTTP_Request class ( http://pear.php.net/ )
Content-Type: application/x-www-form-urlencoded
Connection: close
Host: www.example.com
Content-Length: 10

affe=onkel
```

Wenn Sie cURL verwenden, setzen Sie die Option `CURLOPT_HEADER`, um damit die Response-Header in die Ausgabe von `curl_exec( )` aufzunehmen:

```
$c = curl_init('http://www.example.com/submit.php');
curl_setopt($c, CURLOPT_HEADER, 1);
curl_setopt($c, CURLOPT_POST, 1);
curl_setopt($c, CURLOPT_POSTFIELDS, 'affe=onkel&nashorn=tante');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
$response_headers_and_page = curl_exec($c);
curl_close($c);
```

Um die Response-Header direkt in eine Datei zu schreiben, öffnen Sie eine Datei mit `fopen( )` und setzen `CURLOPT_WRITEHEADER` auf das Handle der Datei:

```
$fh = fopen('/tmp/curl-response-headers.txt','w') or die($php_errormsg);
$c = curl_init('http://www.example.com/submit.php');
curl_setopt($c, CURLOPT_POST, 1);
curl_setopt($c, CURLOPT_POSTFIELDS, 'affe=onkel&nashorn=tante');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($c, CURLOPT_WRITEHEADER, $fh);
$page = curl_exec($c);
curl_close($c);
fclose($fh) or die($php_errormsg);
```

Mit der Option `CURLOPT_VERBOSE` des cURL-Moduls veranlassen Sie `curl_exec( )` und `curl_close( )` dazu, Debugging-Informationen einschließlich des Requests auf dem Standard-Fehlerausgang auszugeben:

```
$c = curl_init('http://www.example.com/submit.php');
curl_setopt($c, CURLOPT_VERBOSE, 1);
curl_setopt($c, CURLOPT_POST, 1);
curl_setopt($c, CURLOPT_POSTFIELDS, 'affe=onkel&nashorn=tante');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
$page = curl_exec($c);
curl_close($c);
```

Die Ausgabe sieht dann so aus:

```
* Connected to www.example.com (208.77.188.166)
> POST /submit.php HTTP/1.1
Host: www.example.com
Pragma: no-cache
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Content-Length: 23
```

```
Content-Type: application/x-www-form-urlencoded

affe=onkel&nashorn=tante* Connection #0 left intact
* Closing connection #0
```

Da cURL die Debugging-Information in den Standard-Fehlerausgang und nicht in die Standardausgabe schreibt, ist ein Abfangen per Ausgabe-Pufferung nicht möglich. Allerdings können Sie eine Datei zum Schreiben öffnen, `CURLOUT_STDERR` auf das Datei-Handle setzen und damit die Debugging-Informationen in eine Datei umleiten:

```
$fh = fopen('/tmp/curl.out','w') or die($php_errormsg);
$c = curl_init('http://www.example.com/submit.php');
curl_setopt($c, CURLOPT_VERBOSE, 1);
curl_setopt($c, CURLOPT_POST, 1);
curl_setopt($c, CURLOPT_POSTFIELDS, 'affe=onkel&nashorn=tante');
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($c, CURLOPT_STDERR, $fh);
$page = curl_exec($c);
curl_close($c);
fclose($fh) or die($php_errormsg);
```

Siehe auch

Rezept 10.12 zum Puffern der Ausgabe; die Dokumentation zu `curl_setopt( )` unter *http://www.php.net/curl-setopt*; die PEAR-Klasse `HTTP_Request` unter *http://pear.php.net/package-info.php?package=HTTP_Request*; die Syntax eines HTTP-Requests ist in RFC 2616 definiert und verfügbar unter *http://www.faqs.org/rfcs/rfc2616.html*.

13.9 Eine Webseite mit Markup versehen

Problem

Sie möchten eine Webseite, zum Beispiel mit Suchergebnissen, so anzeigen, dass bestimmte Wörter hervorgehoben sind.

Lösung

Verwenden Sie `preg_replace( )` mit einem Array aus Mustern und Ersetzungstexten:

```
$patterns = array('\bHund\b/', '\bKatze\b');
$replacements = array('<b style="color:black;background-color=#FFFF00">Hund</b>',
                      '<b style='color:black;background-color=#FF9900">Katze</b>');
while ($page) {
    if (preg_match('{^([^<]*)?(</?[^>]+?>)?(.*)$}',$page,$matches)) {
        print preg_replace($patterns,$replacements,$matches[1]);
        print $matches[2];
        $page = $matches[3];
    }
}
```

Diskussion

Der mit `preg_match( )` verwendete reguläre Ausdruck findet eine Übereinstimmung mit einem möglichst langen Text vor einem HTML-Tag, dem HTML-Tag und schließlich dem Rest des Inhalts. Auf den Text vor dem HTML-Tag wird die Hervorhebung angewandt, das HTML-Tag wird ohne Hervorhebung ausgegeben, und der Rest des Inhalts wird nach denselben Übereinstimmungen durchsucht. Damit wird vermieden, dass Wörter, die sich innerhalb von HTML-Tags befinden (z.B. in URLs oder `alt`-Text), hervorgehoben werden und dadurch die Seite nicht mehr korrekt angezeigt wird.

Das folgende Programm liest die URL in `$url` aus und hebt die Wörter im Array `$words` hervor. Wörter werden nicht hervorgehoben, wenn sie Teil eines längeren Worts sind, da die Übereinstimmungen mit dem Perl-kompatiblen Operator `\b` für die Suche nach Wortgrenzen gesucht werden.

```
$colors = array('FFFF00','FF9900','FF0000','FF00FF',
                '99FF33','33FFCC','FF99FF','00CC33');

// Regex-Ausdrücke zum Suchen und Ersetzen bilden.
$patterns = array();
$replacements = array();
for ($i = 0, $j = count($words); $i < $j; $i++) {
    $patterns[$i] = '/\b'.preg_quote($words[$i], '/').'\b/';
    $replacements[$i] = '<b style="color:black;background-color:#' .
                        $colors[$i % 8] .'">' . $words[$i] . '</b>';
}

// Seite einlesen.
$s = file_get_contents($url);

if ($j) {
    while ($s) {
        if (preg_match('{^([^<]*)?(</?[^>]+?>)?(.*)$}s',$s,$matches)) {
            print preg_replace($patterns,$replacements,$matches[1]);
            print $matches[2];
            $s = $matches[3];
        }
    }
} else {
    print $s;
}
```

Siehe auch

Rezept 16.7 mit Informationen zum Herausfiltern von Text innerhalb von HTML-Tags; die Dokumentationen zu `preg_match( )` unter *http://www.php.net/preg-match* und `preg_replace( )` unter *http://www.php.net/preg-replace*.

13.10 Links aus einer HTML-Datei extrahieren

Problem

Sie möchten die innerhalb eines HTML-Dokuments enthaltenen URLs herausziehen.

Lösung

Verwenden Sie die in Beispiel 13-10 dargestellte Funktion `pc_link_extractor( )`.

Beispiel 13-10: pc_link_extractor()

```
function pc_link_extractor($s) {
  $a = array();
  if (preg_match_all('/<a\s+.*?href=[\"\']?([^\"\' >]*)[\"\']?[^>]*>(.*?)<\/a>/i',
                     $s,$matches,PREG_SET_ORDER)) {
    foreach($matches as $match) {
      array_push($a,array($match[1],$match[2]));
    }
  }
  return $a;
}
```

Zum Beispiel:

```
$links = pc_link_extractor($page);
```

Diskussion

Die Funktion `pc_link_extractor( )` gibt ein Array zurück. Jedes Element dieses Arrays ist selbst wiederum ein Array, dessen erstes Element das Ziel des Links und dessen zweites Element den verlinkten Text enthält. Ein Beispiel:

```
$links=<<<END
Klicken Sie <a href="http://www.oreilly.com">hier</a>, um einen Verlag
f&uuml;r Computerb&uuml;cher aufzusuchen. Klicken Sie
<a href="http://www.sklar.com">an dieser Stelle</a>, um den Autor
eines Computerbuchs aufzusuchen.
END;

$a = pc_link_extractor($links);
print_r($a);
Array
(
    [0] => Array
        (
            [0] => http://www.oreilly.com
            [1] => hier
        )
    [1] => Array
        (
```

```
            [0] => http://www.sklar.com
            [1] => an dieser Stelle
        )
)
```

Der reguläre Ausdruck in `pc_link_extractor( )` erkennt nicht alle Links, zum Beispiel nicht solche, die mit JavaScript oder mit hexadezimalen Escape-Sequenzen gebildet werden. Aber bei der Mehrzahl der einigermaßen wohlgeformten HTML-Seiten sollte es funktionieren.

Die Verwendung von regulären Ausdrücken ist nicht immer leicht zu verstehen, und auch die Geschwindigkeit lässt oft zu wünschen übrig. Wenn Ihre PHP-Installation über die Tidy-Erweiterung verfügt, können Sie auch folgenden Code zur Extraktion verwenden:

```
$doc = new DOMDocument();
$opts = array('output-xml' => true,
              // Verhindern, dass DOMDocument von Entities irritiert ist
              'numeric-entities' => true);
$doc->loadXML(tidy_repair_file('linklist.html',$opts));
$xpath = new DOMXPath($doc);
// $xpath über XHTML namespace informieren
$xpath->registerNamespace('xhtml','http://www.w3.org/1999/xhtml');
foreach ($xpath->query('//xhtml:a/@href') as $node) {
    $link = $node->nodeValue;
    print $link . "\n";
}
```

Im gezeigten Beispiel wird ein HTML-Dokument eingelesen und mittels Tidy in validen XML-Code umgewandelt. Danach kann per XPath eine Abfrage nach den im XML enthaltenen Links (a-Elemente) ausgeführt werden.

Siehe auch

Rezept 16.7 mit Informationen zum Herausfiltern von Text innerhalb von HTML-Tags; die Dokumentation zu `preg_match_all( )` unter *http://www.php.net/preg-match-all*.

13.11 ASCII in HTML konvertieren

Problem

Sie möchten einfachen Text in brauchbar formatierten HTML-Code verwandeln.

Lösung

Codieren Sie zuerst die HTML-Entities mit `htmlentities( )`; transformieren Sie dann den Text in verschiedene HTML-Strukturen. Die in Beispiel 13-11 dargestellte Funktion `pc_ascii2html( )` führt einige grundlegende Transformationen für Links und Absätze durch.

Beispiel 13-11: pc_ascii2html()

```
function pc_ascii2html($s) {
  $s = htmlentities($s);
  $grafs = split("\n\n",$s);
  for ($i = 0, $j = count($grafs); $i < $j; $i++) {
    // Links für alles herstellen, was wie eine HTTP- oder FTP-URL aussieht.
    $grafs[$i] = preg_replace('/((ht|f)tp:\/\/[^\s&]+)/',
                              '<a href="$1">$1</a>',$grafs[$i]);

    // Links für E-Mail-Adressen.
    $grafs[$i] = preg_replace('/[^@\s]+@([-a-z0-9]+\.)+[a-z]{2,}/i',
        '<a href="mailto:$1">$1</a>',$grafs[$i]);

    // Neuen Absatz beginnen.
    $grafs[$i] = '<p>'.$grafs[$i].'</p>';
  }
  return join("\n\n",$grafs);
}
```

Diskussion

Je mehr Sie über den Aufbau des ASCII-Texts wissen, desto mehr kann Ihre HTML-Konvertierung leisten. Wenn beispielsweise Hervorhebungen durch *Sterne* oder /Schrägstriche/ vor und hinter den Wörtern gekennzeichnet sind, können Sie das folgendermaßen mithilfe dieser zusätzlicher Regeln berücksichtigen:

```
$grafs[$i] = preg_replace('/(\A|\s)\*([^*]+)\*(\s|\z)/',
                          '$1<b>$2</b>$3',$grafs[$i]);
$grafs[$i] = preg_replace('{(\A|\s)/([^/]+)/(\s|\z)}',
                          '$1<i>$2</i>$3',$grafs[$i]);
```

Siehe auch

Die Dokumentation zu `preg_replace( )` unter *http://www.php.net/preg-replace*.

13.12 HTML in ASCII konvertieren

Problem

Sie müssen eine HTML-Seite in lesbaren, formatierten ASCII-Text umwandeln.

Lösung

Wenn Sie über ein externes Programm wie *lynx* verfügen, das HTML als ASCII formatiert, können Sie es folgendermaßen aufrufen:

```
$file = escapeshellarg($file);
$ascii = `lynx -dump $file`;
```

Diskussion

Verfügen Sie über kein externes Programm zum Formatieren, können Sie mit der Funktion `pc_html2ascii()` aus Beispiel 13-12 immerhin eine Untermenge der HTML-Codes verarbeiten (allerdings keine Tabellen und Frames).

Beispiel 13-12: pc_html2ascii()

```
function pc_html2ascii($s) {
  // Links konvertieren.
  $s = preg_replace('/<a\s+.*?href="?([^\" >]*)"?[^>]*>(.*?)<\/a>/i',
                    '$2 ($1)', $s);

  // Die Tags <br>, <hr>, <p>, <div> in Zeilenumbrüche umwandeln.
  $s = preg_replace('@<(b|h)r[^>]*>@i',"\n",$s);
  $s = preg_replace('@<p[^>]*>@i',"\n\n",$s);
  $s = preg_replace('@<div[^>]*>(.*)</div>@i',"\n".'$1'."\n",$s);

  // Fett- und Kursivschrift umwandeln.
  $s = preg_replace('@<b[^>]*>(.*?)</b>@i','*$1*',$s);
  $s = preg_replace('@<i[^>]*>(.*?)</i>@i','/$1/',$s);

  // Benannte Entities dekodieren.
  $s = strtr($s,array_flip(get_html_translation_table(HTML_ENTITIES)));

  // Numerische Entities dekodieren.
  $s = preg_replace('//e','chr(\\1)',$s);

  // Alle übrigen Tags entfernen.
  $s = strip_tags($s);

  return $s;
}
```

Eine weitere Möglichkeit besteht in der Verwendung der Klasse `html2text`, die als externe Bibliothek von der Seite *http://www.chuggnutt.com/html2text.php* bezogen werden kann:

```
require_once 'class.html2text.inc';
$html = file_get_contents('http://www.example.com/article.html');
$converter = new html2text($html);
$plain_text = $converter->get_text();
```

Die `html2text`-Klasse verfügt über eine Vielzahl eingebauter Formatierungsmöglichkeiten, um Absätze, Überschriften usw. zu formatieren. Weitere Informationen dazu finden Sie auf der genannten Webseite.

Siehe auch

Rezept 11.8 mit weiteren Informationen zu `get_html_translation_table()`; die Dokumentationen zu `preg_replace()` unter *http://www.php.net/preg-replace*, zu `get_html_translation_table()` unter *http://www.php.net/get-html-translation-table* und zu `strip_`

tags( ) unter *http://www.php.net/strip-tags*. Mehr Informationen zu `html2text` unter *http://www.chuggnutt.com/html2text.php*.

13.13 HTML- und PHP-Tags entfernen

Problem

Sie möchten alle HTML- und PHP-Tags aus einem String oder einer Datei entfernen.

Lösung

Mit `strip_tags( )` entfernen Sie HTML- und PHP-Tags aus einem String:

```
$html = '<a href="http://www.oreilly.com">Ich <b>mag Computer-Literatur.</b></a>';
print strip_tags($html);
Ich mag Computer-Literatur.
```

Diskussion

Wenn Sie eine gesamte Datei einlesen wollen:

```
$no_tags = strip_tags(file_get_contents('test.html'));
```

`strip_tags( )` kann angewiesen werden, bestimmte Tags nicht zu entfernen, indem diese als letztes Argument angegeben werden. Die Angabe dieser Tags ist unabhängig von der Groß- und Kleinschreibung, und bei Tag-Paaren brauchen Sie nur das öffnende Tag anzugeben. Beispielsweise werden folgendermaßen alle Tags außer `<b></b>` aus `$html` entfernt:

```
$html = '<a href="http://www.oreilly.com">Ich <b>mag</b> Computer-Literatur.</a>';
print strip_tags($html,'<b>');
Ich <b>mag</b> Computer-Literatur.
```

Siehe auch

Die Dokumentation zu `strip_tags( )` unter *http://www.php.net/strip-tags*.

13.14 Die Protokolldatei eines Webservers analysieren

Problem

Sie möchten Berechnungen auf der Grundlage von Informationen durchführen, die sich in der Zugriffs-Protokolldatei Ihres Webservers befinden.

Lösung

Öffnen Sie die Datei und verarbeiten Sie die Zeilen mithilfe regulärer Ausdrücke, die dem Format der Protokolldatei entsprechen. Der folgende reguläre Ausdruck passt zum *NCSA Combined Log Format*:

```
$pattern = '/^([^ ]+) ([^ ]+) ([^ ]+) (\[[^\]]+\]) "(.*) (.*) (.*)" ([0-9\-]+)
    ([0-9\-]+) "(.*)" "(.*)"$/';
```

Diskussion

Das folgende Programm liest dem NCSA Combined Log Format entsprechende Zeilen und zeigt eine Liste von Seiten an, die nach der Anzahl der Anfragen pro Seite sortiert ist:

```
$log_file = '/usr/local/apache/logs/access.log';
$pattern = '/^([^ ]+) ([^ ]+) ([^ ]+) (\[[^\]]+\]) "(.*) (.*) (.*)" ([0-9\-]+)
    ([0-9\-]+) "(.*)" "(.*)"$/';

$fh = fopen($log_file,'r') or die($php_errormsg);
$i = 1;
$requests = array();
while (! feof($fh)) {
    // Zeilen einlesen und führende/nachfolgende Leerzeichen entfernen.
    if ($s = trim(fgets($fh,16384))) {
        // Das Muster auf die Zeile anwenden.
        if (preg_match($pattern,$s,$matches)) {
            /* Jeden übereinstimmenden Teil in einer passend
             * benannten Variablen speichern. */
            list($whole_match,$remote_host,$logname,$user,$time,
                 $method,$request,$protocol,$status,$bytes,$referer,
                 $user_agent) = $matches;
             // Zähler für die Anzahl der Anfragen mitführen.
            $requests[$request]++;
        } else {
            // Melden, wenn die Zeile nicht zum Muster passt.
            error_log("Can't parse line $i: $s");
        }
    }
    $i++;
}
fclose($fh) or die($php_errormsg);

// Das Array nach den Anfragezahlen absteigend sortieren.
arsort($requests);

// Ergebnisse formatiert ausgeben.
foreach ($requests as $request => $accesses) {
    printf("%6d   %s\n",$accesses,$request);
}
```

Das in preg_match( ) verwendete Muster passt auf Zeilen im Combined Log Format wie diese:

```
10.1.1.162 - david [20/Jul/2001:13:05:02 -0400] "GET /sklar.css HTTP/1.0" 200
278 "-" "Mozilla/4.77 [en] (WinNT; U)"
10.1.1.248 - - [14/Mar/2002:13:31:37 -0500] "GET /php-cookbook/colors.html
HTTP/1.1" 200 460 "-" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
```

In der ersten Zeile ist 10.1.1.162 die IP-Adresse, von der die Anfrage kam. Abhängig von der Serverkonfiguration, kann an deren Stelle auch ein Host-Name stehen. Wenn das Array $matches der Liste einzelner Variablen zugewiesen wird, wird der Host-Name in $remote_host abgelegt. Der folgende Bindestrich (-) zeigt an, dass der entfernte Host keinen Benutzernamen per *identd*[1] geliefert hat, dementsprechend wird $logname auf - gesetzt.

Der String david ist ein Benutzername, der vom Browser per HTTP-Basic-Authentifizierung gesendet worden ist, und wird in $user gespeichert. Das Datum und die Zeit der Anfrage, die in $time gespeichert werden, befinden sich in eckigen Klammern. Dieses Datums- und Uhrzeitformat ist für strtotime( ) nicht verständlich; wenn Sie also Kalkulationen mit der Anfragezeit durchführen möchten, müssen Sie mithilfe einiger zusätzlicher Berechnungen die einzelnen Teile des formatierten Zeit-Strings extrahieren. Dann folgt in Anführungszeichen die erste Zeile des Requests. Diese besteht aus der Methode (GET, POST, HEAD usw.), die in $method gespeichert wird, dem angeforderten URI, der in $request, und dem Protokoll, das in $protocol gespeichert wird. Bei GET-Requests ist der Query-String Teil des URI. Bei POST-Requests befinden sich die Variablen im Request-Body, der nicht protokolliert wird.

Nach dem Request folgt der Request-Status, abgelegt in $status. Der Status 200 bedeutet, dass die Anfrage erfolgreich war. Nach dem Status kommt die Länge der Antwort in Bytes, die in $bytes gespeichert wird. Die letzten beiden Elemente der Zeile, jedes in Anführungszeichen, sind die verweisende Seite, die, soweit vorhanden, in $referer[2] gespeichert wird, und der User-Agent-String, der den Browser identifiziert, von dem die Anfrage kommt, und in $user_agent gespeichert wird.

Nachdem Sie die Protokollzeile in einzelne Variablen zerlegt haben, können Sie die gewünschten Berechnungen durchführen. In diesem Fall wird nur mit einem Zähler im Array $requests festgestellt, wie oft jeder URI angefordert worden ist. Wenn alle Zeilen der Datei durchlaufen sind, gibt das Programm eine sortierte und formatierte Liste der Anfragen und der Zähler aus.

---

1 *identd* definiert in RFC 1413, soll eine gute Möglichkeit zur Identifizierung entfernter Benutzer darstellen. Allerdings ist sie nicht besonders sicher oder zuverlässig. Eine gute Erklärung dafür, warum dies so ist, finden Sie unter *http://www.clock.org/~fair/opinion/identd.html*.

2 Die korrekte Schreibweise dieses Worts lautet »referrer«. Da es aber in der ursprünglichen HTTP-Spezifikation (RFC 1945) fälschlich »referer« geschrieben wurde, wird die Schreibweise mit drei R häufig in diesem Kontext verwendet.

Man kann auf diese Weise zwar einfach statistische Auswertungen aus den Webserver-Zugriffsprotokollen berechnen, allerdings ist diese Vorgehensweise nicht besonders flexibel. Das Programm müsste noch modifiziert werden, damit man verschiedene Arten von Berichten erzeugen, Zeiträume begrenzen, die Formatierung verändern und weitere Features nutzen kann. Eine bessere Lösung für umfassende Website-Statistiken ist die Verwendung eines Programms wie *analog*, das kostenlos unter *http://www.analog.cx* erhältlich ist. Es kennt viele Typen von Berichten und viele Konfigurationsoptionen, damit sollte es so ziemlich jeden Informationsbedarf befriedigen können.

Siehe auch

Die Dokumentation zu `preg_match()` unter *http://www.php.net/preg-match*; Informationen über standardisierte Protokollformate sind unter *http://httpd.apache.org/docs/logs.html* verfügbar.

13.15 Auf eine AJAX-Anfrage antworten

Problem

Sie nutzen JavaScript, um in einer Seite mit `XMLHTTPRequest` eine eingebettete Anfrage durchzuführen, und müssen Daten zur Beantwortung derartiger Anfragen senden.

Lösung

Setzen Sie einen geeigneten `Content-Type`-Header und verschicken Sie dann die entsprechend formatierten Daten. Beispiel 13-13 sendet ein kleines XML-Dokument als Antwort.

Beispiel 13-13: Eine XML-Antwort versenden

```
<?php header('Content-Type: text/xml'); ?>
<menu>
 <dish type="appetizer">Hühnersuppe</dish>
 <dish type="main course">Frittiertes Affenhirn</dish>
</menu>
```

Beispiel 13-14 nutzt das PEAR-Paket Services_JSON, um eine JSON-Antwort zu versenden.

Beispiel 13-14: Eine JSON-Antwort versenden

```
<?php
require_once 'Services/JSON.php';
$menu = array();
$menu[] = array('type' => 'appetizer',
                'dish' => 'Hühnersuppe');
$menu[] = array('type' => 'main course',
                'dish' => 'Frittiertes Affenhirn');
```

Beispiel 13-14: Eine JSON-Antwort versenden (Fortsetzung)

```
header('Content-Type: application/json');
$json = new Services_JSON();
print $json->encode($menu);
?>
```

Beispiel 13-15 nutzt die Erweiterung `ext/json` (die ab PHP 5.2 und später integriert ist), um eine JSON-Antwort zu versenden.

Beispiel 13-15: Mit ext/json eine JSON-Antwort versenden

```
<?php
$menu = array();
$menu[] = array('type' => 'appetizer',
                'dish' => 'Hühnersuppe');
$menu[] = array('type' => 'main course',
                'dish' => 'Frittiertes Affenhirn');
header('Content-Type: application/json');
print json_encode($menu);
?>
```

Diskussion

Aus PHP-Perspektive unterscheidet sich das Versenden einer Antwort auf eine `XMLHTTP-Request`-basierte Anfrage in keinerlei Weise vom Versenden jeder anderen Antwort. Sie versenden die erforderlichen Header und schicken dann etwas Text hinterher. Was anders ist, sind die Header selbst und meist auch die Form des Texts.

JSON ist ein besonders praktisches Format bei derartigen Antworten, da es unglaublich einfach ist, die JSON-formatierten Daten in JavaScript zu verarbeiten. Die Ausgabe von Beispiel 13-14 sieht folgendermaßen aus:

```
[{"type":"appetizer","dish":"Hühnersuppe"},
{"type":"main course","dish":"Frittiertes Affenhirn"}]
```

Das kodiert ein JavaScript-Array mit Hashes, das zwei Elemente hat. Das PEAR-Modul Services_JSON ist ein handliches Werkzeug, um PHP-Datenstrukturen (Skalare, Arrays und Objekte) in JSON-Strings umzuwandeln und umgekehrt. Da es ein PEAR-Modul ist, können Sie es sogar nutzen, wenn Sie keinen Zugriff auf die *php.ini*-Datei haben oder keine binären Erweiterungen installieren können. Wenn Sie eigene Erweiterungen installieren können, sollten sie erwägen, stattdessen die PECL-Erweiterung `json` zu verwenden, die einen erheblichen Geschwindigkeitsvorteil bietet. Ihre Funktionen `json_encode( )` und `json_decode( )` wandeln PHP-Datenstrukturen in JSON-Strings um und JSON-Strings in PHP-Datenstrukturen.

Bei derartigen Antworten ist es auch wichtig, dass Sie auf das Caching achten. Die verschiedenen Browser bieten eine Vielzahl von Caching-Strategien in Bezug auf Anfragen, die aus JavaScript gemacht werden. Wenn Ihre Antwort dynamische Daten enthält (was meist der Fall ist), dann möchten Sie wahrscheinlich vermeiden, dass sie zwischengespei-

chert werden. Die Werkzeuge in Ihrem Anti-Caching-Kit sind Header und URL-Vergiftung. Beispiel 13-16 zeigt die vollständige Sammlung an Anti-Caching-Headern, die Sie aus PHP versenden können, um zu verhindern, dass ein Browser eine Antwort im Cache ablegt.

Beispiel 13-16: Anti-caching headers

```
<?php
header("Expires: 0");
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
header("Cache-Control: no-store, no-cache, must-revalidate");
// Einige IE-spezifische Optionen ergänzen
header("Cache-Control: post-check=0, pre-check=0", false);
// Für HTTP/1.0
header("Pragma: no-cache");
?>
```

Die andere Anti-Caching-Technik, URL-Vergiftung, erfordert die Kooperation des JavaScripts, das die Anfrage durchführt. Sie fügt dem Query-String jeder Anfrage ein Name/Wert-Paar mit einem beliebigen Wert hinzu. Das sorgt dafür, dass die URL bei jeder Anfrage anders aussieht und verhindert, dass sich ein aufdringlicher Cache dazwischen schiebt. Die JavaScript-Funktion `Math.random()` ist gut geeignet, um derartige Werte zu generieren.

Siehe auch

Dokumentation zu `header()` unter *http://www.php.net/header*. Mehr zu XMLHTTP-Request erfahren Sie unter *http://en.wikipedia.org/wiki/XMLHttpRequest*, zu JSON unter *http://www.json.org*, zu Services_JSON unter *http://pear.php.net/pepr/pepr-proposal-show.php?id=198* und zur PECL-Erweiterung `json` unter *http://pecl.php.net/package/json*. Michael Radwins »HTTP Caching and Cache-Busting for Content Publishers« (*http://public.yahoo.com/~radwin/talks/http-caching-apachecon2005.htm*) ist eine gute Einführung in HTTP-Caching. Abschnitt 13 von RFC 2616 (*http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html#sec13*) bietet alle Details zu HTTP-Caching.

13.16 Zusammenarbeit mit JavaScript

Problem

Sie möchten Teile Ihrer Seite mit serverseitigen Daten auffrischen, ohne die gesamte Seite neu zu laden. Sie wollen beispielsweise eine Liste mit Suchergebnissen füllen.

Lösung

Nutzen Sie ein JavaScript-Toolkit wie Dojo, um die Clientseite der Geschichte so einzurichten, dass eine bestimmte Benutzeraktion (wie ein Klick auf einen Button), eine

Anfrage an den Server anstößt. Schreiben Sie den erforderlichen PHP-Code zur Generierung einer Antwort mit den gewünschten Daten. Nutzen Sie dann das JavaScript-Toolkit, um die Ergebnisse wie gewünscht in die Seite einzufügen.

Beispiel 13-17 zeigt ein einfaches HTML-Dokument, das Dojo lädt, und Beispiel 13-18 den Client-Code. Beispiel 13-18 ist der JavaScript-Code, der die Anfrage an den Server verschickt, wenn auf den Suchen-Button geklickt wird, und dafür sorgt, dass die Ergebnisse an der richtigen Stelle der Seite landen, wenn sie vom Server geliefert werden. Beispiel 13-19 ist der PHP-Code, der die Suchoperation durchführt und eine mit JSON-formatierte Antwort zurücksendet.

Beispiel 13-17: HTML-Fundament für die JavaScript-Integration

```
<!-- Load Dojo -->
<script type="text/javascript" src="/dojo.js"></script>
<!-- JavaScript laden -->
<script type="text/javascript" src="/search.js"></script>

<!-- Einige Eingabeelemente -->
<input type="text" id="q" />
<input type="button" id="go" value="Suchen"/>
<hr/>
<!-- Hier kommt die Ausgabe hin -->
<div id="output"></div>
```

Beispiel 13-18: JavaScript-Brücke für die Integration

```
// Diesen Code ausführen, wenn die Seite geladen wird
dojo.addOnLoad(function() {
    // Funktion search() aufrufen, wenn auf den Suchen-Button geklickt wird
    dojo.event.connect(dojo.byId('go'), 'onclick', 'search');
});

function search() {
    // Text im Textfeld auslesen?
    var q = dojo.byId('q').value;
    // Anfrage an den Server senden
    // Die URL sollte auf den Ort der Suchseiten verweisen
    dojo.io.bind({ 'url': '/search.php',
                   'content': { 'q': q },
                   // Typ der Antwort
                   'mimetype': 'text/json',
                   // Aufzurufende Funktion, wenn Antwort ankommt
                   'load': showResults });
}

// Ergebnisse verarbeiten
function showResults(type, results, evt) {
    var html = '';
    // Wenn es Ergebnisse gibt ...
```

Beispiel 13-18: JavaScript-Brücke für die Integration (Fortsetzung)

```
    if (results.length > 0) {
        html = '<ul>';
        // Eine Liste mit ihnen
        for (var i in results) {
            html += '<li>' + dojo.string.escapeXml(results[i]) + '</li>';
        }
        html += '</ul>';
    } else {
        html = 'Keine Ergebnisse.';
    }
    // Das Ergebnis-HTML in die Seite stecken
    dojo.byId('output').innerHTML = html;
}
```

Beispiel 13-19: PHP zur Generierung der Antwort für JavaScript

```
<?php
// JSON initialisieren
require_once 'Services/JSON.php';
$json = new Services_JSON();

$results = array();
$q = isset($_GET['q']) ? $_GET['q'] : '';

// Mit Datenbank aus Kapitel 12 verbinden
$db = new PDO('sqlite:/usr/local/data/zodiac.db');

// Abfrage ausführen
$st = $db->prepare('SELECT symbol FROM zodiac WHERE planet LIKE ? ');
$st->execute(array($q.'%'));

// Ein Array mit den Ergebnissen aufbauen
while ($row = $st->fetch()) {
    $results[] = $row['symbol'];
}

// Den gesamten Anti-Caching-Kram ausgeben
header("Expires: 0");
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
header("Cache-Control: no-store, no-cache, must-revalidate");
// Einige IE-spezifische Optionen ergänzen
header("Cache-Control: post-check=0, pre-check=0", false);
// Für HTTP/1.0
header("Pragma: no-cache");

// Die Antwort ist JSON
header('Content-Type: application/json');

// Die JSON-Daten ausgeben
print $json->encode($results);
?>
```

Diskussion

Das HTML in Beispiel 13-17 ist mit Absicht äußerst spartanisch. Es gibt nichts, außer ein paar Elementen und Aufrufen zum Laden der externen Skripten. Die Trennung von JavaScript und HTML ist eine empfehlenswerte Entwicklungstechnik – ähnlich wie die Trennung der Präsentations- und der Geschäftslogik auf der Serverseite. Das erste `<script/>`-Tag in Beispiel 13-17 sollte auf den Ort zeigen, an dem Sie Dojo installiert haben. Das zweite Skript sollte auf den Code in Beispiel 13-18 zeigen. Die darin enthaltenen JavaScript-Funktionen schlagen die Brücke zwischen den HTML-Elementen in Beispiel 13-17 und dem serverseitigen Code in Beispiel 13-19. Der erste Aufruf von `dojo.addOnLoad()` sagt dem Webbrowser, dass er, nachdem das Laden der Seite abgeschlossen ist, den JavaScript-Code ausführen soll, der seinerseits dem Webbrowser sagt, dass er die Funktion `search()` ausführen soll, wenn auf den Suchen-Button geklickt wird.

JavaScript-Programme sind häufig *Event-basiert* – d.h. sie basieren auf Regeln wie »führe die Funktion aus, wenn das passiert«. Mit JavaScript ausgestattete Webseite kennen keinen strengen, durchgängigen Programmablauf mit einem Anfang und einem Ende. Stattdessen bieten Sie dem Benutzer eine Vielzahl von Möglichkeiten – er kann auf Buttons klicken, Text in Textfelder eingeben, auf Links klicken und so weiter. Dazu richtet JavaScript-Code üblicherweise verschiedene *Event-Handler* ein – Funktonen die als Reaktion auf einen Klick, eine Texteingabe oder andere Ereignisse ausgeführt werden.

Die `search()`-Funktion in Beispiel 13-18 nutzt die Dojo-Funktion `dojo.io.bind()`, um eine Anfrage an den Server zu senden und dabei über den Abfrageparameter `q` den Inhalt des Textfeld zu übergeben. Die anderen Argument an `dojo.io.bind()` geben an, dass eine JSON-Antwort erwartet wird und, wenn sie ankommt, an die Funktion `showResults()` übergeben werden soll.

Die Funktion `showResults()` ihrerseits nimmt die Ergebnisse und baut aus ihnen eine HTML-Liste auf. Ist das erfolgt, setzt sie den Inhalt des `output-<div/>`s auf das entsprechende HTML.

Beispiel 13-19 ist der vertraute Teil dieses Triumvirat. Es hat große Ähnlichkeit mit jedem anderen "Suche mir mal was auf Basis dieser Eingabe raus"-PHP-Skript, von der Art auf die die Ergebnisse zurückgeliefert werden abgesehen. Statt HTML auszugeben, nutzt es die in Rezept 13.15 beschriebenen Techniken, um eine nicht cachebare JSON-Antwort zurückzuschicken.

Das Schreiben von Anwendungen, die sich auf JavaScript-basierte clientseitige Aktivitäten stützen, erfordert ein anderes Programmierschema als eine übliche PHP-Anwendung. Anstatt sich darüber Gedanken zu machen, wie Sie dynamisch eine vollständige Seite generieren, müssen Sie überlegen, wie Sie dynamisch Datenteile generieren, die die clientseitige Logik ohne großen Aufwand anzeigen oder verarbeiten kann. Ein Toolkit wie Dojo stellt Ihnen eine robuste Plattform, auf der man derartige Anwendungen errichten kann. Es abstrahiert viele der verzwickten praktischen Anforderungen der JavaScript-Program-

mierung – Browser-Inkompatibilitäten, die Interna der asynchronen Kommunikation und andere elementare Verwaltungsaufgaben.

Es gibt auch PHP-orientierte JavaScript-Toolkits wie das PEAR-Paket HTML_Ajax und xajax. Mit ihnen können Sie PHP-Funktionen schreiben und dann leicht von JavaScript aufrufen, während sie sich darum kümmern, die JavaScript-Funktionen auf die jeweiligen PHP-Funktionen abzubilden. Diese Toolkits bieten zwar mehr Annehmlichkeiten auf PHP-Ebene, die aber zu Lasten der Robustheit auf JavaScript-Ebene gehen. HTML_Ajax und xajax können beide praktisch und bequem sein, wenn man schnell etwas PHP-Code mit clientseitiger Logik verbinden muss, sie sind aber nicht für den Aufbau von Anwendungen geeignet, die von Grund auf clientorientiert konzipiert sind.

Nachdem wir das klargestellt haben, sollten wir ergänzen, dass die komfortable Interaktion von PHP und JavaScript ein Problem ist, für das viele Leute aktiv nach Lösungen suchen. In Zukunft werden sich zweifelsohne leichtere Wege öffnen.

Siehe auch

Rezept 13.15 befasst sich mit dem Senden von JSON-Antworten. Dojo finden Sie unter *http://www.dojotoolkit.org/*, xajax unter *http://www.xajaxproject.org/* und HTML_Ajax unter *http://pear.php.net/package/HTML_Ajax*. »Getting Rich with PHP« (*http://talks.php.net/show/tek06*) untersucht die Leistungsauswirkungen, die das Reagieren auf viele JavaScript-basierte Anfragen hat. Andere JavaScript-Toolkits sind script.aculo.us (*http://script.aculo.us/*), Prototype (*http://www.prototypejs.org*) und die Yahoo! User Interface Library (*http://developer.yahoo.com/yui/index.html*).

13.17 Programm: Veraltete Links finden

Das Programm *stale-links.php* in Beispiel 13-20 erstellt eine Liste von in einer Seite enthaltenen Links und deren jeweilige Zustände. Sie sagt Ihnen, ob die Links in Ordnung sind, sich inzwischen woanders befinden oder nicht mehr gültig sind. Sie starten das Programm, indem Sie ihm eine URL übergeben, die nach Links durchsucht werden soll:

```
% stale-links.php http://www.oreilly.com/
http://www.oreilly.com/index.html: OK
http://www.oreillynet.com: OK
http://conferences.oreilly.com: OK
http://international.oreilly.com: OK
http://safari.oreilly.com: MOVED: mainhom.asp?home
...
```

Das Programm *stale-links.php* benutzt zum Laden der Webseiten die cURL-Erweiterung. Zuerst liest es die in der Befehlszeile angegebene URL aus. Nach dem Einlesen einer Seite ermittelt es mithilfe der Funktion `pc_link_extractor()` aus Rezept 13.10 alle Links, die sich in der Seite befinden. Nachdem dann jedem Link, falls nötig, eine Basis-URL vorangestellt worden ist, werden die Links einzeln abgerufen. Da nur die Header der Antworten

benötigt werden, verwenden wir die HEAD-Methode anstelle von GET, indem wir die Option CURLOPT_NOBODY setzen. Aufgrund der Option CURLOPT_HEADER gibt curl_exec( ) die Response-Header in einem String zurück. Entsprechend dem Response-Code wird der Status des Links ausgegeben und, wenn er verschoben wurde, auch die neue Adresse.

Beispiel 13-20: stale-links.php

```
function_exists('curl_exec') or die('CURL-Erweiterung wird benötigt');

function pc_link_extractor($s) {
    $a = array();
    if (preg_match_all('/<A\s+.*?HREF=[\"\']?([^\"\' >]*)[\"\']?[^>]*>(.*?)<\/A>/i',
                       $s,$matches,PREG_SET_ORDER)) {
        foreach($matches as $match) {
            array_push($a,array($match[1],$match[2]));
        }
    }
    return $a;
}

$url = $_SERVER['argv'][1];

// URL auslesen.
$c = curl_init($url);
curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($c, CURLOPT_FOLLOWLOCATION,1);
$page = curl_exec($c);
$info = curl_getinfo($c);
curl_close($c);

// Basis-URL aus der URL ermitteln.
// Ein <base>-Tag in der Seite bleibt dabei unberücksichtigt.
$url_parts = parse_url($info['url']);
if ('' == $url_parts['path']) { $url_parts['path'] = '/'; }
$base_path = preg_replace('<^(.*/)([^/]*)$>','\\1',$url_parts['path']);
$base_url = sprintf('%s://%s%s%s',
                    $url_parts['scheme'],
                    ($url_parts['username'] || $url_parts['password']) ?
                    "$url_parts[username]:$url_parts[password]@" : '',
                    $url_parts['host'],
                    $url_parts['path']);

// Besuchte Links notieren, damit die Links nur einmal besucht werden.
$seen_links = array();

if ($page) {
    $links = pc_link_extractor($page);
    foreach ($links as $link) {
        // Relative Links auflösen.
        if (! (preg_match('{^(http|https|mailto):}',$link[0]))) {
            $link[0] = $base_url.$link[0];
        }
```

Beispiel 13-20: stale-links.php (Fortsetzung)

```
        // Link auslassen, wenn wir ihn bereits gesehen haben.
        if ($seen_links[$link[0]]) {
            continue;
        }

        // Diesen Link als bekannt markieren.
        $seen_links[$link[0]] = true;

        // Den besuchten Link ausgeben.
        print $link[0].': ';
        flush();

        // Link aufsuchen.
        $c = curl_init($link[0]);
        curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($c, CURLOPT_NOBODY, 1);
        curl_setopt($c, CURLOPT_HEADER, 1);
        $link_headers = curl_exec($c);
        $curl_info = curl_getinfo($c);
        curl_close($c);

        switch (intval($curl_info['http_code']/100)) {
        case 2:
            // Response-Codes 2xx bedeuten, die Seite ist in Ordnung.
            $status = 'OK';
            break;
        case 3:
            // Response-Codes 3xx bedeuten Umleitung.
            $status = 'MOVED';
            if (preg_match('/^Location: (.*)$/m',$link_headers,$matches)) {
                $location = trim($matches[1]);
                $status .= ": $location";
            }
            break;
        default:
            // Andere Response-Codes bedeuten Fehler.
            $status = "ERROR: $curl_info[http_code]";
            break;
        }

        print "$status\n";
    }
}
```

13.18 Programm: Aktualisierte Links herausfinden

Beispiel 13-21, *fresh-links.php*, ist eine Variante des Rezepts 13.17, die eine Liste der Links mit dem Datum der letzten Änderung ausgibt. Wenn der Server, auf dem sich eine URL befindet, kein Datum der letzten Änderung liefert, meldet das Programm stattdes-

sen den Zeitpunkt der URL-Abfrage. Kann das Programm die URL nicht erfolgreich abrufen, gibt es den beim Leseversuch erhaltenen Status-Code zurück. Sie starten das Programm, indem Sie die URL übergeben, die nach Links durchsucht werden soll:

```
% fresh-links.php http://www.oreilly.com
http://www.oreilly.com/index.html: Fri Aug 16 16:48:34 2002
http://www.oreillynet.com: Mon Aug 19 10:18:54 2002
http://conferences.oreilly.com: Fri Aug 16 19:41:46 2002
http://international.oreilly.com: Fri Mar 29 18:06:32 2002
http://safari.oreilly.com: 302
http://www.oreilly.com/catalog/search.html: Tue Apr  2 19:05:57 2002
http://www.oreilly.com/oreilly/press/: 302
...
```

Diese Ausgabe stammt aus einem Programmlauf am 19. August 2002 gegen 10:20 Uhr EDT. Der Link zu *http://www.oreillynet.com* ist sehr aktuell, während die anderen unterschiedlich alt sind. Bei dem Link *http://www.oreilly.com/oreilly/press/* ist keine Zeitangabe für die letzte Änderung angegeben, sondern stattdessen ein HTTP-Status-Code (302). Dies bedeutet, dass die Seite irgendwohin umgezogen ist, wie es auch die Ausgabe von *stale-links.php* in Rezept 13.17 meldet.

Das Programm, mit dem man aktuelle Links findet, ist konzeptionell fast identisch mit dem Programm zum Suchen nach veralteten Links. Es verwendet die Funktion `pc_link_extractor( )` aus Rezept 13.11; allerdings verwendet es die Klasse `HTTP_Request` anstelle von cURL zum Lesen der URLs. Der Code, mit dem die in der Befehlszeile angegebene Basis-URL ermittelt wird, befindet sich innerhalb einer Schleife, sodass er gegebenenfalls zurückgegebenen Umleitungen folgen kann.

Nach dem Auslesen einer Seite stellt das Programm mithilfe der Funktion `pc_link_extractor( )` eine Liste der in der Seite vorhandenen Links zusammen. Wenn den Links, soweit erforderlich, eine Basis-URL vorangestellt worden ist, wird `sendRequest( )` mit jedem in der ursprünglichen Seite gefundenen Link aufgerufen. Da nur die Header der Antworten benötigt werden, verwenden wir die HEAD-Methode anstelle von GET. Allerdings gibt das Programm dann bei umgezogenen Links nicht die neue Adresse aus, sondern stattdessen eine formatierte Version des `Last-Modified`-Headers, sofern ein solcher verfügbar ist.

Beispiel 13-21: fresh-links.php

```
require 'HTTP/Request.php';

function pc_link_extractor($s) {
    $a = array();
    if (preg_match_all('/<A\s+.*?HREF=[\"\']?([^\"\' >]*)[\"\']?[^>]*>(.*?)<\/A>/i',
                       $s,$matches,PREG_SET_ORDER)) {
        foreach($matches as $match) {
            array_push($a,array($match[1],$match[2]));
        }
    }
```

Beispiel 13-21: fresh-links.php (Fortsetzung)

```
    return $a;
}

$url = $_SERVER['argv'][1];

// URLs in einer Schleife auslesen, um Umleitungen zu folgen.
$done = 0;
while (! $done) {
    $req = new HTTP_Request($url);
    $req->sendRequest();
    if ($response_code = $req->getResponseCode()) {
        if ((intval($response_code/100) == 3) &&
            ($location = $req->getResponseHeader('Location'))) {
            $url = $location;
        } else {
            $done = 1;
        }
    } else {
        return false;
    }
}

// Basis-URL aus der URL ermitteln.
// Ein <base>-Tag in der Seite bleibt dabei unbeachtet.
$base_url = preg_replace('{^(.*/)([^/]*)$}','\\1',$req->_url->getURL());

// Die gesehenen Links vermerken, damit jeder nur einmal besucht wird.
$seen_links = array();

if ($body = $req->getResponseBody()) {
    $links = pc_link_extractor($body);
    foreach ($links as $link) {
        // HTTPS-URLs auslassen.
        if (preg_match('{^https://}',$link[0])) {
            continue;
        }
        // Relative Links auflösen.
        if (! (preg_match('{^(http|mailto):}',$link[0]))) {
            $link[0] = $base_url.$link[0];
        }
        // Link auslassen, wenn wir ihn schon gesehen haben.
        if ($seen_links[$link[0]]) {
            continue;
        }

        // Diesen Link als bekannt markieren.
        $seen_links[$link[0]] = true;

        // Den gesehenen Link ausgeben.
        print $link[0].': ';
        flush();
```

Beispiel 13-21: fresh-links.php (Fortsetzung)

```
        // Link aufsuchen.
        $req2 = new HTTP_Request($link[0],
                                 array('method' => HTTP_REQUEST_METHOD_HEAD));
        $now = time();
        $req2->sendRequest();
        $response_code = $req2->getResponseCode();

        // Wenn Lesen erfolgreich war...
        if ($response_code == 200) {
            // ...Last-Modified-Header holen.
            if ($lm = $req2->getResponseHeader('Last-Modified')) {
                $lm_utc = strtotime($lm);
            } else {
                // Oder Datum auf aktuelle Uhrzeit setzen.
                $lm_utc = $now;
            }
            print strftime('%c',$lm_utc);
        } else {
            // Andernfalls den Response-Code ausgeben.
            print $response_code;
        }
        print "\n";
    }
}
```

KAPITEL 14

XML

14.0 Einführung

In der letzten Zeit hat XML als Format für den Austausch von Daten und für die Übertragung von Nachrichten weite Verbreitung gefunden. Und seitdem Webservices immer häufiger genutzt werden, spielt XML eine sogar noch größere Rolle im Leben des Entwicklers. PHP ermöglicht mithilfe einiger Erweiterungen, XML in allen möglichen Situationen zu lesen und zu schreiben.

Mit XML können Entwickler in einer strukturierten Weise Daten mit Tags auszeichnen und in einer baumartigen Hierarchie anordnen. Für einen möglichen Verwendungszweck kann man XML als eine Art verbessertes CSV betrachten. Sie können mit XML Datensätze speichern, die in eine Reihe von Feldern aufgeteilt sind. Dabei trennen Sie aber nicht nur einfach die Felder durch Kommata, sondern können darüber hinaus die Daten mit Feldnamen, Typen und Attributen versehen.

Man kann XML aber auch als eine Sprache zur Repräsentation von Dokumenten ansehen. Zum Beispiel wurde das *PHP Kochbuch* ursprünglich in XML geschrieben. Das Buch ist in Kapitel aufgeteilt, jedes Kapitel in Rezepte und jedes Rezept wiederum in die Abschnitte »Problem«, »Lösung« und »Diskussion«. Innerhalb jedes einzelnen Abschnitts findet eine weitere Untergliederung in Absätze, Tabellen, Abbildungen und Beispiele statt. Ein Artikel auf einer Webseite kann auf ähnliche Weise in die Bestandteile Seitentitel, Überschrift, die Autoren des Schriftstücks, den eigentlichen Text sowie Seitenbalken, zugehörige Links und weitere Inhalte zerlegt werden.

Ein XML-Text sieht ähnlich aus wie HTML. Beide benutzen zum Auszeichnen des Texts Tags, die in < und > eingeklammert sind. Aber XML ist zugleich strenger und lockerer als HTML. Strenger insofern, als alle Tags, die einen Inhalt haben, ordentlich geschlossen werden müssen. Es sind keine öffnenden Elemente ohne korrespondierende schließende Tags erlaubt. Lockerer, weil Sie nicht auf eine festgelegte Liste von Tags beschränkt sind wie `<a>`, `<img>` und `<h1>`. Stattdessen haben Sie die Freiheit, eine Reihe von Tag-Namen auszuwählen, die Ihre Daten am besten beschreiben.

Weitere wichtige Unterschiede zwischen XML und HTML sind die Unterscheidung zwischen Groß- und Kleinschreibung, der Gebrauch von Anführungszeichen bei Attributen und die Bedeutung der Leerzeichen. Für HTML sind `<B>` und `<b>` ein und dasselbe Tag zur Kennzeichnung von Fettschrift, für XML sind es zwei verschiedene Tags. Bei HTML können Sie häufig die Anführungszeichen vor und hinter den Attributen weglassen, XML dagegen verlangt sie. Sie müssen also immer schreiben:

```
<element attribute="value">
```

Außerdem ignorieren HTML-Parser im Allgemeinen Whitespace; 20 aufeinander folgende Whitespace werden genau so behandelt wie ein einzelnes Leerzeichen. XML-Parser dagegen berücksichtigen Whitespace, sofern sie nicht anders angewiesen werden. Da alle Elemente geschlossen werden müssen, enden leere Elemente immer mit `/>`. So ist beispielsweise in HTML ein Zeilenumbruch `<br>`, während er in XML als `<br />` geschrieben werden muss.[1]

Es gibt noch eine weitere Restriktion bei XML-Dokumenten. Da XML-Dokumente zu einem Baum von Elementen zerlegt werden können, wird das äußerste Element als *Wurzelelement* bezeichnet. Genau wie ein Baum nur einen Stamm hat, muss ein XML-Dokument genau ein Wurzelelement besitzen. Für das Buch-Beispiel von oben bedeutet dies, dass die Kapitel innerhalb eines Buch-Tags gebündelt werden müssen. Wenn das Dokument mehrere Bücher umfassen soll, müssen Sie diese in ein Buchregal oder einen anderen Behälter packen. Diese Beschränkung gilt nur für die Wurzel des Dokuments. So wie bei Bäumen mehrere Äste an einem Stamm wachsen können, ist es auch zulässig, mehrere Bücher in ein Buchregal zu legen.

Es ist nicht das Ziel dieses Kapitels, Ihnen die Grundlagen von XML nahe zu bringen; einen Einstieg dazu finden Sie in der *Einführung in XML* von Erik T. Ray. Ein grundlegender Führer durch alle XML-Aspekte ist *XML in a Nutshell* von Elliotte Rusty Harold und W. Scott Means. Beide Bücher sind im O'Reilly Verlag erschienen.

Nach den Regeln folgt jetzt ein Beispiel: Angenommen, Sie sind ein Bibliothekar und möchten Ihren Karteikartenkatalog in eine XML-Form bringen, dann beginnen Sie mit dem folgenden Satz grundlegender XML-Tags:

```
<buch>
    <titel>PHP Kochbuch</titel>
    <autor>Sklar, David und Trachtenberg, Adam</autor>
    <thema>PHP</thema>
</buch>
```

Sie können nun weitere Elemente hinzufügen oder bestehende modifizieren. Beispielsweise kann `<autor>` in Vor- und Zunamen aufgeteilt werden, oder Sie können mehrere Einträge zulassen, damit nicht zwei Autoren in einem Feld eingetragen werden müssen.

1 Aus diesem Grund gibt `nl2br( )` für den Zeilenumbruch `<br />` aus; das Ergebnis ist XML-kompatibel.

Bei der Entwicklung von PHP 5 lag ein Schwerpunkt auf den XML-Funktionalitäten, da diese in PHP 4 nicht ausreichend implementiert wurden. Alle XML-Erweiterungen in PHP 5 basieren auf der *libxml2*-Bibliothek, die kompatibel zu vielen aktuellen Standards ist. Mit der Veröffentlichung von PHP 5.1 wird der XML-Support noch verbessert. Die PECL-Extension xmlReader, die Thema von Rezept 14.8 ist, wird standardmäßig mit PHP ausgeliefert.

Die erste Hälfte dieses Kapitels behandelt das Lesen und Schreiben von XML-Daten. Dabei werden verschiedene XML-APIs wie SAX, DOM und SimpleXML verwendet, aber auch PECL-Extensions und PEAR-Pakete können für diese Aufgaben verwendet werden.

XML dient jedoch keinem Selbstzweck. Haben Sie alle Ihre XML-Daten beisammen, lautet die eigentliche Frage: »Was stellen Sie damit an?« Bei einem ereignisgesteuerten Parser, wie er in Rezept 14.5 beschrieben wird, lösen die Element-Tags Aktionen aus, die beispielsweise Daten in leicht manipulierbare Strukturen speichern oder Texte umformatieren.

Bei der Verwendung von XSLT können Sie mithilfe eines XSL-Stylesheets XML in lesbare Texte umwandeln. Wenn Sie den Inhalt von der Präsentation trennen, können Sie ein Stylesheet für Webbrowser, ein anderes für PDAs und ein drittes für Mobiltelefone benutzen, und all das, ohne den Inhalt als solchen zu ändern. Dies ist das Thema der Rezepte 14.9, 14.10 und 14.17. Möchten Sie XML hauptsächlich nutzen, um Daten zu speichern, und wollen wissen, wie Sie möglichst schnell auf einzelne Teile des Dokuments zugreifen können, hilft Ihnen hierzu Rezept 14.11, das die Abfragesprache XPath behandelt.

In den Rezepten 14.12, 14.13 und 14.14 werden die zwei PEAR-Pakete XML_Beautifier und XML_Serializer vorgestellt, die Ihnen dort weiterhelfen, wo die XML-Erweiterungen von PHP 5 nicht mehr genügend Funktionalitäten bieten.

Wenn ein XML-Dokument den syntaktischen Regeln von XML folgt, spricht man von *wohlgeformtem* XML. Neben der syntaktischen Korrektheit des XML-Dokuments sollte das XML auch *valide* sein. Ein XML-Dokument wird als valide betrachtet, wenn es einer gewissen Spezifikation entspricht (einer Dokumenttyp-Definition (DTD) oder einer Schema-Definition). In gewissen Fällen ist es notwendig, die Konformität zu den XML-Syntaxregeln sicherzustellen. Dadurch wird es Webbrowsern, RSS-Readern oder Ihren eigenen Skripten erleichtert, diese XML-Daten zu verarbeiten. Wie Sie ein XML-Dokument validieren können, zeigt Rezept 14.15.

Eine der Schwächen von PHP 5 ist die Behandlung von Zeichensätzen und Encodings. Zeichenketten sind in PHP 5 nicht mit einem spezifischen Encoding verbunden. Alle XML-Erweiterungen erwarten aber UTF-8-Eingaben und liefern UTF-8-Ausgaben. Wenn Sie mit einem UTF-8-inkompatiblen Zeichensatz arbeiten, müssen Sie daher Ihre Daten selbst nach UTF-8 konvertieren, bevor Sie diese an eine XML-Erweiterung weiterreichen. Umgekehrt müssen Sie die UTF-8-kodierten Daten, die Sie von der XML-Erweiterung erhalten, ebenfalls in Ihren Zeichensatz zurück überführen. Rezept 14.16 gibt Ihnen Hilfestellung zu diesem Vorgehen.

14.1 XML manuell generieren

Problem

Sie möchten XML generieren. Beispielsweise möchten Sie eine XML-Version Ihrer Daten erzeugen, die von einem anderen Programm gelesen werden kann.

Lösung

Lassen Sie Ihre Daten in einer Schleife durchlaufen und geben Sie sie in passende XML-Tags eingeschlossen aus:

```
header('Content-Type: text/xml');
print '<?xml version="1.0"?>' . "\n";
print "<shows>\n";

$shows = array(array('name'     => 'Simpsons',
                     'kanal'    => 'FOX',
                     'beginn'   => '20:00',
                     'dauer'    => '30'),

               array('name'     => 'Law & Order',
                     'kanal'    => 'NBC',
                     'beginn'   => '20:00',
                     'dauer'    => '60'));

foreach ($shows as $show) {
    print "    <show>\n";
    foreach($show as $tag => $data) {
        print "        <$tag>" . htmlspecialchars($data) . "</$tag>\n";
    }
    print "    </show>\n";
}

print "</shows>\n";
```

Diskussion

Die manuelle Ausgabe von XML-Daten erfordert meistens viele `foreach`-Schleifen zum Abarbeiten der Arrays. Es gibt aber noch einige wichtige Details. Als Erstes müssen Sie `header( )` aufrufen, um den korrekten `Content-Type`-Header für Ihr Dokument zu setzen. Da Sie XML senden und nicht HTML, sollte dies `text/xml` sein.

Abhängig von der Einstellung der Konfigurationsdirektive `short_open_tag`, kann außerdem der Versuch, die XML-Deklaration auszugeben, unbeabsichtigt die PHP-Verarbeitung anstoßen. Da das `<?` in `<?xml version="1.0"?>` mit dem kurzen öffnenden PHP-Tag identisch ist, müssen Sie, um die Deklaration an den Browser ausgeben zu können, entweder die Direktive deaktivieren oder die Zeile mit einer PHP-Anweisung ausgeben. In der Lösung wenden wir den zweiten Weg an.

Schließlich müssen noch Entities in Escape-Sequenzen umgewandelt werden. Beispielsweise muss das `&` in der Show `Law & Order` zu `&` werden. Um Ihre Daten entsprechend zu konvertieren, rufen Sie `htmlspecialchars( )` auf.

Die Ausgabe des Beispiels in der Lösung sieht folgendermaßen aus:

```
<?xml version="1.0"?>
<shows>
    <show>
        <name>Simpsons</name>
        <kanal>FOX</kanal>
        <beginn>20:00</beginn>
        <dauer>30</dauer>
    </show>
    <show>
        <name>Law &amp; Order</name>
        <kanal>NBC</kanal>
        <beginn>20:00</beginn>
        <dauer>60</dauer>
    </show>
</shows>
```

PEAR::XML_Util ist ein Paket, das Ihnen bei der Erzeugung von XML-Dokumenten sehr viel Arbeit abnehmen kann. Es stellt Hilfsmethoden bereit, mit denen einzelne Fragmente von XML-Dokumenten, wie z.B. Tags oder auch eine XML-Deklaration, mit einem Funktionsaufruf erzeugt werden können. Der Code zur Erzeugung desselben XML-Dokuments sieht unter Verwendung von XML_Util folgendermaßen aus:

```
require_once 'XML/Util.php';

header('Content-Type: text/xml');
print XML_Util::getXMLDeclaration() . "\n";
print XML_Util::createStartElement('shows') . "\n";

$shows = array(array('name'     => 'Simpsons',
                     'kanal'    => 'FOX',
                     'beginn'   => '20:00',
                     'dauer'    => '30'),

               array('name'     => 'Law & Order',
                     'kanal'    => 'NBC',
                     'beginn'   => '20:00',
                     'dauer'    => '60'));

foreach ($shows as $show) {
    print "    ".XML_Util::createStartElement('show')."\n";
    foreach($show as $tag => $data) {
        print "        ".XML_Util::createTag($tag, array(), $data)."\n";
    }
    print "    ".XML_Util::createEndElement('show')."\n";
}

print XML_Util::createEndElement('shows');
```

Nachdem Sie über require_once das Paket eingebunden haben, werden alle Teile, in denen Sie über die String-Funktionen ein XML-Tag erzeugt haben, durch Methodenaufrufe auf XML_Util ersetzt. Dazu stellt die Klasse statische Methoden wie createStartElement(), createEndElement() oder auch createTag() zur Verfügung. XML_Util kümmert sich dabei automatisch um das Ersetzen der Entities. Der Einsatz dieser Klasse hat mehrere Vorteile:

- Der Code wird lesbarer, da sofort ersichtlich ist, was erzeugt werden soll.
- Sie können keine Flüchtigkeitsfehler bei der Erzeugung der Tags machen.
- Sie können genauso einfach Tags mit Attributen erzeugen.

XML_Util bietet nicht nur Methoden, um Tags zu erstellen, Sie können damit auch CData-Sections und DTDs erstellen oder überprüfen, ob ein String ein gültiger Name für ein XML-Tag ist.

Siehe auch

Rezept 14.2 zum Generieren von XML mit DOM; Rezept 14.4 zum Generieren von XML mit xmlWriter; Rezept 14.13 zur Verwendung von XML_Serializer; Rezept 24.3 zur Installation von PEAR-Paketen; die Dokumentation zu htmlspecialchars( ) unter *http://www.php.net/htmlspecialchars* und die Dokumentation zu XML_Util unter *http://pear.php.net/manual/en/package.xml.xml-util.php*.

14.2 XML mit DOM generieren

Problem

Sie möchten XML generieren, aber Sie möchten es auf eine strukturierte Weise tun und nicht einfach durch Schleifen mit print-Ausgaben.

Lösung

Verwenden Sie die PHP 5-Erweiterung DOM zum Erzeugen eines DOM-Objekts und rufen Sie dann saveXML() oder save() auf, um wohlgeformte XML-Dokumente zu generieren:

```
// Neues Dokument erzeugen.
$dom = new DOMDocument('1.0', 'iso-8859-1');
$dom->formatOutput = true;

// Wurzelelement <book> anlegen und an das Dokument hängen.
$book = $dom->appendChild($dom->createElement('buch'));

// Titel-Element erzeugen und an $book hängen.
$title = $book->appendChild($dom->createElement('titel'));
```

```
// Text und Einband-Attribute zu $title setzen.
$title->appendChild($dom->createTextNode('PHP 5 Kochbuch'));
$title->setAttribute('einband', 'Hardcover');

// Autor-Elemente erzeugen und an $book hängen.
$sklar = $book->appendChild($dom->createElement('autor'));
// Den Text zu jedem Element erzeugen und anhängen.
$sklar->appendChild($dom->createTextNode('Sklar'));

$trachtenberg = $book->appendChild($dom->createElement('autor'));
$trachtenberg->appendChild($dom->createTextNode('Trachtenberg'));

$schmidt = $book->appendChild($dom->createElement('autor'));
$schmidt->appendChild($dom->createTextNode('Schmidt'));

$speidel = $book->appendChild($dom->createElement('autor'));
$speidel->appendChild($dom->createTextNode('Speidel'));

$lucke = $book->appendChild($dom->createElement('autor'));
$lucke->appendChild($dom->createTextNode('Lucke'));

$brusdeylins = $book->appendChild($dom->createElement('autor'));
$brusdeylins->appendChild($dom->createTextNode('Brusdeylins'));

print $dom->saveXML();
<?xml version="1.0" encoding="iso-8859-1"?>
<buch>
  <titel einband="Hardcover">PHP Kochbuch</titel>
  <autor>Sklar</autor>
  <autor>Trachtenberg</autor>
  <autor>Schmidt</autor>
  <autor>Speidel</autor>
  <autor>Lucke</autor>
  <autor>Brusdeylins</autor>
</buch>
```

Diskussion

Ein einzelnes Element wird als *Knoten* bezeichnet. Es gibt ein Dutzend verschiedener Typen von Knoten, aber die am meisten benutzten Typen sind Elemente, Attribute und Texte. Nehmen wir zum Beispiel diese Zeile:

```
<buch einband="Hardcover">PHP Kochbuch</buch>
```

Die DOM-Funktionen in PHP bezeichnen den Typ von `buch` als `XML_ELEMENT_NODE`, `einband="Hardcover"` wird auf einen `XML_ATTRIBUTE_NODE` abgebildet, und `PHP Kochbuch` ist ein `XML_TEXT_NODE`.

Die DOM-Extension ist in PHP 5 standardmäßig mit einkompiliert, sofern Sie sie nicht mit `--disable-xml` beim Kompilieren deaktiviert haben.

Die DOM-Funktionen in PHP 5 folgen dem Standard DOM-Level 2. Sie erzeugen ein Element entweder als Element oder als Textknoten, fügen alle gewünschten Attribute hinzu beziehungsweise setzen sie und hängen es dann an der Stelle in den Baum ein, an die es gehört.

Bevor Sie Elemente erzeugen, legen Sie erst einmal ein neues Dokument an und übergeben dabei die XML-Version und das gewünschte Encoding:

```
$dom = new DOMDocument('1.0', 'iso-8859-1');
```

Danach können Sie Eigenschaften des Dokuments festlegen. Durch Setzen der `$dom->formatOutput`-Eigenschaft auf den Wert `true` bestimmen Sie, dass die Ausgabe des XML-Baums in formatierter Form (also mit Zeilenumbrüchen und Einrückungen) geschehen soll. Setzen Sie diesen Wert auf `false`, wird das ganze Dokument in einer Zeile ausgegeben. Über weitere Objekt-Eigenschaften können Sie nachträglich das Encoding oder die XML-Versionsnummer setzen oder auch definieren, ob Whitespace-Zeichen ignoriert werden sollen.

Nun erzeugen Sie die zum Dokument gehörenden Elemente. Unabhängig davon, dass die Knoten zu einem bestimmten Dokument gehören, werden sie erst dadurch Teil des Dokumentbaums, dass sie angehängt werden:

```
$bookElement = $dom->createElement('buch');
$book = $dom->appendChild($bookElement);
```

Hier wird ein neues Element `buch` erzeugt und dem Objekt `$bookElement` zugeordnet. Um die Dokumentwurzel zu erzeugen, hängen Sie `$bookElement` als Kind an das Dokument `$dom` an. Das Ergebnis, `$book`, bezeichnet das spezielle Element und seine Position innerhalb des DOM-Objekts.

Alle Knoten werden durch Methodenaufrufe des `$dom` erzeugt. Ist ein Knoten einmal erzeugt, kann er an jedes beliebige Element des Baums gehängt werden. Das Element, bei dem die Methode `appendChild( )` aufgerufen wird, legt die Position im Baum fest, an der der Knoten angeordnet ist. Im obigen Fall wird `$bookElement` an `$dom` gehängt, dadurch wird es zum Top-Level-Knoten oder *Wurzelknoten*.

Sie können nun ein weiteres Kindelement an `$book` hängen. Da `$book` ein Kind von `$dom` ist, ist das neue Element gewissermaßen ein Enkel von `$dom`:

```
$titleElement = $dom->createElement('titel');
$title = $book->append_child($titleElement);
```

Durch den Aufruf von `$book->appendChild( )` wird das Element `$titleElement` unter dem Element `$book` angeordnet.

Um den Text innerhalb der Tags `<titel></titel>` einzufügen, erzeugen Sie mit `createTextNode( )` einen Textknoten und hängen diesen an `$title`:

```
$textNode = $dom->createTextNode('PHP Kochbuch');
$title->appendChild($text_node);
```

Da `$title` bereits zum Dokument hinzugefügt worden ist, besteht keine Notwendigkeit, dieses Element noch einmal an `$book` zu hängen.

Die Reihenfolge, in der Sie die Kindelemente an die Knoten hängen, spielt keine Rolle. Die folgenden vier Zeilen, die den Textknoten erst an `$titleElement` und dann an `$book` hängen, sind gleichbedeutend mit dem obigen Code:

```
$titleElement = $dom->createElement('titel');
$textNode = $dom->createTextNode('PHP Kochbuch');

$titleElement->appendChild($textNode);
$book->appendChild($titleElement);
```

Um ein Attribut hinzuzufügen, rufen Sie die Methode `setAttribute( )` des Knotens auf und übergeben den Attributnamen und den Wert als Argumente:

```
$title->setAttribute('einband', 'Hardcover');
```

Wenn Sie das Titel-Element jetzt ausgeben, sieht es folgendermaßen aus:

```
<titel einband="Hardcover">PHP Kochbuch</titel>
```

Nun können Sie das Dokument als String ausgeben oder in einer Datei speichern:

```
// Die String-Repräsentation des XML-Dokuments in $books speichern.
$books = $dom->saveXML( );

// Das XML-Dokument in die Datei buch.xml schreiben.
$dom->save('buch.xml');
```

Der einzige Parameter, den `saveXML( )` annimmt, ist ein optionaler XML-Knoten. Dadurch können Sie auch nur einen Teil des XML-Baums ausgeben:

```
// Nur ein Tag ausgeben.
print $dom->saveXML($schmidt);
```

Dieser Aufruf führt zum folgenden XML-Dokument:

```
<autor>Schmidt</autor>
```

Die Methode `save( )` wird verwendet, um das XML-Dokument in eine Datei zu schreiben, dazu müssen Sie als einzigen Parameter den Dateinamen angeben.

Siehe auch

Rezept 14.1 zum XML-Schreiben ohne DOM; Rezept 14.3 zum Erzeugen von XML-Dokumenten mit xmlWriter; Rezept 14.4 zum Parsen von XML mit DOM; die Dokumentation zu den DOM-Funktionen unter *http://www.php.net/dom*; weitere Informationen zur DOM-Spezifikation unter *http://w3c.org/DOM/*.

14.3 XML-Dokumente mit xmlWriter generieren

Problem

Sie möchten ein XML-Dokument erzeugen und sicherstellen, dass es wohlgeformt ist. Sie wollen dabei aber nicht auf die komplexe DOM-Erweiterung zurückgreifen.

Lösung

Verwenden Sie die PECL-Extension xmlWriter, die ab PHP-Version 5.1 voraussichtlich Teil der PHP-Standard-Distribution sein wird.

```
$shows = array(array('name'     => 'Simpsons',
                     'kanal'    => 'FOX',
                     'beginn'   => '20:00',
                     'dauer'    => '30'),

               array('name'     => 'Law & Order',
                     'kanal'    => 'NBC',
                     'beginn'   => '20:00',
                     'dauer'    => '60'));

$xw = xmlwriter_open_memory();

xmlwriter_set_indent($xw, 1);
xmlwriter_set_indent_string($xw, '    ');

xmlwriter_start_document($xw, '1.0');
xmlwriter_start_element($xw, 'shows');

foreach ($shows as $show) {
    xmlwriter_start_element($xw, 'show');
    foreach($show as $tag => $data) {
        xmlwriter_start_element($xw, $tag);
        xmlwriter_text($xw, $data);
        xmlwriter_end_element($xw);
    }
    xmlwriter_end_element($xw);
}

xmlwriter_end_element($xw);
xmlwriter_end_document($xw);

$xml = xmlwriter_output_memory($xw, true);
echo $xml;
```

Diskussion

xmlWriter ist eine Extension für PHP 5, mit der sehr einfach und sehr schnell wohlgeformte XML-Dokumente erzeugt werden können. Im Gegensatz zu DOM wird hierbei

keine Baumstruktur, sondern stattdessen ein XML-Stream erzeugt. Das heißt, die Funktionen zum Erstellen öffnender und schließender Tags müssen in der Reihenfolge aufgerufen werden, in der die Tags im XML-Dokument eingefügt werden sollen. Es ist nicht mehr möglich, im Dokument zurückzuspringen und ein weiteres Tag einzufügen.

Zum Erzeugen eines neuen Dokuments brauchen Sie zunächst ein xmlWriter-Resource-Handle, das Sie durch den Aufruf der Funktion `xmlwriter_open_memory()` erhalten können. Bei allen weiteren Funktionsaufrufen müssen Sie dieses Handle immer als ersten Parameter übergeben, damit xmlWriter weiß, mit welchem Dokument Sie arbeiten möchten. Nach Erzeugen des Handles legen Sie zuerst fest, wie die Tags im XML-Dokument eingerückt werden sollen:

```
xmlwriter_set_indent($xw, 1);
xmlwriter_set_indent_string($xw, '    ');
```

`xmlwriter_set_indent($xw, 1)` aktiviert die Einrückung des Tags, und mit `xmlwriter_set_indent_string()` geben Sie an, wie eingerückt werden soll.

Danach beginnen Sie ein XML-Dokument durch den Aufruf von `xmlwriter_start_document()`. Neben dem Handle übergeben Sie hier die Versionsnummer des XML-Dokuments, weitere optionale Parameter sind das zu verwendende Encoding und die Angabe, ob es sich um ein Standalone-Dokument handelt. Diese Parameter werden zum Erzeugen der XML-Deklaration gebraucht.

Jetzt endlich können Sie damit beginnen, das eigentliche XML-Dokument zu erzeugen. Dazu benötigen Sie nur drei Funktionen:

- `xmlwriter_start_element()`
- `xmlwriter_end_element()`
- `xmlwriter_text()`

Da das Dokument mit einem öffnenden Tag `<shows>` beginnen soll, erzeugen Sie dieses über

```
xmlwriter_start_element($xw, 'shows');
```

Danach durchlaufen Sie, analog zum Beispiel, in dem Sie XML nur mithilfe der String-Funktionen von PHP erzeugt haben, einfach mit einer `foreach`-Schleife das Array, aus dem ein XML-Dokument erzeugt werden soll.

Für jeden Eintrag in diesem Array erstellen Sie mit `xmlwriter_start_element()` ein öffnendes Tag `<show>` und durchlaufen danach das assoziative Array mit den Daten der Show:

```
xmlwriter_start_element($xw, 'show');
    foreach($show as $tag => $data) {
        xmlwriter_start_element($xw, $tag);
        xmlwriter_text($xw, $data);
        xmlwriter_end_element($xw);
    }
    xmlwriter_end_element($xw);
```

Dabei erzeugen Sie aus den Array-Keys jeweils ein öffnendes Tag und verwenden die Werte des Arrays als Inhalt des Tags, indem Sie es der Funktion `xmlwriter_text()` übergeben. Danach können Sie durch einen Aufruf von `xmlwriter_end_element()` das zuletzt geöffnete Tag wieder schließen. Natürlich brauchen Sie hier keinen Tag-Namen anzugeben, da die Funktion weiß, welches Tag zuletzt geöffnet wurde. Somit verhindert die Extension, dass das resultierende Dokument nicht wohlgeformt ist.

Wenn das Array mit den Daten einer Show durchlaufen wurde, schließen Sie nachfolgend das entsprechende `<show>`-Tag durch einen erneuten Aufruf von `xmlwriter_end_element()`.

Am Ende müssen Sie nur noch das Wurzel-Tag schließen und mit `xmlwriter_end_document()` das Dokument beenden. Hierbei werden automatisch alle noch geöffneten Elemente geschlossen. Um das erzeugte XML-Dokument auszugeben, verwenden Sie dann die Methode `xmlwriter_output_memory()`, die das XML-Dokument zurückliefert:

```
$xml = xmlwriter_output_memory($xw, true);
echo $xml;
```

Der zweite Parameter veranlasst xmlWriter, das erzeugte Dokument dabei aus dem Speicher zu löschen, wodurch Sie dasselbe Handle verwenden können, um weitere Dokumente zu erzeugen.

Möchten Sie das XML-Dokument nicht ausgeben, sondern in eine Datei schreiben, könnten Sie zwar die Funktion `file_put_contents()` verwenden, jedoch gibt es einen weitaus schnelleren Weg. Anstatt das Resource-Handle mit `xmlwriter_open_memory()` zu holen, verwenden Sie einfach `xmlwriter_open_uri()` und übergeben den Namen der Datei, in die das Dokument geschrieben werden soll. Beim Abschließen des Dokuments mit `xmlwriter_end_document()` wird dieses dann automatisch in die Datei geschrieben.

Die xmlWriter-Extension bietet natürlich nicht nur Methoden, um Tags und Text in das Dokument einzufügen, genauso können Sie auch Attribute oder XML-Kommentare hinzufügen:

```
$xw = xmlwriter_open_memory();

xmlwriter_set_indent($xw, 1);
xmlwriter_set_indent_string($xw, '   ');

xmlwriter_start_document($xw, '1.0');
xmlwriter_write_comment($xw, 'Erzeugt mit xmlWriter');
xmlwriter_start_element($xw, 'shows');

foreach ($shows as $show) {
    xmlwriter_start_element($xw, 'show');
    foreach($show as $att => $data) {
        xmlwriter_write_attribute($xw, $att, $data);
    }
    xmlwriter_end_element($xw);
}
```

```
xmlwriter_end_element($xw);
xmlwriter_end_document($xw);

$xml = xmlwriter_output_memory($xw, true);
echo $xml;

<?xml version="1.0"?>
<!--Erzeugt mit xmlWriter-->
<shows>
   <show name="Simpsons" kanal="FOX" beginn="20:00" dauer="30"/>
   <show name="Law &amp; Order" kanal="NBC" beginn="20:00" dauer="60"/>
</shows>
```

Mit `xmlwriter_write_comment()` fügen Sie einen neuen Kommentar ein, `xmlwriter_write_attribute()` fügt ein Attribut zum aktuell offenen Tag hinzu. Weiterhin bietet die Extension die Möglichkeit, Tags oder Attribute einem XML-Namespace zuzuordnen oder DTDs zu erzeugen.

Siehe auch

Rezept 14.1 zum manuellen Erzeugen von XML-Dokumenten; Rezept 14.2 zum Generieren von XML-Dokumenten mit DOM; die xmlwriter-Homepage unter *http://pecl.php.net/package/xmlwriter*.

14.4 Komplexe XML-Dokumente einlesen (DOM)

Problem

Sie möchten eine XML-Datei mithilfe der DOM-API zerlegen. Dabei wird die Datei in einem Baum abgespeichert, den Sie durch DOM-Funktionen verarbeiten können. Mit DOM ist es einfach, Elemente zu suchen und auszulesen, die bestimmten Kriterienmengen entsprechen.

Lösung

Verwenden Sie die PHP 5-Erweiterung DOM. Hier sehen Sie, wie Sie XML aus einer Datei lesen können:

```
$dom = DOMDocument::load('buecher.xml');
```

Und so lesen Sie XML aus einer Variablen:

```
$dom = DOMDocument::loadXML($books);
```

Sie können auch nach einem einzelnen Knoten suchen. Auf diese Weise erhalten Sie zum Beispiel den Wurzelknoten:

```
$root = $dom->documentElement;
```

Folgendermaßen führen Sie eine Depth-First-Rekursion zur Verarbeitung aller Knoten eines Dokuments aus:

```
function process_node($node) {
    if ($node->hasChildNodes()) {
        foreach($node->childNodes as $n) {
            process_node($n);
        }
    }

    // Blätter verarbeiten.
    if ($node->nodeType == XML_TEXT_NODE) {
        $content = rtrim($node->nodeValue);
        if (!empty($content)) {
            print "$content\n";
        }
    }

}
process_node($root);
```

Diskussion

Das W3C-DOM bietet eine plattform- und sprachneutrale Methode zur Spezifikation der Struktur und des Inhalts eines Dokuments. Mithilfe des DOM können Sie ein XML-Dokument in eine aus Knoten bestehende Baumstruktur einlesen. Danach haben Sie die Möglichkeit, durch diesen Baum zu navigieren, um Informationen zu einem bestimmten Element oder zu mehreren Elementen zu lokalisieren, die Ihren Kriterien entsprechen. Dies wird als *baumbasiertes Parsen* bezeichnet.

Außerdem können Sie die Struktur verändern, indem Sie Knoten erzeugen, editieren oder löschen. Oder stellen Sie mithilfe der DOM-Funktionen ein völlig neues XML-Dokument zusammen; sehen Sie dazu Rezept 14.2.

Einer der wesentlichen Vorzüge von DOM besteht darin, dass viele Sprachen der W3C-Spezifikation folgen und die DOM-Funktionen auf ähnliche Weise implementieren. Dadurch wird die Übersetzung der Logik und der Instruktionen von einer Anwendung zu einer anderen beträchtlich vereinfacht. PHP 5 implementiert den DOM-Level-2-Standard nahezu 100%ig, Sie können also Code, den Sie in einer anderen Sprache, die DOM unterstützt (z.B. JavaScript), geschrieben haben, vollständig in PHP übernehmen und müssen lediglich die Syntax ein wenig anpassen.

DOM ist umfangreich und komplex. Wenn Sie mehr darüber wissen möchten, lesen Sie die Spezifikation unter *http://www.w3.org/DOM/*, oder besorgen Sie sich ein Exemplar von *XML in a Nutshell* (O'Reilly Verlag), Kapitel 19 behandelt DOM.

Die DOM-Funktionen in PHP sind objektorientiert. Um sich von einem Knoten zum anderen zu bewegen, verwenden Sie Eigenschaften wie `$node->childNodes`, die ein DOM-NodeList-Objekt beinhaltet, das alle Kindobjekte speichert, oder `$node->parentNode`, die

das Elternknoten-Objekt speichert. Durch Iteratoren ist es möglich, ein DOMNodeList-Objekt wie ein Array mit foreach zu iterieren. Um einen Knoten zu verarbeiten, prüfen Sie also seinen Typ und rufen die korrespondierende Methode auf:

```
// $node ist der DOM-Knoten <buch einband="Hardcover">PHP Kochbuch</buch>.
$type = $node->nodeType;

switch($type) {
case XML_ELEMENT_NODE:
    // Tag-Element mit der Eigenschaft tagname.
    print $node->nodeName;  // Eigenschaft tagname: "buch"
    print $node->nodeValue; // null
    break;
case XML_ATTRIBUTE_NODE:
    // Attibut mit den Eigenschaften name und value.
    print $node->nodeName;  // Eigenschaft name: "einband"
    print $node->nodeValue; // Eigenschaft value: "hardcover"
    break;
case XML_TEXT_NODE:
    // Ein Stück Text innerhalb eines Elements
    // mit den Eigenschaften name und content.
    print $node->nodeName;  // Eigenschaft name: "#text"
    print $node->nodeValue; // Eigenschaft content: "PHP Kochbuch"
    break;
default:
    // Ein anderer Typ.
    break;
}
```

Mit `getElementsByTagname()` können Sie einen DOM-Baum automatisch nach bestimmten Elementen durchsuchen. Hier sehen Sie, wie Sie dies bei mehreren Bucheinträgen tun:

```
<buecher>
    <buch>
        <titel>PHP Kochbuch</titel>
        <autor>Sklar</autor>
        <autor>Trachtenberg</autor>
        <thema>PHP</thema>
    </buch>
    <buch>
        <titel>Perl Kochbuch</titel>
        <autor>Christiansen</autor>
        <autor>Torkington</autor>
        <thema>Perl</thema>
    </buch>
</buecher>
```

Folgendermaßen finden Sie alle Autoren heraus:

```
// Alle Autoren suchen und ausgeben.
$authors = $dom->getElementsByTagname('autor');

// Autor-Elemente in einer Schleife durchlaufen.
```

```
foreach ($authors as $author) {
    print $author->textContent;
    print "\n";
}
```

Die Funktion `getElementsByTagname()` gibt ein DOMNodeList-Objekt mit Elementknoten-Objekten zurück. Diese können Sie mit einer Schleife durchlaufen. Die gewünschten Informationen stehen in den Kindknoten der `autor`-Elemente, auf die Sie eigentlich über die Eigenschaft `$autor->childNodes` zugreifen müssten. Stattdessen wird hier eine PHP 5-spezifische Erweiterung des DOM verwendet, die Eigenschaft `$autor->textContent`, die den Inhalt der Textknoten unter dem Element zurückliefert, in diesem Fall die Namen von Buchautoren wie `Sklar` und `Trachtenberg`.

Siehe auch

Rezept 14.2 zum Schreiben mit DOM; Rezept 14.5 zum ereignisgesteuerten XML-Parsen; die Dokumentationen zu `DOMDocument::load()` unter *http://www.php.net/domdocument.load*, `DOMDocument::loadXML()` unter *domdocument.loadxml* und die DOM-Funktionen allgemein unter *http://www.php.net/dom*.

14.5 Große XML-Dokumente einlesen (SAX)

Problem

Sie möchten ein XML-Dokument lesen und auf der Basis von Ereignissen formatieren. Ein solches Ereignis tritt etwa ein, wenn der Parser auf ein neues öffnendes oder schließendes Tag trifft. Zum Beispiel möchten Sie einen RSS-Feed in HTML umwandeln.

Lösung

Verwenden Sie die Parser-Funktionen in der XML-Erweiterung zu PHP:

```
$xml = xml_parser_create();
$obj = new Parser_Object;  // eine Klasse zur Parser-Unterstützung

xml_set_object($xml,$obj);
xml_set_element_handler($xml, 'start_element', 'end_element');
xml_set_character_data_handler($xml, 'character_data');
xml_parser_set_option($xml, XML_OPTION_CASE_FOLDING, false);

$fp = fopen('data.xml', 'r') or die("Kann XML-Daten nicht lesen.");
while ($data = fread($fp, 4096)) {
  xml_parse($xml, $data, feof($fp)) or die("Kann XML-Daten nicht parsen.");
}
fclose($fp);

xml_parser_free($xml);
```

Diskussion

Diese Funktionen waren bereits in PHP 4 verfügbar, da PHP 5 jedoch auf Basis einer anderen XML-Bibliothek implementiert wurde als PHP 4, stellten die Entwickler eine Kompatibilitätsschicht zur Verfügung, um PHP 4-Skripten kompatibel zu den neuen Versionen zu halten.

Die XML-Funktionen verarbeiten XML-Dokumente und ermöglichen Ihnen, den Parser so zu konfigurieren, dass er beim Erreichen bestimmter Stellen in der Datei Funktionen aufruft. Solche Stellen sind zum Beispiel öffnende und schließende Element-Tags oder Zeichendaten (die Texte zwischen den Tags). Abhängig vom Namen eines Tags können Sie entscheiden, ob Sie es formatieren oder die Daten ignorieren. Im Gegensatz zum baumorientierten DOM wird dies als *ereignisorientiertes Parsen* bezeichnet.

Eine bekannte API für ereignisorientiertes XML-Parsen ist SAX (*Simple API for XML*). SAX wurde ursprünglich nur für Java entwickelt, hat sich aber auf andere Sprachen ausgeweitet. Die XML-Funktionen in PHP folgen den SAX-Konventionen. Mehr über die neuste SAX-Version – SAX2 – finden Sie in dem Buch *SAX2* von David Brownell (O'Reilly).

PHP unterstützt zwei Schnittstellen für SAX: eine prozedurale und eine objektorientierte. Bei der Arbeit mit der prozeduralen Schnittstelle muss man in der Praxis häufig globale Variablen verwenden, wenn man ernsthafte Aufgaben lösen möchte. Daher bevorzugen wir die objektorientierte Version. Bei der objektorientierten Schnittstelle können Sie ein Objekt an den Parser binden und während der XML-Verarbeitung mit dem Objekt interagieren. Auf diese Weise können Sie Objekt-Eigenschaften anstelle globaler Variablen verwenden.

Hier folgt ein Beispiel für eine Anwendung, die zeigt, wie man einen RSS-Feed verarbeitet und in HTML verwandelt. Mehr über RSS finden Sie in Rezept 15.7. Das Skript beginnt mit dem standardmäßigen Code für die XML-Verarbeitung, dem die Objekte für das RSS-spezifische Parsen folgen:

```
$xml = xml_parser_create( );
$rss = new pc_RSS_parser;

xml_set_object($xml, $rss);
xml_set_element_handler($xml, 'start_element', 'end_element');
xml_set_character_data_handler($xml, 'character_data');
xml_parser_set_option($xml, XML_OPTION_CASE_FOLDING, false);

$feed = 'http://pear.php.net/rss.php';
$fp = fopen($feed, 'r') or die("Kann RSS-Daten nicht lesen.");
while ($data = fread($fp, 4096)) {
  xml_parse($xml, $data, feof($fp)) or die("Kann RSS-Daten nicht parsen.");
}
fclose($fp);

xml_parser_free($xml);
```

Nachdem Sie einen neuen XML-Parser und eine Instanz der Klasse `pc_RSS_parser` erzeugt haben, konfigurieren Sie den Parser. Zuerst binden Sie das Objekt an den Parser; dadurch greift der Parser auf die Methoden des Objekts zu und nicht auf globale Funktionen. Dann rufen Sie `xml_set_element_handler( )` und `xml_set_character_data_handler( )` auf, um dem Parser die Namen der Methoden mitzuteilen, die er aufrufen soll, wenn er auf Elemente und Zeichendaten trifft. Das erste Argument ist bei beiden Funktionen die Parser-Instanz; die anderen Argumente sind die Funktionsnamen. Bei `xml_set_element_handler( )` bezeichnen das mittlere und das letzte Argument die Funktionen, die bei öffnenden beziehungsweise schließenden Tags aufgerufen werden sollen. Die Funktion `xml_set_character_data_handler( )` hat nur ein zusätzliches Argument – die zur Verarbeitung von Zeichendaten aufzurufende Funktion.

Da hier dem Parser ein Objekt zugeordnet wurde, ruft der Parser folgende Methoden auf, wenn er den String `<tag>data</tag>` vorfindet: `$rss->start_element( )` beim Erreichen von `<tag>`, `$rss->character_data( )` beim Erreichen von `data` und `$rss->end_element( )` beim Erreichen von `</tag>`. Man kann den Parser nicht so konfigurieren, dass er für jedes Tag eine spezifische Methode aufruft; vielmehr müssen Sie dies selbst regeln. Allerdings bietet das PEAR-Paket `XML_Parser` eine einfache Möglichkeit, Handler-Methoden für jedes Tag zuzuweisen.

Die letzte Konfigurationsoption des XML-Parsers veranlasst den Parser dazu, nicht alle Tags automatisch in Großbuchstaben zu konvertieren. Standardmäßig verwandelt der Parser alle Tags in Großbuchstaben, sodass `<tag>` und `<TAG>` zum selben Element werden. Da XML zwischen Klein- und Großschreibung unterscheidet und die meisten Feeds Elementnamen in Kleinbuchstaben verwenden, sollte diese Option abgeschaltet werden.

Nachdem der Parser konfiguriert ist, übergeben Sie ihm die Daten:

```
$feed = 'http://pear.php.net/rss.php';
$fp = fopen($feed, 'r') or die("Kann RSS-Daten nicht lesen.");
while ($data = fread($fp, 4096)) {
  xml_parse($xml, $data, feof($fp)) or die("Kann RSS-Daten nicht parsen.");
}
fclose($fp);
```

Um den Speicherbedarf zu begrenzen, laden Sie die Datei in Blöcken von 4.096 Bytes und übergeben dem Parser jeden Block einzeln. Dadurch ist es notwendig, Handler-Funktionen zu schreiben, die auch mit über mehrere Aufrufe verteilten Texten umgehen können und nicht darauf angewiesen sind, dass der ganze String in einem Stück kommt.

Zwar räumt PHP am Ende der Anfrage alle offenen Parser ab, aber Sie können auch am Ende den Parser durch einen Aufruf von `xml_parser_free( )` manuell schließen.

Wenn das Parsen nun korrekt vorbereitet ist, fügen Sie die in den Beispielen 14-1 und 14-2 dargestellten Klassen `pc_RSS_item` und `pc_RSS_parser` hinzu, damit ein RSS-Dokument verarbeitet werden kann.

Beispiel 14-1: pc_RSS_item

```
class pc_RSS_item {

  public $title = '';
  public $description = '';
  public $link = '';

  public function display() {
    printf('<p><a href="%s">%s</a><br />%s</p>',
            $this->link,htmlspecialchars($this->title),
            htmlspecialchars($this->description));
  }
}
```

Beispiel 14-2: pc_RSS_parser

```
class pc_RSS_parser {

  private $tag;
  private $item;

  public function start_element($parser, $tag, $attributes) {
    if ('item' == $tag) {
      $this->item = new pc_RSS_item;
    } elseif (!empty($this->item)) {
      $this->tag = $tag;
    }
  }

  public function end_element($parser, $tag) {
    if ('item' == $tag) {
      $this->item->display();
      unset($this->item);
    }
  }

  public function character_data($parser, $data) {
    if (!empty($this->item)) {
      if (isset($this->item->{$this->tag})) {
        $this->item->{$this->tag} .= trim($data);
      }
    }
  }
}
```

Die Klasse `pc_RSS_item` stellt eine Schnittstelle für ein einzelnes Feed-Element zur Verfügung. Sie lagert die Details der Darstellung einzelner Elemente aus dem allgemeinen Parser-Code aus und erleichtert das Zurücksetzen der Daten für ein neues Element durch einen Aufruf von `unset()`.

Die Methode `pc_RSS_item::display()` gibt ein HTML-formatiertes RSS-Element aus. Sie ruft `htmlspecialchars()` auf, um alle notwendigen Entities wieder zu kodieren, da *expat* sie beim Parsen des Dokuments zu normalen Zeichen dekodiert hat. Dieses Neukodieren funktioniert allerdings nicht bei Feeds, die im Titel HTML enthalten und nicht nur einfachen Text.

Im Inneren von `pc_RSS_parser()` nimmt die Methode `start_element()` drei Parameter an: den XML-Parser, den Namen des Tags sowie ein Array aus Attribut/Wert-Paaren (sofern sie vorhanden sind) aus dem Element. PHP liefert diese Werte als Teil des Zerlegungsprozesses an die Handler-Methode.

Die Methode `start_element()` prüft den Wert von `$tag`. Ist es ein Item, hat der Parser ein neues RSS-Element gefunden, und es wird ein neues `pc_RSS_item`-Objekt instantiiert. Andernfalls prüft sie, ob `$this->item` leer ist; ist dies nicht der Fall, befindet sich der Parser innerhalb eines Item-Elements. Dann ist es notwendig, den Namen des Tags aufzuzeichnen, damit die Methode `character_data()` weiß, welcher Eigenschaft ihr Wert zugeordnet werden muss. Ist es dagegen leer, ist dieser Teil des RSS offenbar nicht für diese Anwendung vorgesehen und wird ignoriert.

Wenn der Parser ein schließendes `item`-Tag findet, gibt die korrespondierende Methode `end_element()` erst das RSS-Item aus und räumt dann auf, indem sie das Objekt löscht.

Die Methode `character_data()` schließlich ist dafür verantwortlich, dem RSS-Item die Werte von `title`, `description` und `link` zuzuweisen. Nachdem sie sichergestellt hat, dass sie sich innerhalb eines `item`-Elements befindet, überprüft sie, ob das aktuelle Tag eine der Eigenschaften des `pc_RSS_item` bezeichnet. Ohne diese Abfrage würde der Wert auch dem Objekt zugewiesen werden, wenn der Parser auf ein anderes als eines dieser drei Elemente stößt. Die { } sind notwendig, um die Reihenfolge festzulegen, in der die Objekt-Eigenschaften dereferenziert werden. Beachten Sie, wie `trim($data)` an die Eigenschaft angehängt und ihr nicht direkt zugewiesen wird. Dadurch werden auch Daten korrekt verarbeitet, die über mehrere der von `fread()` gelesenen 4.096-Byte-Blöcke verteilt sind; außerdem wird dadurch ein den RSS-Feed umgebender Leerraum entfernt.

Wenn Sie das Programm mit diesem beispielhaften RSS-Feed laufen lassen:

```
<?xml version="1.0"?>
<rss version="0.93">
<channel>
  <title>PHP-Ank&uuml;ndigungen</title>
  <link>http://www.php.net/</link>
  <description>Die neuesten Informationen zu PHP.</description>

  <item>
    <title>PHP 5.0 freigegeben!</title>
    <link>http://www.php.net/downloads.php</link>
    <description>Die neueste PHP-Version ist jetzt verf&uuml;gbar.</description>
  </item>
</channel>
</rss>
```

wird folgender HTML-Code erzeugt:

```
<p><a href="http://www.php.net/downloads.php">PHP 5.0 freigegeben!</a><br />
Die neueste PHP-Version ist jetzt verf&uuml;gbar.</p>
```

Siehe auch

Rezept 14.4 zum baumorientierten XML-Parsen mit DOM; Rezept 15.7 mit weiteren Informationen über das Parsen von RSS; die Dokumentationen zu `xml_parser_create()` unter *http://www.php.net/xml-parser-create*, `xml_set_element_handler()` unter *http://www.php.net/manual/function.xml-set-character-data-handler.php*, `xml_set_character_data_handler()` unter *http://www.php.net/manual/function.xml-set-element-handler.php*, `xml_parse()` unter *http://www.php.net/xml-parse* und die XML-Funktionen allgemein unter *http://www.php.net/xml*; die offizielle SAX-Site unter *http://www.saxproject.org/*.

14.6 XML mit SimpleXML parsen

Problem

Sie möchten ein XML-Dokument auf sehr einfache Art und Weise verarbeiten, von dem Sie die Struktur bereits kennen, wie zum Beispiel ein RSS-Dokument.

Lösung

Verwenden Sie die neue SimpleXML-Erweiterung in PHP 5.

```
$feed = 'http://pear.php.net/rss.php';
$rss  = simplexml_load_file($feed);

if ($rss === false) {
    die("Konnte RSS-Feed nicht laden.");
}
foreach ($rss->item as $item) {

    printf('<p><a href="%s">%s</a><br />%s</p>',
            $item->link, htmlspecialchars($item->title),
            htmlspecialchars($item->description));
}
```

Diskussion

Die SimpleXML-Erweiterung bietet eine vollkommen neue API für XML-Dokumente, die in PHP 5 zum allerersten Mal implementiert wurde. Ziel dieser Extension ist es, Benutzern Zugriff auf XML-Dokumente zu geben, die mit XML nicht vertraut und denen die DOM- und SAX-basierten APIs zu kompliziert sind. Dazu erzeugt SimpleXML eine Repräsentation des XML-Dokuments, auf das zugegriffen werden kann, als wäre es eine

Ansammlung von verschachtelten PHP-Objekten. Als Erstes muss ein Dokument mit `simpleXML` eingelesen werden:

```
$feed = 'http://pear.php.net/rss.php';
$rss  = simplexml_load_file($feed);
```

Wenn Sie die Rückgabe der Funktion `simplexml_load_file()` auf den Wert `false` überprüfen, können Sie ermitteln, ob die XML-Datei erfolgreich eingelesen werden konnte.

```
if ($rss === false) {
    die("Konnte RSS-Feed nicht laden.");
}
```

War die Aktion erfolgreich, enthält die Variable `$rss` ein PHP-Objekt, mit dem Sie Zugriff auf alle in XML vorhandenen Daten haben. Um auf die einzelnen Daten zugreifen zu können, müssen Sie wissen, wie die XML-Datei strukturiert ist. Dazu öffnen Sie die URL *http://pear.php.net/rss.php* z.B. einfach in Ihrem Browser, der Ihnen dann den XML-Code anzeigt:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://purl.org/rss/1.0/"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
  <channel rdf:about="http://pear.php.net/">
    <link>http://pear.php.net/</link>
    <dc:creator>pear-webmaster@lists.php.net</dc:creator>
    <dc:publisher>pear-webmaster@lists.php.net</dc:publisher>
    <dc:language>en-us</dc:language>
    <items>
      <rdf:Seq>
        <rdf:li rdf:resource="http://pear.php.net/package/DB_DataObject/download/1.7.5/" />
        <rdf:li rdf:resource="http://pear.php.net/package/HTML_TreeMenu/download/1.2.0/" />
        <rdf:li rdf:resource="http://pear.php.net/package/DB_DataObject_FormBuilder/
                                                download/0.11.4/" />
      </rdf:Seq>
    </items>
    <title>PEAR: Latest releases</title>
    <description>The latest releases in PEAR.</description>
  </channel>
  <item rdf:about="http://pear.php.net/package/DB_DataObject/download/1.7.5/">
    <title>DB_DataObject 1.7.5</title>
    <link>http://pear.php.net/package/DB_DataObject/download/1.7.5/</link>
    <description>A few releases where skipped, this should be the amalgamated
                      upgrade</description>
    <dc:date>2005-03-01T22:52:57-05:00</dc:date>
  </item>
  <item rdf:about="http://pear.php.net/package/HTML_TreeMenu/download/1.2.0/">
    <title>HTML_TreeMenu 1.2.0</title>
    <link>http://pear.php.net/package/HTML_TreeMenu/download/1.2.0/</link>
    <description>Various bug fixes.</description>
    <dc:date>2005-03-01T21:52:20-05:00</dc:date>
  </item>
```

```
  <item rdf:about="http://pear.php.net/package/DB_DataObject_FormBuilder/download/0.11.4/">
    <title>DB_DataObject_FormBuilder 0.11.4</title>
    <link>http://pear.php.net/package/DB_DataObject_FormBuilder/download/0.11.4/</link>
    <description>Bugfixes</description>
    <dc:date>2005-03-01T20:02:24-05:00</dc:date>
  </item>
</rdf:RDF>
```

Der Zugriff auf die einzelnen Tags ist nun denkbar einfach: Wollen Sie z.B. den Titel des Channels ausgeben, müssen Sie lediglich den Pfad zum Titel im XML-Dokument auf das PHP-Objekt übertragen. In XML steht der Titel innerhalb der Tags `<rdf:RDF>`, `<channel>` und `<title>`, in SimpleXML verwenden Sie nun einfach:

```
echo $rss->channel->title;
```

wodurch PHP den String "`PEAR: Latest releases`" ausgibt. Genau nach dem gleichen Schema erfolgt auch der Zugriff auf die anderen Tags innerhalb des Dokuments:

```
echo $rss->channel->link;
```

gibt z.B. die URL des Channels aus.

Natürlich wollen Sie auf Ihrer Website nicht nur den Titel des RSS-Channels ausgeben, sondern auch die aktuellsten Nachrichten des Channels auflisten. Jede dieser Nachrichten steht innerhalb eines `<item>`-Tags, von denen beliebig viele in einer RSS-Datei auftreten können. Ein Zugriff über `$rss->item->title` ist also nicht möglich, da SimpleXML nicht weiß, auf welchen der `<item>`-Tags Sie zugreifen möchten.

Aber auch hier bietet SimpleXML eine seinem Namen entsprechende Lösung: Tritt ein Tag mehr als einmal auf derselben Ebene auf, wird dieser automatisch in ein Array konvertiert, und der Zugriff kann dann über die Array-Syntax erfolgen:

```
echo $rss->item[0]->title . "\n";
echo $rss->item[1]->title . "\n";
```

Damit werden die Textdaten innerhalb der `<title>`-Tags der ersten beiden Einträge ausgegeben, in diesem Fall erhalten Sie als Ausgabe also:

```
DB_DataObject 1.7.5
HTML_TreeMenu 1.2.0
```

Dieses Array kann natürlich auch in einer `foreach`-Schleife durchlaufen werden, damit alle Einträge nacheinander verarbeitet werden können:

```
foreach ($rss->item as $item) {
    // Item verarbeiten.
}
```

Innerhalb der Schleife enthält die Variable `$item` wiederum ein SimpleXML-Objekt, über das Sie auf alle Unterelemente über Objekt-Eigenschaften zugreifen können. So erhalten Sie über `$item->title` z.B. den Inhalt des `<title>`-Tags. Mit einer einfachen Schleife ist es also möglich, alle Einträge der RSS-Datei als HTML-Code auszugeben.

```
foreach ($rss->item as $item) {

    printf('<p><a href="%s">%s</a><br />%s</p>',
            $item->link, htmlspecialchars($item->title),
            htmlspecialchars($item->description));
}
```

SimpleXML ist nicht darauf beschränkt, XML aus Dateien zu laden, Sie können SimpleXML auch verwenden, um dynamisch erzeugte XML-Dokument zu parsen:

```
$xml = <<<EOD
<team name="JLA">
  <held name="Clark Kent">Superman</held>
  <held name="Bruce Wayne">Batman</held>
  <held name="Barry Allen">Der rote Blitz</held>
</team>
EOD;

$team = simplexml_load_string($xml);
if ($team === false) {
    die("Konnte XML nicht parsen.");
}
```

Steht der XML-Code in einer Variablen zu Verfügung, verwenden Sie statt `simplexml_load_file()` einfach `simplexml_load_string()` und übergeben die Variable, die den XML-Code enthält. Dieses Beispiel zeigt auch gleich ein weiteres Feature von SimpleXML. Neben den Decknamen der Superhelden enthalten die Tags auch zusätzlich den richtigen Namen im `name`-Attribut. Auf diese Attribute kann wiederum über die Array-Syntax zugegriffen werden:

```
echo "Name des Teams: " . $team['name'] . "\n";
foreach ($team->held as $held) {
    echo sprintf( "- %s ist %s\n", $held['name'], $held);
}
```

Statt eine Zahl als Key beim Array-Zugriff anzugeben, verwenden Sie hier den Namen des Attributs, auf das Sie zugreifen möchten. Nach Ausführen des Skripts erhalten Sie die folgende Ausgabe:

```
Name des Teams: JLA
- Clark Kent ist Superman
- Bruce Wayne ist Batman
- Barry Allen ist Der rote Blitz
```

SimpleXML ermöglicht nicht nur lesenden Zugriff auf XML-Dokumente, es ist auch möglich, bestehende Tags und Attribute zu verändern, wobei Sie ebenfalls nur die bekannte PHP-Syntax verwenden müssen:

```
$team->held[3]['name'] = 'Wally West';
```

Diese Zeile ändert den Namen des dritten Superhelden im Team in Wally West. Nachdem Sie den Wert des Attributs geändert haben, können Sie daraus auch wieder ein XML-Dokument erstellen, indem Sie die `asXML()`-Methode des `$team`-Objekts nutzen:

```
echo $team->asXML();
```

Übergeben Sie dieser Methode einen Dateinamen, wird das erzeugte XML-Dokument zusätzlich noch unter diesem Dateinamen gespeichert:

```
echo $team->asXML("jla.xml");
```

Die Datei *jla.xml* enthält dann dieses XML-Dokument:

```
<?xml version="1.0"?>
<team name="JLA">
  <held name="Clark Kent">Superman</held>
  <held name="Bruce Wayne">Batman</held>
  <held name="Wally West">Der rote Blitz</held>
</team>
```

Leider ist es mit SimpleXML nicht möglich, neue Knoten an ein XML-Dokument zu hängen.

Siehe auch

Rezept 14.4 zum Parsen von XML mit DOM; Rezept 14.5 zum Parsen von XML mit SAX; Rezept 14.8 zum Parsen von XML mit xmlReader; die Dokumentation zu `SimpleXML` unter *http://www.php.net/simplexml*.

14.7 Daten zwischen DOM und SimpleXML austauschen

Problem

Sie möchten XML-Dokumente mit DOM und SimpleXML verarbeiten und dabei immer die einfachere API für das aktuelle Problem nutzen.

Lösung

Verwenden Sie die Funktionen `simplexml_import_dom()` und `dom_import_simplexml()`, um XML-Knoten zwischen den beiden Erweiterungen auszutauschen.

```
$xmlJla = <<<EOD
<team name="JLA"><held name="Clark Kent">Superman</held><held name="Bruce Wayne">Batman</
held></team>
EOD;

$xmlAvengers = <<<EOD
<team name="Avengers"><held name="Steve Rogers">Captain America</held><held name="Tony
Stark">Iron Man</held></team>
EOD;

$simpleXmlJla = simplexml_load_string($xmlJla);
$simpleXmlAvengers = simplexml_load_string($xmlAvengers);
```

```
$dom   = new DOMDocument();
$dom->formatOutput = true;
$team = $dom->appendChild($dom->createElement('team'));
$team->setAttribute('name', 'Amalgam');

$superman = dom_import_simplexml($simpleXmlJla->held[0]);
$superman = $dom->importNode($superman, true);
$team->appendChild($superman);

$ironman  = dom_import_simplexml($simpleXmlAvengers->held[1]);
$ironman  = $dom->importNode($ironman, true);
$team->appendChild($ironman);

echo $dom->saveXML();
```

Diskussion

Auch wenn die SimpleXML-Erweiterung für viele Aufgaben die einfachste Lösung bietet, weist sie einige Schwächen auf. So ist es z.B. nicht möglich, neue Tags mithilfe der SimpleXML-Erweiterung zu erzeugen oder zwei XML-Dokumente zu einem zusammenzufügen. Mit der DOM-Erweiterung wäre diese Aufgabe allerdings innerhalb weniger Zeilen sehr einfach zu lösen. Da SimpleXML lediglich einen Wrapper um die DOM-Funktionen darstellt und mit den gleichen Datenstrukturen arbeitet, bieten beide Extensions Funktionen an, mit deren Hilfe Daten zwischen den Erweiterungen ausgetauscht werden können.

Im obigen Beispiel wird der Weg von SimpleXML zu DOM demonstriert. Es soll aus zwei XML-Dokumenten, die jeweils ein Superhelden-Team repräsentieren, ein Teil extrahiert und zu einem neuen XML-Dokument zusammengefügt werden.

Dazu werden zuerst beide XML-Dokumente in SimpleXML-Objekte eingelesen. Danach erstellen Sie ein neues `DOMDocument`-Objekt, das später das neue XML-Dokument werden soll. In diesem Dokument erstellen Sie zusätzlich ein neues Tag und hängen dieses als Wurzelelement an den XML-Baum:

```
$dom   = new DOMDocument();
$dom->formatOutput = true;
$team = $dom->appendChild($dom->createElement('team'));
$team->setAttribute('name', 'Amalgam');
```

Mit der Methode `setAttribute()` wird zusätzlich noch ein Attribut für den Wurzelknoten erzeugt.

An dieses Element möchten Sie nun die beiden Superhelden-Teams anhängen, die nicht als DOM-Dokumente, sondern nur als SimpleXML-Objekte zur Verfügung stehen. Dazu bietet PHP 5 die Funktion `dom_import_simplexml()`, die aus einem SimpleXML-Element ein DOMElement erzeugen kann.

```
// SimpleXML Element in DOMElement konvertieren.
$superman = dom_import_simplexml($simpleXmlJla->held[0]);
```

Dieser Methode kann entweder ein komplettes XML-Dokument oder auch nur wie in diesem Fall ein Teilknoten übergeben werden. Die Funktion liefert ein Objekt vom Typ `DOMElement` zurück, also das gleiche, das auch von der Methode `DOMDocument::createElement()` erzeugt wird. Damit Sie dieses DOMElement noch im zuvor erzeugten DOM-Baum verwenden können, müssen Sie diesen zusätzlich in den DOM-Baum importieren:

```
// DOMElement in den DOM-Baum importieren.
$superman = $dom->importNode($superman, true);
```

Danach verhält sich dieses DOMElement so, als wäre es durch das Dokument in `$dom` erzeugt worden. Um zu überprüfen, ob Sie das gewünschte Element erfolgreich importiert haben, können Sie es mit dem folgenden Code als XML-Fragment ausgeben:

```
echo $dom->saveXML($superman);
```

Die Ausgabe dieses Methodenaufrufs sieht folgendermaßen aus:

```
<held name="Clark Kent">Superman</held>
```

Nun bleibt nur noch, den Knoten an der richtigen Stelle in das Dokument einzuhängen und danach die gleiche Prozedur mit einem Knoten aus dem zweiten Dokument zu wiederholen:

```
// Knoten einhängen.
$team->appendChild($superman);

// SimpleXML-Element in DOMElement konvertieren.
$ironman  = dom_import_simplexml($simpleXmlAvengers->held[1]);
// DOMElement in den DOM-Baum importieren.
$ironman  = $dom->importNode($ironman, true);
// Knoten einhängen.
$team->appendChild($ironman);
```

Das erzeugte DOM-Dokument beinhaltet nun Elemente aus den beiden anderen XML-Dokumenten, die zuvor mit `simplexml_load_string()` eingelesen wurden, und kann sehr einfach als XML ausgegeben werden:

```
echo $dom->saveXML();

<?xml version="1.0"?>
<team name="Amalgam">
  <held name="Clark Kent">Superman</held>
  <held name="Tony Stark">Iron Man</held>
</team>
```

Bei der Erstellung des Teams haben Sie nun das Beste der beiden XML-Erweiterungen DOM und SimpleXML verwendet. Während Sie SimpleXML nutzen, um direkt auf einzelne Knoten des Dokuments zuzugreifen, nutzen Sie DOM, um ein neues Dokument zu erzeugen und abzuspeichern.

Möchten Sie nun gern über alle Mitglieder des neuen Teams iterieren, ist das mit dem DOM-Objekt, das Sie als Ergebnis erhalten haben, sehr aufwendig, wohingegen es mit

einem SimpleXML-Objekt sehr einfach zu erledigen wäre (vergleichen Sie hierzu Rezept 14.6 zum Parsen von XML mit SimpleXML). Natürlich könnten Sie das DOM-Dokument mit der Methode `saveXML()` in XML konvertieren und danach mit `simplexml_load_string()` wieder in ein SimpleXML-Objekt einlesen, allerdings bietet Ihnen PHP 5 hier eine einfachere Lösung durch die Funktion `simplexml_import_dom()`, die aus einem DOMElement ein SimpleXML-Objekt erzeugt:

```
$amalgam = simplexml_import_dom($team);
```

Dieses Objekt können Sie nun benutzen, als hätten Sie ein Dokument über eine der beiden `simplexml_load_*()`-Methoden erzeugt:

```
echo "Name des Teams: " . $amalgam['name'] . "\n";
foreach ($amalgam->held as $held) {
    echo sprintf( "- %s ist %s\n", $held['name'], $held);
}
```

Hätten Sie die gleiche Funktionalität mithilfe von DOM implementieren wollen, wäre der Aufwand sehr viel größer gewesen. Und da beide Erweiterungen mit der gleichen Basis arbeiten, geht beim Konvertieren zwischen DOM und SimpleXML kaum Rechenzeit verloren.

Siehe auch

Rezept 14.2 zum Generieren von XML mit DOM; Rezept 14.6 zum Parsen von XML mit SimpleXML; die Dokumentation zu `simplexml_import_dom()` unter *http://www.php.net/manual/en/function.simplexml-import-dom* und die Dokumentation zu `dom_import_simplexml()` unter *http://www.php.net/manual/en/function.dom-import-simplexml*.

14.8 Große XML-Dokumente einlesen (xmlReader)

Problem

Sie möchten ein XML-Dokument ereignisbasiert verarbeiten, ohne Callbacks schreiben zu müssen.

Lösung

Verwenden Sie die PECL-Extension xmlReader, die ab PHP 5.1 auch Teil der Standard-Distribution sein wird. Diese ereignisbasierte Art der Verarbeitung von XML-Dokumenten ist besonders bei sehr großen Dateien vorteilhaft, da das Dokument nicht vollständig in den Speicher geladen werden muss und dadurch Ressourcen gespart werden können.

```
$reader = new xmlReader();
$reader->open('buecher.xml');
```

```
while ($reader->read()) {
    switch ($reader->nodeType) {
        case XMLREADER_ELEMENT:
            if ($reader->name == 'buecher') {
                continue;
            }
            if ($reader->name == 'buch') {
                print "Neues Buch\n";
                continue;
            }
            print " " . ucfirst($reader->name) . " : ";
            break;
        case XMLREADER_TEXT:
            print $reader->value . "\n";
            break;
    }
}
$reader->close();
```

Diskussion

xmlReader ist eine PECL-Extension. Ab PHP-Version 5.1 wird diese allerdings standardmäßig mit PHP ausgeliefert und muss beim Kompilieren lediglich mit `--with-xmlreader` aktiviert werden. Diese Extension verwendet eine sogenannte XML-Pull-API, die ein Zwischenstück zu DOM und SAX bildet. Wie beim SAX-basierten Parsen muss nicht das ganze Dokument eingelesen werden, sondern die Extension arbeitet cursorbasiert. Dieser Cursor wandert vom Anfang des Dokuments bis ans Ende und kann nicht wieder nach hinten versetzt werden. Das Dokument wird also sequenziell verarbeitet. Allerdings müssen Sie bei xmlReader keine Callbacks registrieren, sondern Sie steuern selbst, wie weit das Dokument geparst wird, und können den Prozess überall abbrechen. Sie können dann an jeder Stelle herausfinden, ob sich der Cursor momentan über einem XML-Element, über Textdaten oder einem Attribut befindet.

Um ein XML-Dokument mit xmlReader zu parsen, erzeugen Sie zunächst eine Instanz der xmlReader-Klasse und lesen das XML-Dokument mit der `open()`-Methode ein:

```
$reader = new xmlReader();
$reader->open('buecher.xml');
```

Danach lassen Sie das Dokument in einer `while`-Schleife bis zum Ende des Dokuments durchlaufen, indem Sie die `read()`-Methode aufrufen, bis diese `false` zurückliefert. Dabei wird der interne Cursor von einem Knoten zum nächsten bewegt. Um festzustellen, auf was für einer Art Knoten (z.B. Element oder Text) sich der Zeiger im Moment befindet, verwenden Sie die Eigenschaft `$reader->nodeType`. Ist dieser Wert auf `XMLREADER_ELEMENT` gesetzt, befindet sich der Cursor über einem öffnenden XML-Tag, eine Liste aller vordefinierten Konstanten und deren Entsprechung finden Sie in Tabelle 14-1.

Tabelle 14-1: nodeType-Werte in xmlReader

Konstante	Knotentyp
XMLReader::NONE	kein Knoten unter dem Cursor
XMLReader::ELEMENT	öffnendes Tag
XMLReader::ATTRIBUTE	Attribut
XMLReader::TEXT	Text zwischen zwei Tags
XMLReader::CDATA	Character Data (<![CDATA[...]]>)
XMLReader::ENTITY_REF	XML-Entity Referenz
XMLReader::ENTITY	XML-Entity
XMLReader::PI	Processing Instruction (z.B. <?php ... ?>)
XMLReader::COMMENT	XML-Kommentar
XMLReader::DOC	Dokumentknoten
XMLReader::DOC_TYPE	Dokumenttyp-Knoten
XMLReader::DOC_FRAGMENT	Dokumentfragment-Knoten
XMLReader::WHITESPACE	Whitespace Knoten
XMLReader::SIGNIFICANT_WHITESPACE	Significant Whitespace Knoten
XMLReader::END_ELEMENT	End Element
XMLReader::END_ENTITY	schließendes Tag
XMLReader::XML_DECLARATION	XML-Deklaration

Je nach Knotentyp können Sie auf andere Eigenschaften des xmlReader-Objekts zugreifen, um weitere Informationen über den aktuellen Knoten auszulesen. Befindet sich der Cursor aktuell über einem XML-Element, können Sie z.B. über die Eigenschaft `$reader->name` den Namen des Elements auslesen. Befindet sich der Cursor über einem Textknoten, erhalten Sie den Text über die Eigenschaft `$reader->value`. Folgendes Skript gibt die Struktur eines XML-Dokuments aus:

```
$reader = new xmlReader();
$reader->open('buecher.xml');

while ($reader->read()) {
    switch ($reader->nodeType) {
        case XMLREADER_ELEMENT:
            printf("Öffnendes Tag: %s\n", $reader->name);
            break;
        case XMLREADER_END_ELEMENT:
            printf("Schließendes Tag: %s\n", $reader->name);
            break;
        case XMLREADER_TEXT:
            printf("Textknoten: %s\n", $reader->value);
            break;
      }
}
$reader->close();
```

Dabei wird für jedes öffnende oder schließende Tag und für jeden Textknoten eine Zeile ausgegeben. Als Beispiel wird die Datei *buecher.xml* verwendet, die bereits beim Parsen von XML-Dokumenten mit DOM (Rezept 14.4) verwendet wurde:

```
<buecher>
    <buch>
        <titel>PHP Kochbuch</titel>
        <autor>Sklar</autor>
        <autor>Trachtenberg</autor>
        <thema>PHP</thema>
    </buch>
    <buch>
        <titel>Perl Kochbuch</titel>
        <autor>Christiansen</autor>
        <autor>Torkington</autor>
        <thema>Perl</thema>
    </buch>
</buecher>
```

Dabei erhalten Sie folgende Ausgabe:

```
Öffnendes Tag: buecher
Öffnendes Tag: buch
Öffnendes Tag: titel
Textknoten: PHP Kochbuch
Schließendes Tag: titel
Öffnendes Tag: autor
Textknoten: Sklar
Schließendes Tag: autor
Öffnendes Tag: autor
Textknoten: Trachtenberg
Schließendes Tag: autor
Öffnendes Tag: thema
Textknoten: PHP
Schließendes Tag: thema
Schließendes Tag: buch
...
Schließendes Tag: buecher
```

Wurde das ganze Dokument durchlaufen, schließen Sie die geöffnete Datei durch einen Aufruf der `close()`-Methode. xmlReader kann natürlich ein XML-Dokument direkt aus einer Variablen lesen, falls dieses dynamisch generiert wurde. Dazu verwenden Sie die Methode `XML()` anstatt `open()`:

```
$xml = <<<EOD
<team name="JLA">
    <held alias="Superman">
        <name>Clark Kent</name>
        <stadt>Metropolis</stadt>
    </held>
    <held alias="Batman">
        <name>Bruce Wayne</name>
        <stadt>Gotham City</stadt>
```

```
    </held>
    <held alias="Der Rote Blitz">
        <name>Wally West</name>
        <stadt>Keystone City</stadt>
    </held>
</team>
EOD;

$reader = new xmlReader();
$reader->XML($xml);
```

Neben der Anforderung, ein Dokument direkt aus einem String zu lesen, kommt hier noch eine weitere auf das Skript zu, da in diesem Dokument wichtige Informationen auch in Attributen gespeichert werden. xmlReader bietet Ihnen zwei Möglichkeiten, auf Attribute zuzugreifen:

- Wenn Sie wissen, wie die gesuchten Attribute heißen, können Sie auf den Wert eines Attributs über den Namen zugreifen, indem Sie die Methode `getAttribute("name")` verwenden.
- Wissen Sie nichts über die Struktur des XML-Dokuments und kennen somit auch die Namen der Attribute nicht, bietet Ihnen xmlReader die Methoden `moveToFirstAttribute()`, und `moveToNextAttribute()` um durch die einzelnen Attribute zu iterieren.

In diesem Beispiel wissen Sie, dass jeder `<held>`-Tag ein Attribut `alias` hat, und können deshalb die Methode `getAttribute` verwenden, um auf das Attribut zuzugreifen. Möchten Sie nun den Inhalt der XML-Datei ausgeben, durchlaufen Sie das Dokument nach dem Parsen erneut mit einer `while`-Schleife. In dieser Schleife verwenden Sie eine `switch`-Anweisung, um zwischen öffnenden Tags und Textknoten zu unterscheiden.

```
while ($reader->read()) {
    switch ($reader->nodeType) {
        case XMLREADER_ELEMENT:
            break;
        case XMLREADER_TEXT:
            print $reader->value . "\n";
            break;
      }
}
```

Handelt es sich um einen Textknoten, wird der Text einfach ausgegeben. Handelt es sich jedoch um ein Tag, muss zuerst überprüft werden, um was für ein Tag es sich handelt, bevor es verarbeitet werden kann:

- Das `<team>`-Tag kann einfach ignoriert werden.
- Das `<held>`-Tag kann ignoriert werden, allerdings muss zuvor der Name des Helden aus dem Attribut `"alias"` extrahiert und ausgegeben werden.
- Bei allen anderen Tags soll der Name des Tags ausgegeben werden, wobei allerdings der erste Buchstabe des Tags in einen Großbuchstaben konvertiert wird.

Diese Abfrage wird in PHP folgendermaßen umgesetzt:

```
if ($reader->name === 'team') {
    continue;
}
if ($reader->name === 'held') {
    print $reader->getAttribute('alias') . "\n";
    continue;
}
print ucfirst($reader->name) . " : ";
```

Nach Beendigung der Schleife muss nur noch das geöffnete XML-Dokument geschlossen werden:

```
$reader->close();
```

Starten Sie nun das Skript, erhalten Sie die folgende Ausgabe:

```
Superman
Name : Clark Kent
Stadt : Metropolis
Batman
Name : Bruce Wayne
Stadt : Gotham City
Der Rote Blitz
Name : Wally West
Stadt : Keystone City
```

Die xmlReader-Erweiterung bietet jedoch noch weitere Funktionalitäten. In vielen Fällen wollen Sie nur einen Teil eines XML-Dokuments verarbeiten. Angenommen, Sie möchten nur die Daten des Roten Blitz ausgeben. Da xmlReader cursorbasiert arbeitet, müssen Sie trotzdem das vollständige Dokument Tag für Tag durchlaufen, was die Performance Ihres Skripts verschlechtert. Denn eigentlich können Sie beim Auffinden des Tags `<held alias="Superman">` schon entscheiden, dass alle Tags, die innerhalb dieses Tags verwendet werden, nicht eingelesen werden müssten. Die `read()`-Methode liest jedoch Tag für Tag des Dokuments ein, wodurch der Cursor auch bei den Tags `<name>` und `<stadt>` sowie den darunter liegenden Textknoten stoppen würde. Aus diesem Grund wurde die `next()`-Methode implementiert, die den Cursor direkt zum nächsten Knoten derselben Ebene weiterbewegt. In diesem Fall würde der Cursor also erst wieder beim nächsten `<held>`-Tag anfangen. Es ist also möglich, die Performance des Skripts enorm zu verbessern, indem Sie die Schleife leicht modifizieren:

```
while ($reader->read()) {
    switch ($reader->nodeType) {
        case XMLREADER_ELEMENT:
            if ($reader->name === 'team') {
                continue;
            }
            if ($reader->name === 'held') {
                if ($reader->getAttribute('alias') !== 'Der Rote Blitz') {
                    $reader->next();
                    continue;
```

```
                }
                print $reader->getAttribute('alias') . "\n";
                continue;
            }
            print ucfirst($reader->name) . " : ";
            break;
        case XMLREADER_TEXT:
            print $reader->value . "\n";
            break;
    }
}
```

Steht der Cursor nun über einem öffnenden `<held>`-Tag, wird überprüft, ob das Attribut `"alias"` den Wert `"Der Rote Blitz"` hat. Ist dies nicht der Fall, wird der Cursor zum nächsten `<held>`-Tag weiterbewegt, und die Schleife wird fortgesetzt.

Siehe auch

Rezept 14.4 zum Parsen von XML mit DOM; Rezept 14.5 zum Parsen von XML mit SAX; Rezept 14.6 zum Parsen von XML mit SimpleXML; Rezept 14.11 zum Selektieren einzelner Daten aus XML; die xmlReader-Homepage unter *http://pecl.php.net/package/xmlreader*.

14.9 XML mit XSLT transformieren

Problem

Sie haben ein XML-Dokument und ein XSL-Stylesheet. Sie möchten das Dokument mit XSLT transformieren und das Ergebnis ausgeben. Auf diese Weise können Sie Stylesheets auf Ihre Daten anwenden und unterschiedliche Versionen Ihrer Inhalte für verschiedene Medien erzeugen.

Lösung

Verwenden Sie die XSL-Erweiterung zu PHP:

```
// XML-Dokument laden.
$xml = new DOMDocument;
$xml->load('data.xml');

// XSL-Stylesheet laden.
$xsl = new DOMDocument;
$xsl->load('stylesheet.xsl');

// Neuen XSLT-Prozessor erzeugen und Stylesheet einbinden.
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl);
```

```
$ergebnis = $proc->transformToXML($xml);
if ($ergebnis === false) {
    die('Transformation nicht möglich');
}
print $ergebnis;
```

Der transformierte Text wird in `$ergebnis` gespeichert.

Diskussion

XML-Dokumente beschreiben die Inhalte von Daten, aber sie enthalten keine Informationen darüber, wie diese Daten dargestellt werden sollen. Werden XML-Inhalte jedoch mit einem Stylesheet kombiniert, das in XSL (*eXtensible Stylesheet Language*) beschrieben ist, kann der Inhalt nach bestimmten visuellen Regeln angezeigt werden.

Die Verbindung zwischen XML und XSL bildet XSLT (eXtensible Stylesheet Language Transformations). Derartige Transformationen wenden auf Ihre XML-Daten eine Reihe von Regeln an, die in dem Stylesheet aufgeführt sind. Genau so, wie PHP Ihren Code zerlegt und mit Benutzereingaben kombiniert, um eine dynamische Seite zu erzeugen, verwendet ein XSLT-Programm XSL und XML zur Ausgabe einer neuen Seite, die neuen XML- oder HTML-Code oder ein anderes von Ihnen beschriebenes Format enthält.

Es sind einige XSLT-Programme verfügbar, von denen jedes andere Eigenschaften und Beschränkungen aufweist. Die XSLT-Unterstützung in PHP 5 basiert auf der libxslt-Bibliothek[2] und wird standardmäßig mit PHP 5 ausgeliefert, sie muss lediglich mit *--enable-xsl* aktiviert werden.

Zur Verarbeitung der Dokumente sind einige wenige Schritte erforderlich. Als Erstes müssen Sie einen XSLT-Prozessors erzeugen. Sowohl das zu transformierende XML-Dokument als auch das XML-Stylesheet müssen als DOM-Objekt vorliegen:

```
// XML-Dokument laden.
$xml = new DOMDocument;
$xml->load('data.xml');

// XSL-Stylesheet laden.
$xsl = new DOMDocument;
$xsl->load('stylesheet.xsl');

// Neuen XSLT-Prozessor erzeugen.
$proc = new XSLTProcessor;
```

Mit der Methode `XSTLProcessor::importStylesheet()` übergeben Sie den DOM-Baum, der das Stylesheet enthält, an den XSLT-Prozessor:

```
$proc->importStyleSheet($xsl);
```

2 *http://xmlsoft.org/XSLT/*

Danach verwenden Sie die Methode `transformToXml()` und übergeben den DOM-Baum des XML-Dokuments, das Sie transformieren möchten. Diese Methode liefert entweder das Ergebnis der Transformation als String zurück oder `false`, falls bei der Transformation ein Fehler aufgetreten ist. Sie wird meistens verwendet, wenn XSLT genutzt wird, um XML in HTML zu transformieren und dieses dann an den Browser zu schicken.

Oft möchten Sie aber auch das entstandene XML- oder HTML-Dokument zwischenspeichern, bevor Sie es an den Browser schicken, dazu können Sie die Methode `transformToURI()` verwenden:

```
$proc->transformToURI($xml, 'file://ausgabe.html');
```

Als zweiten Parameter geben Sie hier den Dateinamen an. Sie können dabei auch die Streams-Funktionalität von PHP verwenden.

Es gibt noch eine letzte Möglichkeit, wie XML-Dokumente transformiert werden können, und zwar kann der XSLT-Prozessor auch wieder einen DOM-Baum des transformierten Dokuments zurückliefern, anstatt diesen in einen String zu serialisieren. Diese Methode kann mehrere Vorteile haben:

- Sie möchten das Dokument mehrmals transformieren.
- Sie verwenden XSLT nicht, um eine Ausgabe in HTML zu erreichen, sondern um Inhalte in XML-Dokumenten zu sortieren oder neu zu organisieren. Danach möchten Sie über DOM Zugriff auf die Daten erhalten.

Wollen Sie anstatt eines Strings einen DOM-Baum erhalten, verwenden Sie:

```
$xmlNeu = $proc->transformToDoc($xml)
$buecher = $xmlNeu->getElementsByTagName('buch')
```

Möchten Sie im XSL-Stylesheet auf Variablen reagieren, die von Ihrem PHP-Skript gesetzt werden können, verwenden Sie dazu Parameter, die mit der Methode `XSLTProcessor::setParameter()` gesetzt werden können:

```
$proc->setParameter( '', 'autor', 'Stephan Schmidt' );
$proc->importStyleSheet($xsl);
$ergebnis = $proc->transformToXML($xml);
```

Als Erstes übergeben Sie hierbei den Namespace, in dem der Parameter gelten soll, als Zweites den Namen und als letzten Parameter den Wert, den der Parameter annehmen soll.

In Ihrem Stylesheet können Sie diese Parameter z.B. in den Tags `<xsl:if/>` oder `<xsl:apply-templates/>` verwenden:

```
<xsl:template match="buecher">
    <xsl:apply-templates select="buch/autor=$autor"/>
</xsl:template>
```

Damit transformieren Sie z.B. nur Bücher, die von dem Autor geschrieben wurden, der vom PHP-Skript übergeben wurde.

Siehe auch

Rezept 14.4 zum Parsen von XML mit DOM; Rezept 14.10 zum Verwenden von PHP-Funktionen in XSL-Stylesheets; die Dokumentation zu den XSL-Funktionen unter *http://www.php.net/xsl*; *XSLT* von Doug Tidwell (O'Reilly Verlag).

14.10 PHP-Funktionen in XSL-Stylesheets verwenden

Problem

Sie möchten PHP-Funktionen in einem XSL-Stylesheet verwenden, da Sie z.B. ein Datum in ein lesbares Format konvertieren möchten.

Lösung

Aktivieren Sie die Verwendung von PHP-Funktionen in XSL-Stylesheets mit der Methode `XSLTProcessor::registerPhpFunctions()` und verwenden Sie den Namespace *http://www.php.net/xsl*, um auf PHP-Funktionen in Ihren Stylesheets zuzugreifen.

```
function formatiereDatum($datum)
{
    // Datum in Timestamp konvertieren.
    $timestamp = strtotime($datum);
    // Timestamp in deutsches Datum konvertieren.
    $datum     = date('d.m.Y H:i:s', $timestamp);
    return $datum;
}

// XSL und Daten laden.
$xsl = DOMDocument::load("stylesheet-php.xsl");
$xml = DOMDocument::load("daten.xml");

// XSLT-Prozessor erzeugen.
$proc = new XLSTProcessor();
// Verwendung von PHP-Funktionen erlauben.
$proc->registerPhpFunctions();

$xsl = $proc->importStylesheet($xsl);
$newdom = $proc->transformToDoc($xml);

print $newdom->saveXML();
```

Greifen Sie nun auf die Funktion `formatiereDatum()` aus dem XSL-Stylesheet zu:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                                                    xmlns:php="http://php.net/xsl">
<xsl:output method="html" encoding="ISO-8859-1" indent="yes"/>
<xsl:template match="daten">
```

```
    <ul>
    <xsl:for-each select="datum">
        <li><xsl:value-of select="php:functionString('formatiereDatum', .)"/></li>
    </xsl:for-each>
    </ul>
</xsl:template>
</xsl:stylesheet>
```

Diskussion

Die XSL-Erweiterung ermöglicht Ihnen, PHP-Funktionen aus Ihren XSL-Stylesheets heraus aufzurufen. Dabei können sowohl PHP-interne Funktionen wie z.B. `strrev()`, aber auch benutzerdefinierte Funktionen verwendet werden. Diese Funktionalität öffnet natürlich Sicherheitslücken, da Sie Ihren XSL-Programmierern ermöglichen, PHP-Code auszuführen. Besonders riskant ist es, wenn Sie unbekannten Benutzern ermöglichen, Stylesheets auf Ihren Server mit einem Datei-Upload-Formular hochzuladen. Aus diesem Grund muss diese Funktionalität zuerst für jeden XSLT-Prozessor aktiviert werden.

Sie möchten ein Dokument transformieren, das verschiedene Daten in einem für Computer lesbaren Format beinhaltet, wie es z.B. von MySQL verwendet wird:

```
<daten>
    <datum>1974-05-12 06:34:00</datum>
    <datum>2005-03-03 15:37:34</datum>
</daten>
```

Diese Daten sollen nun durch ein XSL-Stylesheet in eine HTML-Liste transformiert werden, wozu Sie das folgende XSL-Dokument nehmen:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="ISO-8859-1"/>
<xsl:template match="daten">
    <ul>
    <xsl:for-each select="datum">
        <li><xsl:value-of select="."/></li>
    </xsl:for-each>
    </ul>
</xsl:template>
</xsl:stylesheet>
```

Das Ergebnis der Transformation ist dieses HTML-Dokument:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<ul>
    <li>1974-05-12 06:34:00</li>
    <li>2005-03-03 15:37:34</li>
</ul>
```

Dabei wird einfach jedes `<datum>`-Tag in ein `<li>`-Tag konvertiert, und diese werden wiederum von `<ul>`-Tags eingeschlossen. `<html>`-, `<head>`- und `<body>`-Tags wurden aus Gründen der Lesbarkeit weggelassen.

Das Datumsformat ist zwar optimal, wenn es durch Programme weiterverarbeitet oder sortiert werden soll, wird es allerdings einem Benutzer dargestellt, erwartet diesen eher ein Datum in der Form: Do, 3.3.2005. Als erfahrener PHP-Programmierer ist es für Sie ein Leichtes, aus dem ursprünglichen Format das gewünschte Format zu machen. PHP bietet hierzu die Funktion `strtotime()`, um das Datum in einen Unix-Zeitstempel zu konvertieren, und danach können Sie die Funktion `date()` verwenden, um diesen Zeitstempel nach Ihren Wünschen zu formatieren:

```
$timestamp = strtotime($datum);
$datum = date('D, j.n.Y', $timestamp);
```

Diese Funktionalität in XSL zu programmieren ist nahezu unmöglich, der beste Weg ist also, direkt aus dem XSL-Stylesheet das Datum an PHP zu übergeben und das neu formatierte Datum in das resultierende XHMTL-Dokument einzufügen. Dazu müssen Sie zuerst eine PHP-Funktion schreiben, die die benötigte Funktionalität bereitstellt:

```
function formatiereDatum($datum)
{
    $timestamp = strtotime($datum);
    $datum = date('D, j.n.Y', $timestamp);
    return $datum;
}
```

Damit Sie diese PHP-Funktion nun in Ihren Stylesheets nutzen können, sind zwei Änderungen an Skript und Stylesheet nötig. Als Erstes müssen Sie nach Erzeugen des XSLT-Prozessors die Nutzung von PHP-Funktionen freischalten:

```
$proc = new XLSTProcessor();
$proc->registerPhpFunctions();
```

Danach passen Sie das Stylesheet an, indem Sie einen Namespace für PHP deklarieren:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                                                    xmlns:php="http://php.net/xsl">
```

Um einen Namespace für PHP-Funktionen zu deklarieren, müssen Sie die URL *http://www.php.net/xsl* nutzen, in diesem Fall wird das Kürzel `php` verwendet. Damit Ihre Funktion nun mit den einzelnen Daten aufgerufen wird, muss auch das Stylesheet angepasst werden:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                                                    xmlns:php="http://php.net/xsl">
<xsl:output method="html" encoding="ISO-8859-1" indent="yes"/>
<xsl:template match="daten">
    <ul>
    <xsl:for-each select="datum">
        <li><xsl:value-of select="php:functionString('formatiereDatum', .)"/></li>
    </xsl:for-each>
    </ul>
</xsl:template>
</xsl:stylesheet>
```

Anstatt innerhalb des -Tags nur den Inhalt des aktuellen Tags einzufügen, verwenden Sie `php:functionString('formatiereDatum', .)`. Durch den Namespace `php` weiß der XSLT-Prozessor, dass eine Funktionalität von PHP genutzt werden soll. Danach können Sie eine Funktion übergeben, die genutzt werden soll, im Moment können Sie hier zwei Werte verwenden:

- `function()`, um eine PHP-Funktion aufzurufen und einen Knoten des Dokuments als DOM-Fragment zu übergeben
- `functionString()`, um eine PHP-Funktion aufzurufen und einen Knoten als String zu übergeben.

In beiden Fällen müssen Sie als Erstes einen String mit dem Namen der PHP-Funktion übergeben, die aufgerufen werden soll. Als zweiten Parameter geben Sie einen XPath-Ausdruck an, der den Knoten bestimmt, der an die PHP-Funktion übergeben werden soll. Im Beispiel soll einfach der Inhalt das aktuellen Knotens als String übergeben werden; deshalb wird `php:functionString()` verwendet und als zweiten Parameter `"."` übergeben, das den aktuellen Knoten bezeichnet. Der String, den die PHP-Funktion dann zurückliefert, wird an dieser Stelle im Ausgabestrom eingefügt, wodurch folgender XHTML-Code entsteht:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<ul xmlns:php="http://php.net/xsl">
    <li>Son, 12.5.1974</li>
    <li>Don, 3.3.2005</li>
</ul>
```

Je nach Einstellung Ihres Servers werden auch englische Kürzel für die Wochentage verwendet.

Auf diese Art können beliebige PHP-Funktionen in XSL-Stylesheets eingesetzt werden. Sie sollten allerdings immer daran denken, dass diese Stylesheets dann leider nicht mehr portabel sind, sondern nur noch mit PHP verwendet werden können.

Falls Sie Texte mit Umlauten verwenden, sollten Sie darauf achten, dass der XSLT-Prozessor die Texte UTF-8-kodiert an die PHP-Funktionen übergibt und dass diese zuerst dekodiert werden müssen, bevor Sie sie weiterverarbeiten. Möchten Sie z.B. eine Funktion verwenden, die den ersten Buchstaben eines Texts in Großbuchstaben konvertiert, sollte diese folgendermaßen aussehen:

```
function ucFirstXsl($text)
{
    $text = utf8_decode($text);
    $text = ucfirst($text);
    $text = utf8_encode($text);
    return $text;
}
```

Siehe auch

Rezept 14.9 zum Transformieren von XML mit XSLT; die Dokumentation zu `XSLTProcessor::registerPHPFunctions()` unter *http://www.php.net/manual/xsltprocessor.registerphpfunctions.php*; die Dokumentationen zu `utf8_encode()` und `utf8_encode()` unter *http://www.php.net/manual/de/function.utf8-encode* und *http://www.php.net/manual/de/function.utf8-decode*.

14.11 Informationen aus einem XML-Dokument selektieren (XPath)

Problem

Sie möchten auf einfache Art auf bestimmte Informationen in einem XML-Dokument abhängig von Tag-Namen und Attributwerten zugreifen.

Lösung

Verwenden Sie die XPath-Funktionen der DOM-Erweiterung, um einzelne Knoten zu extrahieren und weiterzuverarbeiten.

```
$xml = <<<EOD
<team name="JLA">
    <held alias="Superman">
        <name>Clark Kent</name>
        <stadt>Metropolis</stadt>
    </held>
    <held alias="Batman">
        <name>Bruce Wayne</name>
        <stadt>Gotham City</stadt>
    </held>
    <held alias="Der Rote Blitz">
        <name>Wally West</name>
        <stadt>Keystone City</stadt>
    </held>
</team>
EOD;

// Neues Dokument erzeugen.
$dom = new DOMDocument();
$dom->preserveWhiteSpace = false;
$dom->formatOutput = true;

// XML parsen.
$dom->loadXML($xml);

// XPath-Objekt erzeugen.
```

```
$xpath = new DOMXPath($dom);

// Alle Städte extrahieren.
$query = '//stadt';
$nodes = $xpath->query($query);

// Namen der Städte ausgeben.
foreach ($nodes as $stadt) {
   echo $stadt->textContent . "\n";
}

// Nur Batman extrahieren.
$query = '/team/held[@alias="Batman"]';
$batman = $xpath->query($query);

// XML des Batman-Knotens erzeugen.
print $dom->saveXML($batman->item(0));
```

Oder nutzen Sie die `xpath()`-Methode eines SimpleXML-Objekts:

```
$sx = simplexml_load_string($xml);

// Alle Städte extrahieren.
$query = '//stadt';
$nodes = $sx->xpath($query);

// Namen der Städte ausgeben.
foreach ($nodes as $stadt) {
   print $stadt . "\n";
}

// Nur Batman extrahieren.
$query = '/team/held[@alias="Batman"]';
$batman = $sx->xpath($query);

// XML des Batman-Knotens erzeugen.
print $batman[0]->asXML();
```

Natürlich macht SimpleXML auch bei der Anwendung von XPath seinem Namen alle Ehre und wartet mit einer einfacheren API auf.

Diskussion

XPath ist eine Sprache, mit der Sie auf Teile eines XML-Dokuments gezielt zugreifen können. XPath macht sich die Baumstruktur von XML zunutze und ermöglicht, einzelne Äste oder Knoten zu adressieren. Damit können Sie z.B. alle Tags mit dem Namen `stadt` extrahieren, unabhängig von der Position in XML-Dokument (`//stadt`), oder auch alle Tags, bei denen ein bestimmtes Attribut einen vordefinierten Wert hat (`//held [@name="Batman"]`).

Damit ist XPath die Entsprechung zu SQL bei relationalen Datenbanken. Eine Einführung in XPath finden Sie in *XML in a Nutshell*, das im O'Reilly Verlag erschienen ist.

Die DOM-Erweiterung von PHP 5 bietet eine DOMXPath-Klasse, mit der Sie XPath-Anfragen an ein XML-Dokument stellen können. Dazu muss das XML-Dokument zuerst als DOM-Baum zur Verfügung stehen:

```
$dom = new DOMDocument();
$dom->preserveWhiteSpace = false;
$dom->formatOutput = true;

// XML parsen.
$dom->loadXML($xml);
```

Danach erzeugen Sie ein neues DOMXPath-Objekt und übergeben im Konstruktor das DOM-Objekt, auf das Sie die Abfragen anwenden möchten:

```
$xpath = new DOMXPath($dom);
```

Dieses Objekt bietet in PHP 5.0 bisher nur die Methode query(), der Sie einen XPath-Ausdruck übergeben müssen. Wollen Sie wissen, in welchen Städten bereits Superhelden wirken, können Sie das über den XPath-Ausdruck '//stadt' extrahieren. Die zwei Slashes (//) stehen für einen beliebigen Pfad im XML-Dokument und stadt steht für das XML-Tag <stadt>. Diesen Ausdruck übergeben Sie nun an die query()-Methode des DOMXPath-Objekts:

```
$query = '//stadt';
$nodes = $xpath->query($query);
```

Diese Methode liefert immer ein DOMNodeList-Objekt zurück, auch wenn nur ein XML-Knoten vom XPath-Ausdruck adressiert wurde. Durch Objekt-Überladung können Sie dieses Objekt in einer foreach-Schleife allerdings wie ein Array verwenden:

```
foreach ($nodes as $stadt) {
   print $stadt->textContent . "\n";
}
```

Das DOMNodeList-Objekt enthält beliebig viele Objekte vom Typ DOMNode, über die Eigenschaft $nodes->length können Sie ermitteln, wie viele Knoten zurückgegeben wurden. Bei dieser Abfrage handelt es sich bei jedem Knoten um ein XML-Element, da der XPath-Ausdruck ja nur Knoten mit dem Tag-Namen <stadt> extrahiert. Um nun den Text innerhalb des Elementknotens auszugeben, können Sie die PHP-spezifischere Erweiterung des DOMs textContent verwenden. Möchten Sie portableren Code schreiben, verwenden Sie firstChild->nodeValue:

```
foreach ($nodes as $stadt) {
   print $stadt->firstChild->nodeValue . "\n";
}
```

Wollen Sie nur einen Superhelden aus dem Dokument extrahieren, können Sie auch Attribute im XPath-Ausdruck verwenden. Der Ausdruck "/team/held" extrahiert z.B. alle <held>-Tags aus dem Dokument. Möchten Sie diese Abfrage auf den Helden beschränken, dessen Alias "Batman" ist, erweitern Sie den XPath-Ausdruck zu "/team/held[@alias="Batman"]". Durch die eckigen Klammern können Sie Bedingungen definie-

ren, und durch das @-Zeichen greifen Sie auf Attribute des aktuellen Knoten zu. Die Anfrage wird wiederum mit der Methode query() auf das XML-Dokument angewandt:

```
$query = '/team/held[@alias="Batman"]';
$batman = $sx->xpath($query);
```

Auch dieses Mal liefert die Methode ein DOMNodeList-Objekt zurück. Allerdings wollen Sie dieses Mal nicht alle Knoten in einer Schleife durchlaufen lassen, da Sie davon ausgehen, dass nur ein Knoten zurückgegeben wurde. Durch die Methode item() können Sie direkt auf einen Knoten der Liste zugreifen und diesen z.B. wieder in XML konvertieren, indem Sie den Knoten an die saveXML()-Methode des DOM-Objekts übergeben:

```
// XML des Batman-Knotens erzeugen.
print $dom->saveXML($batman->item(0));
```

Das Ergebnis dieser Methode ist:

```
<held alias="Batman">
  <name>Bruce Wayne</name>
  <stadt>Gotham City</stadt>
</held>
```

Seit PHP 5.1 bietet die DOM-Erweiterung eine weitere Methode, um XPath-Abfragen zu senden. Mit der Methode evaluate() können auch Anfragen gestellt werden, die keine DOMNodelist zurückliefern, da z.B. die count()-Funktion von XPath verwendet wird, um Knoten zu zählen. Möchten Sie z.B. wissen, wie viele <stadt>-Elemente in einem Dokument auftauchen, können Sie dazu den folgenden Code verwenden:

```
// Städte zählen.
$query = 'count(//stadt)';
$anzahl = $xpath->evaluate($query);

print "$anzahl\n";
```

Unter PHP 5.1 gibt die Methode evaluate() den Integer-Wert "3" zurück, würden Sie den XPath-Ausdruck mit den query()-Methode verwenden, bekämen Sie ein leeres DOMNodeList-Objekt zurück.

Da die SimpleXML-Erweiterung auf der DOM-Erweiterung basiert, bietet auch SimpleXML Unterstützung für XPath. Dazu bietet SimpleXML die Methode xpath(), mit der XPath-Anfragen an das Dokument gestellt werden können. Als Erstes müssen Sie natürlich das XML-Dokument mit simplexml_load_file() oder simplexml_load_string() einlesen:

```
$sx = simplexml_load_string($xml);
```

Danach können Sie genau die gleichen XPath-Ausdrücke verwenden, die Sie auch schon bei der Verwendung von DOM genutzt haben.

```
// Alle Städte extrahieren.
$query = '//stadt';
$nodes = $sx->xpath($query);
```

Diese Methode liefert natürlich kein DOMNodeList-Objekt zurück, sondern ein ganz normales PHP-Array, das SimpleXML-Objekte enthält:

```
array(3) {
  [0]=>
  object(SimpleXMLElement)#2 (1) {
    [0]=>
    string(10) "Metropolis"
  }
  [1]=>
  object(SimpleXMLElement)#3 (1) {
    [0]=>
    string(11) "Gotham City"
  }
  [2]=>
  object(SimpleXMLElement)#4 (1) {
    [0]=>
    string(13) "Keystone City"
  }
}
```

Um zu ermitteln, wie viele Knoten zum XPath-Ausdruck passen, verwenden Sie einfach die count()-Funktion, die hier das Gleiche zurückliefert wie das Abfragen der DOMNodeList::length-Eigenschaft unter DOM. Möchten Sie nun alle Städte ausgeben, die auf den XPath-Ausdruck passen, genügt auch hier wieder eine einfache Schleife:

```
foreach ($nodes as $stadt) {
   print $stadt . "\n";
}
```

Dasselbe Resultat ist hier also einfacher über SimpleXML als über DOM zu erreichen.

Analog dazu können Sie auch den XPath-Ausdruck verwenden, der nur den Helden mit dem Alias "Batman" aus dem XML-Dokument extrahiert:

```
$query = '/team/held[@alias="Batman"]';
$batman = $sx->xpath($query);
```

Auch hier ist die Rückgabe wiederum ein PHP-Array, das dieses Mal nur ein SimpleXML-Objekt enthält, auf das Sie über die PHP-Array-Syntax zugreifen können:

```
print $batman[0]->asXML();
```

Da Sie bei der Verwendung von SimpleXML nicht so komfortabel definieren können, wie der Parser mit Leerzeichen umgehen kann, ist die Ausgabe nicht so gut strukturiert wie bei der Verwendung von DOM:

```
<held alias="Batman">
            <name>Bruce Wayne</name>
                <stadt>Gotham City</stadt>
        </held>
```

Außerdem bietet SimpleXML auch keine Möglichkeit, Ausdrücke mit XPath-Funktionen wie z.B count() auszuwerten. Für komplexere Abfragen ist also DOM besser geeig-

net, für einfachere Funktionen ist jedoch die XPath-Unterstützung in SimpleXML ausreichend. Rezept 14.7 zeigt Ihnen, wie Sie Daten zwischen DOM und SimpleXML austauschen und somit, sobald Sie komplexe Anfragen stellen müssen, zur Arbeit mit DOM wechseln können.

Siehe auch

Rezept 14.4 zum Parsen von XML mit DOM; Rezept 14.6 zum Parsen von XML mit SimpleXML; Rezept 14.7 zum Austauschen von Daten zwischen DOM und SimpleXML; die Dokumentation zu XPath unter *http://www.php.net/domxpath* und *http://www.php.net/manual/en/function.simplexml-element-xpath*; *XML in a Nutshell* aus dem O'Reilly Verlag für weitere Informationen zu XPath.

14.12 XML-Dokumente für Menschen lesbar machen

Problem

Sie möchten ein maschinell erzeugtes XML-Dokument mit Zeilenumbrüchen und Einrückungen versehen, um es auch für Menschen lesbarer zu gestalten.

Lösung

Verwenden Sie das PEAR-Paket XML_Beautifier, das XML-Dokumente neu formatieren kann.

```
require_once 'XML/Beautifier.php';

$xml = <<<EOD
<team abbrev="JLA"
   name="Justice League of America"       mitglieder="3">
    <held alias="Superman">      <name>Clark Kent</name><stadt>Metropolis</stadt></held>
<held alias="Batman">
<name>Bruce Wayne
</name><stadt>Gotham City</stadt>
        </held>
    <held alias=
"Der Rote Blitz">
            <name>Wally West</name>
        <stadt>Keystone City</stadt>
</held></team>
EOD;

// Optionen für die "Aufräumarbeiten"
$optionen = array(
                    'removeLineBreaks' => true,
                    'indent' => '    ',
```

```
                    'linebreak' => "\n",
                    'multilineTags' => true
                );

$beautifier = new XML_Beautifier($optionen);
// String neu formatieren
$neu = $beautifier->formatString($xml);

// auf Fehler testen
if (PEAR::isError($neu)) {
    die($neu->getMessage());
}
print $neu;
```

Diskussion

Ein großer Vorteil von XML-Dokumenten ist, dass sie sowohl von Programmen als auch von Menschen leicht zu interpretieren und zu verarbeiten sind. Programme können XML-Dokumente einfach verarbeiten, da diese bestimmten Regeln folgen müssen, um wohlgeformt zu sein. Für Menschen sind sie leicht verständlich, da sprechende Namen für Tags verwendet werden können und die logische Struktur durch Zeilenumbrüche und Einrückungen grafisch unterstützt wird.

Oft sind allerdings diese Layout-Informationen in von Programmen erzeugten XML-Dokumenten nicht vorhanden und diese somit für Menschen nahezu unleserlich. Wenn Sie diese Informationen nachträglich einfügen wollen, kann das vom PEAR-Paket XML_Beautifier für Sie erledigt werden. Das folgende XML-Dokument ist zwar wohlgeformt, jedoch ist weder diese Wohlgeformtheit noch die Struktur für das menschliche Auge auf den ersten Blick sichtbar:

```
<team abbrev="JLA"
   name="Justice League of America"       mitglieder="3">
    <held alias="Superman">      <name>Clark Kent</name><stadt>Metropolis</stadt></held>
<held alias="Batman">
<name>Bruce Wayne
</name><stadt>Gotham City</stadt>
        </held>
    <held alias=
"Der Rote Blitz">
            <name>Wally West</name>
        <stadt>Keystone City</stadt>
</held></team>
```

Mit dem XML_Beautifier können Sie mithilfe weniger Zeilen PHP-Code ein gut strukturiertes Dokument erschaffen:

```
require_once 'XML/Beautifier.php';

$beautifier = new XML_Beautifier();
$neu = $beautifier->formatString($xml);
```

```
if (PEAR::isError($neu)) {
    die($neu->getMessage());
}
echo $neu;
```

Nachdem Sie das Paket mit `require_once` eingebunden haben, erzeugen Sie ein neues `XML_Beautifier`-Objekt. Um einen XML-String neu zu formatieren, verwenden Sie die Methode `formatString()` und übergeben den String als einzigen Parameter. Die Rückgabe ist entweder ein `PEAR_Error`-Objekt, das einen Fehler signalisiert (z.B. wenn das Dokument nicht wohlgeformt ist), oder ein String mit dem neu formatierten XML. Bei Anwendung der Methode auf das unstrukturierte XML-Dokument erhalten Sie folgende Ausgabe:

```
<team abbrev="JLA" mitglieder="3" name="Justice League of America">
    <held alias="Superman">
        <name>Clark Kent</name>
        <stadt>Metropolis</stadt>
    </held>
    <held alias="Batman">
        <name>Bruce Wayne</name>
        <stadt>Gotham City</stadt>
    </held>
    <held alias="Der Rote Blitz">
        <name>Wally West</name>
        <stadt>Keystone City</stadt>
    </held>
</team>
```

Bei diesem Dokument erkennt man sofort, dass das Dokument wohlgeformt ist, und auch, in welcher Art und Weise die Inhalte strukturiert wurden.

Bei der Umformatierung des XML-Dokuments akzeptiert XML_Beautifier eine Reihe von Optionen, die beeinflussen, wie das Dokument formatiert wird. Diese werden als assoziatives Arrays im Konstruktor übergeben:

```
// Optionen für die "Aufräumarbeiten"
$optionen = array(
                    'indent' => '  ',
                    'multilineTags' => true,
                    'caseFolding' => true,
                );

$beautifier = new XML_Beautifier($optionen);
// String neu formatieren.
$neu = $beautifier->formatString($xml);
```

Mit diesen Optionen können Sie z.B. die Einrücktiefe bestimmen (*indent*), definieren, ob Tags auf mehrere Zeilen verteilt werden (*multilineTags*) oder auch die Schreibweise aller Tags verändern (*caseFolding*). Eine Liste aller Optionen finden Sie unten in Tabelle Tabelle 14-2. Bei der Anwendung dieser Optionen sieht das resultierende Dokument folgendermaßen aus:

```
<TEAM ABBREV="JLA"
      MITGLIEDER="3"
      NAME="Justice League of America">
 <HELD ALIAS="Superman">
  <NAME>Clark Kent</NAME>
  <STADT>Metropolis</STADT>
 </HELD>
 <HELD ALIAS="Batman">
  <NAME>Bruce Wayne</NAME>
  <STADT>Gotham City</STADT>
 </HELD>
 <HELD ALIAS="Der Rote Blitz">
  <NAME>Wally West</NAME>
  <STADT>Keystone City</STADT>
 </HELD>
</TEAM>
```

Möchten Sie unterschiedliche Optionen für verschiedene Dokumente übergeben, ist dies über die Methode `setOptions()` möglich, die dasselbe assoziative Array erwartet wie der Konstruktor.

Tabelle 14-2: Optionen der XML_Beautifier-Klasse

Option	Default	Beschreibung
removeLineBreaks	`false`	Gibt an, ob Zeilenumbrüche aus Textdaten entfernt werden sollen oder nicht.
indent	4 Leerzeichen	Setzt den String, der für Einrückungen verwendet werden sollen. Ein Tabulator kann über `"\t"` gesetzt werden.
linebreak	\n	Zeichenkette für Zeilenumbrüche, z.B. `"\r\n"` für Windows-Zeilenumbrüche.
caseFolding	false	Definiert, ob die Groß- und Kleinschreibung der Tags verändert werden soll. Wenn dieser Wert auf `true` gesetzt ist, kann mit der Option *caseFoldingTo* das Verhalten beeinflusst werden.
caseFoldingTo	uppercase	Gibt an, ob Tags in Großschreibung ("uppercase") oder Kleinschreibung ("lowercase") konvertiert werden sollen. Nur aktiv, wenn die *caseFolding*-Option auf `true` gesetzt wird.
normalizeComments	false	Gibt an, ob Zeilenumbrüche und unnötige Leerzeichen aus XML-Kommentaren entfernt werden sollen.
maxCommentLine	–1	Maximale Zeilenlänge für Kommentare, am besten in Verbindung mit *normalizeComments* auf `true`.
multilineTags	false	Gibt an, ob Tags auf mehrere Zeilen verteilt werden sollen. Wenn diese Option auf `true` gesetzt wird, wird jedes Attribut in eine eigene Zeile geschrieben.

Möchten Sie eine XML-Datei neu formatieren, können Sie die Methode `formatFile()` verwenden. Diese erwartet den Namen der ursprünglichen Datei als ersten und den Namen der Zieldatei als zweiten Parameter. Möchten Sie die Originaldatei überschreiben, übergeben Sie die Konstante `XML_BEAUTIFIER_OVERWRITE` als zweiten Parameter:

```
$beautifier = new XML_Beautifier();
$ok = $beautifier->formatFile("alt.xml", "neu.xml");
if ($ok === true) {
   print "Aktion erfolgreich.\n";
}
```

Diese Methode gibt `true` zurück, wenn die Aktion erfolgreich war, und ansonsten ein `PEAR_Error`-Objekt.

Siehe auch

Kapitel 24 zur Arbeit mit PEAR-Paketen; die Dokumentation zu XML_Beautifier unter *http://pear.php.net/manual/en/package.xml.xml-beautifier.php*.

14.13 XML-Dokumente aus PHP-Datenstrukturen erzeugen

Problem

Sie möchten aus einem Array oder PHP-Objekt ein XML-Dokument erzeugen.

Lösung

Verwenden Sie das PEAR-Paket XML_Serializer.

```
require_once 'XML/Serializer.php';

$shows = array(array('name'     => 'Simpsons',
                     'kanal'    => 'FOX',
                     'beginn'   => '20:00',
                     'dauer'    => '30'),

               array('name'     => 'Law & Order',
                     'kanal'    => 'NBC',
                     'beginn'   => '20:00',
                     'dauer'    => '60'));

$serializer = new XML_Serializer();
$serializer->setOption('indent', '    ');
$serializer->setOption('rootName', 'shows');
$serializer->setOption('defaultTagName', 'show');

$success = $serializer->serialize($shows);
if (PEAR::isError($success)) {
    die($success->getMessage());
}
print $serializer->getSerializedData();
```

Diskussion

Obwohl PHP 5 verschiedene Möglichkeiten bietet, XML-Dokumente zu erzeugen, ist dafür unabhängig von der genutzten Erweiterung immer noch sehr viel Programmieraufwand nötig, da die PHP-Datenstrukturen, aus denen Sie ein XML-Dokument erzeugen möchten, meistens in mehreren Schleifen durchlaufen werden müssen.

Das PEAR-Paket XML_Serializer wurde entwickelt, um mit möglichst wenig Aufwand aus beliebigen Datenstrukturen XML-Dokumente zu generieren, die stets wohlgeformt sind. Bevor Sie das Paket nutzen, müssen Sie es zunächst über `require_once` einbinden. Das Paket benötigt außerdem die PEAR-Pakete XML_Util und XML_Parser, diese müssen also installiert sein. Nach dem Einbinden der Klasse erzeugen Sie zunächst ein `XML_Serializer`-Objekt:

```
require_once 'XML/Serializer.php';

$serializer = new XML_Serializer();
```

Um nun aus einem PHP-Array ein XML-Dokument zu machen, verwenden Sie einfach die Methode `serialize()` des Objekts und übergeben die Variable, die das Array enthält:

```
$success = $serializer->serialize($shows);
if (PEAR::isError($success)) {
    die($success->getMessage());
}
```

Falls der Inhalt der Variablen nicht in XML konvertiert werden kann, gibt diese Methode ein `PEAR_Error`-Objekt zurück. Nutzen Sie die Methode `PEAR::isError()`, um den Rückgabewert zu überprüfen. Wurde kein Fehler zurückgeliefert, können Sie mit `getSerializedData()` den erzeugten XML-Code zurückbekommen:

```
<array>
<XML_Serializer_Tag>
<name>Simpsons</name>
<kanal>FOX</kanal>
<beginn>20:00</beginn>
<dauer>30</dauer>
</XML_Serializer_Tag>
<XML_Serializer_Tag>
<name>Law &amp;amp; Order</name>
<kanal>NBC</kanal>
<beginn>20:00</beginn>
<dauer>60</dauer>
</XML_Serializer_Tag>
</array>
```

Wie Sie schnell erkennen können, nutzt XML_Serializer die Keys des Arrays als Tag-Namen und schreibt die Werte des Arrays als Textknoten in das Dokument. Allerdings entspricht dieses Ergebnis nicht genau dem erwarteten:

- Tags sind nicht eingerückt.
- Der Wurzelknoten sollte <shows> sein.
- Die Daten einer Show sollten durch ein <show>-Tag eingeschlossen werden.

Das Verhalten von XML_Serializer kann durch das Setzen verschiedener Optionen verändert werden. Dazu nutzen Sie entweder die Methode `setOption()`, um eine Option zu verändern, oder Sie übergeben ein assoziatives Array mit allen Optionen an den Konstruktor oder die Methode `setOptions()`. Die Einrückung kann über die Option `indent` verändert werden: Wollen Sie vier Leerzeichen als Einrücktiefe nutzen, übergeben Sie hier einfach einen String mit vier Leerzeichen:

```
$serializer->setOption('indent', '    ');
```

Zur Änderung des Namens des Wurzelknotens verwenden Sie die Option `rootName`:

```
$serializer->setOption('rootName', 'shows');
```

So bleibt nun nur noch das Tag, das die Daten einer einzelnen Show begrenzt und im Moment den Namen *XML_Serializer_Tag* trägt. Wie bereits erwähnt, verwendet XML_Serializer die Keys eines Arrays als Tag-Namen. Bei indizierten Arrays ist dies allerdings nicht möglich, da <0>, <1> usw. keine gültigen XML-Tags sind. Aus diesem Grund verwendet die Klasse hier ein Default-Tag, dessen Namen über die Option `defaultTagName` geändert werden kann.

```
$serializer->setOption('defaultTagName', 'show');
```

Wenden Sie alle diese Optionen an, entspricht das erzeugte XML-Dokument dem gewünschten Ergebnis:

```
<shows>
    <show>
        <name>Simpsons</name>
        <kanal>FOX</kanal>
        <beginn>20:00</beginn>
        <dauer>30</dauer>
    </show>
    <show>
        <name>Law &amp;amp; Order</name>
        <kanal>NBC</kanal>
        <beginn>20:00</beginn>
        <dauer>60</dauer>
    </show>
</shows>
```

Die Schachtelungstiefe für Tags ist im Moment nur durch das Limit für Rekursionen in PHP beschränkt, ansonsten können Sie beliebige PHP-Datentypen, also auch Strings, Boolesche Werte oder Objekte, an die `serialize()`-Methode übergeben.

XML_Serializer unterstützt noch weitere Optionen: So können Sie z.B. Informationen über den Datentyp hinzufügen, indem Sie die Option `typeHints` auf den Wert `true` setzen. Durch Verwendung der Option `scalarAsAttributes` können Sie erzwingen, dass ska-

lare Werte wie Zahlen oder Strings als Attribute anstatt als verschachtelte Tags im XML-Dokument erscheinen:

```
$serializer->setOption('scalarAsAttributes', true);

<shows>
    <show beginn="20:00" dauer="30" kanal="FOX" name="Simpsons" />
    <show beginn="20:00" dauer="60" kanal="NBC" name="Law & Order" />
</shows>
```

In der aktuellen Version 0.15.0 unterstützt XML_Serializer bereits über 20 Optionen, und das Paket entwickelt sich stetig weiter.

Siehe auch

Rezept 14.1 zum manuellen Generieren von XML-Dokumenten; Rezept 14.14 zum Einlesen von XML-Dokumenten mit XML_Serializer; Kapitel 24 zur Arbeit mit PEAR; die Dokumentation zu XML_Serializer unter *http://pear.php.net/manual/en/package.xml.xml-serializer.php*.

14.14 XML-Dokumente in PHP-Arrays oder Objekte einlesen

Problem

Sie möchten ein XML-Dokument in PHP-Arrays oder Objekte einlesen, ohne sich mit einer der XML-Erweiterungen befassen zu müssen.

Lösung

Verwenden Sie die Klasse `XML_Unserializer` des PEAR-Pakets XML_Serializer.

```
require_once 'XML/Unserializer.php';

$xml = <<<EOD
<shows>
    <show>
        <name>Simpsons</name>
        <kanal>FOX</kanal>
        <beginn>20:00</beginn>
        <dauer>30</dauer>
    </show>
    <show>
        <name>Law &amp; Order</name>
        <kanal>NBC</kanal>
        <beginn>20:00</beginn>
        <dauer>60</dauer>
```

```
    </show>
</shows>
EOD;

$unserializer = new XML_Unserializer();
$success = $unserializer->unserialize($xml);
if (PEAR::isError($success)) {
    die($success->getMessage());
}
$data = $unserializer->getUnserializedData();

print_r($data);
```

Diskussion

Das PEAR-Paket XML_Serializer bietet neben der in Rezept 14.13 besprochenen `XML_Serializer`-Klasse auch eine passende `XML_Unserializer`-Klasse, mit der beliebige XML-Dokumente in PHP-Datenstrukturen konvertiert werden können. Auch hier muss zu Beginn des Skripts die Klasse eingebunden und ein Objekt erstellt werden:

```
require_once 'XML/Unserializer.php';
$unserializer = new XML_Unserializer();
```

Dieses Objekt bietet nun die Methode `unserialize()`, der Sie das XML-Dokument als String übergeben. Sollte das Dokument nicht wohlgeformt sein oder sonst ein Fehler auftreten, gibt diese Methode ein `PEAR_Error`-Objekt zurück. In diesem Fall geben Sie den Fehler aus und beenden das Skript.

```
$success = $unserializer->unserialize($xml);
if (PEAR::isError($success)) {
    die($success->getMessage());
}
```

Liefert die Methode `true` zurück, konnte das XML-Dokument eingelesen werden, und die Methode `getUnserializedData()` kann verwendet werden, um auf die reinen Daten zuzugreifen:

```
$data = $unserializer->getUnserializedData();
print_r($data);
```

Bei obigem XML-Dokument erhalten Sie die folgenden Daten:

```
Array
(
    [show] => Array
        (
            [0] => Array
                (
                    [name] => Simpsons
                    [kanal] => FOX
                    [beginn] => 20:00
                    [dauer] => 30
                )
```

```
            [1] => Array
                (
                    [name] => Law & Order
                    [kanal] => NBC
                    [beginn] => 20:00
                    [dauer] => 60
                )
        )
)
```

XML_Unserializer verwandelt also ein Tag, das weitere Tags enthält, in ein Array, Tags, die nur Text enthalten, werden in Strings konvertiert. Die Namen der Tags werden bei den resultierenden Arrays als Keys verwendet. Tritt ein Tag mehrmals auf einer Ebene auf, wird automatisch ein indiziertes Array erzeugt, das die Werte der einzelnen Tags enthält.

Auf einzelne Elemente des XML-Dokuments kann nun über die bekannte Array-Syntax von PHP zugegriffen werden:

```
print $data['show'][1]['name'];
```

Wie XML_Serializer arbeitet auch XML_Unserializer mit Optionen, die beeinflussen können, wie das XML-Dokument in Datenstrukturen konvertiert wird. Möchten Sie z.B. Objekte anstatt Arrays verwenden, um verschachtelte XML-Tags zu repräsentieren, setzen Sie die Option `complexType` auf den Wert `"object"`. XML_Unserializer verwendet nun Objekte der Klasse `stdClass` anstatt PHP-Arrays, wodurch Sie eine SimpleXML-ähnliche Syntax für den Zugriff auf Elemente des XML-Dokuments verwenden müssen:

```
print $data->show[1]->name;
```

Bevor XML_Unserializer allerdings ein Objekt vom Typ `stdClass` erzeugt, wird überprüft, ob eine Klasse mit dem Namen des aktuellen Tags existiert. Somit können Sie sich durch XML_Unserializer Objekte aus XML-Dokumenten erzeugen lassen, die zusätzliche Funktionalität bereitstellen und automatisch mit den Daten aus den XML-Dokumenten initialisiert werden. Als Erstes erstellen Sie eine Klasse, die für die einzelnen Shows verwendet wird. Diese Klasse muss wie das Tag für eine Show genannt werden und benötigt vier Eigenschaften, die die Inhalte der Tags innerhalb eines `<show>`-Tags speichern:

```
class Show
{
    public $name;
    public $kanal;
    public $beginn;
    public $dauer;

    public function getName()
    {
        return $this->name;
    }

    public function getKanal()
```

```
    {
        return $this->kanal;
    }

    public function getBeginn()
    {
        return $this->beginn;
    }

    public function getDauer()
    {
        return $this->dauer;
    }

    public function getEnde()
    {
        list($stunde, $minuten) = explode(':', $this->beginn);
        $minuten = $minuten + $this->dauer;
        while ($minuten >= 60) {
            $minuten = $minuten - 60;
            $stunde  = $stunde + 1;
        }
        return sprintf('%02d:%02d', $stunde, $minuten);
    }
}
```

Diese Eigenschaften `$name`, `$kanal`, `$beginn` und `$dauer` werden als `public` deklariert, damit XML_Unserializer diese Eigenschaften setzen kann. Zusätzlich werden noch die Methoden `getName()`, `getKanal()`, `getBeginn()` und `getDauer()` implementiert, um einen einfacheren Zugriff auf die Eigenschaften zu ermöglichen. Diese Funktionalität hätte bisher auch mit einer etwas anderen Art des Zugriffs über Arrays oder Objekte vom Typ `stdClass` ermöglicht werden können. Allerdings implementiert diese Klasse noch die Methode `getEnde()`, die anhand der Eigenschaften `$beginn` und `$dauer` das Ende der Sendung berechnen und zurückgeben kann.

Danach erstellen Sie eine Klasse, die als Container für die einzelnen Shows dient. Diese Klasse muss wie der Wurzelknoten, also `shows`, genannt werden:

```
class Shows
{
    public $show;

    public function getShow($name)
    {
        foreach ($this->show as $show) {
            if ($name === $show->getName()) {
                return $show;
            }
        }
        return false;
    }
}
```

Da das <shows>-Tag mehrere <show>-Tags enthält, wird hier eine Eigenschaft $show deklariert, die dann später automatisch mit einem Array von Instanzen der Klasse Show von XML_Unserializer gefüllt wird. Die Methode getShow() liefert nun ein Objekt dieser Objekte zurück, indem in einer Schleife alle Objekte durchlaufen werden und dabei der übergebene gesuchte Name mit dem tatsächlichen Namen verglichen wird.

Nachdem Sie beide Klassen implementiert haben, kann XML_Unserializer diese automatisch beim Einlesen verwenden, um Objekte zu erzeugen:

```
// Neuen XML_Unserializer erzeugen.
$unserializer = new XML_Unserializer();

// Verschachtelte Tags in Objekte konvertieren.
$unserializer->setOption('complexType', 'object');

// XML einlesen.
$success = $unserializer->unserialize($xml);
if (PEAR::isError($success)) {
    die($success->getMessage());
}
// Shows-Objekt holen.
$shows = $unserializer->getUnserializedData();

// Ein Show-Objekt holen und Ende der Sendung berechnen.
$simpsons = $data->getShow('Simpsons');
print "Simpsons endet um " . $simpsons->getEnde() . "\n";

$lao = $data->getShow('Law & Order');
print "Law & Order endet um " . $lao->getEnde() . "\n";
```

Sie erhalten dabei diese Ausgabe, obwohl die Informationen zum Ende der Sendung nicht in der XML-Datei enthalten sind:

```
Simpsons endet um 20:30
Law & Order endet um 21:00
```

Damit können Sie einen Großteil der niedrigen Logik in Klassen auslagern und diese transparent in Ihren Anwendungen durch XML_Unserializer erzeugen lassen. XML_Unserializer bietet noch viele weitere Features, so ist es z.B. möglich, Attribute zu verarbeiten oder auch alle Textinformationen über eine Callback-Funktion während des Einlesens zu verändern. Natürlich können Sie XML-Dokumente auch direkt aus einer Datei lesen, dazu übergeben Sie der unserialize()-Methode einfach den Dateinamen als ersten Parameter und zusätzlich als zweiten Parameter true, um zu signalisieren, dass es sich nicht um die XML-Daten handelt, sondern lediglich einen Zeiger darauf:

```
$success = $unserializer->unserialize("shows.xml", true);
```

Trotz der großen Anzahl an XML-APIs in PHP 5 gibt es also noch sehr viele Anwendungsfälle, die von PHP selbst nicht abgedeckt werden, jedoch durch PEAR auch sehr einfach gelöst werden können.

Siehe auch

Rezept 14.13 zum Erzeugen von XML-Dokumenten mit XML_Serializer; Rezept 15.8 zum Senden und Empfangen von REST-Requests; Kapitel 24 zur Arbeit mit PEAR; die Dokumentation zu XML_Serializer unter *http://pear.php.net/manual/en/package.xml.xml-serializer.php*.

14.15 XML-Dokumente validieren

Problem

Sie möchten sicherstellen, dass Ihr XML-Dokument einem Schema wie von XML Schema, einem RelaxNG-Schema oder einer DTD genügt.

Lösung

Nutzen Sie die DOM-Erweiterung.

Rufen Sie für bestehende DOM-Objekte `DOMDocument::schemaValidate( )` oder `DOMDocument::relaxNGValidate( )` auf:

```
$file = 'address-book.xml';
$schema = 'address-book.xsd';
$ab = new DOMDocument;
$ab->load($file);
if ($ab->schemaValidate($schema)) {
    print "$file ist gültig.";
} else {
    print "$file ist ungültig.";
}
```

Wenn Ihr XML-Dokument eine Dokumenttyp-Deklaration enthält, rufen Sie `DOMDocument::validate( )` auf, um es anhand der DTD zu prüfen.

Bei XML in einem String rufen Sie `DOMDocument::schemaValidateSource( )` oder `DOMDocument::relaxNGValidateSource( )` auf:

```
$xml = '<person><firstname>Adam</firstname></person>';
$schema = 'address-book.xsd';
$ab = new DOMDocument
$ab->load($file);
if ($ab->schemaValidateSource($schema)) {
    print "XML ist gültig.";
} else {
    print "XML ist ungültig.";
}
```

Diskussion

Schemata sind ein Mittel, Spezifikationen für XML-Dokumente zu definieren. Während das Ziel in allen Fällen das gleiche ist, gibt es einige unterschiedliche Wege, Schemata zu kodieren, die jeweils unterschiedliche Syntaxformen nutzen.

Einige beliebte Formate sind DTD (Document Type Definition), XML Schema und RelaxNG. DTDs werden bereits am längsten verwendet, werden aber nicht in XML formuliert und haben noch andere Haken, die die Arbeit mit ihnen verkomplizieren können. XML Schema und RelaxNG sind jüngere Schema-Formen, die versuchen, einige der Probleme zu lösen, die DTDs mit sich bringen.

Die Validierungsunterstützung in PHP 5 basiert auf der `libxml2`-Bibliothek. Deswegen können Sie unter PHP 5 Dateien anhand aller drei Typen prüfen. Am flexibelsten ist es bei der Arbeit mit XML Schema und RelaxNG, die XML Schema-Unterstützung ist allerdings unvollständig. Bei den meisten XML Schema-Dokumenten sollten Sie keine Probleme haben, es kann aber passieren, dass Sie feststellen müssen, dass `libxml2` komplexere Schemata oder Schemata, die esoterischere Funktionen einsetzen, nicht verarbeiten kann.

In PHP unterstützt die DOM-Erweiterung DTD-, XML Schema- und RelaxNG-Validierung, während SimpleXML nur eine XML Schema-Validierung bietet.

Die Validierung einer Datei über DOM läuft, vom verwendeten Schema-Format unabhängig, jeweils auf ähnliche Weise. Zur Validierung rufen Sie eine Validierungsmethode auf einem DOM-Objekt auf (siehe Beispiel 14-3). Diese liefert `true`, wenn die Datei Schemakonform ist. Tritt ein Fehler auf, liefert die Methode `false` und gibt eine Meldung ins Fehler-Log aus. Es gibt keine Methode zum »Einfangen« der Fehlermeldung.

Beispiel 14-3: Ein XML-Dokument validieren

```
$file = 'address-book.xml';
$schema = 'address-book.xsd';
$ab = new DOMDocument;
$ab->load($file);
if ($ab->schemaValidate($schema)) {
    print "$file ist gültig.";
} else {
    print "$file ist ungültig.";
}
```

Ist das Schema in einem String gespeichert, nutzen Sie `DOMDocument::schemaValidateSource()` statt `schemaValidate()`.

Tabelle 14-3 führt alle Validierungsmethoden auf.

Tabelle 14-3: DOM-Schema-Validierungsmethoden

Methodenname	Schema-Typ	Datenort
`schemaValidate`	XML Schema	Datei
`schemaValidateSource`	XML Schema	String
`relaxNGValidate`	RelaxNG	Datei
`relaxNGValidateSource`	RelaxNG	String
`validate`	DTD	N/A

Diese Validierungsmethoden verhalten sich alle in ähnlicher Weise. Sie müssten also im vorangehenden Beispiel nur den Methodennamen ändern, um zu einer anderen Schema-Validierung überzugehen.

XML Schema und RelaxNG unterstützen beide eine Validierung von Dateien und Strings. Gegen eine DTD können Sie ein DOM-Objekt nur prüfen, wenn die DTD am Anfang des Dokuments definiert ist.

Siehe auch

Die XML Schema-Spezifikation unter *http://www.w3.org/XML/Schema*; die RelaxNG-Spezifikation unter *http://www.relaxng.org/*.

14.16 Die Inhaltskodierung steuern

Problem

PHP XML-Erweiterungen nutzen UTF-8, aber Ihre Daten sind auf andere Weise kodiert.

Lösung

Nutzen Sie `iconv`-Bibliothek, um Ihre Daten zu konvertieren, bevor Sie sie an eine XML-Erweiterung übergeben:

```
$utf_8 = iconv('ISO-8859-1', 'UTF-8', $iso_8859_1);
```

Und kehren Sie die Kodierung um, wenn Sie fertig sind:

```
$iso_8859_1 = iconv('UTF-8', 'ISO-8859-1', $utf_8);
```

Diskussion

Die Zeichenkodierung ist eine der großen Schwächen von PHP. Glücklicherweise ist die Unicode-Unterstützung aber auch der wichtigste Ansporn bei der Entwicklung von PHP 6. Da PHP 6 allerdings immer noch im Entstehen ist, können Sie Schwierigkeiten bekom-

men, wenn Sie versuchen, XML-Erweiterungen mit Daten in frei gewählten Kodierungen zu verwenden.

Der Einfachheit halber nutzen alle XML-Erweiterungen ausschließlich die UTF-8-Zeichenkodierung. Das bedeutet, dass sie Daten in UTF-8 erwarten und Daten in UTF-8 ausgeben. Wenn Sie reine ASCII-Daten haben, müssen Sie sich keine Sorgen machen, da UTF-8 eine Obermenge zu ASCII ist. Aber wenn Sie mit anderen Kodierungen arbeiten, werden Sie früher oder später auf Probleme stoßen.

Sie können diese Probleme umgehen, indem Sie die `iconv`-Erweiterung nutzen, um Ihre Daten zwischen Ihrer Zeichenkodierung und UTF-8 zu konvertieren. Machen Sie beispielsweise Folgendes, um von ISO-8859-1 nach UTF-8 zu konvertieren:

```
$utf_8 = iconv('ISO-8859-1', 'UTF-8', $iso_8859_1);
```

Die `iconv`-Funktion unterstützt zwei spezielle Modifizierer für die Zielkodierung: `//TRANSLIT` und `//IGNORE`. Die erste Option sagt `iconv`, dass es Zeichen, die es in der Zielkodierung nicht exakt reproduzieren kann, so gut wie möglich durch eine Folge anderer Zeichen darstellen soll. Die andere Option sorgt dafür, dass `iconv` nicht konvertierbare Zeichen stillschweigend ignoriert.

Enthält der String `$geb` die Zeichenfolge `"Gödel, Escher, Bach."`, führt eine einfache Konvertierung zu ASCII zu einem Fehler:

```
echo iconv('UTF-8', 'ASCII', $geb);
PHP Notice:  iconv(): Detected an illegal character in input string...
```

Die Aktivierung von `//IGNORE` ermöglicht die Durchführung der Konvertierung:

```
echo iconv('UTF-8', 'ASCII//IGNORE', $geb);
```

Die Ausgabe ist allerdings weniger schön, weil das ö fehlt:

```
Gdel, Escher, Bach
```

Die beste Lösung liefert die Verwendung von `//TRANSLIT`:

```
echo iconv('UTF-8', 'ASCII//TRANSLIT', $geb);
```

Das liefert uns diesen einigermaßen ansprechenden String:

```
G"odel, Escher, Bach
```

Sie sollten allerdings aufpassen, wenn Sie `//TRANSLIT` einsetzen, da es dazu führen kann, dass sich die Anzahl an Zeichen erhöht. Das eine ö-Zeichen wird, wie Sie gesehen haben, beispielsweise zu den beiden Zeichen `"` und `o`.

Siehe auch

Weitere Informationen zur Arbeit mit UTF-8-Text finden Sie in Rezept 19.11; die Dokumentation zu *iconv* unter *http://www.php.net/iconv*; die Homepage zu GNU libiconv unter *http://www.gnu.org/software/libiconv/*.

14.17 XSLT-Parameter aus PHP setzen

Problem

Sie möchten Parameter für Ihr XSLT-Stylesheet aus PHP setzen.

Lösung

Nutzen Sie die Methode `XSLTProcessor::setParameter( )`:

```
// Das könnte auch aus $_GET['city'] kommen.
$city = 'San Francisco';
$dom  = new DOMDocument;
$dom->load('address-book.xml');
$xsl  = new DOMDocument;
$xsl->load('stylesheet.xsl');
$xslt = new XSLTProcessor();
$xslt->importStylesheet($xsl);
$xslt->setParameter(NULL, 'city', $city);
print $xslt->transformToXML($dom);
```

Dieser Code setzt den XSLT-Parameter `city` auf den in der PHP-Variablen `$city` gespeicherten Wert.

Diskussion

Mit der Methode `setParameter( )` können Sie Daten aus PHP an Ihr XSLT-Stylesheet übergeben. Auf diese Weise filtern Sie beispielsweise in Ihrem Stylesheet Daten auf Basis von Benutzereingaben.

Das Programm in Beispiel 14-4 ermöglicht Ihnen, Personen auf Basis der Stadt zu finden.

Beispiel 14-4: XSLT-Parameter aus PHP setzen

```
// Das könnte auch aus $_GET['city'] kommen.
$city = 'San Francisco';
$dom  = new DOMDocument
$dom->load('address-book.xml');
$xsl  = new DOMDocument
$xsl->load('stylesheet.xsl');
$xslt = new XSLTProcessor();
$xslt->importStylesheet($xsl);
$xslt->setParameter(NULL, 'city', $city);
print $xslt->transformToXML($dom);
```

Das Programm nutzt das folgende Stylesheet:

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="@*|node()">
```

```
  <xsl:copy>
    <xsl:apply-templates select="@*|node()"/>
  </xsl:copy>
</xsl:template>
<xsl:template match="/address-book/person">
  <xsl:if test="city=$city">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:if>
</xsl:template>
</xsl:stylesheet>
```

Gemeinsam erzeugen Programm und Stylesheet folgendes Ergebnis:

```
<?xml version="1.0"?>
<address-book>
        <person id="2">
        <!--Adam Trachtenberg-->
        <firstname>Adam</firstname>
        <lastname>Trachtenberg</lastname>
        <city>San Francisco</city>
        <state>CA</state>
        <email>amt@php.net</email>
    </person>
</address-book>
```

Das PHP-Skript führt eine gewöhnliche XSL-Transformation durch, ruft zuvor aber `$xslt->setParameter(NULL, 'city', $city)` auf. Das erste Argument ist der Namensraum des Parameters, das zweite der Parametername, das dritte der Parameterwert.

Hier wird der in der PHP-Variablen `$city` gespeicherte Wert – in diesem Fall `San Francisco` – dem XSLT-Parameter `city` zugewiesen, der sich im Default-Namensraum befindet. Das entspricht folgender Zeile in einer XSLT-Datei:

```
<xsl:param name="city">San Francisco</xsl:param>
```

In einem Stylesheet greifen Sie auf Parameter normalerweise zu wie auf eine PHP-Variable: indem Sie seinem Namen ein Dollarzeichen (`$`) voranstellen. Das Stylesheet-Beispiel erstellt ein Template, das `/address-book/person`-Knoten findet.

Im Template prüfen Sie, ob `city=$city` gilt, d.h., ob das `city`-Kind des aktuellen Knotens dem Wert des Parameters `city` entspricht. Bei einem Treffer werden die Kinder kopiert, andernfalls wird der Datensatz übersprungen.

Hier ist `city` gleich `San Francisco`: Davids Datensatz wird also entfernt, der von Adam übernommen.

Siehe auch

Dokumentation zu `XSLTProcessor::setParameter` unter *http://www.php.net/manual/xslt-processor.setparameter.php*; *XSLT* von Doug Tidwell (O'Reilly).

KAPITEL 15

Webservices

15.0 Einführung

Das Internet wächst stetig weiter und damit auch die Anforderungen an die Vernetzung der verschiedenen Applikationen und Inhalte. Dadurch wurden Webservices zu einem der wichtigsten Themen der webbasierten Software-Entwicklung und automatisch auch zu einem der wichtigsten Themen in PHP 5. Webservices erlauben die Kommunikation zwischen verschiedenen Applikationen, die dabei Web-Technologien wie das HTTP-Protokoll oder XML verwenden.

PHP 5 kommt mit einer neuen SOAP-Erweiterung, die das Erstellen von SOAP-Clients oder -Servern zum Kinderspiel macht; die Rezepte 15.4, 15.5 und 15.6 zeigen Ihnen, wie es geht. Sollte Ihnen SOAP zu aufwendig sein, können Sie ein einfacheres Protokoll wie XML-RPC verwenden. Die Rezepte 15.1 und 15.2 machen Sie mit dem Erstellen von XML-RPC-Clients bzw. XML-RPC-Servern vertraut und zeigen Ihnen den Umgang mit der klassischen PEAR XML-RPC-Erweiterung. In PECL existiert die Erweiterung XML-RPCi in der Version 1.0 Beta aus dem Jahr 2005; wie Sie mit ihr arbeiten, ist Thema des Rezepts 15.3.

Einige Unternehmen nutzen eine Technik, die unter der Abkürzung REST zusammengefasst wird. Wie Sie generell mit REST-basierten Webservices umgehen, ist Thema des Rezepts 15.8, das das Erstellen eines Clients für die Yahoo!-API zeigt.

Rezept 15.7 behandelt schließlich die Verarbeitung von RSS-Streams, einem beliebten Format zur Content-Syndication, Rezept 15.9 zeigt Ihnen eine ältere Form des Datenaustauschs, die Verwendung des WDDX-Formats.

Während Webservices in PHP 4 noch sehr aufwendig zu programmieren waren und deren Performance eher dürftig war, sind in PHP 5 kaum noch Wünsche offen. Sollten Sie dennoch einmal mit PHP 5 nicht weiterkommen, bietet PEAR weitere Pakete, die die Webservice-Features von PHP 5 optimal ergänzen.

15.1 XML-RPC-Anfragen senden

Problem

Sie benötigen einen XML-RPC-Client, der Anfragen an einen Server durchführt. Über XML-RPC kann man Funktionsaufrufe an Webserver senden, auch wenn diese kein PHP verwenden. Die erhaltenen Daten werden dann automatisch in PHP-Variablen für den Gebrauch in Ihrer Anwendung konvertiert.

Lösung

Verwenden Sie das PEAR-Paket XML_RPC. Hier ist ein Stück Clientcode, der eine Funktion bei einem XML-RPC-Server aufruft, der Namen von US-Bundesstaaten liefert:

```
require_once 'XML/RPC.php';

// Server-Einstellungen
$host = 'betty.userland.com';
$uri = '/RPC2';

// Anfrage-Einstellungen
// Es wird eine Zahl zwischen 1 und 50 übergeben, und man
// erhält den n-ten Staat in alphabetischer Reihenfolge.
// 1 ist Alabama, 50 ist Wyoming.
$method = 'examples.getStateName';
$state  = 32;

$params  = array(new XML_RPC_Value($state, 'int'));
$message = new XML_RPC_Message($method, $params);

$client   = new XML_RPC_Client($uri, $host);
$response = $client->send($message);

if (!$response) {
    print "Keine Antwort: " . $client->errstr ."\n";
    exit();
}

if (!$response->faultCode()) {
    $val = $response->value();
    $staat = XML_RPC_decode($val);
    print "Ich mag $staat!\n";
} else {
    print "Es ist ein Fehler aufgetreten.\n";
    print 'Fehler Nummer: ' . $response->faultCode() . "\n";
    print 'Fehler Grund : ' . $response->faultString() . "\n";
}
```

Diskussion

XML-RPC ist ein von Userland Software entwickeltes Format, mit dem Sie Anfragen an Webserver über HTTP durchführen können. Eine solche Anfrage ist ein speziell formatiertes XML-Dokument. Als Client stellen Sie eine XML-Anfrage entsprechend der XML-RPC-Spezifikation zusammen und senden diese an den Server, worauf der Server ebenfalls mit einem XML-Dokument antwortet. Dann parsen Sie die XML-Daten der Antwort und finden das Ergebnis. In der Lösung liefert der XML-RPC-Server den Namen eines Staats, daher erzeugt der Code diese Ausgabe:

```
Ich mag New York!
```

Im obigen Beispiel übernimmt das PEAR-Paket XML_RPC das Erstellen und Parsen der XML-Dokumente, Sie müssen lediglich die Klassen dementsprechend konfigurieren. Die Server-Einstellungen teilen PHP mit, mit welcher Website zur Durchführung der Anfrage Kontakt aufgenommen werden soll. In `$host` steht der Hostname des Rechners und in `$uri` der Name des Pfads zu dem XML-RPC-Server, den Sie ansprechen möchten. Diese Anfrage ist gleichbedeutend mit *http://betty.userland.com/RPC2*.

Die Anfrage besteht aus der aufzurufenden Funktion und den Daten, die an diese Funktion übergeben werden sollen. Die Methode `examples.getStateName` nimmt eine Integer-Zahl zwischen 1 und 50 an und liefert einen String mit dem Namen eines US-Staats in alphabetischer Anordnung. Bei XML-RPC können Methodennamen Punkte enthalten, was bei PHP nicht der Fall ist. Wenn man davon absieht, wäre das Äquivalent für die Übergabe von `32` als Argument eines XML-RPC-Aufrufs an `examples.getStateName` der Aufruf einer Funktion namens `examples.getStateName( )`:

```
examples.getStateName(32);
```

Bei XML-RPC sieht dies folgendermaßen aus:

```
<?xml version='1.0' encoding="iso-8859-1" ?>
<methodCall>
    <methodName>examples.getStateName</methodName>
    <params>
        <param>
            <value>
                <int>32</int>
            </value>
        </param>
    </params>
</methodCall>
```

Um diesen Methodenaufruf mit der PEAR-Klasse zu implementieren, müssen Sie zunächst die Datei *XML/RPC.php* einbinden, die verschiedene Klassen für die Implementierung von XML-RPC-Clients enthält. Eine dieser Klassen ist `XML_RPC_Value`, die einen Wert im speziellen XML-RPC-Format repräsentiert. Wenn Sie einen Wert in einem XML-RPC-Aufruf verwenden wollen, müssen Sie dafür eine Instanz dieser Klasse erstellen:

```
$nummer = new XML_RPC_Value(32, "int");
```

Der Konstruktor akzeptiert zwei Parameter, den Wert, der übergeben werden soll, und den Datentyp des Werts. Im obigen Beispiel wird als Typ `"int"` verwendet, da es sich um eine Ganzzahl handelt, andere mögliche Werte für diesen Parameter sind `"boolean"`, `"string"`, `"double"`, `"dateTime.iso8601"`, `"base64"`, `"array"` und `"struct"`. Wenn Sie diesen Parameter nicht übergeben, wird angenommen, es handele sich um einen String-Wert.

Nun können Sie mit der nächsten Klasse `XML_RPC_Message` eine XML-RPC-Nachricht erstellen, die dem Methodenaufruf entspricht:

```
$message = new XML_RPC_Message($method, array($nummer));
```

Dem Konstruktor der Klasse übergeben Sie den Namen der Methode, die Sie aufrufen möchten, sowie ein Array mit allen zuvor erzeugten Value-Objekten. Um diese Methode nun an den Server zu schicken, erzeugen Sie ein neues `XML_RPC_Client`-Objekt und übergeben das Nachricht-Objekt an die Methode `send()`:

```
$client   = new XML_RPC_Client($uri, $host);
$response = $client->send($message);
```

Beim Erzeugen des Clients übergeben Sie den Pfad und den Host, um den Endpunkt für die Anfrage zu definieren. Weitere optionale Parameter sind der zu verwendende Port sowie verschiedene Proxy-Einstellungen.

Ist der Server erreichbar, liefert die `send()`-Methode ein `XML_RPC_Response`-Objekt zurück. Ist ein Fehler aufgetreten (wenn z.B. der Server die aufgerufene Methode nicht unterstützt), enthält dieses Objekt weitere Informationen zum Fehler, ansonsten enthält es den Rückgabewert der Methode. Um zu überprüfen, ob ein Fehler aufgetreten ist, kann die Methode `faultCode()` verwendet werden, die im Fehlerfall die Fehlernummer zurückliefert:

```
if ($response->faultCode()) {
    print "Es ist ein Fehler aufgetreten.\n";
}
```

Konnte der Methodenaufruf fehlerfrei ausgeführt werden, können Sie mit der Methode `value()` das Ergebnis anfordern. Dieses Ergebnis ist allerdings immer ein Objekt vom Typ `XML_RPC_Value`. Um dies in eine PHP-Variable zu konvertieren, verwenden Sie einfach die Funktion `XML_RPC_decode()`:

```
$val = $response->value();
$staat = XML_RPC_decode($val);
print "Ich mag $staat!\n";
```

Diese Hilfsfunktion konvertiert jedes beliebige `XML_RPC_Value`-Objekt in den entsprechenden PHP-Datentyp, sodass Sie es wie jede andere PHP-Variable weiterverwenden können.

Siehe auch

Rezept 15.2 mit weiteren Informationen zu XML-RPC-Server; Rezept 15.3 zur PECL XMLRPCi-Erweiterung; das PEAR-Paket XML_RPC unter *http://pear.php.net/package/XML_RPC*.

15.2 XML-RPC-Anfragen empfangen

Problem

Sie möchten einen XML-RPC-Server aufbauen und auf XML-RPC-Anfragen antworten. Auf diese Weise können XML-RPC-fähige Clients Ihrem Server Fragen stellen, und Sie können mit Daten antworten.

Lösung

Verwenden Sie das PEAR-Paket XML_RPC, das eine Klasse zum Erstellen von XML-RPC-Servern zur Verfügung stellt.

```
require_once 'XML/RPC/Server.php';

// Dies ist die als "get_time()" offen gelegte Funktion.
function return_time($args)
{
   $date = date('Ymd\THis');
   $val  = new XML_RPC_Value($date);
   $response = new XML_RPC_Response($val);
   return $response;
}
$map = array(
              'getTime' => array(
                                  'function' => 'return_time'
                                )
           );
$server = new XML_RPC_Server($map);
```

Diskussion

Das PEAR-Paket XML_RPC bietet neben Klassen für das Erstellen von XML-RPC-Clients auch eine Klasse, um einen XML-RPC-Server zu erstellen. Dazu binden Sie einfach die Datei *XML/RPC/Server.php* ein, nachdem Sie das PEAR-Paket installiert haben.

Die Lösung beginnt mit einer Definition der PHP-Funktion, die mit der XML-RPC-Methode assoziiert werden soll. Der Name der Funktion ist `return_time()`. Sie wird später mit der XML-RPC-Methode `getTime()` verknüpft:

```
function return_time($args)
{
   $date = date('Ymd\THis');
   $val  = new XML_RPC_Value($date);
   $response = new XML_RPC_Response($val);
   return $response;
}
```

Diese Funktion liefert einen nach ISO 8601 formatierten String mit dem aktuellen Datum und der aktuellen Zeit. Versehen Sie das `T` innerhalb des Aufrufs von `date( )` mit einem Backslash, da die Spezifikation erfordert, dass ein literales `T` den Datumsteil vom Zeitteil trennt. Für den 21. August 2005, 15:03:51 Uhr, ist der Rückgabewert `20050821T150351`.

Damit der Wert vom Server an den Client zurückgeschickt werden kann, müssen Sie ihn zunächst in ein `XML_RPC_Value`-Objekt verpacken und daraus dann ein `XML_RPC_Response` erzeugen.

Beim Erzeugen des `XML_RPC_Server`-Objekts übergeben Sie ein Array mit den Definitionen der Methoden, die der Server unterstützen soll. In den Schlüsseln dieser sogenannten *Dispatch-Map* stehen die Namen der XML-RPC-Methoden, für jede Methode wird ein Array verwendet, das weitere Informationen zur Methode definiert:

```
$map = array(
            'getTime' => array(
                              'function' => 'return_time'
                             )
          );
```

Mit dem Wert `function` geben Sie den Namen der PHP-Funktion an, die der XML-RPC-Methode entspricht. Über die optionalen Werte `signature` und `docstring` können Sie noch die Signatur der Methode und eine kurze Beschreibung definieren.

Die erstellte Dispatch-Map übergeben Sie als einzigen Parameter dem Konstruktor des Server-Objekts:

```
$server = new XML_RPC_Server($map);
```

Der Server beginnt nun automatisch mit dem Auswerten der eingehenden Anfragen.

Bei einer eingehenden Anfrage liest der Server die XML-Daten automatisch aus der Server-Variablen `$HTTP_RAW_POST_DATA`. Im aktuellen Beispiel hat eine Anfrage den folgenden Aufbau:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<methodCall>
    <methodName>getTime</methodName>
    <params/>
</methodCall>
```

Der Server reagiert also auf die Methode `getTime( )`, und er erwartet keine Parameter. Um eine Anfrage von PHP an den Server zu schicken, können Sie das folgende Skript verwenden:

```
require_once 'XML/RPC.php';

$client  = new XML_RPC_Client('/pfad/zum/xmlrpc-server.php', 'ihre-domain.de');
$message = new XML_RPC_Message('getTime', array());
$response = $client->send($message);

$value = $response->value();
$zeit  = XML_RPC_decode($value);
print $zeit . "\n";
```

Wollen Sie ermöglichen, dass der Client die aktuelle Uhrzeit auch in anderen Formaten anfordern kann, müssen Sie nur die Funktion `return_time()` erweitern:

```
function return_time($args)
{
   $format = $args->getParam(0)->scalarval();
   $date = date($format);
   $val  = new XML_RPC_Value($date);
   $response = new XML_RPC_Response($val);
   return $response;
}
```

Der Parameter `$args` wird vom `XML_RPC_Server`-Objekt beim Aufruf mit einem Objekt der Klasse `XML_RPC_Message` belegt. Dieses Objekt bietet mit der Methode `getParam()` eine Möglichkeit, auf die einzelnen Parameter der Methode zuzugreifen. In diesem Beispiel wird mit `getParam(0)->scalarval()` der erste Parameter ermittelt und vom zurückgegebenen `XML_RPC_Value`-Objekt direkt der skalare PHP-Wert angefordert.

Dieser String wird nun als Format-String für den folgenden Aufruf der `date()`-Funktion verwendet, wodurch der Client die Möglichkeit hat, das Format des zurückgelieferten Werts zu bestimmen. Da die Methode nun einen Parameter erwartet, muss auch die Client-Anwendung angepasst werden und diesen Parameter beim Aufruf der Methode übergeben:

```
require_once 'XML/RPC.php';

$client  = new XML_RPC_Client('/oreilly/x2/x2.2-xmlrpc.php', 'php5.php-tools.net');
$message = new XML_RPC_Message('getTime', array(new XML_RPC_Value('d.m.Y H:i:s',
                               'string')));
$response = $client->send($message);

$value = $response->value();
$zeit  = XML_RPC_decode($value);
print $zeit . "\n";
```

Siehe auch

Die Rezepte 15.1 und 15.3 mit Informationen zu XML-RPC-Clients; das PEAR-Paket XML_RPC unter der URL *http://pear.php.net/package/XML_RPC*.

15.3 XML-RPC-Anfragen mit der XMLRPCi-Erweiterung senden

Problem

Sie möchten möglichst einfach auf Methoden eines XML-RPC-Service zugreifen, ohne sich dabei mit dem XML-RPC-Protokoll oder einer komplizierten API auseinander setzen zu müssen.

Lösung

Verwenden Sie die PHP 5-Erweiterung XMLRPCi, die in PECL zur Verfügung steht.

```
// Neuen Client erzeugen.
$client = new XMLRPC('http://betty.userland.com/RPC2', 'examples.');

// Methode getStateName aufrufen und den Parameter 32 übergeben.
$state  = $client->getStateName(32);

print "Ich mag $state.\n";
```

Sie müssen diese Erweiterung jedoch mit dem PEAR-Installer installieren oder beim Kompilieren mit `--enable-xmlrpci` aktivieren, nachdem Sie sich die aktuellen Quellen von der Homepage des Projekts heruntergeladen haben.

Diskussion

Bei der Überarbeitung des Webservice-Supports von PHP 5 wurde der Fokus auf SOAP gelegt und XML-RPC leider etwas vernachlässigt. Die neue XMLRPCi-Erweiterung in PECL hat sich zum Ziel gesetzt, die bisherige XML-RPC-Implementierung abzulösen, und soll dabei so leicht zu verwenden zu sein wie die neue SOAP-Erweiterung.

Dazu nutzt die Erweiterung analog zum SOAP-Client Objekt-Überladung, um den Zugriff auf entfernte Methoden in der Art zu ermöglichen, als wären sie Teil des lokalen Objekts. Zuerst muss dazu ein neuer XMLRPC-Client erzeugt und die URL des Servers übergeben werden:

```
$client = new XMLRPC('http://betty.userland.com/RPC2', 'examples.');
```

Als zweiter Parameter kann hier optional ein Präfix für die folgenden Methoden-Aufrufe übergeben werden. Der Userland-XML-RPC-Server bietet nun die Methode `examples.getStateName()`, die zu einem Integer-Wert den Namen des passenden US-Bundesstaats zurückliefert. Da Sie bereits `'examples.'` als Präfix für die folgenden Methoden im Konstruktor übergeben haben, können Sie die Methode ganz einfach aufrufen:

```
$state  = $client->getStateName(32);
```

Der Client ruft nun die Methode `'examples.getStateName'` auf dem Server *http://betty.userland.com/RPC2* auf und übergibt den Integer-Wert `32` als einzigen Parameter. Der

Server liefert hierzu den passenden Bundesstaat zurück, der vom Client automatisch in einen PHP-String konvertiert wird, was dann zur folgenden Ausgabe führt:

```
Ich mag New York.
```

Wie bei der SOAP-Erweiterung können Sie auch hier die `__call()`-Methode verwenden, um die RPC-Methoden aufzurufen. So führt der folgende Code zum gleichen Ergebnis:

```
$client = new XMLRPC('http://betty.userland.com/RPC2', 'examples.');
$state  = $client->__call('examples.getStateName', 32);
print "Ich mag $state.\n";
```

Der `__call()`-Methode übergeben Sie als ersten Parameter den kompletten Methodennamen der Funktion (einschließlich Präfix), die auf dem Server aufgerufen werden soll. Alle folgenden Parameter werden vom XML-RPC-Client an den Server weitergereicht.

Sie können die XMLRPCi-Erweiterung z.B. auch verwenden, um Informationen über PEAR-Pakete vom PEAR-Server abzufragen. Im folgenden Beispiel soll die Beschreibung des Pakets HTTP_Server ausgegeben werden. Zusätzlich wird das Beispiel noch um eine Fehlerbehandlung ergänzt:

```
$package = new XMLRPC('http://pear.php.net/xmlrpc.php', 'package.');

try {
    $result = $package->info('HTTP_Server');
} catch (XMLRPC_Fault $fault) {
    echo "Es ist ein Fehler aufgetreten.\n";
    exit();
}
print $result['description'] . "\n";
```

Konnte die Information zum Paket ermittelt werden, erfolgt diese Ausgabe:

```
HTTP server class that allows you to easily implement HTTP servers by supplying
callbacks.
The base class will parse the request, call the appropriate callback and build a repsonse
based on an array that the callbacks have to return.
```

Tritt während der Anfrage ein Fehler auf, da z.B. die Methodensignatur nicht übereinstimmt, reagiert der XMLRPC-Client mit dem Werfen einer Exception vom Typ `XMLRPC_Fault`, die mit einem `try/catch`-Block verarbeitet werden kann.

Die XMLRPCi-Erweiterung ermöglicht einen sehr intuitiven Umgang mit XML-RPC-Services.

Siehe auch

Rezept 15.1 zum Senden von XML-RPC-Anfragen mit der XML-RPC-Erweiterung; die Homepage der XMLRPCi-Erweiterung unter *http://pecl.php.net/package/XMLRPCi*.

15.4 SOAP-Anfragen mit einem WSDL-Dokument senden

Problem

Sie möchten eine SOAP-Anfrage senden. Indem Sie einen SOAP-Client aufsetzen, können Sie Informationen von SOAP-Servern unabhängig von deren Betriebssystem und Middleware-Software abrufen. Der Betreiber des SOAP-Servers stellt Ihnen dazu eine WSDL-Datei zur Verfügung.

Lösung

Verwenden Sie die SOAP-Erweiterung von PHP 5. Hier sehen Sie einen Ausschnitt aus einem Client-Programm, das den SOAP-Dienst von Google verwendet (die API-Keys bekamen Sie bis 2006 von Google; leider wurde hier der SOAP-Dienst als »veraltet« eingestuft, was dazu führte, dass keine neuen API-Keys verteilt werden):

```
<?php
$apiKey = 'XXXXXXXXXXXX';

$client = new SoapClient('http://api.google.com/GoogleSearch.wsdl');
$result = $client->doGoogleSearch(
        $apiKey,             // API-Key
        "php5",              // Suchbegriff
        0,                   // Erster Eintrag der Ergebnisse
        10,                  // Anzahl der Ergebnisse, die geliefert werden
        false,               // Ähnliche Ergebnisse ausschließen
        '',                  // Auf Themen einschränken
        true,                // Keine Inhalte, die nur für Erwachsene sind
        '',                  // Auf Sprache einschränken
        '', ''               // Sprache und Encoding
        );

printf("Insgesamt ungefähr %d Ergebnisse gefunden\n", $result->
estimatedTotalResultsCount);
$i = 0;
foreach ($result->resultElements as $ergebnis) {
    printf("%d. %s\n", ++$i, utf8_decode($ergebnis->title));
}
?>
```

Diskussion

Wie XML-RPC ermöglicht auch SOAP den Austausch von Informationen über HTTP, wobei SOAP-Nachrichten nicht auf das HTTP-Protokoll beschränkt sind, sondern z.B. auch in E-Mails übermittelt werden können. Es verwendet XML als Nachrichtenformat und ist daher einfach zu erzeugen und zu parsen. In der Folge seiner Plattform- und Sprachunabhängigkeit ist SOAP auf vielen Plattformen und in vielen Sprachen verfügbar,

darunter auch in PHP 5. Die SOAP-Erweiterung wird zwar standardmäßig mit PHP 5 ausgeliefert, jedoch muss sie zur Verwendung in PHP-Versionen vor 5.1 mit `--enable-soap` beim Kompilieren aktiviert werden. Dadurch, dass diese Erweiterung in C programmiert wurde, ist sie auch sehr performant.

Um eine SOAP-Anfrage zu starten, müssen Sie zunächst ein neues `SoapClient`-Objekt erzeugen. Sofern Sie eine WSDL-Datei (Web Service Definition Language) erhalten haben, die den SOAP-Dienst beschreibt, übergeben Sie den Dateinamen oder die URL an den Konstruktor:

```
$client = new SoapClient('http://api.google.com/GoogleSearch.wsdl');
```

Dabei liest PHP die WSDL-Datei ein und erstellt einen Proxy, der Ihnen einen sehr einfachen Zugriff auf die Methoden des SOAP-Diensts ermöglicht. Natürlich möchten Sie nicht die WSDL-Datei bei jedem Aufruf Ihres Skripts neu vom Server abholen und parsen, um nicht zu viel Bandbreite und Rechenzeit zu verwenden. PHP ist deshalb in der Lage, einen lokalen Cache für die WSDL-Datei zu verwenden. Die Einstellungen für das Caching können Sie in der PHP-Konfigurationsdatei *php.ini* vornehmen, Tabelle 15-1 erklärt die einzelnen Optionen.

Tabelle 15-1: Konfiguration der SOAP-Erweiterung

Name	Beschreibung
soap.wsdl_cache_enabled	Definiert, ob ein Cache verwendet werden soll (1) oder nicht (0).
soap.wsdl_cache_dir	Verzeichnis, in dem temporäre Cache-Dateien gespeichert werden sollen.
soap.wsdl_cache_ttl	Lebensdauer der Cache-Dateien in Sekunden.

Damit Sie den SOAP-Client nun auch verwenden können, müssen Sie wissen, welche Methoden er bereitstellt und welche Parameter den einzelnen Methoden übergeben werden müssen. Diese Information entnehmen Sie entweder der Dokumentation des SOAP-Diensts oder direkt der WSDL-Beschreibung. Wenn Sie die WSDL-Datei in einem Editor öffnen, sehen Sie mehrere Tags `<operation name="...">`, die die einzelnen Methoden definieren:

```
<operation name="doGoogleSearch">
      <input message="typens:doGoogleSearch"/>
      <output message="typens:doGoogleSearchResponse"/>
</operation>
```

Diese Operation hat also den Namen `doGoogleSearch`, sie wird verwendet, um eine Google-Suche durchzuführen. Die Methode hat eine Eingabe (`<input/>`) vom Typ `doGoogleSearch` und eine Ausgabe (`<output/>`) vom Typ `doGoogleSearchResponse`. Diese Nachrichten-Typen werden auch in der WSDL-Datei definiert:

```
<message name="doGoogleSearch">
    <part name="key" type="xsd:string"/>
    <part name="q" type="xsd:string"/>
    <part name="start" type="xsd:int"/>
```

```
    <part name="maxResults" type="xsd:int"/>
    <part name="filter" type="xsd:boolean"/>
    <part name="restrict" type="xsd:string"/>
    <part name="safeSearch" type="xsd:boolean"/>
    <part name="lr" type="xsd:string"/>
    <part name="ie" type="xsd:string"/>
    <part name="oe" type="xsd:string"/>
</message>

<message name="doGoogleSearchResponse">
    <part name="return" type="typens:GoogleSearchResult"/>
</message>
```

Die Nachricht, die der SOAP-Server erwartet, besteht also insgesamt aus zehn Parametern: dem API-Key, dem Suchbegriff, dem Start der Ergebnisse, der Anzahl der maximal zurückgelieferten Ergebnisse, einem Filter, einer Beschränkung auf Themen, einem Parameter zum Einschränken der Ergebnisse auf jugendfreie Inhalte, einem Parameter zur Einschränkung auf eine Sprache sowie aus dem Encoding der Eingabe und Ausgabe. Alle Eingabewerte müssen in UTF-8 kodiert werden, der SOAP-Server liefert auch UTF-8-kodierte Daten zurück.

Diese Methode und die Parameter können Sie nun in Ihrem Proxy-Client verwenden, als würde es sich um eine lokale Methode handeln:

```
$result = $client->doGoogleSearch(
        $apiKey,            // API-Key
        "php5",             // Suchbegriff
        0,                  // Erster Eintrag der Ergebnisse
        10,                 // Anzahl der Ergebnisse, die geliefert werden
        false,              // Ähnliche Ergebnisse ausschließen
        '',                 // Auf Themen einschränken
        true,               // Keine Inhalte, die nur für Erwachsene sind
        '',                 // Auf Sprache einschränken
        '', ''              // Sprache und Encoding
        );
```

Diese Methode liefert ein Objekt vom Typ `stdClass` zurück:

```
stdClass Object
(
    [documentFiltering] =>
    [searchComments] =>
    [estimatedTotalResultsCount] => 327000
    [estimateIsExact] =>
    [resultElements] => Array
        (
            [0] => stdClass Object
                (
                    [summary] =>
                    [URL] => http://www.php.net/downloads.php
                    [snippet] => search for in the function list. <b>...</b>
                    [title] => PHP: Downloads
                    [cachedSize] => 12k
                    [relatedInformationPresent] => 1
```

```
                    [hostName] =>
                    [directoryCategory] => stdClass Object
                        (
                            [fullViewableName] =>
                            [specialEncoding] =>
                        )
                    [directoryTitle] =>
                )
            [1] => stdClass Object
                (
                    ....
                )
            ....
}
```

Sie können nun einfach auf Eigenschaften dieses Objekts zugreifen, um Informationen zum Suchergebnis auszugeben:

```
printf("Insgesamt ungefähr %d Ergebnisse gefunden\n", $result->
estimatedTotalResultsCount);
```

Oder Sie iterieren die Eigenschaft `resultElements` des Objekts, die ein Array mit allen Suchergebnissen enthält. Jedes Suchergebnis ist wiederum ein Objekt vom Typ `stdClass`, das Informationen zur gefundenen Seite enthält. Beim Ausgeben der Ergebnisse müssen Sie darauf achten, die Informationen zu dekodieren, da Google die Seitentitel und Beschreibungen UTF-8-kodiert liefert:

```
$i = 0;
foreach ($result->resultElements as $ergebnis) {
    printf("%d. %s\n", ++$i, utf8_decode($ergebnis->title));
}
```

Dadurch erzeugen Sie die folgende Ausgabe

```
Insgesamt ungefähr 327000 Ergebnisse gefunden
1. PHP: Downloads
2. PHP: PHP 5 ChangeLog
3. Zend Technologies - PHP 5 InfoCenter - Information, Articles and <b>...</b>
4. PHP 5 InfoCenter - Zend&#39;s <b>PHP5</b> Coding Contest
5. PHPBuilder.com, the best resource for PHP tutorials, templates <b>...</b>
6. What&#39;s keeping you from <b>PHP5</b>? - SitePoint PHP Blog
7. <b>PHP5</b>: Coming Soon to a Webserver Near You [PHP &amp; MySQL Reviews and <b>...</b>
8. <b>PHP5</b>??????- <b>php5</b>,java,jsp,mysql,pgsql,flash,sql,xml <b>...</b>
9. PHPVolcano
10. ONLamp.com: Why PHP 5 Rocks!
```

Der Gebrauch der anderen Methoden, die Google zur Verfügung stellt (z.B. Schreibweisen für falsch geschriebene Wörter anbieten), erfolgt analog zur Google-Suche.

Siehe auch

Rezept 15.5 zum Senden von SOAP-Anfragen ohne WSDL; Rezept 15.6 zum Empfangen von SOAP-Anfragen; die Dokumentation der SOAP-Erweiterung unter *http://www.php.*

net/manual/de/ref.soap; Webservice-Programmierung mit SOAP von Doug Tidwell, James Snell und Pavel Kulchenko (O'Reilly Verlag).

15.5 SOAP-Anfragen ohne ein WSDL-Dokument senden

Problem

Sie möchten eine SOAP-Anfrage senden. Allerdings steht Ihnen keine WSDL-Beschreibung des SOAP-Diensts zur Verfügung, und Sie müssen die Anfrage von Hand ausführen.

Lösung

Verwenden Sie die SOAP-Erweiterung von PHP 5, übergeben Sie anstelle einer WSDL-URL hier `null` sowie weitere Parameter und rufen Sie den Dienst über die `__soapCall()`-Methode des `SoapClient`-Objekts auf. Hier sehen Sie einen Ausschnitt aus einem Client-Programm, das den SOAP-Dienst von Google verwendet:

```
$client = new SoapClient(null,
                         array(
                               'location' => 'http://api.google.com/search/beta2',
                               'uri'          => 'urn:GoogleSearch',
                               'style'       => SOAP_RPC,
                               'use'         => SOAP_ENCODED
                              )
                        );

$params = array(
                new SoapParam('XXXXXXXXXX', 'key'),
                new SoapParam('php5', 'q'),
                new SoapParam(0, 'start'),
                new SoapParam(10, 'maxResults'),
                new SoapParam(false, 'filter'),
                new SoapParam('', 'restrict'),
                new SoapParam(false, 'safeSearch'),
                new SoapParam('', 'lr'),
                new SoapParam('', 'ie'),
                new SoapParam('', 'oe')
               );

$options = array(
                  'uri'              => 'urn:GoogleSearch',
                  'soapaction' => 'urn:GoogleSearch#doGoogleSearch'
                );

$result = $client->__soapCall('doGoogleSearch', $params, $options);

printf('Insgesamt ungefähr %d Ergebnisse gefunden\n', $result->estimatedTotalResultsCount);
```

```
$i = 0;
foreach ($result->resultElements as $ergebnis) {
    printf('%d. %s\n', ++$i, utf8_decode($ergebnis->title));
}
```

Diskussion

Die SOAP-Erweiterung verwendet Objekt-Überladung, um einen Proxy-Client zu erstellen, für den Methoden verwendet werden können, die den Methoden des SOAP-Diensts entsprechen. Dabei wird intern die `__soapCall()`-Methode aufgerufen, die die eigentliche SOAP-Anfrage stellt. Natürlich ist es möglich, diese Methode direkt aufzurufen, um SOAP-Anfragen zu stellen. Haben Sie keine WSDL-Beschreibung des SOAP-Diensts, ist dies sogar der einzige Weg. Dazu erzeugen Sie zuerst ein neues `SoapClient`-Objekt:

```
$client = new SoapClient(null,
                         array(
                               'location' => 'http://api.google.com/search/beta2',
                               'uri'          => 'urn:GoogleSearch',
                               'style'       => SOAP_RPC,
                               'use'         => SOAP_ENCODED
                              )
                        );
```

Anstatt die URL der WSDL-Datei zu übergeben, verwenden Sie hier `null` und übergeben als zweiten Parameter ein Array mit Optionen für den Client. Neben der URL des Diensts (`location`) und dem Namensraum, der verwendet wird (*`uri`*), geben Sie hierbei noch zwei Parameter an, die definieren, welcher Anfragetyp verwendet werden soll.

Danach legen Sie die Parameter für die kommende SOAP-Anfrage fest. Dazu erzeugen Sie mehrere Objekte vom Typ `SoapParam` und speichern diese in einem Array:

```
$params = array(
                new SoapParam('XXXXXXXXXX', 'key'),
                new SoapParam('php5', 'q'),
                new SoapParam(0, 'start'),
                new SoapParam(10, 'maxResults'),
                new SoapParam(false, 'filter'),
                new SoapParam('', 'restrict'),
                new SoapParam(false, 'safeSearch'),
                new SoapParam('', 'lr'),
                new SoapParam('', 'ie'),
                new SoapParam('', 'oe')
               );
```

Dem Konstruktor der `SoapParam`-Klasse übergeben Sie zwei Parameter, den Wert und den Namen des SOAP-Parameters. Die SOAP-Erweiterung ermittelt dabei selbst den Typ; wenn Sie diese also z.B. von einem Formular erhalten haben, sollten Sie darauf achten, sie von einem String in den richtigen Typ (z.B. Integer) zu konvertieren. Welche Parameter übergeben werden müssen, sollten Sie vom Betreiber des SOAP-Diensts erfahren können oder der Dokumentation entnehmen, sofern diese existiert.

Jetzt müssen Sie nur noch zusätzliche Optionen für die SOAP-Anfrage definieren:

```
$options = array(
                  'uri'              => 'urn:GoogleSearch',
                  'soapaction' => 'urn:GoogleSearch#doGoogleSearch'
                );
```

Hier übergeben Sie erneut den Namensraum, den Sie verwenden möchten, wodurch es möglich ist, in einem SOAP-Client mehrere Namensräume zu nutzen und auch pro SOAP-Server mehrmals die Funktion `doGoogleSearch` anzubieten und diese durch unterschiedliche Namensräume eindeutig zu kennzeichnen. Als zweite Option geben Sie hier zusätzlich `soapaction` an, allerdings ignorieren bisher nahezu alle SOAP-Dienste diesen Parameter. Nachdem Sie auch diese Optionen definiert haben, können Sie die SOAP-Anfrage starten:

```
$result = $client->__soapCall('doGoogleSearch', $params, $options);
```

Der `__soapCall()`-Methode übergeben Sie als Erstes die Methode, die Sie aufrufen möchten, gefolgt von den zuvor definierten Parametern und Optionen für die Anfrage. Sie erhalten dann das Ergebnis der SOAP-Anfrage zurück. Im Fall der Google-Suche ist dies ein Objekt vom Typ `stdClass` (siehe Rezept 15.4), das alle Informationen zum Suchergebnis enthält und das Sie einfach ausgeben können:

```
printf('Insgesamt ungefähr %d Ergebnisse gefunden\n', $result->
estimatedTotalResultsCount);
$i = 0;
foreach ($result->resultElements as $ergebnis) {
    printf('%d. %s\n', ++$i, utf8_decode($ergebnis->title));
}
```

Sie erhalten dabei die Ausgabe:

```
Insgesamt ungefähr 327000 Ergebnisse gefunden
1. PHP: Downloads
2. PHP: PHP 5 ChangeLog
3. Zend Technologies - PHP 5 InfoCenter - Information, Articles and <b>...</b>
4. PHP 5 InfoCenter - Zend&#39;s <b>PHP5</b> Coding Contest
5. PHPBuilder.com, the best resource for PHP tutorials, templates <b>...</b>
6. What&#39;s keeping you from <b>PHP5</b>? - SitePoint PHP Blog
7. <b>PHP5</b>: Coming Soon to a Webserver Near You [PHP &amp; MySQL Reviews and <b>...</b>
8. <b>PHP5</b>??????- <b>php5</b>,java,jsp,mysql,pgsql,flash,sql,xml <b>...</b>
9. PHPVolcano
10. ONLamp.com: Why PHP 5 Rocks!
```

Sollten Sie einen Fehler gemacht und z.B. nicht alle Parameter übergeben haben, reagiert der SOAP-Client mit einer `SoapFault`-Exception:

```
Fatal error: Uncaught SoapFault exception: [SOAP-ENV:Server] Exception while handling
service request: com.google.soap.search.GoogleSearchService.doGoogleSearch(java.lang.
String,java.lang.String,int,int,boolean,java.lang.String,java.lang.String,java.lang.
String,java.lang.String) -- no signature match in /home/schst/oreilly/soap.php:30
Stack trace:
#0 {main}
  thrown in /home/schst/oreilly/soap.php on line 30
```

Um diese abzufangen, müssen Sie den Code leicht modifizieren:

```
try {
    $result = $client->__call('doGoogleSearch', $params, $options);
} catch (SoapFault $e) {
    print "SOAP-Anfrage fehlgeschlagen.\n";
}
```

Möchten Sie die SOAP-Erweiterung verwenden, ohne dabei Exceptions verwenden zu müssen, können Sie beim Erzeugen eines neuen SOAP-Clients angeben, ob dieser im Fehlerfall eine Exception erzeugen soll oder nicht:

```
$client = new SoapClient(null,
                         array(
                               'location'     => 'http://api.google.com/search/beta2',
                               'uri'              => 'urn:GoogleSearch',
                               'exceptions' => 0
                              )
                        );
```

Übergeben Sie für die Option `'exceptions'` den Wert `0`, liefert das Objekt bei einem Fehler ein Objekt vom Typ `SoapFault` zurück, anstatt eine Exception zu werfen. Mit der Funktion `is_soap_fault()` können Sie leicht testen, ob eine Rückgabe einen Fehler signalisiert:

```
$result = $client->__call('doGoogleSearch', $params, $options);

if (is_soap_fault($result)) {
    print "SOAP-Anfrage fehlgeschlagen.\n";
    print $result->faultstring . "\n";
    exit();
}
```

Siehe auch

Rezept 15.4 zum Senden von SOAP-Anfragen mit WSDL; Rezept 15.6 zum Empfangen von SOAP-Anfragen; die Dokumentation der SOAP-Erweiterung unter *http://www.php.net/manual/de/ref.soap; Webservice-Programmierung mit SOAP* von Doug Tidwell, James Snell und Pavel Kulchenko (O'Reilly Verlag).

15.6 SOAP-Anfragen empfangen

Problem

Sie möchten einen SOAP-Server einrichten und mit diesem auf SOAP-Anfragen antworten. Wenn Ihr Server auf SOAP-Anfragen reagiert, kann jeder im Internet, der über einen SOAP-Client verfügt, Anfragen an Ihren Server stellen.

Lösung

Erstellen Sie eine WSDL-Datei für den Service und verwenden Sie die SOAP-Erweiterung von PHP 5, mit der Sie die Anfragen an eine PHP-Klasse weiterleiten können:

```
<?xml version ='1.0' encoding ='UTF-8' ?>
<definitions name='Encrypt'
    targetNamespace='http://example.org/Encrypt'
    xmlns:tns=' http://example.org/Encrypt '
    xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
    xmlns:xsd='http://www.w3.org/2001/XMLSchema'
    xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
    xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
    xmlns='http://schemas.xmlsoap.org/wsdl/'>

    <message name='encryptRequest'>
        <part name='passwort' type='xsd:string'/>
        <part name='methode' type='xsd:string'/>
    </message>
    <message name='encryptResponse'>
        <part name='Result' type='xsd:string'/>
    </message>

    <portType name='encryptPortType'>
        <operation name='encrypt'>
            <input message='tns:encryptRequest'/>
            <output message='tns:encryptResponse'/>
        </operation>
    </portType>

    <binding name='encryptBinding' type='tns:encryptPortType'>
        <soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http'/>
        <operation name='encrypt'>
            <soap:operation soapAction='urn:xmethods-delayed-quotes#encrypt'/>
            <input>
                <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
                           encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
            </input>
            <output>
                <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
                           encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
            </output>
        </operation>
    </binding>

    <service name='encryptService'>
        <port name='encryptPort' binding='encryptBinding'>
            <soap:address location='http://ihre.domain.de/server.php'/>
        </port>
    </service>
</definitions>
```

Nun benötigen Sie lediglich eine Klasse, die die SOAP-Anfragen verarbeitet:

```
/**
 * Klasse zum Verschlüsseln von Passwörtern
 */
class CryptServer
{
   /**
    * Passwort verschlüsseln.
    *
    * @param    string    Passwort
    * @param    string    Verschlüsselungstyp
    * @return    string    Verschlüsseltes Passwort
    * @throws   SoapFault
    */
    function encrypt($passwort, $type)
    {
        switch ($type) {
            case 'md5':
                return md5($passwort);
                break;
            case 'md5rev':
                return strrev(md5($passwort));
                break;
            default:
                throw new SoapFault('Server', 'Unbekannter Verschlüsselungsalgorithmus.');
        }
    }
}

// Neuen Server auf Basis der WSDL-Datei erzeugen.
$server = new SoapServer('crypt-server.wsdl');
// Callback-Klasse setzen.
$server->setClass('CryptServer');
// Anfrage verarbeiten.
$server->handle();
```

Diskussion

Um mit der SOAP-Erweiterung von PHP 5 einen eigenen SOAP-Server zur Verfügung stellen zu können, sind die folgenden Schritte nötig:

1. Erstellen Sie eine Klasse und implementieren Sie die Methoden, die Sie als SOAP-Service zur Verfügung stellen möchten.
2. Erstellen Sie eine WSDL-Datei, die den SOAP-Service beschreibt.
3. Erzeugen Sie eine neue Instanz der `SoapServer`-Klasse und übergeben Sie dabei den Pfad zur WSDL-Datei.
4. Übergeben Sie dem Server-Objekt den Namen der Klasse, die die Methoden des SOAP-Service implementiert.

Mit der neuen SOAP-Erweiterung wird das Erstellen von SOAP-Servern mit PHP zum Kinderspiel. Als Erstes erstellen Sie eine neue Klasse und implementieren die Methoden, die Sie öffentlich anbieten möchten. Verwenden Sie hierbei die gleichen Methodennamen, die Sie später auch für den SOAP-Server nutzen möchten. Im Beispiel der `CryptServer`-Klasse wurde eine Methode `encrypt()` implementiert. Der dazugehörige SOAP-Server wird später dann auch eine Methode `encrypt()` zur Verfügung stellen. In dieser Methode können Sie wie gewohnt jede PHP-Funktion verwenden sowie auch beliebige Parameter akzeptieren und beliebige Werte zurückliefern, Sie müssen hier also keine SOAP-spezifischen Regeln beachten. Lediglich die Rückgabe eines Fehlers sollte durch das Werfen einer Exception der Klasse `SoapFault` signalisiert werden.

Nachdem Sie alle Methoden implementiert haben, erstellen Sie eine neue WSDL-Datei, die die zuvor implementierten Methoden beschreibt und auch die URL angibt, unter der Sie später den SOAP-Service anbieten möchten. Sollten Sie Probleme mit dem Erstellen von WSDL-Dateien bekommen, gibt es verschiedene Tools, die diesen Vorgang automatisieren, wie z.B. den WebService Helper unter *http://jool.nl/new/index.php* oder auch das Framework Cerebral Cortex. Diese PHP 5-Anwendungen können WSDL-Dateien aus PHP-Klassen erstellen und somit PHP-Funktionen ohne zusätzlichen Aufwand als Webservice-Methoden anbieten.

Nachdem Sie nun ein WSDL-Dokument manuell oder automatisiert erstellt haben, erzeugen Sie eine neue Instanz der Klasse `SoapServer` und übergeben als ersten Parameter den Dateinamen der WSDL-Datei:

```
$server = new SoapServer('crypt-server.wsdl');
```

Als optionalen zweiten Parameter können Sie ein Array mit Optionen für den SOAP-Server übergeben. Dazu gehören z.B. die SOAP-Version oder auch ein Array, mit dem Sie SOAP-Datentypen auf PHP-Objekte mappen können.

Danach übergeben Sie dem SOAP-Server den Namen der Klasse, die die SOAP-Anfragen verarbeiten soll.

```
$server->setClass('CryptServer');
```

Optional können Sie beliebige Parameter übergeben, die automatisch an den Konstruktor der Klasse weitergegeben werden, wenn dieser vom SOAP-Server aufgerufen wird.

Schließlich müssen Sie nur noch eine Methode aufrufen, um die Abarbeitung der eingehenden SOAP-Anfrage anzustoßen:

```
$server->handle();
```

Beim Aufruf dieser Methode können Sie die SOAP-Anfrage als String übergeben. Übergeben Sie keinen String, so verwendet der Server automatisch die Daten in `$HTTP_RAW_POST_DATA`.

Um Ihren SOAP-Server nun zu testen, erstellen Sie einen neuen SOAP-Client mit der SOAP-Erweiterung von PHP:

```
$client = new SoapClient('http://ihre.domain.de/server.php');
```

Dieser Client bietet Ihnen jetzt automatisch Zugriff auf die `encrypt()`-Methode, die Sie zuvor im Server implementiert haben:

```
$crypt  = $client->encrypt('geheimesPasswort', 'md5');
print "MD5 verschlüsselt: $crypt\n";
```

Wenn Sie dieses Skript ausführen, erhalten Sie die Ausgabe:

```
MD5 verschlüsselt: b76b84abdc024df568b72aee7cad42f0
```

Wenn Sie als zweiten Parameter `'md5rev'` verwenden, können Sie einen anderen Verschlüsselungsalgorithmus nutzen:

```
$crypt  = $client->encrypt('geheimesPasswort', 'md5rev');
print "MD5 verschlüsselt und gespiegelt: $crypt\n";

MD5 verschlüsselt und gespiegelt: 0f24dac7eea27b865fd420cdba48b67b
```

Übergeben Sie jedoch als zweiten Parameter einen unbekannten Algorithmus, so reagiert der SOAP-Server mit dem Werfen einer `SoapFault`-Exception. Diese wird über das SOAP-Protokoll an den Client übermittelt, wo sie wiederum in eine PHP-Exception verwandelt wird.

Der Fehler kann also abgefangen werden, als würde es sich um einen lokalen Fehler handeln:

```
try {
    $crypt  = $client->encrypt('geheimesPasswort', 'sha-1');

} catch (SoapFault $f) {
    print "Es ist ein Fehler aufgetreten.\n";
    print $f;
    exit();
}
```

Der SOAP-Server ermöglicht Ihnen nun also, einen Verschlüsselungsalgorithmus für Passwörter zentral an einer Stelle zu halten und diesen dann von verschiedenen Anwendungen zu nutzen.

Siehe auch

Die Rezepte 15.4 und 15.5 für Informationen zu SOAP-Clients; die Dokumentation der SOAP-Erweiterung unter *http://www.php.net/manual/de/ref.soap*; *Webservice-Programmierung mit SOAP* von Doug Tidwell, James Snell und Pavel Kulchenko (O'Reilly Verlag).

15.7 RSS-Feeds lesen

Problem

Sie möchten einen RSS-Feed empfangen und sich seinen Inhalt ansehen. Auf diese Weise können Sie Schlagzeilen von verschiedenen Websites in Ihre Anwendung integrieren.

Lösung

Verwenden Sie die PEAR-Klasse XML_RSS. Hier ist ein Beispiel, das den RSS-Feed der Mailingliste *php.announce* liest:

```
require 'XML/RSS.php';

$feed = 'http://news.php.net/group.php?group=php.announce&format=rss';

$rss =& new XML_RSS($feed);
$rss->parse();

print "<ul>\n";
foreach ($rss->getItems() as $item) {
    print '<li><a href="' . $item['link'] . '">' . $item['title'] . "</a></li>\n";
}
print "</ul>\n";
```

Diskussion

RSS steht für *RDF Site Summary* und ist ein in XML formuliertes Format für den Austausch von Schlagzeilen oder Artikeln, das einfach zu handhaben ist.[1] Viele Websites mit Nachrichten, zum Beispiel *Slashdot* und O'Reillys *Meerkat*, bieten RSS-Feeds, die bei der Veröffentlichung jeder neuen Meldung aktualisiert werden. Auch Weblog-Seiten haben das RSS-Format entdeckt und bieten RSS-Feeds als standardmäßiges Feature an. Die PHP-Website publiziert ebenfalls RSS-Feeds für die meisten PHP-Mailinglisten.

Einen RSS-Feed kann man problemlos empfangen und zerlegen:

```
$feed = 'http://news.php.net/group.php?group=php.announce&format=rss';

$rss =& new XML_RSS($feed);
$rss->parse();
```

Dieses Beispiel macht `$rss` zu einem neuen `XML_RSS`-Objekt und setzt den Nachrichteneingang auf den RSS-Feed der Mailingliste *php.announce*. Dieser Nachrichteneingang wird dann durch `XML_RSS::parse( )` zerlegt und intern innerhalb von `$rss` abgelegt.

Danach werden die RSS-Items mithilfe von `XML_RSS:getItems( )` in der Form eines assoziativen Arrays ausgelesen:

```
print "<ul>\n";

foreach ($rss->getItems() as $item) {
    print '<li><a href="' . $item['link'] . '">' . $item['title'] . "</a></li>\n";
}

print "</ul>\n";
```

1 RDF ist die Abkürzung für *Resource Definition Framework*. RSS ist auch eine Abkürzung für *Rich Site Summary*.

Diese foreach-Schleife erzeugt eine ungeordnete Liste von Items, wobei jeder Item-Titel wieder zurück auf die URL weist, die mit dem kompletten Artikel verknüpft ist, wie Abbildung 15-1 zeigt. Neben den erforderlichen Feldern `title` und `link` kann ein Item noch ein optionales Feld `description` haben, das eine kurze Zusammenfassung zu dem Item enthält.

Abbildung 15-1: RSS-Feed php.announce

Jeder Nachrichtenkanal verfügt auch über eine Eintragung mit Informationen zu dem Feed, wie in Abbildung 15-2 dargestellt. Diese Daten lesen Sie mithilfe von `XML_RSS::getChannelInfo( )` aus:

```
$feed = 'http://news.php.net/group.php?group=php.announce&format=rss';
$rss = new XML_RSS($feed);

$rss->parse();

print "<ul>\n";

foreach ($rss->getChannelInfo() as $key => $value) {
    print "<li>$key: $value</li>\n";
}

print "</ul>\n";
```

Abbildung 15-2: RSS-Kanalinformationen zu php.announce

Siehe auch

Rezept 14.5 zur direkten Verarbeitung der XML-Daten eines RSS-Feeds und dessen Umwandlung zu HTML; die PEAR-Klasse `XML_RSS` unter *http://pear.php.net/package-info.php?package=XML_RSS*.

15.8 REST-Anfragen senden

Problem

Sie möchten einen REST-basierten Webservice wie z.B. den von Yahoo! ansprechen.

Lösung

Verwenden Sie die PEAR-Pakete HTTP_Request und XML_Serializer, um einen Client für den REST-basierten Webservice zu implementieren.

```
// Die benötigten PEAR-Bibliotheken einbinden.
require_once 'HTTP/Request.php';
require_once 'XML/Unserializer.php';

// Basis-URL der Yahoo!-Suche
$url = 'http://api.search.yahoo.com/WebSearchService/V1/webSearch';

// Neuen Request vorbereiten.
$request = new HTTP_Request($url);

// Applikations-Id an die URL anhängen.
$request->addQueryString('appid', 'YahooDemo');
// Suchbegriff an die URL anhängen.
$request->addQueryString('query', 'PHP5');

// HTTP-Request abschicken.
$request->sendRequest();

// Antwort überprüfen.
if ($request->getResponseCode() !== 200) {
    die("Anfrage konnte nicht gesendet werden.\n");
}
```

```
// Zurückgelieferten XML-Code holen.
$xml = $request->getResponseBody();

// Neues XML_Unserializer-Objekt erzeugen.
$us = new XML_Unserializer();

// Optionen setzen.
$us->setOption('encoding', 'UTF-8');
$us->setOption('targetEncoding', 'ISO-8859-1');
$us->setOption('parseAttributes', true);

// XML einlesen.
$result = $us->unserialize($xml);
if (PEAR::isError($result)) {
    die("XML konnte nicht gelesen werden.\n");
}

$ergebnis = $us->getUnserializedData();

// Ergebnis ausgeben.
print "Die Suche nach PHP5 auf Yahoo! ergab:\n";
foreach ($ergebnis['Result'] as $seite) {
    print $seite['Title'] . " (".$seite['Url'].")\n";
}
```

Diskussion

Obwohl mit SOAP und XML-RPC bereits zwei anerkannte Standards für Webservices existieren, gibt es noch einen dritten Ansatz, der in letzter Zeit immer mehr an Bedeutung gewinnt. REST (*Representational State Transfer*) entstand ursprünglich aus einer Diplomarbeit und macht sich bereits bestehende Standards zu Nutze: das HTTP-Protokoll und XML als Datenformat.

Anstatt bereits die Anfrage in ein XML-Dokument zu verpacken, wie bei SOAP und XML-RPC praktiziert, wird dazu einfach das HTTP-Protokoll verwendet, als Verben können dabei `GET`, `POST`, `PUT` oder auch `DELETE` verwendet werden, die das HTTP-Protokoll schon von Haus aus unterstützt. Parameter werden wie gewohnt bei HTTP-Anfragen an die angeforderte URL angehängt oder direkt in den Pfad der URL eingefügt. Eine typische URL für eine REST-Anfrage könnte also folgendermaßen aussehen:

```
http://www.example.de/xml/suche?begriff=PHP5&seite=2
```

Wenn man diese URL liest, ist jedem sofort klar, dass hier eine Suche nach dem Begriff »PHP5« ausgeführt und die Seite 2 der Ergebnisliste zurückgegeben werden soll.

Ein REST-Webservice gibt dieses Ergebnis als XML-Dokument zurück, jedoch existiert kein Standard, der festlegt, wie dieses XML-Dokument aufgebaut sein muss. Stattdessen überlässt REST dies der Implementierung des Service. So könnte der oben genannte imaginäre Service ein XML-Dokument im folgenden Format zurückliefern:

```
<ergebnis seite="2">
    <seite url="http://www.php.net" titel="PHP Homepage"/>
    <seite url="http://www.zend.com/php5" titel="Zend.com PHP5 Area"/>
    ...
</ergebnis>
```

Dieses XML-Dokument kann nun relativ einfach mit einer der XML-Erweiterungen in PHP verarbeitet werden.

Die Suchmaschine Yahoo! bietet Ihnen die Möglichkeit, Suchanfragen über einen REST-Service zu senden und somit sehr einfach eine Websuche in Ihre Webseite einzubinden. Um eine einfache Suche durchzuführen, müssen Sie *http://api.search.yahoo.com/WebSearchService/V1/webSearch* als Basis-URL verwenden. An diese URL müssen nun mehrere Parameter angehängt werden:

- `appid` definiert Ihre Applikations-Id. Diese erhalten Sie nach der Registrierung für den Yahoo!-Webservice unter der URL *http://api.search.yahoo.com/webservices/register_application*.
- `query` wird verwendet, um den eigentlichen Suchbegriff zu übermitteln.

Neben diesen beiden Parametern können Sie noch weitere optionale Parameter an die URL anhängen, um Ihre Suche weiter zu spezifizieren; so kann diese z.B. über `language` auf eine Sprache eingeschränkt werden, oder durch Verwendung des Parameters `format` kann die Suche auf PDF-Dokumente begrenzt werden. Eine vollständige URL für eine Suche nach »PHP5« sieht also folgendermaßen aus:

```
http://api.search.yahoo.com/WebSearchService/V1/webSearch?appid=YahooDemo&query=PHP5
```

Yahoo! reagiert auf diese Anfrage mit der Auslieferung des folgenden XML-Dokuments:

```
<?xml version="1.0" encoding="UTF-8"?>
<ResultSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns="urn:yahoo:srch"
           xsi:schemaLocation="urn:yahoo:srch http://api.search.yahoo.com/.../V1/
                               WebSearchResponse.xsd"
           totalResultsAvailable="414045" totalResultsReturned="10"
                                  firstResultPosition="1">
    <Result>
        <Title>PHP: Hypertext Preprocessor</Title>
        <Summary>
            search for in the. What is PHP? PHP is a widely-used general-purpose
            scripting language that is especially suited for Web development and can be
            embedded into HTML. ...
            Curso de PHP Avanzado en Bilbao. 7. Formation PHP5. 7. PHP/MySQL in San
            Diego...
        </Summary>
        <Url>http://www.php.net/</Url>
        <ClickUrl>http://rds.yahoo.com/S=96857.../*-http%3A//www.php.net/</ClickUrl>
        <ModificationDate>1111305600</ModificationDate>
        <MimeType>text/html</MimeType>
    </Result>
```

```
    ...
</ResultSet>
```

Möchten Sie nun eine Yahoo!-Suche in Ihre Webseite integrieren, müssen in PHP lediglich die folgenden Schritte abarbeiten:

1. Erzeugen der URL für den Yahoo!-Service.
2. Absenden eines HTTP-Requests.
3. Parsen des zurückgelieferten XML-Dokuments.

Bei diesen Aufgaben bietet es sich an, zwei Pakete aus PEAR zu verwenden: HTTP_Request zum Absenden des HTTP-Requests und XML_Unserializer, um das zurückgelieferte XML-Dokument zu verarbeiten. Die Klasse `HTTP_Request` ermöglicht Ihnen, sehr einfach eine HTTP-Anfrage an einen Server zu senden, im Konstruktor übergeben Sie dazu die Basis-URL, an die die Abfrage geschickt werden soll:

```
// Basis-URL der Yahoo!-Suche
$url = 'http://api.search.yahoo.com/WebSearchService/V1/webSearch';

// Neuen Request vorbereiten.
$request = new HTTP_Request($url);
```

Danach können Sie zusätzliche Parameter, die per `GET` oder `POST` übermittelt werden sollen, mit der Methode `addQueryString()` anhängen:

```
$request->addQueryString('appid', 'YahooDemo');
$request->addQueryString('query', 'PHP5');
```

In diesem Fall werden nur die Parameter `'appid'` und `'query'` übergeben, weitere Parameter können Sie mit der gleichen Methode anhängen. Danach schicken Sie den Request mit der Methode `sendRequest()` an den Server. Nun können Sie den vom Server zurückgelieferten Statuscode mithilfe der Methode `getResponseCode()` ermitteln:

```
if ($request->getResponseCode() !== 200) {
    die("Anfrage konnte nicht gesendet werden.\n");
}
```

Wenn die Anfrage erfolgreich vom Server verarbeitet werden konnte, reagiert dieser mit dem Statuscode 200; wird ein anderer Code zurückgeliefert, wird das Skript an dieser Stelle abgebrochen. Danach können Sie ganz auf den Inhalt der Antwort zugreifen:

```
$xml = $request->getResponseBody();
```

Um dieses XML-Dokument in ein PHP-Array zu verwandeln, mit dem Sie komfortabel weiterarbeiten können, kann die Klasse `XML_Unserializer` aus dem Paket XML_Serializer verwendet werden. Diese Klasse liest beliebige XML-Dokumente ein und erzeugt daraus PHP-Arrays oder -Objekte.

```
$us = new XML_Unserializer();

// Optionen setzen.
$us->setOption('encoding', 'UTF-8');
```

```
$us->setOption('targetEncoding', 'ISO-8859-1');
```

Nach dem Instantiieren des Objekts können Sie verschiedene Optionen setzen, die das Verhalten von XML_Unserializer steuern. Als Erstes wird das aktuelle Encoding des Dokuments übergeben und definiert, in welches Encoding die Textinformationen innerhalb des XML-Baums konvertiert werden sollen. Dadurch werden Umlaute korrekt dargestellt.

Als Zweites wird dem Objekt mitgeteilt, dass Attribute der XML-Tags auch in die PHP-Datenstruktur übernommen werden sollen:

```
$us->setOption('parseAttributes', true);
```

Schließlich müssen Sie nur noch das XML-Dokument, das Sie von Yahoo! zurückgeliefert bekommen haben, an XML_Unserializer übergeben:

```
$result = $us->unserialize($xml);
if (PEAR::isError($result)) {
    die("XML konnte nicht gelesen werden.\n");
}
```

Der Rückgabewert der Methode `unserialize()` wird auf das Auftreten eines Fehlers überprüft. Falls Yahoo! ein ungültiges XML-Dokument zurückgeliefert haben sollte, kann dieser Fehler hier abgefangen werden. Konnte das XML-Dokument fehlerfrei eingelesen werden, liefert die Methode `getUnserializedData()` das erzeugte Array zurück:

```
Array
(
    [xmlns:xsi] => http://www.w3.org/2001/XMLSchema-instance
    [xmlns] => urn:yahoo:srch
    [xsi:schemaLocation] => urn:yahoo:srch http://api.search.yahoo.com/.../V1/
                                                    WebSearchResponse.xsd
    [totalResultsAvailable] => 412068
    [totalResultsReturned] => 10
    [firstResultPosition] => 1
    [Result] => Array
        (
            [0] => Array
                (
                    [Title] => PHP: Hypertext Preprocessor
                    [Summary] => search for in the. What i... in San Diego ...
                    [Url] => http://www.php.net/
                    [ClickUrl] => http://rds.yahoo.com/.../*-http%3A//www.php.net/
                    [ModificationDate] => 1111478400
                    [MimeType] => text/html
                )
            .....
    )
)
```

Dieses Array ähnelt in seiner Struktur dem zurückgelieferten XML-Dokument und enthält die gleichen Inhalte. Sie können nun das Array mit einer Schleife durchlaufen und somit alle Suchergebnisse ausgeben lassen:

```
print "Die Suche nach PHP5 auf Yahoo! ergab:\n";
foreach ($ergebnis['Result'] as $seite) {
    print $seite['Title'] . " (".$seite['Url'].")\n";
}
```

Die Ausgabe dieses Skripts ist:

```
Die Suche nach PHP5 auf Yahoo! ergab:
PHP: Hypertext Preprocessor (http://www.php.net/)
PHP: Downloads (http://www.php.net/downloads.php)
Zend Technologies - PHP 5 InfoCenter - Information, Articles and Tutorials from the PHP
Experts (http://www.zend.com/php5)
FAQTs - Knowledge Base - faqts : Computers : Programming : Languages : Php : General
Information : PHP5 (http://www.faqts.com/knowledge_base/index.phtml/fid/1150)
PHPBuilder.com, the best resource for PHP tutorials, templates, PHP manuals, content
management systems, scripts, ... (http://www.phpbuilder.com/columns/argerich20030411.
php3)
PHP5: Coming Soon to a Webserver Near You [PHP & MySQL Reviews and Apps] (http://www.
sitepoint.com/article/coming-soon-webserver-near)
Beginning PHP5 - Linux Central the /root for Linux resources (http://linuxcentral.com/
catalog/index.php3?prod_code=T000-273)
EvilWalrus.com (http://php5.evilwalrus.com/index.php)
PHP Snapshots (http://snaps.php.net/)
PHP5????????? - php5,java,jsp,mysql,pgsql,flash,sql,xml,php5???????? (http://www.php5.
idv.tw/)
```

Mit den Klassen `HTTP_Request` und `XML_Unserializer` kann nicht nur der Yahoo!-Webservice genutzt werden; das erstellte Skript kann mit sehr wenig Aufwand an nahezu jeden anderen REST-basierten Service angepasst werden. Einige Pakete der Webservice-Kategorie in PEAR verwenden genau diese Herangehensweise.

Natürlich ist es in PHP 5 auch sehr einfach, selbst REST-Anfragen entgegenzunehmen und somit beliebige Funktionalitäten als REST-Service zur Verfügung zu stellen. PHP stellt Ihnen die übermittelten Request-Parameter ja bereits in der superglobalen Variablen $_REQUEST zur Verfügung. Nachdem Sie diese extrahiert und die Ergebnisdaten der Anfrage ermittelt haben, müssen Sie das Ergebnis nur noch als XML-Dokument zurückliefern. Kapitel 14 zeigt Ihnen verschiedene Arten, wie XML-Dokumente erstellt werden können.

Siehe auch

Rezept 14.14 zur Arbeit mit XML_Unserializer; die PEAR-Pakete XML_Serializer unter *http://pear.php.net/package/XML_Serializer* und HTTP_Request unter *http://pear.php.net/package/HTTP_Request*; Informationen zu den Yahoo!-Webservices unter *http://developer.yahoo.net/*.

15.9 Daten mit WDDX austauschen

Problem

Sie möchten Daten im WDDX-Format serialisieren, um sie zu übertragen, oder Sie möchten WDDX-Daten deserialisieren, die Sie empfangen haben. Auf diese Weise können Sie mit jedem kommunizieren, der das WDDX-Format versteht.

Lösung

Verwenden Sie die WDDX-Erweiterung zu PHP. Serialisieren Sie mehrere Variablen durch `wddx_serialize_vars( )`:

```
$a = 'String-Daten';
$b = 123;
$c = 'Roggen';
$d = 'Pastrami';
$array = array('c', 'd');

$wddx = wddx_serialize_vars('a', 'b', $array);
```

Sie können auch mit `wddx_packet_start( )` ein WDDX-Paket beginnen und mit `wddx_add_vars( )` nach Bedarf Daten hinzufügen:

```
$wddx = wddx_packet_start('Einige meiner liebsten Dinge');

// Die Daten in einer Schleife durchlaufen.
while ($array = mysql_fetch_array($r)) {
    $thing = $array['thing'];
    wddx_add_vars($wddx, 'thing');
}

$wddx = wddx_packet_end($wddx);
```

Mit `wddx_deserialize( )` können Sie die Daten wieder deserialisieren:

```
// $wddx enthält eine WDDX-Paket.
$vars = wddx_deserialize($wddx);
```

Diskussion

WDDX steht für *Web Distributed Data eXchange* und war eines der ersten XML-Formate für einen sprachneutralen Datenaustausch. Das Format wurde von der Firma Allaire entwickelt, von der auch ColdFusion stammt, und wurde im Jahr 1999 recht populär. Gegenwärtig hat es jedoch keine allzu große Bedeutung mehr.

Inzwischen haben viele damit begonnen, SOAP als Ersatz für WDDX einzusetzen. Aber der Vorteil von WDDX besteht in seiner Einfachheit; wenn es darum geht, elementare Informationen auszutauschen, kann WDDX immer noch eine gute Wahl darstellen. Außerdem ist es aufgrund seines Ursprungs sehr einfach, WDDX-Pakete mit ColdFusion

zu lesen und zu schreiben. Daher kann WDDX nützlich sein, wenn Sie mit einer ColdFusion-Anwendung kommunizieren.

WDDX benötigt die *expat*-Bibliothek, die in Apache 1.3.7 und höher oder unter *http://www.jclark.com/xml/expat.html* verfügbar ist. Konfigurieren Sie PHP mit `--with-xml` und `--enable-wddx`.

Das Beispiel in der Lösung erzeugt den folgenden XML-Code (hier zur besseren Lesbarkeit formatiert dargestellt):

```
<wddxPacket version='1.0'>
<header/>
<data>
    <struct>
        <var name='a'><string>String-Daten</string></var>
        <var name='b'><number>123</number></var>
        <var name='c'><string>Roggen</string></var>
        <var name='d'><string>Pastrami</string></var>
    </struct>
</data>
</wddxPacket>
```

Variablen werden in `<var>`-Tags verpackt, wobei der Variablenname als Wert des Attributs `name` zugewiesen wird. Darin gibt es einen weiteren Satz von Tags, der den Variablentyp angibt: `string`, `number`, `dateTime`, `boolean`, `array`, `binary` oder `recordSet`. Und schließlich haben Sie die Daten als solche.

Sie können auch eine Variable einzeln mit `wddx_serialize_value` serialisieren:

```
// Eine Variable
$s = wddx_serialize_value('Serialisiert', 'Optionaler Kommentar');
```

Daraus resultiert der folgende XML-Code:

```
<wddxPacket version='1.0'>
<header>
    <comment>Optionaler Kommentar</comment>
</header>
<data>
    <string>Serialisiert</string>
</data>
</wddxPacket>
```

Siehe auch

Die Dokumentation zu WDDX unter *http://www.php.net/wddx*; weitere Informationen in Kapitel 20, »Sharing Data with WDDX« aus *Programming ColdFusion* von Rob Brooks-Bilson (O'Reilly).

KAPITEL 16

Reguläre Ausdrücke

16.0 Einführung

Reguläre Ausdrücke sind ein leistungsstarkes Hilfsmittel für die Suche nach Textübereinstimmungen (*Matching*) und für die Bearbeitung von Texten. Sie sind nicht so schnell wie einfache String-Vergleiche, dafür aber extrem flexibel. Mit ihrer Hilfe erstellen Sie in einer einfachen, aber knappen und etwas undurchsichtigen Syntax Muster, um praktisch jede denkbare Kombination von Zeichen finden zu können.

PHP verfügt über Funktionen für reguläre Ausdrücke, mit denen Sie Texte nach Stellen durchsuchen können, die mit bestimmten Kriterien übereinstimmen. Wenn Sie solche Stellen gefunden haben, können Sie sie verändern, oder Sie können alle oder einen Teil der übereinstimmenden Teil-Strings ersetzen. Der folgende reguläre Ausdruck verwandelt beispielsweise E-Mail-Adressen in `mailto`:-Hyperlinks:

```
$html = preg_replace('/[^@\s]+@([-a-z0-9]+\.)+[a-z]{2,}/i',
                     '<a href="mailto:$0">$0</a>', $text);
```

Wie Sie sehen, sind reguläre Ausdrücke hilfreich bei der Umwandlung von einfachem Text in HTML und umgekehrt. Da dies häufig erforderlich ist, verfügt PHP glücklicherweise für solche Aufgaben über viele eingebaute Funktionen. Rezept 11.8 erläutert, wie man HTML-Entities in Escape-Sequenzen verwandelt, 13.13 behandelt das Entfernen von HTML-Tags, und die Rezepte 13.11 und 13.12 zeigen, wie man ASCII in HTML beziehungsweise HTML in ASCII konvertieren kann. Mehr über das Suchen und Prüfen von E-Mail-Adressen finden Sie in Rezept 16.5.

Im Laufe der Jahre ist die Funktionalität der regulären Ausdrücke gegenüber ihren ersten Anfängen stark weiterentwickelt worden und umfasst jetzt viele zusätzliche Möglichkeiten. Als Konsequenz bietet PHP zwei verschiedene Gruppen von Funktionen für reguläre Ausdrücke. Die erste enthält die traditionellen Funktionen (nach POSIX), die alle mit `ereg` beginnen (für *extended regular expressions*; die `ereg`-Funktionen sind selbst bereits eine Erweiterung des ursprünglichen Funktionsumfangs). Die andere Gruppe umfasst

die Familie der Perl-Funktionen mit einem vorangestellten preg (für Perl-kompatible reguläre Ausdrücke).

Die preg-Funktionen setzen auf einer Bibliothek auf, die reguläre Ausdrücke wie in der Programmiersprache Perl interpretiert. Der Vorteil besteht darin, dass Sie nun in PHP reguläre Ausdrücke für eine Vielzahl nützlicher Dinge verwenden können, darunter die nicht-gierige Suche nach Übereinstimmungen (*nongreedy matching*), Vorwärts- und Rückwärts-Assertions und sogar rekursive Muster.

Eigentlich spricht nichts mehr dafür, die ereg-Funktionen zu verwenden. Sie bieten einen geringeren Funktionsumfang und sind langsamer als die preg-Funktionen. Allerdings gab es in PHP die ereg-Funktionen schon viele Jahre vor der Einführung der preg-Funktionen, daher werden sie von vielen Programmierern wegen alter Programme oder aus Gewohnheit noch immer verwendet. Zum Glück sind die Funktionsprototypen beider Bibliotheken identisch, sodass man einfach und ohne allzu große Verwirrung zwischen beiden hin- und herschalten kann. (In Rezept 16.1 zeigen wir Ihnen, wie Sie dabei einige Fallen vermeiden.)

Die Grundlagen der regulären Ausdrücke sind leicht zu verstehen. Zunächst kombinieren Sie eine Folge von Zeichen zu einem Muster. Dann vergleichen Sie Text-Strings mit diesem Muster und suchen nach übereinstimmenden Stellen. In dem Muster repräsentieren die meisten Zeichen sich selbst. Folgendermaßen können Sie zum Beispiel feststellen, ob ein HTML-String ein Bild enthält:

```
if (preg_match('/<img /', $html)) {
    // Öffnendes Tag für ein Bild gefunden.
}
```

Die Funktion preg_match() vergleicht das Muster "<img " mit dem Inhalt von $html. Wenn sie einen Match findet, gibt sie 1 zurück, andernfalls 0. Die Schrägstriche (/) werden als Muster-Begrenzungszeichen (*Pattern-Delimiter*) bezeichnet, sie kennzeichnen den Anfang und das Ende des Musters. Hier können Sie auch andere Zeichen an ihrer Stelle angeben.

Allerdings gibt es einige spezielle Zeichen. Die Besonderheiten dieser Zeichen verleihen einem regulären Ausdruck die über strstr() und strpos() hinausgehenden Fähigkeiten. Man nennt diese Zeichen *Metazeichen*. Zu den am häufigsten verwendeten Metazeichen gehören der Punkt (.), der Stern (*), das Pluszeichen (+) und das Fragezeichen (?). Wenn Sie nach Übereinstimmungen mit einem Metazeichen als solchem suchen, setzen Sie einen Backslash (\) davor.

- Der Platzhalter »Punkt« liefert eine Übereinstimmung für ein beliebiges Zeichen; das Muster /.ein/ passt dementsprechend zu mein, dein und auch Wein.
- Der Quantifikator »Stern« besagt, dass eine zuvor festgelegte Übereinstimmung beliebig oft vorkommen kann. Eine fehlende Übereinstimmung ist somit auch erlaubt. Das Muster /.*ein/ passt auf Pein und Schwein, aber auch auf ein.

- Der Quantifikator »Pluszeichen« ähnelt dem Stern. Er gibt an, dass eine zuvor definierte Übereinstimmung mindestens einmal, aber auch mehrmals vorkommen darf. Deshalb passt `/.+ein/` auf `Pein`, `Schwein`, aber nicht auf `ein`. Damit das Muster mit `ein` übereinstimmen kann, muss das + durch ein * im Suchmuster ersetzt werden.
- Der Quantifikator »Fragezeichen« legt die Trefferanzahl der Übereinstimmungen auf »0 oder 1« fest. Das Muster `/.?ein/` passt somit auf `Pein` und `ein`, nicht aber auf `Schwein`.

Um `*`, `+` und `?` auf Übereinstimmungen anzuwenden, die länger als ein Zeichen sind, umgeben Sie die entsprechende Zeichenfolge mit runden Klammern, um sie zu gruppieren. Darüber hinaus haben Klammerungen auch die wichtige Eigenschaft, die mit ihnen korrespondierenden Teiltreffer des Musters einzufangen. Eine eingefangene Zeichenfolge kann dann in `preg_replace()` referenziert und modifiziert werden. Außerdem können alle eingefangenen Sequenzen in einem Array gespeichert werden, das den Funktionen `preg_match()` und `preg_match_all()` jeweils als dritter Parameter übergeben wird. Die Funktion `preg_match_all()` ähnelt der Funktion `preg_match()`, sucht aber nach allen möglichen Übereinstimmungen innerhalb eines Strings und bricht nicht nach der ersten Übereinstimmung ab. Hier folgen einige Beispiele:

```
if (preg_match('/<title>.+<\/title>/', $html)) {
    // Die Seite hat einen Titel.
}

if (preg_match_all('/<li>/', $html, $matches)) {
    print 'Die Seite hat ' . count($matches[0]) . " Listenelemente\n";
}

// Fettschrift in kursiv umwandeln.
$italics = preg_replace('/(<\/?)b(>)/', '$1i$2', $bold);
```

Wenn Sie Strings auf Übereinstimmungen mit einer bestimmten Menge von Zeichen prüfen möchten, erzeugen Sie eine Zeichenklasse mit den gewünschten Buchstaben. Eine *Zeichenklasse* ist eine Folge von Zeichen, die innerhalb eckiger Klammern steht. Das Caret-Zeichen (`^`) und das Dollarzeichen (`$`) verankern das Muster am Anfang beziehungsweise am Ende des Strings. Ohne sie kann die Übereinstimmung irgendwo innerhalb des Strings auftreten. Um also auf Vokale zu prüfen, erzeugen Sie eine Zeichenklasse aus `a`, `e`, `i`, `o` und `u`, beginnen das Muster mit `^` und beenden es mit `$`:

```
preg_match('/^[aeiou]+$/', $string); // nur Vokale
```

Manchmal ist es einfacher, das Gesuchte durch sein Gegenteil zu definieren. Eine Zeichenklasse, die mit allen außer den aufgeführten Zeichen übereinstimmen soll, beginnt mit einem Caret. Ein Caret außerhalb einer Zeichenklasse verankert ein Muster am Anfang eines Strings, ein Caret innerhalb einer Zeichenklasse bedeutet, dass diese Klasse mit allem übereinstimmt, was nicht innerhalb der eckigen Klammern aufgeführt ist.

```
preg_match('/^[^aeiou]+$/', $string) // nur Nicht-Vokale
```

Beachten Sie, dass das Gegenteil von [aeiou] nicht etwa [bcdfghjklmnpqrstvwxyz] ist. Die Zeichenklasse [^aeiou] passt auch auf Vokale in Großbuchstaben wie AEIOU, Zahlen wie 123, URLs wie http://www.cnpq.br/ und sogar auf Emoticons wie :).

Der senkrechte Balken (|), auch Pipe-Zeichen genannt, benennt Alternativen. Ein Beispiel:

```
// gif oder jpeg finden.
preg_match('/(gif|jpeg)/', $images);
```

Neben Metazeichen gibt es auch *Metasymbole*. Diese ähneln den Metazeichen, sind aber länger als ein Zeichen. Einige nützliche Metasymbole sind \w (alle Zeichen, die zu einem Wort gehören können [a-zA-Z0-9_]); \d (alle Ziffern [0-9]); \s (alle Zeichen, die Leerraum ergeben) und \b (Wortgrenze). Folgendermaßen finden Sie alle Zahlen, die nicht Teil eines anderen Worts sind:

```
// Alle Ziffern finden, die keine anderen Wörter berühren.
preg_match_all('/\b\d+\b/', $html, $matches);
```

Dies passt auf 123, 76! und 38-jährig, aber nicht auf 2ter.

Das folgende Muster ist ein Äquivalent zu trim() in der Form eines regulären Ausdrucks:

```
// Führenden und nachfolgenden Leerraum entfernen.
$trimmed = preg_replace('/(^\s+)|(\s+$)/', '', $string);
```

Schließlich gibt es noch Modifikatoren für Muster. Modifikatoren beeinflussen das ganze Muster, nicht nur ein Zeichen oder eine Gruppe von Zeichen. Die Muster-Modifikatoren stehen hinter dem schließenden Begrenzungszeichen des Musters. Der Buchstabe i beispielsweise führt dazu, dass der reguläre Ausdruck nicht zwischen Groß- und Kleinschreibung unterscheidet:

```
// Passt exakt nur auf Image-Tags in Kleinbuchstaben (XHTML-konform).
if (preg_match('/<img[^>]+>/', $html)) {
   ...
}

// Passt auf Image-Tags in Groß- und Kleinbuchstaben.
if (preg_match('/<img[^>]+>/i', $html)) {
   ...
}
```

Wir haben uns hier nur mit einem kleinen Ausschnitt aus der Welt der regulären Ausdrücke befasst. Die folgenden Rezepte enthalten noch weitere Einzelheiten, aber auch auf der PHP-Website finden Sie einige sehr nützliche Informationen zu regulären Ausdrücken nach POSIX unter *http://www.php.net/regex* und zu Perl-kompatiblen regulären Ausdrücken unter *http://www.php.net/pcre*. Die Links in der letzteren Seite zu den Themen »Pattern Modifiers« und »Pattern Syntax« sind besonders detailliert und informativ.

Die besten Bücher zu diesem Thema sind *Reguläre Ausdrücke* von Jeffrey Friedl und *Programmieren mit Perl* von Larry Wall, Tom Christiansen und Jon Orwant, beide erschienen im O'Reilly Verlag. (Da die Perl-kompatiblen regulären Ausdrücke auf den regulären

Ausdrücken von Perl basieren, haben wir kein allzu schlechtes Gewissen, ein Buch über Perl zu empfehlen.)

16.1 Von ereg zu preg wechseln

Problem

Sie möchten ein Programm so konvertieren, dass es preg-Funktionen anstelle von ereg-Funktionen verwendet.

Lösung

Zunächst müssen Sie Ihre Muster mit Begrenzungszeichen (*Delimiters*) versehen:

```
preg_match('/pattern/', 'string')
```

Anstelle der Funktion `eregi()`, die nicht zwischen Groß- und Kleinschreibung unterscheidet, verwenden Sie den Modifikator `/i`:

```
preg_match('/pattern/i', 'string');
```

Wenn Sie in Mustern oder Ersetzungswerten Integer-Zahlen verwenden, die Strings repräsentieren sollen, konvertieren Sie jede Zahl in eine Hexadezimalzahl und geben sie als Escape-Sequenz an:

```
$hex = dechex($number);
preg_match("/\x$hex/", 'string');
```

Diskussion

Es gibt einige wesentliche Unterschiede zwischen ereg und preg. Zunächst einmal ist bei den preg-Funktionen das Muster nicht einfach nur der String `Muster`. Zusätzlich benötigt es noch wie bei Perl Begrenzungszeichen, deshalb muss es stattdessen `/Muster/` lauten.[1] Somit wird aus dem Aufruf

```
ereg('Muster', 'String');
```

der Aufruf:

```
preg_match('/Muster/', 'String');
```

Bei der Auswahl der Begrenzungszeichen für das Muster müssen Sie darauf achten, dass das Begrenzungszeichen nicht innerhalb des regulären Ausdrucksmusters auftritt, da sonst das Muster zu früh geschlossen wird. Wenn Sie dieses Problem nicht anders vermeiden können, müssen Sie alle Stellen, an denen Ihre Begrenzungszeichen auftreten, mit einem Backslash versehen. Statt dies manuell zu tun, rufen Sie einfach `addcslashes()` auf.

1 Oder {}, <>, ||, ## oder was auch immer Ihre bevorzugten Begrenzungszeichen sind. PHP unterstützt sie alle.

Wenn Sie beispielsweise / als Begrenzungszeichen verwenden:

```
$ereg_pattern = '<b>.+</b>';
$preg_pattern = addcslashes($ereg_pattern, '/');
```

Der Inhalt von `$preg_pattern` ist anschließend `<b>.+<\/b>`.

Zu den preg-Funktionen existiert keine parallele Reihe von Funktionen, die unabhängig von der Groß-/Kleinschreibung arbeiten. Stattdessen gibt es einen Modifikator für den Fall, dass das Matching nicht von der Schreibweise abhängen soll. Zur Konvertierung ersetzen Sie:

```
eregi('Muster', 'String');
```

durch:

```
preg_match('/Muster/i', 'String');
```

Das zusätzliche `i` hinter dem schließenden Begrenzungszeichen macht den Unterschied aus.

Schließlich gibt es noch eine letzte, unscheinbare Differenz. Wenn Sie bei `ereg_replace()` eine Zahl (nicht einen String) als Muster oder Ersetzungswert verwenden, wird angenommen, dass Sie den ASCII-Wert eines Zeichens meinen. Da 9 die ASCII-Repräsentation des Tabulator-Zeichens ist (d.h. `\t`), fügt der folgende Code an allen Zeilenanfängen Tabulatoren ein:

```
$tab = 9;
$replaced = ereg_replace('^', $tab, $string);
```

Hier sehen Sie, wie man Zeilenvorschübe an den Zeilenenden konvertiert:

```
$converted = ereg_replace(10, 12, $text);
```

Um diese Eigenschaft von ereg-Funktionen zu vermeiden, schreiben Sie stattdessen:

```
$tab = '9';
```

Dagegen behandelt `preg_replace()` die Zahl 9 als die Zahl 9 und nicht als Ersatz für das Tab-Zeichen. Um solche Zeichen-Codes für die Verwendung mit `preg_replace()` zu konvertieren, verwandeln Sie diese in Hexadezimalwerte und stellen `\x` voran. Aus `9` wird beispielsweise `\x9` oder `\x09`, und aus `12` wird `\x0c`. Alternativ können Sie auch `\t`, `\r` und `\n` als Tab-Zeichen, Wagenrücklauf (*Carriage Return*) oder Zeilenvorschub verwenden.

Siehe auch

Die Dokumentationen zu `ereg()` unter *http://www.php.net/ereg*, `preg_match()` unter *http://www.php.net/preg-match* und `addcslashes()` unter *http://www.php.net/addcslashes*.

16.2 Wörter suchen

Problem

Sie möchten alle Wörter aus einem String herausfiltern.

Lösung

Der Schlüssel liegt in der sorgfältigen Definition dessen, was Sie unter einem Wort verstehen. Wenn Sie Ihre Definition haben, erzeugen Sie einen regulären Ausdruck mithilfe der speziellen Zeichentypen:

```
/\S+/         // alles, was kein Leerraum ist
/[A-Z'-]+/i   // alle Groß- und Kleinbuchstaben, Apostrophe und Bindestriche
```

Diskussion

Die einfache Frage: »Was ist ein Wort?« ist überraschend kompliziert. Zwar kennen Perl-kompatible reguläre Ausdrücke einen eingebauten Wort-Zeichentyp, der mit `\w` angegeben wird, aber Sie müssen genau verstehen, wie PHP ein Wort definiert. Ansonsten sind die Resultate anders, als Sie es erwarten.

Da `\w` direkt aus Perls Definition eines Wortes abgeleitet ist, umfasst es normalerweise alle Buchstaben, Ziffern und Unterstriche. Dementsprechend ist `a_z` ein Wort, die E-Mail-Adresse `php@example.com` jedoch nicht.

In diesem Beispiel betrachten wir nur Wörter der englischen Sprache, aber andere Sprachen verwenden möglicherweise andere Alphabete. Da die Einstellungen für Perl-kompatible reguläre Ausdrücke anhand der aktuellen Ländereinstellung (*Locale*) definiert werden, kann die Veränderung des Locale die Definition eines Buchstabens verändern und damit auch die Bedeutung dessen, was ein Wort ist.

Um das zu verhindern, sollten Sie möglicherweise die zu einem Wort gehörenden Zeichen innerhalb einer Zeichenklasse explizit aufführen. Um ein nicht-standardmäßiges Zeichen hinzuzufügen, verwenden Sie dabei `\ddd`, wobei `ddd` der Oktal-Code des Zeichens ist.

Siehe auch

Rezept 19.2 mit Informationen zu den Locale-Einstellungen.

16.3 Den n-ten Match finden

Problem

Sie möchten die *n*-te Wortübereinstimmung finden und nicht die erste.

Lösung

Mit `preg_match_all()` übernehmen Sie alle Übereinstimmungen in ein Array. Dann filtern Sie diejenigen Übereinstimmungen heraus, an denen Sie interessiert sind:

```
preg_match_all ("/$muster/$modifikatoren", $string, $matches)

foreach($matches[1] as $match) {
    print "$match\n";
}
```

Diskussion

Anders als in Perl unterstützen die Perl-kompatiblen regulären Ausdrücke in PHP nicht den Modifikator `/g`, mit dem Sie in einem String eine Übereinstimmung nach der anderen suchen können. Sie müssen `preg_match_all()` anstelle von `preg_match()` verwenden.

Die Funktion `preg_match_all()` gibt ein zweidimensionales Array zurück. Das erste Element enthält ein Array mit Übereinstimmungen zum vollständigen Muster. Das zweite Element enthält ebenfalls ein Array mit Übereinstimmungen, hier sind es aber die eingeklammerten Unter-Übereinstimmungen innerhalb jeder vollständigen Übereinstimmung. Um also das Wort vor dem dritten `potato` zu finden, greifen Sie auf das dritte Element des zweiten Elements des Arrays `$matches` zu:

```
$potatoes = 'one potato two potato three potato four';
preg_match_all("/(\w+)\s+potato\b/", $potatoes, $matches);
print $matches[1][2];
three
```

Anstelle eines Arrays, das in die kompletten Übereinstimmungen und dann in die Unter-Übereinstimmungen aufgeteilt ist, kann `preg_match_all()` auch ein Array liefern, das in die Gesamt-Übereinstimmungen aufgeteilt ist, innerhalb deren sich die Unter-Übereinstimmungen befinden. Um dieses Verhalten auszulösen, übergeben Sie `PREG_SET_ORDER` als viertes Argument. Nun steht `three` nicht wie zuvor in `$matches[1][2]`, sondern in `$matches[2][1]`.

Prüfen Sie den Rückgabewert von `preg_match_all()`, um die Anzahl der Treffer herauszufinden:

```
print preg_match_all("/(\w+)\s+potato\b/", $potatoes, $matches);
3
```

Beachten Sie, dass es drei Übereinstimmungen gibt und nicht vier, da im String auf das Wort `four` kein `potato` mehr folgt.

Siehe auch

Die Dokumentation zu `preg_match_all()` unter *http://www.php.net/preg-match-all*.

16.4 Zwischen gierigem und nicht-gierigem Matching wählen

Problem

Sie möchten mit Ihrem Muster die kleinstmögliche Übereinstimmung finden und nicht die größtmögliche.

Lösung

Fügen Sie ein `?` nach einem Quantifikator ein, um diesen Teil des Musters zu modifizieren:

```
// Alle fett geschriebenen Abschnitte finden.
preg_match_all('#<b>.+?</b>#', $html, $matches);
```

Oder verwenden Sie den Muster-Modifikator `U`, um alle Quantifikatoren von gierig auf nicht-gierig umzuschalten:

```
// Alle fett geschriebenen Abschnitte finden.
preg_match_all('#<b>.+</b>#U', $html, $matches);
```

Diskussion

Standardmäßig entspricht das Verhalten von regulären Ausdrücken in PHP dem, was man als *gierig* (greedy) bezeichnet. Dies bedeutet, dass ein Quantifikator stets versucht, eine Übereinstimmung mit so vielen Zeichen wie möglich zu finden.

Nehmen Sie als Beispiel das Muster `p.*`, das auf ein `p` passt, auf das null oder mehrere Zeichen folgen, und prüfen Sie, ob es im String `php` vorkommt. Ein gieriger regulärer Ausdruck findet genau eine Übereinstimmung, da er sich das `p` am Anfang greift, dann weiterläuft und auch eine Übereinstimmung mit `hp` feststellt. Ein nicht-gieriger regulärer Ausdruck dagegen findet zwei Übereinstimmungen. Wie zuvor stimmt er mit dem `p` und dem folgenden `h` überein, aber anstatt weiterzusuchen, bricht er ab und lässt das abschließende `p` unbeachtet. Eine zweite Übereinstimmung fährt dann fort und übernimmt den letzten Buchstaben.

Der folgende Code zeigt, dass die gierige Übereinstimmung nur einen Treffer, die nicht-gierige jedoch zwei erzielt:

```
$results = array();
print preg_match_all('/p.*/', "php", $results);  // gierig
print preg_match_all('/p.*?/', "php", $results); // nicht-gierig
print preg_match_all('/p.*/U', "php", $results); // nicht-gierig
1
2
2
```

Gieriges Matching wird auch als *maximales Matching* und das nicht-gierige als *minimales Matching* bezeichnet, da jeweils nach Übereinstimmungen mit der maximal oder minimal möglichen Anzahl von Zeichen gesucht wird.

Anfänglich waren regulären Ausdrücke grundsätzlich gierig. Daher können Sie die obige Syntax nicht mit `ereg()` oder `ereg_replace()` verwenden. Nicht-gieriges Matching wird durch eine für diese Funktionen verwendete ältere Engine nicht unterstützt; daher müssen Sie hierfür Perl-kompatible Funktionen verwenden.

Nicht-gieriges Matching ist besonders hilfreich beim simplen Zerlegen von HTML. Angenommen, Sie möchten alle Texte finden, die zwischen Fettschrift-Tags stehen, dann erhalten Sie bei gierigem Matching dies:

```
$html = '<b>Ich bin fett.</b> <i>Ich bin kursiv.</i> <b>Ich bin auch fett.</b>';
preg_match_all('#<b>(.+)</b>#', $html, $bolds);
print_r($bolds[1]);
Array
(
    [0] => Ich bin fett.</b> <i>Ich bin kursiv.</i> <b>Ich bin auch fett.
)
```

Da hier noch ein zweites Mal Tags für Fettschrift auftreten, reicht das Muster über das erste `</b>` hinaus, und es ist unmöglich, den HTML-Code korrekt aufzubrechen. Wenn Sie minimales Matching verwenden, bleibt jedes Tag-Paar in sich erhalten:

```
$html = '<b>Ich bin fett.</b> <i>Ich bin kursiv.</i> <b>Ich bin auch fett.</b>';
preg_match_all('#<b>(.+?)</b>#', $html, $bolds);
print_r($bolds[1]);
Array
(
    [0] => Ich bin fett.
    [1] => Ich bin auch fett.
)
```

Das funktioniert natürlich nur, wenn Ihr Markup zu 100 Prozent korrekt ist und es keine einzeln verstreuten Fettschrift-Tags gibt.[2] Wenn es nur darum geht, alle (oder einige) HTML-Tags aus einem Textblock zu entfernen, verwenden Sie besser keinen regulären

2 Auch mit gültigem HTML kann es noch Probleme geben, wenn Sie beispielsweise Tags für Fettschrift innerhalb eines Kommentars haben. Ein richtiger HTML-Parser würde einen solchen Abschnitt ignorieren, aber unser Muster nicht.

Ausdruck, sondern die eingebaute Funktion `strip_tags();` sie ist schneller und funktioniert korrekt. In Rezept 13.13 finden Sie weitere Einzelheiten.

Es sei noch angemerkt, dass, obwohl die Idee des nicht-gierigen Matching von Perl stammt, der Modifikator `-U` nicht mit Perl kompatibel ist und eine Besonderheit der regulären Ausdrücke in PHP darstellt. Dieser Modifikator verkehrt alle Quantifikatoren, indem er sie von gierig nach nicht-gierig und umgekehrt umschaltet. Um also einen gierigen Quantifikator innerhalb eines Musters zu erhalten, das ein nachfolgendes `U` enthält, fügen Sie einfach ein `?` am Ende hinzu, genau wie Sie normalerweise einen gierigen Quantifikator in einen nicht-gierigen verwandeln.

Siehe auch

Rezept 16.7 mit weiteren Informationen zum Auslesen von Text innerhalb von HTML-Tags; Rezept 13.13 mit weiteren Informationen zum Entfernen von HTML-Tags; die Dokumentation zu `preg_match_all()` unter *http://www.php.net/preg-match-all*.

16.5 E-Mail-Adressen validieren

Problem

Sie möchten prüfen, ob eine E-Mail-Adresse gültig ist.

Lösung

Dies ist ein verbreitetes Problem, zu dem jeder eine andere Antwort hat, die von der jeweiligen Definition der Gültigkeit abhängt. Wenn »gültig« für Sie bedeutet, dass es sich um die Mailbox eines legitimierten Benutzers mit einem existierenden Host-Namen handelt, können Sie dies eigentlich nicht korrekt überprüfen; also machen Sie sich erst gar keine Gedanken darüber. Allerdings kann bisweilen ein regulärer Ausdruck helfen, einfache Schreibfehler oder offensichtliche Fälschungsversuche auszusieben. Unter dieser Voraussetzung sieht unser bevorzugtes Muster, das keiner weiteren Pflege bedarf, folgendermaßen aus:

```
/^[^@\s]+@([-a-zO-9]+\.)+[a-z]{2,}$/i
```

Wenn die IMAP-Erweiterung aktiviert ist, können Sie auch `imap_rfc822_parse_adrlist()` verwenden:

```
$parsed = imap_rfc822_parse_adrlist($email_address, $default_host)
if ('INVALID_ADDRESS' == $parsed['mailbox']) {
    // ungültige Adresse
}
```

Kurioserweise liefert diese Funktion, gerade weil sie so RFC-konform ist, nicht unbedingt die erwarteten Ergebnisse.

Diskussion

Das Muster in der Lösung akzeptiert jede E-Mail-Adresse, die einen Namen aus einer beliebigen Zeichenfolge ohne `@` und Leerzeichen hat. Nach dem `@` benötigen Sie mindestens einen Domainnamen, der aus den Buchstaben `a–z`, den Ziffern `0–9` und dem Bindestrich bestehen kann und der durch beliebig viele, durch Punkte getrennte Subdomains ergänzt sein kann. Am Ende steht dann ein zweistelliger Country-Code oder eine andere Top-Level-Domain wie `.com` oder `.edu`.

Das Muster in der Lösung ist praktisch, da es auch dann noch funktioniert, wenn ICANN zusätzliche Top-Level-Domains hinzufügt. Allerdings lässt es auch einige definitiv falsche Eingaben durch. Das folgende strengere Muster führt die aktuellen, nicht länderbezogenen Top-Level-Domains einzeln auf:

```
/
    ^               # Am Anfang verankern.
    [^@\s]+         # Name besteht aus allen Zeichen außer @ und Leerzeichen.
    @               # Das @ trennt den Namen von der Domain.
    (
        [-a-z0-9]+  # (Sub-)Domains sind Buchstaben, Ziffern und Bindestriche.
        \.          # getrennt durch einen Punkt,
    )+              # und wir können mehrere von ihnen haben.
    (
        [a-z]{2}    # TLDs können ein zweistelliger alphabetischer Country-Code
        |com|net    # oder einer von
        |edu|org    # vielen
        |gov|mil    # möglichen
        |int|biz    # Kombinationen aus
        |pro        # drei Buchstaben sein
        |info|arpa  # oder sogar
        |aero|coop  # einige
        |name       # mit vier Buchstaben
        |museum     # und einer, der sechs Buchstaben lang ist!
    )
    $               # Am Ende verankern,
/ix                 # und die Groß-/Kleinschreibung wird ignoriert.
```

Beide Muster sind absichtlich liberal formuliert. Sie reichen aus, wenn Sie nur sicherstellen möchten, dass niemand die Top-Level-Domain versehentlich weglässt oder etwas Abwegiges einträgt wie »keine Angabe«. Beispielsweise gibt es keine Domain `"-.com"`, aber `"foo@-.com"` wird klaglos akzeptiert. (Es wäre nicht schwierig, das Muster so zu verändern, dass auch dies ausgeschlossen würde, aber das möchten wir Ihnen als Übung überlassen.) Andererseits ist eine Adresse wie `"Tim O'Reilly@oreilly.com"` zulässig, die von unserem Muster nicht akzeptiert wird. Allerdings sind Leerzeichen in E-Mail-Adressen selten; und da ein Leerzeichen fast immer ein Fehler ist, kennzeichnen wir eine solche Adresse als falsch.

Die kanonische Definition einer gültigen Adresse ist in RFC 822 dokumentiert; allerdings ist es ziemlich aufwendig, Code zu schreiben, der alle Fälle berücksichtigt. Hier sehen Sie

ein Beispiel dafür, was Sie berücksichtigen müssen: Es ist erlaubt, in die Adressen Kommentare einzufügen! Kommentare befinden sich in Klammern, und deswegen ist Folgendes gültig:

```
Tim (is the man @ computer books) @ oreilly.com
```

Dies ist gleichbedeutend mit "tim@oreilly.com". (Und auch bei dieser Adresse versagt das Muster.)

Als alternative Möglichkeit bietet die IMAP-Erweiterung einen RFC-822-konformen Adress-Parser. Diese Funktion navigiert korrekt durch Leerzeichen, Kommentare und andere Besonderheiten. Aber sie lässt offensichtliche Fehler zu, da sie davon ausgeht, dass Adressen ohne Host-Namen lokal sind:

```
$email = 'stephen(sein Konto)@ example(sein Host)';
$parsed = imap_rfc822_parse_adrlist($email,'');
print_r($parsed);
Array
(
    [0] => stdClass Object
        (
            [mailbox] => stephen
            [host] => example
            [personal] => sein Host
        )

)
```

Wenn Sie Mailbox und Host wieder zusammensetzen, erhalten Sie "stephen@example", was Sie sich wahrscheinlich nicht wünschen. Der leere String, den Sie als zweites Argument übergeben müssen, nimmt Ihnen die Möglichkeit, auf gültige Host-Namen zu prüfen.

Manche Menschen führen gern hinter den Kulissen DNS-Lookups durch, um die Gültigkeit von Adressen zu prüfen. Diese Technik ist nicht besonders sinnvoll, da sie nicht immer funktioniert, und möglicherweise weisen Sie vollkommen korrekte Menschen von Ihre Site zurück, die dafür nichts können. (Es ist auch nicht zu erwarten, dass ein Mail-Administrator seinen Umgang mit der E-Mail ändert, nur um die E-Mail-Validierung auf einer Webseite zu umgehen.)

Beim Validieren von E-Mail-Adressen ist auch zu bedenken, dass es für einen Benutzer nicht sehr schwierig ist, eine vollkommen korrekte und funktionierende Adresse einzugeben, die nicht seine eigene ist. Beispielsweise hatte einer der Autoren die schlechte Angewohnheit, sich bei der Microsoft-Website unter "billg@microsoft.com" anzumelden, denn: »Hey! Vielleicht kennt Bill diese neue Version des Internet Explorer noch nicht?«

Wenn das Hauptproblem darin besteht, Schreibfehler zu vermeiden, können Sie auch die Anwender bitten, ihre Adressen zweimal einzugeben, und die beiden Versionen miteinander vergleichen. Wenn die Eingaben übereinstimmen, ist die Adresse vermutlich korrekt. Außerdem sollten Sie auch häufig verwendete unsinnige Adressen wie

"president@whitehouse.gov" und den oben erwähnten "billg@microsoft.com" herausfiltern. (Dies hat allerdings den Nachteil, dass sich weder der Präsident der Vereinigten Staaten von Amerika noch Bill Gates auf Ihrer Site anmelden können.)

Wenn Sie jedoch sicher sein möchten, dass die Anwender wirklich Zugriff auf die von ihnen angegebene E-Mail-Adresse haben, haben Sie auch die Möglichkeit, eine Nachricht an diese Adresse zu senden. In dieser Nachricht bitten Sie, entweder auf die Nachricht zu antworten oder eine Seite auf Ihrer Site aufzurufen und dort einen speziellen Code zur Bestätigung der Anmeldung einzugeben, den Sie in den Text der Nachricht eingefügt haben. Wenn Sie den Weg mit dem speziellen Code wählen, sollten Sie besser keine zufällig gebildete Buchstabenkette wie HSD5nbADl8 generieren. Da sie keinen Sinn ergibt, kann man sie nur schwer korrekt abschreiben. Verwenden Sie stattdessen eine Liste von Wörtern und erzeugen Sie Code-Wörter wie fernsehen4schichtregal. Es kann zwar passieren, dass jemand aus einer solchen Kombination eine heimliche Mitteilung herausliest, aber Sie können auf diese Weise Ihre Fehlerraten und Ihre Support-Kosten vermindern.

Siehe auch

Rezept 10.22 mit einem Programm zur Deaktivierung von Website-Benutzerkonten; die Dokumentation zu imap_rfc822_parse_adrlist() unter *http://www.php.net/imap-rfc822-parse-adrlist*.

16.6 Alle zu einem Muster passenden Zeilen in einer Datei finden

Problem

Sie möchten alle Zeilen in einer Datei finden, die zu einem bestimmten Muster passen.

Lösung

Lesen Sie die Datei in ein Array ein und verwenden Sie preg_grep().

Diskussion

Es gibt dafür zwei Möglichkeiten, dies ist der schnellere Weg:

```
$pattern = "/\bo'reilly\b/i"; // nur Bücher von O'Reilly
$ora_books = preg_grep($pattern, file('/pfad/zu/ihrer/datei.txt'));
```

Mit file() können Sie automatisch alle Zeilen der Datei in Array-Elemente laden und mit preg_grep() die unerwünschten Zeilen herausfiltern.

Hier ist die effizientere Methode:

```
$fh = fopen('/pfad/zu/ihrer/datei.txt', 'r') or die($php_errormsg);
while (!feof($fh)) {
    $line = fgets($fh, 4096);
    if (preg_match($pattern, $line)) { $ora_books[] = $line; }
}
fclose($fh);
```

Da die erste Methode alles auf einmal einliest, ist sie ungefähr dreimal schneller als die zweite, die die Datei Zeile für Zeile untersucht, aber weniger Speicher benötigt. Ein Nachteil besteht allerdings darin, dass die zweite Methode keine Strings findet, die sich über mehrere Zeilen erstrecken, da der reguläre Ausdruck auf jede Zeile einzeln angewendet wird.

Siehe auch

Rezept 21.4 zum Einlesen von Dateien in Strings; die Dokumentation zu `preg_grep()` unter *http://www.php.net/preg-grep*.

16.7 Text innerhalb von HTML-Tags finden

Problem

Sie möchten Text auslesen, der sich innerhalb von HTML-Tags befindet. Zum Beispiel möchten Sie alle Überschriften in einem HTML-Dokument finden.

Lösung

Lesen Sie die HTML-Datei in einen String ein und führen Sie nicht-gieriges Matching mit Ihrem Muster durch:

```
$html = join('',file($file));
preg_match('#<h([1-6])>(.+?)</h\1>#is', $html, $matches);
```

In diesem Beispiel enthält `$matches[2]` ein Array mit den ausgelesenen Überschriften.

Diskussion

Es ist schwierig, HTML-Code mit einfachen regulären Ausdrücken korrekt zu analysieren. Einer der Vorteile von XHTML ist, dass es deutlich einfacher zu validieren und zu zerlegen ist.

Zu Beginn des Suchmusters sehen wir, dass das Ergebnis der Zahlengruppe `[1-6]` mit runden Klammern eingefangen wird. Im Bereich des schließenden Tags referenzieren wir

dann auf das Ergebnis dieser (ersten) Klammerung: `\1`. So stellen wir sicher, dass immer das passende schließende Tag zum zuvor gefundenen öffnenden Tag abgefragt wird.

Das Muster in der Lösung ist dadurch so intelligent, dass es zueinander passende Tags für Überschriften findet; so wird `<h1>Dr. Seltsam</h1>` erkannt, weil es in `<h1>`-Tags verpackt ist, aber nicht `<h2>Wie ich lernte, die Bombe zu lieben</h3>`, denn das öffnende Tag `<h2>` passt hier nicht zum schließenden.

Diese Technik funktioniert auch, wenn man alle Texte finden möchte, die zwischen Tags für Fett- oder Kursivschrift liegen:

```
$html = join('',file($file));
preg_match('#<([bi])>(.+?)</\1>#is', $html, $matches);
```

Bei verschachtelten Tags versagt sie allerdings. Wenn Sie diesen regulären Ausdruck auf folgende Zeile anwenden

```
<b>Dr. Seltsam oder: <i>Wie ich lernte, die Bombe zu lieben</i></b>
```

liefert er den Text innerhalb der `<i>`-Tags nicht als gesondertes Element.

Dies war oben kein Problem, denn Überschriften sind Block-Elemente, die nicht geschachtelt werden dürfen. Das Schachteln von Tags für fette und kursive Schrift ist aber zulässig, da es sich hier um Inline-Tags handelt.

Sie können den ausgelesenen Text verarbeiten, indem Sie das Array mit den Übereinstimmungen in einer Schleife durchlaufen lassen. Beispielsweise durchsucht der folgende Code ein Dokument nach Überschriften und gibt sie ordentlich formatiert und entsprechend der jeweiligen Ebene eingerückt aus:

```
$html = join('',file($file));
preg_match('#<h([1-6])>(.+?)</h\1>#is', $html, $matches);

for ($i = 0, $j = count($matches[0]); $i < $j; $i++) {
  print str_repeat(' ', 2 * ($matches[1][$i] - 1)) . $matches[2][$i] . "\n";
}
```

Mit der folgenden Darstellung dieses Kapitels in HTML:

```
$html =<<<_END_
<h1>PHP Kochbuch</h1>

Andere Kapitel
<h2>Reguläre Ausdrücke</h2>

Andere Rezepte
<h3>Text innerhalb von HTML-Tags finden</h3>

<h4>Problem</h4>
<h4>Lösung</h4>
<h4>Diskussion</h4>
<h4>Siehe auch</h4>

_END_;
```

```
preg_match_all('#<h([1-6])>(.+?)</h\1>#is', $html, $matches);

for ($i = 0, $j = count($matches[0]); $i < $j; $i++) {
  print str_repeat(' ', 2 * ($matches[1][$i] - 1)) . $matches[2][$i] . "\n";
}
```

erhalten Sie:

```
PHP Kochbuch
  Reguläre Ausdrücke
    Text innerhalb von HTML-Tags finden
      Problem
      Lösung
      Diskussion
      Siehe auch
```

Wenn Sie neben dem Text der Überschrift auch deren Ebene auslesen, können Sie direkt auf diese Ebene zugreifen und sie bei der Berechnung der Einrückung als Integer-Zahl behandeln. Damit nicht alle Zeilen um zwei Leerzeichen eingerückt werden, ziehen Sie 1 von der Ebene ab.

Reguläre Ausdrücke können bei kleinen HTML-Parsing-Problemen sehr hilfreich sein. Will man hingegen eine HTML-Quelle vollständig und vor allem robust parsen, bietet es sich an, die PECL Erweiterung `tidy` einzusetzen. Mit ihr können Sie HTML-Quellen reparieren, manipulieren und natürlich auch elementweise durchstöbern.

Siehe auch

Rezept 13.9 mit Informationen darüber, wie man eine Webseite mit Markup versieht, und Rezept 13.10 zum Extrahieren von Links aus einer HTML-Datei; die Dokumentationen zu `preg_match()` unter *http://www.php.net/preg-match*, zu `str_repeat()` unter *http://www.php.net/str-repeat* und zu `tidy` unter *http://www.php.net/tidy*.

16.8 In regulären Ausdrücken Sonderzeichen verwenden

Problem

Sie möchten, dass in einem regulären Ausdruck Zeichen wie * und + als normale Zeichen und nicht als Metazeichen behandelt werden. Dies ist sinnvoll, wenn Sie den Benutzern die Eingabe von Suchtexten ermöglichen wollen, um diese innerhalb eines regulären Ausdrucks zu verwenden.

Lösung

Verwenden Sie `preg_quote()`, um Metazeichen für Perl-kompatible reguläre Ausdrücke zu kennzeichnen:

```
$pattern = preg_quote('The Education of H*Y*M*A*N K*A*P*L*A*N').':(\d+)';
if (preg_match("/$pattern/",$book_rank,$matches)) {
```

```
    print "Das Buch von Leo Rosten hat den Rang: ".$matches[1];
}
```

Verwenden Sie quotemeta() zum Kennzeichnen von POSIX-Metazeichen:

```
$pattern = quotemeta('M*A*S*H').':[0-9]+';
if (ereg($pattern,$tv_show_rank,$matches)) {
    print 'Radar, Hot Lips und die Gang haben den Rang: '.$matches[1];
}
```

Diskussion

Folgende Zeichen werden von preg_quote() berücksichtigt:

```
. \ + * ? ^ $ [ ] ( ) { } < > = ! | :
```

Und diese Zeichen werden von quotemeta() behandelt:

```
. \ + * ? ^ $ [ ] ( )
```

Beide Funktionen versehen die angegebenen Metazeichen mit einem Backslash.

Die Funktion quotemeta() findet nicht alle POSIX-Metazeichen heraus. Beispielsweise sind auch die Zeichen {, } und | gültige Metazeichen, werden aber nicht konvertiert. Dies ist ein weiterer guter Grund, preg_match() anstelle von ereg() zu verwenden.

Sie können preg_quote() noch ein weiteres zu berücksichtigendes Zeichen als zweites Argument übergeben. Sinnvollerweise übergeben Sie das Begrenzungszeichen Ihres Musters (normalerweise /) als drittes Argument, damit es ebenfalls mit einem Backslash versehen wird. Dies ist wichtig, wenn Sie Benutzereingaben in ein Muster für einen regulären Ausdruck einfügen. Der folgende Code erwartet $_REQUEST['search_term'] aus einem Webformular und sucht in einem String $s nach Wörtern, die mit $_REQUEST['search_term'] beginnen:

```
$search_term = preg_quote($_REQUEST['search_term'],'/');
if (preg_match("/\b$search_term/i",$s)) {
   print 'Gefunden!';
}
```

Mit preg_quote() stellen Sie sicher, dass der reguläre Ausdruck auch dann korrekt interpretiert wird, wenn beispielsweise ein Magnum-Fan den Suchbegriff t.c. eingibt. Ohne preg_quote() stimmt dies mit tic, tucker ebenso überein wie mit allen Wörtern, deren erster Buchstabe ein t und deren dritter Buchstabe ein c ist. Wenn Sie auch das Muster-Begrenzungszeichen an preg_quote() übergeben, stellen Sie sicher, dass Benutzereingaben mit Schrägstrichen, wie etwa CP/M, korrekt behandelt werden.

Siehe auch

Die Dokumentation zu preg_quote() unter *http://www.php.net/preg-quote* und quotemeta() unter *http://www.php.net/quotemeta*.

16.9 Datensätze lesen, bei denen ein Muster als Separator dient

Problem

Sie möchten Datensätze aus einer Datei lesen, wobei die Datensätze durch ein Muster getrennt sind, auf das ein regulärer Ausdruck passt.

Lösung

Lesen Sie die ganze Datei in einen String ein und trennen Sie diesen mithilfe des regulären Ausdrucks auf:

```
$filename = '/path/to/your/file.txt';
$fh = fopen($filename, 'r') or die($php_errormsg);
$contents = fread($fh, filesize($filename));
fclose($fh);

$records = preg_split('/[0-9]+\) /', $contents);
```

Diskussion

Dieser Code zerteilt eine nummerierte Liste und platziert die einzelnen Listenelemente in Array-Elemente. Wenn Sie also eine Liste wie diese haben:

```
1) Gödel
2) Escher
3) Bach
```

bekommen Sie ein Array mit vier Elementen, wobei das erste Element leer ist. Dies liegt daran, dass `preg_split()` davon ausgeht, dass sich die Trennzeichen zwischen den Elementen befinden, aber in diesem Fall stehen die Zahlen vor den Elementen:

```
Array
(
    [0] =>
    [1] => Gödel
    [2] => Escher
    [3] => Bach
)
```

Unter einem bestimmten Aspekt ist dies ein Feature und kein Bug: das *n*-te Array-Element enthält das *n*-te Listenelement. Wenn Sie aber das Array komprimieren möchten, können Sie das erste Element entfernen:

```
$records = preg_split('/[0-9]+\) /', $contents);
array_shift($records);
```

Vielleicht möchten Sie auch als weitere Änderung die Zeilenvorschübe aus den Elementen entfernen und durch einen leeren String ersetzen:

```
$records = preg_split('/[0-9]+\) /', str_replace("\n",'',$contents));
array_shift($records);
```

Sie können mit PHP keine anderen Zeichen zum Trennen von Eingabe-Datensätzen verwenden als den Zeilenvorschub. Deshalb ist diese Technik auch nützlich, wenn Sie Datensätze teilen möchten, die durch Strings getrennt sind. Sollten Sie aber feststellen, dass Sie als Trenner einen String und keinen regulären Ausdruck verwenden, ersetzen Sie `preg_split()` durch `explode()`, da dies effizienter ist.

Siehe auch

Rezept 21.4 zum Lesen aus Dateien; Rezept 1.11 zum Zerlegen von CSV-Dateien.

16.10 Verhindern, dass Klammern Text fangen

Problem

In einem Muster haben Sie Klammern zum Gruppieren verwendet, wollen aber nicht, dass der vom Ausdruck in der Klammer gefundene Text im Array mit den eingefangenen Treffern erscheint.

Lösung

Geben Sie wie in Beispiel 16-1 hinter der öffnenden Klammer `?:` an.

Beispiel 16-1: Das Einfangen von Text verhindern

```
<?php
$html = '<link rel="icon" href="http://www.example.com/icon.gif"/>
<link rel="prev" href="http://www.example.com/prev.xml"/>
<link rel="next" href="http://www.example.com/next.xml"/>';
preg_match_all('/rel="(prev|next)" href="([^"]*?)"/', $html, $bothMatches);
preg_match_all('/rel="(?:prev|next)" href="([^"]*?)"/', $html, $linkMatches);
print '$bothMatches ist: '; var_dump($bothMatches);
print '$linkMatches ist: '; var_dump($linkMatches);
?>
```

Die Variable `$bothMatches` enthält hier die Werte der Attribute `rel` und `href`. `$linkMatches` jedoch enthält nur die Werte des `href`-Attributs. Der Code gibt Folgendes aus:

```
$bothMatches ist: array(3) {
  [0]=>
  array(2) {
    [0]=>
    string(49) "rel="prev" href="http://www.example.com/prev.xml""
    [1]=>
    string(49) "rel="next" href="http://www.example.com/next.xml""
  }
```

```
    [1]=>
    array(2) {
      [0]=>
      string(4) "prev"
      [1]=>
      string(4) "next"
    }
    [2]=>
    array(2) {
      [0]=>
      string(31) "http://www.example.com/prev.xml"
      [1]=>
      string(31) "http://www.example.com/next.xml"
    }
  }
  $linkMatches ist: array(2) {
    [0]=>
    array(2) {
      [0]=>
      string(49) "rel="prev" href="http://www.example.com/prev.xml""
      [1]=>
      string(49) "rel="next" href="http://www.example.com/next.xml""
    }
    [1]=>
    array(2) {
      [0]=>
      string(31) "http://www.example.com/prev.xml"
      [1]=>
      string(31) "http://www.example.com/next.xml"
    }
  }
```

Diskussion

Das Verhindern des Einfangens ist besonders nützlich, wenn ein Teilmuster optional ist. Da es im Array mit den eingefangenen Treffern eventuell nicht auftaucht, kann es die Anzahl der Treffer beeinflussen. Das kann es erschweren, einen bestimmten Treffer über einen festgelegten Index zu referenzieren. Indem Sie dafür sorgen, dass das optionale Teilmuster nicht einfangend ist, verhindern Sie dieses Problem. Beispiel 16-2 illustriert diesen Unterschied.

Beispiel 16-2: Ein nicht einfangendes optionales Teilmuster

```
<?php
$html = '<link rel="icon" href="http://www.example.com/icon.gif"/>
<link rel="prev" title="Previous" href="http://www.example.com/prev.xml"/>
<link rel="next" href="http://www.example.com/next.xml"/>';
preg_match_all('/rel="(?:prev|next)"(?: title="[^"]+?")? href=
"([^"]*?)"/', $html, $linkMatches);
print '$bothMatches ist: '; var_dump($linkMatches);
?>
```

Siehe auch

Die PCRE Pattern Syntax-Dokumentation unter *http://php.net/reference.pcre.pattern.syntax*.

16.11 In einem regulären Ausdruck eine PHP-Funktion nutzen

Problem

Sie möchten gefundenen Text mit einer PHP-Funktion verarbeiten. Sie wollen zum Beispiel alle HTML-Entities in eingefangenen Teilmustern dekodieren.

Lösung

Nutzen Sie `preg_replace_callback( )`. Geben Sie anstelle eines Ersetzungsmusters eine Callback-Funktion an. Dieser Callback-Funktion wird ein Array mit allen gefundenen Teilmustern übergeben, und sie sollte einen geeigneten Ersetzungsstring zurückliefern. Beispiel 16-3 dekodiert die Entities zwischen `<code></code>`-Tags.

Beispiel 16-3: Ersetzungsstrings mit einer Callback-Funktion generieren

```
<?php
$html = 'Das &lt;b&gt;-Tag macht Text fett: <code>&lt;b&gt;fett&lt;/b&gt;</code>';
print preg_replace_callback('@<code>(.*?)</code>@','decode', $html);
// $matches[0] ist der gesamte gefundene String.
// $matches[1] ist das erste gefundene Teilmuster.
function decode($matches) {
    return html_entity_decode($matches[1]);
}
?>
```

Beispiel 16-3 liefert folgende Ausgabe:

```
Das &lt;b&gt;-Tag macht Text fett: <b>fett</b>
```

Diskussion

Das zweite Argument für `preg_replace_callback( )` gibt die Funktion an, die zur Berechnung der Ersetzungsstrings aufgerufen werden soll. Wie üblich wird dabei der PHP-Pseudotyp »Callback« genutzt. Das Argument kann ein String oder ein Array sein. Nutzen Sie einen String, um einen Funktionsnamen anzugeben. Wenn Sie die Instanzmethode eines Objekts als Callback einsetzen wollen, übergeben Sie ein Array, dessen erstes Element das Objekt und dessen zweites Element ein String mit dem Namen der zu verwendenden

Methode ist. Wollen Sie eine statische Klassenmethode als Callback verwenden, nutzen Sie ein Array mit zwei Strings: dem Klassen- und dem Methodennamen.

Der Callback-Funktion wird ein Argument übergeben: ein Array mit den Treffern. Element 0 dieses Arrays ist immer der Text, der vom vollständigen Muster gefunden wurde. Enthält das an `preg_replace_callback( )` übergebene Muster Teilmuster in Klammern, enthalten die nachfolgenden Elemente des Arrays die Texte, die von diesen Teilmustern gefunden wurden. Die Schlüssel für das Treffer-Array sind numerisch – auch dann, wenn das Muster benannte Teilmuster enthält.

Die PHP-Manpage zu `preg_replace_callback( )` schlägt vor, `create_function( )` einzusetzen, um eine anonyme Funktion zu erstellen, die als Callback verwendet wird. Obwohl das praktisch sein kann, beansprucht es mehr Speicher, wenn der Aufruf von `create_function( )` in den Aufruf von `preg_replace_callback( )` eingebettet ist und dieser in einer Schleife steht. Wollen Sie mit `preg_replace_callback( )` eine anonyme Funktion verwenden, sollten Sie `create_function( )` einmal aufrufen und die anonyme Callback-Funktion in einer Variablen speichern. Übergeben Sie diese Variable `preg_replace_callback( )` dann als Callback-Funktion. Beispiel 16-4 nutzt eine anonyme Funktion, um die Transformation in Beispiel 16-3 auf alle Zeilen einer Datei anzuwenden.

Beispiel 16-4: Ersetzungsstrings mit einer anonymen Callback-Funktion erzeugen

```
<?php
$callbackFunction = create_function('$matches',
                        'return html_entity_decode($matches[1]);');
$fp = fopen('html-to-decode.html','r');
while (! feof($fp)) {
    $line = fgets($fp);
    print preg_replace_callback('@<code>(.*?)</code>@',$callbackFunction, $line);
}
fclose($fp);
?>
```

Als Alternative zu `preg_replace_callback( )` kann der Mustermodifikator e eingesetzt werden. Dieser bewirkt, dass der Ersetzungsstring als PHP-Code ausgewertet wird. Wir empfehlen Ihnen aufgrund der unten erläuterten Problemen mit Rückwärtsreferenzen allerdings, stattdessen `preg_replace_callback( )` zu verwenden.

Beispiel 16-5 nutzt den Modifikator e, um die gleiche Entity-Auflösung durchzuführen wie in Beispiel 16-3.

Beispiel 16-5: Entity-Kodierung in gefundenem Text

```
<?php
$html = 'Das &lt;b&gt;-Tag macht Text fett: <code>&lt;b&gt;fett&lt;/b&gt;</code>';
print preg_replace('@<code>(.*?)</code>@e',"html_entity_decode('$1')", $html);
?>
```

Der Einsatz des Modifikators e kann zu Verwicklungen führen, wenn der Ersetzungsstring Rückwärtsreferenzen enthält. Dabei muss man die unterschiedlichen Maskierungsebenen im Blick haben (und in einigen Fällen umgehen).

Die erste Maskierungsstufe bildet PHPs gewöhnliches Verhalten, das immer im Einsatz ist, wenn Sie einen String aufbauen. (Beachten Sie allerdings, dass das Dollarzeichen in Beispiel 16-5, obwohl der gesamte Ersetzungsstring in doppelten Anführungszeichen steht, nicht maskiert werden muss, da `$1` kein gültiger Variablenname ist.)

Die zweite Maskierungsstufe betrifft die Begrenzung des über die Rückwärtsreferenz in den Ersetzungsstring eingefügten Texts. In Beispiel 16-5 wird `html_entity_decode('$1')` zu `html_entity_decode('<code><b>bold</b></code>')`. `html_entity_decode( )` wird also mit einem Argument aufgerufen, einem String in einfachen Anführungszeichen.

Sowohl die einfachen als auch die doppelten Anführungszeichen im gefundenen Treffer werden mit Backslashs maskiert. Die Ersetzung mit der Rückwärtsreferenz sieht etwas anders aus, wenn der gefundene Text selbst einfache oder doppelte Anführungszeichen enthält. Schauen Sie sie sich Beispiel 16-6 an.

Beispiel 16-6: Maskierung von Anführungszeichen in Rückwärtsreferenz-Ersetzungen

```
<?php
$html = "<code>&lt;b&gt; Das ist 'fett'.&lt;/b&gt;</code>";
print preg_replace('@<code>(.*?)</code>@e',"html_entity_decode('$1')", $html);
print "\n";
$html = '<code>&lt;i&gt; Das ist "kursiv". &lt;/i&gt;</code>';
print preg_replace('@<code>(.*?)</code>@e',"html_entity_decode('$1')", $html);
print "\n";
?>
```

Beispiel 16-6 liefert folgende Ausgabe:

```
<b> Das ist 'fett'. </b>
<i> Das ist \"kursiv\". </i>
```

Irgendwie haben sich in die zweite Zeile Backslashs eingeschlichen. Das resultiert aus der Arbeitsweise des Modifikators e. Wie oben gesagt, werden sowohl die einfachen als auch die doppelten Anführungszeichen im gefundenen Treffer mit Backslashs maskiert. Das bedeutet, dass beim ersten Aufruf von `preg_replace( )` in Beispiel 16-6 Folgendes ausgeführt wird, um die Ersetzung zu berechnen:

```
html_entity_decode('<code>&lt;b&gt; Das ist \'fett\'.&lt;/b&gt;</code>)
```

`html_entity_decode( )` wird ein String in einfachen Anführungszeichen übergeben, der mit einem Backslash maskierte einfache Anführungszeichen enthält. Kein Problem also – `'Das ist \'fett\'.'` ist einfach `Das ist 'fett'.`. Aber das zweite `preg_replace( )` ist problematisch. Zur Berechnung der Ersetzung wird dabei dieses ausgeführt: `html_entity_decode('<code><i> Das ist \"kursiv\". </i></code>')`. In einem String in einfachen Anführungszeichen gibt ein Backslash vor einem doppelten Anführungszeichen

aber kein literales Anführungszeichen, sondern die zwei Zeichen enthaltende Zeichenfolge \" an.

Dieses Problem umgehen Sie, indem Sie mit str_replace( ) in dem zur Ermittlung des Ersetzungswerts eingesetzten Code die Zeichenfolge \" durch " ersetzen. (Nutzen Sie nicht stripslashes( ), da damit auch die Backslashs vor anderen Zeichen entfernt werden, was wir hier nicht haben wollen.) Beispiel 16-7 umgibt html_entity_decode( ) mit einer Funktion, die genau das macht.

Beispiel 16-7: Das Maskieren von Anführungszeichen in Rückwärtsreferenz-Ersetzungen reparieren

```
<?php
$html = "<code>&lt;b&gt; Das ist 'fett'. &lt;/b&gt;</code>";
print preg_replace('@<code>(.*?)</code>@e',"preg_html_entity_decode('$1')", $html);
print "\n";
$html = '<code>&lt;i&gt; Das ist "kursiv". &lt;/i&gt;</code>';
print preg_replace('@<code>(.*?)</code>@e',"preg_html_entity_decode('$1')", $html);
print "\n";
function preg_html_entity_decode($s) {
    $s = str_replace('\\"','"', $s);
    return html_entity_decode($s);
}
?>
```

Der Einsatz der Funktion preg_html_entity_decode( ) in Beispiel 16-7 sichert, dass die richtigen Ergebnisse ausgeben werden:

```
    <b> Das ist 'fett'. </b>
    <i> Das ist "kursiv". </i>
```

Ein letzter Hinweis zum Maskieren und dem Mustermodifikator e: Achten Sie darauf, dass Sie in dem Ausdruck, der den Ersetzungswert berechnet, einfache (keine doppelten) Anführungszeichen verwenden, um die Strings einzufassen, die Rückwärtsreferenzwerte enthalten. Das bedeutet: Nutzen Sie preg_html_entity_decode('$1'), nicht preg_html_entity_decode("$1"). Doppelte Anführungszeichen können zu Problemen führen, wenn der Rückwärtsreferenzwert etwas enthält, das wie ein gültiger Variablenname aussieht. Beispiel 16-8 illustriert dieses Problem.

Beispiel 16-8: Variablennamen und Strings in einfachen Anführungszeichen

```
<?php
$text = '<code>if ($temperature &lt; 12) { fever(); }</code>';
print "Gut: \n";
print preg_replace('@<code>(.*?)</code>@e',"preg_html_entity_decode('$1')", $text);
print "\n Schlecht: \n";
print preg_replace('@<code>(.*?)</code>@e','preg_html_entity_decode("$1")'  , $text);
function preg_html_entity_decode($s) {
    $s = str_replace('\\"','"', $s);
    return html_entity_decode($s);
}
?>
```

Beispiel 16-8 liefert folgende Ausgabe:

```
Gut:
if ($temperature < 12) { fever(); }
 Schlecht:
Notice: Undefined variable: temperature in example.php(6) : regexp code on line 1
if ( < 12) { fever(); }
```

Da geeignete Anführungszeichen verwendet werden, funktioniert das erste `preg_replace()` wie erwartet: Die einzige Veränderung an `$text` ist, dass `<` durch < ersetzt wird. Das zweite `preg_replace()`, mit `$1` in doppelten Anführungszeichen, funktioniert nicht. Der PHP-Interpreter denkt, dass der an `preg_html_entity_decode()` zu übergebende String "`if ($temperature < 12) { fever(); }`" ist. Da das ein String in doppelten Anführungszeichen ist, versucht der PHP-Interpreter, `$temperature` durch den Wert der entsprechenden Variablen zu ersetzen, die es natürlich nicht gibt.

Die Moral der »Der e-Modifikator und `preg_replace()`«-Geschichte hat also zwei Aspekte: Korrigieren Sie mit Backslash maskierte doppelte Anführungszeichen und nutzen Sie einfache Anführungszeichen zur Begrenzung der Strings in Ihrem Codeausdruck, um eine unerwünschte Variableninterpolation zu vermeiden. Dieses komplizierte Maskierungs- und Interpolationsverhalten macht `preg_replace_callback()` zur netteren Option.

Siehe auch

Die Dokumentationen zu `preg_replace_callback()` unter *http://www.php.net/preg_replace_callback*, zu `preg_replace()` unter *http://www.php.net/preg_replace*, zu `create_function()` unter *http://www.php.net/create_function* und zum Callback-Pseudotyp unter *http://www.php.net/language.pseudo-types#language.types.callback*.

KAPITEL 17

Verschlüsselung und Sicherheit

17.0 Einführung

In einer perfekten Welt wäre Verschlüsselung unnötig. Neugierige Mitmenschen würden ihre Blicke auf ihre eigenen Daten beschränken, und eine durchs Internet flitzende Kreditkartennummer würde keine besondere Aufmerksamkeit auf sich ziehen. Da unsere Welt aber in vielerlei Hinsicht nicht so ganz perfekt ist, brauchen wir Verschlüsselung.

PHP stellt Ihnen Werkzeuge zur Verfügung, mit denen Sie Ihre Daten verschlüsseln und absichern können. Einige Werkzeuge, wie beispielsweise die Funktionen `crypt()` und `md5()`, gehören zu den eingebauten PHP-Grundfunktionen, andere wiederum sind Erweiterungen, die explizit mit einbezogen werden müssen, wenn PHP kompiliert wird (z.B. *mcrypt*, *mhash* und cURL).

Die `crypt()`-Funktion führt Einwegverschlüsselungen durch. Dieses Verfahren erzeugt aus einem Inhalt beliebiger Länge einen Hash-Wert mit fester Länge. Dadurch wird bei langen Ursprungstexten der Informationsgehalt reduziert, was eine Entschlüsselung unmöglich macht. Dies ist bei Hash-Wert-Algorithmen aber auch nicht erwünscht. Hash-Werte werden als Fingerabdruck eines Inhalts angesehen. Theoretisch ist der Fingerabdruck, aufgrund der oben genannten Eigenschaft, nicht zu 100% eindeutig. Die Hash-Wert-Algorithmen minimieren allerdings die Wahrscheinlichkeit eines doppelten Treffers sehr.

Sie übergeben der Funktion `crypt()` den zu verschlüsselnden Klartext (sowie ein sogenanntes »Salz« oder »salt« – eine Zeichenkette, die die Verschlüsselung zusätzlich variiert), und sie gibt Ihnen den Hash-Wert mit dem »Salz« am Anfang zurück. Wenn Sie kein »Salz« übergeben, erzeugt PHP hierfür einen zufälligen Wert:

```
print crypt('shrimp','34');
34b/4qaoXmcoY
```

Wenn die Konstante `CRYPT_MD5` auf 1 gesetzt ist, wird `crypt()` alternativ den Hash-Wert-Algorithmus MD5 einsetzen. Sie können hierfür auch das »Salz« mit `$1$` beginnen lassen:

```
print crypt('shrimp','$1$seasalt!');
$1$seasalt!$C8bRD475BC3T4EvjjmR9I.
```

Neben den Algorithmen DES (`CRYPT_STD_DES` mit 2-Zeichen-Salt oder `CRYPT_EXT_DES` mit 9-Zeichen-Salt) und MD5 (`CRYPT_MD5` mit 12-Zeichen-Salt, beginnend mit 1) unterstützt PHP auch noch BLOWFISH (`CRYPT_BLOWFISH` mit 16-Zeichen-Salt, beginnend mit 2 oder $2a$). Seit PHP 5.3.0 existieren für all diese Algorithmen eigene Implementierungen. Diese werden eingesetzt, wenn das System die erwünschte Verschlüsselung selbst nicht unterstützt. Rezept 17.4 bespricht `crypt()`. Diese Funktion wird meistens dazu verwendet, Passwörter zu verschlüsseln.

mcrypt ist eine Verschlüsselungsbibliothek mit etwas vollständigerem Funktionsumfang, die Ihnen verschiedene Algorithmen und Verschlüsselungsmodi anbietet. Da sie eine Reihe von Verschlüsselungstechniken unterstützt, ist *mcrypt* gerade dann besonders hilfreich, wenn Sie Daten mit anderen Systemen oder Programmen austauschen müssen, die nicht in PHP geschrieben sind. *mcrypt* wird in Rezept 17.7 zum Ver- und Entschlüsseln von Daten eingesetzt.

PHP gibt Ihnen zwar das Werkzeug, Ihre Daten robust verschlüsseln zu können, jedoch ist das nur ein Teil des umfangreichen und oft komplexen Themas Sicherheit. Da sich Ihre verschlüsselten Daten mit einem Schlüssel wieder entschlüsseln lassen, ist es sehr wichtig, diesen Schlüssel zu schützen. Wenn Unberechtigte auf Ihre Schlüssel zugreifen können (weil sie beispielsweise in einer Datei abgespeichert sind, auf die über Ihren Webserver zugegriffen werden kann oder die anderen Benutzern in einer gemeinschaftlich genutzten Hosting-Umgebung offen steht oder Ihr Programm durch geschickte Parametrisierung ausgetrickst wurde), sind Ihre Daten gefährdet – egal, wie wasserdicht Ihr gewählter Verschlüsselungsalgorithmus dabei ist. Angriffe werden häufig durch an die Anwendung übermittelte Daten bzw. Parameterwerte durchgeführt. Ab Rezept 17.12 werden hierzu unterschiedliche Themen behandelt, die solch eine Art von Angriff verhindern können.

Sie werden sich überlegen müssen, wie sicher Ihre Daten sein sollen. Verschlüsselung ist sicherer als einfache Codierung, ist aber auch komplexer. Einfacheres Codieren schützt Ihre Daten vor einfachen neugierigen Blicken, bietet aber wenig Sicherheit. Auf der anderen Seite ist aber keine Verschlüsselung oder eine andere Sicherheitstechnik absolut. Diese Auswahl einer passenden Sicherungsmethode bedeutet daher, einen Platz im Spektrum zwischen Bequemlichkeit und Schutz zu finden. Die bequemeren (oder vom Rechenaufwand her günstigen) Sicherheitstechniken bieten im Regelfall weniger Schutz. Manchmal wollen Sie die Daten auch gar nicht vor neugierigen Blicken bewahren, sondern lediglich den Anschein vermeiden, dass hier etwas nicht mit rechten Dingen zugeht. Wenn Ihre Benutzer in Ihrem Formular (oder in einer URL) ein Klartextfeld namens »Password« sehen, könnte das für mehr Unruhe sorgen als dieselben Daten in Base64-Codierung. Rezept 17.2 zeigt, wie man Daten mit Base64 verschleiert.

Vertrauliche Daten müssen nicht nur auf dem Server geschützt werden, sondern auch auf ihrem Weg durchs Netzwerk zwischen Ihrem Server und Ihren Benutzern. Daten, die über reguläres HTTP verschickt werden, lassen sich von jedem beobachten, der an irgendeinem Punkt zwischen Ihrem Server und einem Benutzer Zugang zum Netzwerk hat. Rezept 17.10 bespricht, wie Sie HTTP über SSL laufen lassen können, um Netzwerkschnüffler vom Ausspionieren vorbeireisender Daten abzuhalten.

Darüber hinaus unterliegt strenge Sicherheit auch noch einer Menge nicht-technischer Voraussetzungen. Das Ausgeben von Passwörtern, die aus zufällig zusammengewürfelten Buchstaben, Zahlen und Satzzeichen bestehen, hilft Ihnen auch nicht weiter, wenn diese Passwörter so schwer im Kopf zu behalten sind, dass Benutzer sie mit Stickern an ihre Monitore kleben. Wie bereits gesagt, gibt es keine absolute Sicherheit, sondern lediglich einen Kompromiss zwischen Bequemlichkeit und Schutz. Wenn Sie die Rezepte aus diesem Kapitel verwenden wollen, müssen Sie sich entscheiden, welches Risiko für Ihre Daten akzeptabel ist, und es gegen die Unannehmlichkeiten aufwiegen, die Sicherheitsmechanismen mit sich bringen.[1]

17.1 Passwörter aus den Dateien Ihrer Site heraushalten

Problem

Zum Verbindungsaufbau mit Ihrer Datenbank brauchen Sie beispielsweise ein Passwort. Sie wollen es aber nicht in die PHP-Dateien schreiben, die Sie auf Ihrer Site verwenden, für den Fall, dass die Dateien kompromittiert werden und nach außen gelangen.

Lösung

Speichern Sie das Passwort in einer Umgebungsvariablen in einer Datei, die der Webserver beim Start lädt. Verweisen Sie dann in Ihrem Skript einfach auf die Umgebungsvariable:

```
mysql_connect('localhost',$_SERVER['MYSQL_USER'],$_SERVER['MYSQL_PASSWORD']);
```

Diskussion

Diese Technik entfernt die Passwörter zwar aus dem Quellcode Ihrer Seiten, macht sie aber an anderen Stellen zugänglich, die zu schützen sind. Am wichtigsten für Sie ist es sicherzustellen, dass es keine öffentlich sichtbaren Seiten gibt, die `phpinfo( )` aufrufen. Da diese Funktion die den Skripten zugänglichen Umgebungsvariablen anzeigt, stellt sie auch die Passwörter bloß, die in Umgebungsvariablen abgespeichert sind.

1 *Practical Unix and Internet Security* von Simson Garfinkel und Gene Spafford (O'Reilly) gibt hilfreichen und (was nicht überraschend ist) praktischen Rat, wie Sie den Balanceakt Risikomanagement angehen können.

Insbesondere wenn Sie ein gemeinschaftlich genutztes Webhosting-System verwenden (virtuelles Webhosting), sollten Sie sicherstellen, dass die Umgebungsvariablen so gesetzt werden, dass sie nur Ihrem virtuellen Host zugänglich sind und nicht allen gemeinsam auf dem Server untergebrachten Benutzern. Unter Apache setzen Sie dazu die Variablen in einer getrennten Datei, d.h. nicht in der Hauptkonfigurationsdatei:

```
SetEnv  MYSQL_USER     "susanne"
SetEnv  MYSQL_PASSWORD "y23a!t@ce8"
```

Innerhalb der <VirtualHost>-Direktive für die Site in der Hauptkonfigurationsdatei können Sie diese separate Datei dann wie folgt mit einbeziehen:

```
Include "/usr/local/apache/database-passwords"
```

Stellen Sie sicher, dass die separate Datei mit den Passwörtern (z.B. */usr/local/apache/database-passwords*) nur von dem Benutzer gelesen werden kann, der den entsprechenden virtuellen Host kontrolliert – und von sonst niemandem. Wenn Apache startet und die Konfigurationsdateien liest, läuft das Programm üblicherweise als root, sodass es die einzubeziehende Datei lesen kann.

Siehe auch

Die Dokumentation der Include-Direktive von Apache unter *http://httpd.apache.org/docs/mod/core.html#include*.

17.2 Daten durch Kodierung verschleiern

Problem

Sie möchten verhindern, dass Daten im Klartext eingesehen werden können. Zum Beispiel möchten Sie versteckte Formulardaten nicht einfach verraten, wenn jemand den Quelltext einer Webseite anschaut.

Lösung

Kodieren Sie die Daten mit base64_encode( ):

```
$personal_data = array('code' => 5123, 'blood_type' => 'O');
$info = base64_encode(serialize($personal_data));
print '<input type="hidden" name="info" value="'.$info.'">';
<input type="hidden" name="info"
value="YToyOntzOjQ6ImNvZGUiO2k6NTEyMztzOjEwOiJibG9vZF90eXBlIjtzOjE6Ik8iO30=">
```

Dekodieren Sie die Daten mit base64_decode( ):

```
$personal_data = unserialize(base64_decode($_REQUEST['info']));
get_transfusion($personal_data['blood_type']);
```

Diskussion

Der Base64-Algorithmus kodiert Daten als eine Zeichenkette, die aus Buchstaben, Nummern und Satzzeichen besteht. Damit eignet er sich ideal dazu, binäre Daten in einfachen Text umzuwandeln, oder auch zur Tarnung der Daten.

Siehe auch

Die Dokumentationen zu `base64_encode()` unter *http://www.php.net/base64-encode* und `base64_decode()` unter *http://www.php.net/base64-decode*. Der Base64-Algorithmus wird im RFC 2045 definiert, der unter *http://www.faqs.org/rfcs/rfc2045.html* zu finden ist.

17.3 Daten durch Prüfsummen verifizieren

Problem

Sie wollen sicherstellen, dass Benutzer die Daten, die Sie ihnen in einem Cookie oder Formularelement geschickt haben, nicht verändern.

Lösung

Senden Sie zusammen mit den Daten eine MD5-Prüfsumme (Hash) der mit einem Geheimwort kombinierten Daten. Wenn Sie die Daten zurückerhalten, berechnen Sie die Prüfsumme der erhaltenen Werte mit dem gleichen Geheimwort. Wenn die beiden Prüfsummen nicht übereinstimmen, hat der Benutzer die Daten verändert.

So können Sie eine Prüfsumme in einem versteckten Formularfeld ausgeben:

```
$geheimwort = 'flyingturtle';
$id = 2836;
$hash = md5($geheimwort . $id);

print<<<_HTML_
<input type="hidden" name="id" value="$id">
<input type="hidden" name="idhash" value="$hash">
_HTML_;
```

Wenn es zurückgegeben wird, überprüfen Sie das versteckte Formularfeld wie folgt:

```
$geheimwort = 'flyingturtle';

if (md5($geheimwort. $_REQUEST['id']) == $_REQUEST['idhash']) {
    $id = $_REQUEST['id'];
} else {
    die("Ungueltige Daten in $_REQUEST[id]");
}
```

Diskussion

Wenn Sie die zurückgegebenen Formulardaten verarbeiten, berechnen Sie den Hash-Wert des zurückgegebenen Inhalts von `$_REQUEST['id']` in Kombination mit dem Geheimwort. Wenn die beiden Prüfsummen übereinstimmen, hat der Benutzer den Inhalt von `$_RE-QUEST['id']` nicht verändert. Stimmen sie nicht überein, wissen Sie, dass der empfangene Wert von `$_REQUEST['id']` von dem ausgegebenen Wert abweicht.

Um eine Prüfsumme zusammen mit einem Cookie zu verwenden, fügen Sie mit `join()` den Hash an den Cookie-Wert an:

```
$geheimwort = 'flyingturtle';
$cookie_wert = 'Ellen';
$hash = md5($geheimwort . $cookie_wert);

setcookie('name',join('|',array($cookie_wert,$hash)));
```

Beim Empfang des Cookies können Sie den Hash mit `explode()` aus dem Wert des Cookies extrahieren:

```
$geheimwort = 'flyingturtle';
list($cookie_wert,$cookie_hash) = explode('|',$_COOKIE['name'],2);
if (md5($geheimwort . $cookie_wert) == $cookie_hash) {
    $name = $cookie_wert;
} else {
    die('Ungueltige Daten in $_COOKIE[name]');
}
```

Die Verwendung einer Hash-Prüfsumme in einem Formular oder Cookie hängt natürlich von dem in der Hash-Berechnung benutzten Geheimwort ab. Wenn ein böswilliger Benutzer Ihr Geheimwort entdeckt, bietet der Hash keinen Schutz. Neben der sicheren Aufbewahrung des Geheimworts ist es eine gute Idee, dieses häufig zu ändern. Um zusätzlichen Schutz zu erreichen, können Sie mehrere Geheimwörter verwenden. Das spezielle Wort, das Sie für den Hash auswählen, können Sie durch Eigenschaften des `$id`-Werts auswählen (z.B. zehn verschiedene Wörter, die durch `$id%10` ausgewählt werden). Dadurch können Sie den Schaden in Grenzen halten, wenn eines der Wörter herauskommt.

Wenn Sie das *mhash*-Modul installiert haben, müssen Sie sich nicht auf MD5-Prüfsummen beschränken. *mhash* unterstützt mehrere Hash-Algorithmen. Nähere Informationen zu *mhash* finden Sie im *mhash*-Material im Online-PHP-Handbuch oder auf der *mhash*-Homepage unter *http://mhash.sourceforge.net/*.

Siehe auch

Rezept 10.10 benutzt eine Verifizierungs-Prüfsumme für die Cookie-basierte Authentifizierung; Rezept 11.3 zeigt ein Beispiel für die Verwendung von Hash-Werten mit versteckten Formularvariablen. Eine Dokumentation zu `md5()` finden Sie unter *http://www.php.net/md5* und die der *mhash*-Erweiterung unter *http://www.php.net/mhash*.

17.4 Passwörter speichern

Problem

Sie müssen die Passwörter der Benutzer verwalten, damit diese sich in Ihre Webseite einloggen können.

Lösung

Wenn sich ein Benutzer anmeldet, erstellen Sie mit `crypt()` einen Hash-Wert aus dem von ihm gewählten Passwort und speichern diesen in Ihrer Benutzerdatenbank:

```
// Passwort verschlüsseln.
$verschluesseltes_passwort = crypt($_REQUEST['passwort']);

// $verschluesseltes_passwort in der Benutzerdatenbank speichern.
$dbh->query('INSERT INTO benutzer (benutzername,passwort) VALUES (?,?)',
            array($_REQUEST['benutzername'],$verschluesseltes_passwort));
```

Wenn der Benutzer dann versucht, sich in Ihre Webseite einzuloggen, verschlüsseln Sie das von ihm angegebene Passwort ebenfalls mit `crypt()` und vergleichen es mit dem gespeicherten Hash-Wert. Wenn diese beiden Werte übereinstimmen, hat er das richtige Passwort eingegeben:

```
$verschluesseltes_passwort =
    $dbh->getOne('SELECT passwort FROM benutzer WHERE benutzername = ?',
                 array($_REQUEST['benutzername']));

if (crypt($_REQUEST['passwort'],$verschluesseltes_passwort) ==
                    $verschluesseltes_passwort) {
  // Login erfolgreich
} else {
  // Login fehlgeschlagen
}
```

Diskussion

Das Speichern verschlüsselter Passwörter (oder wie in diesem Fall der Hash-Werte der Passwörter) verhindert, dass Benutzerkonten kompromittiert werden, falls eine unberechtigte Person Einblick in Ihre Benutzernamen- und Passwort-Datenbank bekommt. (Solche unberechtigten Einblicke können allerdings auch Vorboten anderer Sicherheitsprobleme sein.)

Wenn das Passwort beim ersten Abspeichern mit `crypt()` zu einem Hash-Wert verschlüsselt wird, werden diesem Wert zufällig erstellte Salz-Zeichen vorangestellt. Verwenden Sie `$verschluesseltes_passwort` bei der Verschlüsselung eines vom Benutzer eingereichten Passworts, wird `crypt()` dazu aufgefordert, wieder die gleichen Salz-Zeichen zu benutzen. Dadurch werden beide Hash-Werte vergleichbar.

Das Salz vermindert Ihre Verwundbarkeit bei Wörterbuch-Attacken, bei denen jemand das verschlüsselte Passwort mit verschlüsselten Versionen von gebräuchlichen Wörtern vergleicht. Trotzdem ist es sinnvoll, Benutzer von der Verwendung einfacher Wörter oder einfach zu knackender Kombinationen für Passwörter abzuhalten. Rezept 17.5 stellt eine Funktion zur Verfügung, die leicht zu erratende Passwörter ausfiltert.

Die `crypt()`-Funktion benutzt einen Einweg-Algorithmus, um einen Hash-Wert zu erstellen. Der Hash-Wert hat dabei in der Regel eine feste Länge, die unter Umständen kürzer als der Originaltext sein kann. Der Informationsgehalt ist in solch einem Fall dann auch geringer. Diese Eigenschaft impliziert, dass Hash-Werte nicht zurückberechnet werden können, aber auch, dass unterschiedliche Passwörter den gleichen Hash-Wert ergeben können. Die Hash-Algorithmen minimieren allerdings diese Wahrscheinlichkeit sehr.

Weil eine Entschlüsselung nicht möglich ist, bedeutet das auch, dass Sie nicht an den Klartext des Benutzerpassworts herankommen, selbst wenn Sie es einmal brauchen. Wenn also beispielsweise ein Benutzer sein Passwort vergisst, können Sie ihm das Passwort nicht mitteilen. Sie können es lediglich auf einen neuen Wert setzen und dem Benutzer dann dieses neue Passwort übermitteln. Eine Methode zum Umgang mit verloren gegangenen Passwörtern wird in Rezept behandelt.

Siehe auch

Rezept 17.8 zur Speicherung von verschlüsselten Daten; die Dokumentation zu `crypt()` unter *http://www.php.net/crypt*.

17.5 Überprüfung der Passwortsicherheit

Problem

Sie möchten sicherstellen, dass die Benutzer schwer zu erratende Passwörter auswählen.

Lösung

Überprüfen Sie die Passwortauswahl eines Benutzers mit der Funktion `pc_passwordcheck()`, die in Beispiel 17-1 unten erklärt wird. Ein Beispiel:

```
if ($err = pc_passwordcheck($_REQUEST['benutzername'],$_REQUEST['passwort'])) {
    print "Ungeeignetes Passwort: $err";
    // Lassen Sie den Benutzer ein anderes Passwort auswählen.
}
```

Diskussion

Die in Beispiel 17-1 verwendete Funktion `pc_passwordcheck()` führt ein paar Tests an den von den Benutzern eingegebenen Passwörtern durch, um sicherzustellen, dass sie schwe-

rer zu knacken sind. Falls das Passwort die Kriterien nicht erfüllt, gibt die Funktion einen String zurück, der das Problem beschreibt. Das Passwort muss mindestens sechs Zeichen lang sein und eine Kombination aus Großbuchstaben, Kleinbuchstaben, Nummern und speziellen Zeichen enthalten. Das Passwort darf den Benutzernamen weder in richtiger noch in umgedrehter Reihenfolge enthalten. Außerdem darf es kein Wort aus dem Wörterbuch enthalten. Der Dateiname für die Wörterliste, die als Wörterbuch verwendet wird, wird in `$word_file` gespeichert (in diesem Fall ein englisches Wörterbuch).

Die Tests für den Benutzernamen und für Wörter aus dem Wörterbuch im Passwort werden auch für Versionen des Passworts durchgeführt, bei denen Buchstaben durch ähnlich aussehende Nummern ersetzt worden sind. Wenn das eingegebene Passwort beispielsweise `w0rd$%` ist, testet die Funktion auch, ob der String `word$%` der Benutzername oder ein Wort aus dem Wörterbuch ist. Das Zeichen »0« wird durch »o« ersetzt. Genauso wird »5« durch »s«, »3« durch »e« und sowohl »1« als auch »!« durch »l« (el) ersetzt.

Beispiel 17-1: pc_passwordcheck()

```
function pc_passwordcheck($user,$pass) {
    $word_file = '/usr/share/dict/words';

    $lc_pass = strtolower($pass);
    // Passwort auch auf durch Ziffern oder Interpunktion ersetzte Buchstaben prüfen.
    $denum_pass = strtr($lc_pass,'53O1!','seoll');
    $lc_user = strtolower($user);

    // Das Passwort muss mindestens 6 Zeichen lang sein.
    if (strlen($pass) < 6) {
        return 'Das Passwort ist zu kurz.';
    }

    // Das Passwort darf nicht dem Benutzernamen entsprechen, auch wenn man es umdreht.
    if (($lc_pass == $lc_user) || ($lc_pass == strrev($lc_user)) ||
        ($denum_pass == $lc_user) || ($denum_pass == strrev($lc_user))) {
        return 'Das Passwort beruht auf dem Benutzernamen.';
    }

    // Groß-/Kleinbuchstaben und Ziffern im Passwort zählen.
    $uc = 0; $lc = 0; $num = 0; $other = 0;
    for ($i = 0, $j = strlen($pass); $i < $j; $i++) {
        $c = substr($pass,$i,1);
        if (preg_match('/^[[:upper:]]$/',$c)) {
            $uc++;
        } elseif (preg_match('/^[[:lower:]]$/',$c)) {
            $lc++;
        } elseif (preg_match('/^[[:digit:]]$/',$c)) {
            $num++;
        } else {
            $other++;
        }
    }
```

Beispiel 17-1: pc_passwordcheck() (Fortsetzung)

```
    // Das Passwort muss mehr als zwei Zeichen mindestens zweier
    // verschiedener Typen enthalten.
    $max = $j - 2;
    if ($uc > $max) {
        return "Das Passwort hat zu viele Großbuchstaben.";
    }
    if ($lc > $max) {
        return "Das Passwort hat zu viele Kleinbuchstaben.";
    }
    if ($num > $max) {
        return "Das Passwort hat zu viele Ziffern.";
    }
    if ($other > $max) {
        return "Das Passwort hat zu viele Sonderzeichen.";
    }

    // Das Passwort darf kein Wort aus dem Wörterbuch enthalten.
    if (is_readable($word_file)) {
        if ($fh = fopen($word_file,'r')) {
            $found = false;
            while (! ($found || feof($fh))) {
                $word = preg_quote(trim(strtolower(fgets($fh,1024))),'/');
                if (preg_match("/$word/",$lc_pass) ||
                    preg_match("/$word/",$denum_pass)) {
                    $found = true;
                }
            }
            fclose($fh);
            if ($found) {
                return 'Das Passwort beruht auf einem Wort aus dem Wörterbuch.';
            }
        }
    }

    return false;
}
```

17.6 Was tun bei verlorenen Passwörtern?

Problem

Sie wollen ein Passwort an einen Benutzer ausgeben, der angibt, sein Passwort verloren zu haben.

Lösung

Erzeugen Sie ein neues Passwort und senden Sie es an die E-Mail-Adresse des Benutzers (die Sie irgendwo hinterlegt haben sollten):

```
// Neues Passwort erzeugen.
$neues_passwort = '';
$i = 8;
while ($i--) { $neues_passwort .= chr(mt_rand(33,126)); }

// Neues Passwort verschlüsseln.
$verschluesseltes_passwort = crypt($neues_passwort);

// Neues verschlüsseltes Passwort in der Datenbank speichern.
$dbh->query('UPDATE benutzer SET passwort = ? WHERE benutzername = ?',
            array($verschluesseltes_passwort,$benutzername));

// Neues Klartext-Passwort per E-Mail an den Benutzer schicken.
mail($email,"Neues Passwort","Ihr neues Passwort ist $neues_passwort");
```

Diskussion

Wenn ein Benutzer sein Passwort vergisst und wenn Sie, wie in Rezept 17.4 empfohlen, verschlüsselte Passwörter speichern, können Sie das vergessene Passwort nicht herausgeben. Da `crypt( )` nur in eine Richtung funktioniert, haben Sie keinen Zugriff auf das Klartext-Passwort.

Stattdessen erstellen Sie ein neues Passwort und senden es an seine bereits bekannte E-Mail-Adresse. Wenn Sie das neue Passwort an eine Adresse senden, die Sie für diesen Benutzer noch nicht gespeichert haben, können Sie nicht verifizieren, dass die neue Adresse auch wirklich dem Benutzer gehört. Es könnte ja sein, dass jemand eine Attacke durchführt und vorgibt, der echte Benutzer zu sein.

Da die E-Mail mit dem neuen Passwort nicht verschlüsselt ist, enthält der Code in der Lösung keinen Benutzernamen in der E-Mail-Nachricht. Das verringert die Wahrscheinlichkeit, dass ein Angreifer, der sich Zugang zu der E-Mail-Nachricht verschafft hat, das Passwort stehlen kann. Um das Offenlegen eines neuen Passworts per E-Mail-Nachricht ganz zu vermeiden, können Sie den Benutzer auffordern, sich selbst ohne Passwort zu identifizieren, indem er eine oder mehrere persönliche Fragen beantwortet (die Antworten haben Sie in einer Datei gespeichert). Solche Fragen sind zum Beispiel: »Was war der Name Ihres ersten Haustiers?«, oder: »Was ist der Mädchenname Ihrer Mutter?« – eben alles, was ein bösartiger Angreifer sehr wahrscheinlich nicht weiß. Wenn der Benutzer die richtigen Antworten auf Ihre Fragen liefert, können Sie ihn ein neues Passwort wählen lassen.

Eine Möglichkeit, die einen Kompromiss zwischen Sicherheit und Lesbarkeit bietet, ist ein Passwort für einen Benutzer, das aus wirklichen Wörtern besteht, die von einigen Ziffern unterbrochen werden.

```
$woerter =
array('kugeln','freddy','papier','geiger','wieder','elende','angelt','trumpf',
      'siegel','raufen','staube','glocke','zeiger','dient','mutige','kommt',
      'drusus','brille','pommes','gabeln','fertig','dattel','bremse','nuesse',
      'schild','fester','schafe','hobbit','klumpt','lampen','latent','verbal',
      'pelzig','ritual','trasse','finger','truhen','bremen','sauren','kuebel');
```

```
mt_srand((double) microtime() * 1000000);
$wortanzahl = count($woerter);

$passwort = sprintf('%s%02d%s',
                    $woerter[mt_rand(0,$wortanzahl - 1)],
                    mt_rand(0,99),
                    $woerter[mt_rand(0,$wortanzahl - 1)]);

print $passwort;
```

Dieser Code erstellt Passwörter, die aus zwei Wörtern mit jeweils sechs Buchstaben und zwei Ziffern dazwischen bestehen, wie z.B. `gabeln43trumpf` oder `fester08hobbit`. Die Passwörter sind zwar lang, dank der Wörter, die sie enthalten, kann man sie sich aber leichter merken.

Siehe auch

Rezept 17.4 informiert über das Speichern von verschlüsselten Passwörtern, Rezept 17.5 bespricht das Testen der Passwort-Sicherheit im Detail.

17.7 Daten ver- und entschlüsseln

Problem

Sie wollen mit einem von mehreren gebräuchlichen Algorithmen Daten ver- und entschlüsseln.

Lösung

Benutzen Sie die PHP-Erweiterung *mcrypt*:

```
$schluessel  = 'That golden key that opes the palace of eternity.';
$daten = 'The chicken escapes at dawn. Send help with Mr. Blue.';
$alg  = MCRYPT_BLOWFISH;
$modus = MCRYPT_MODE_CBC;

$iv = mcrypt_create_iv(mcrypt_get_iv_size($alg,$modus),MCRYPT_DEV_URANDOM);
$verschluesselte_daten = mcrypt_encrypt($alg, $schluessel, $daten, $modus, $iv);
$zeichentext = base64_encode($verschluesselte_daten);

print $zeichentext."\n";
$decodiert = mcrypt_decrypt($alg,$key,base64_decode($zeichentext),$modus,$iv);
print $decodiert."\n";
NNB9WnuCYjyd3Y7vUh7XDfWFCWnQYOBsMehHNmBHbGOdJ3cM+yghABb/XyrJ+w3xz9tms74/a7O=
The chicken escapes at dawn. Send help with Mr. Blue.
```

Diskussion

Die *mcrypt*-Erweiterung ist eine Schnittstelle zu *mcrypt*, einer Bibliothek, in der viele verschiedene Verschlüsselungsalgorithmen implementiert sind. Die Daten werden jeweils mit `mcrypt_encrypt()` und `mcrypt_decrypt()` ver- bzw. entschlüsselt. Jede der beiden Funktionen nimmt je fünf Parameter an. Der erste beschreibt den zu benutzenden Algorithmus. Mit `mcrypt_list_algorithms()` finden Sie heraus, welche Algorithmen *mcrypt* auf Ihrem System unterstützt. Tabelle 17-1 enthält die vollständige Liste der *mcrypt*-Algorithmen. Das zweite Argument ist der Schlüssel, und das dritte Argument enthält die zu ver- oder entschlüsselnden Daten. Das vierte Argument ist der Ver- bzw. Entschlüsselungsmodus (eine Liste der unterstützten Modi wird über `mcrypt_list_modes()` ausgegeben). Das fünfte Argument ist ein Initialisierungsvektor (IV), der von manchen Modi als Teil des Ver- oder Entschlüsselungsprozesses verwendet wird.

In Tabelle 17-1 sind alle möglichen *mcrypt*-Algorithmen aufgelistet. Sie enthält den Konstantwert, der den Algorithmus angibt, sowie die Schlüssel- und Blockgrößen in Bit. Außerdem gibt sie an, ob der Algorithmus von *libmcrypt* 2.2.x oder 2.4.x unterstützt wird.

Tabelle 17-1: mcrypt-Algorithmen-Konstanten

Algorithmen-Konstante	Beschreibung	Schlüssel-größe	Block-größe	2.2.x	2.4.x
MCRYPT_3DES	Triple DES (Dreifach-DES)	168 (112 effektiv)	64	Ja	Ja
MCRYPT_TRIPLEDES	Triple DES (Dreifach-DES)	168 (112 effektiv)	64	Nein	Ja
MCRYPT_3WAY	3way (Joan Daemen)	96	96	Ja	Nein
MCRYPT_THREEWAY	3way	96	96	Ja	Ja
MCRYPT_BLOWFISH	Blowfish (Bruce Schneier)	bis zu 448	64	Nein	Ja
MCRYPT_BLOWFISH_COMPAT	Blowfish mit Kompatibilität zu anderen Implementationen	bis zu 448	64	Nein	Ja
MCRYPT_BLOWFISH_128	Blowfish	128	64	Ja	Nein
MCRYPT_BLOWFISH_192	Blowfish	192	64	Ja	
MCRYPT_BLOWFISH_256	Blowfish	256	64	Ja	Nein
MCRYPT_BLOWFISH_448	Blowfish	448	64	Ja	Nein
MCRYPT_CAST_128	CAST (Carlisle Adams und Stafford Tavares)	128	64	Ja	Ja
MCRYPT_CAST_256	CAST	256	128	Ja	Ja
MCRYPT_CRYPT	Einzelrotor-Unix-crypt	104	8		Ja
MCRYPT_ENIGMA	Einzelrotor-Unix-crypt	104	8	Nein	Ja
MCRYPT_DES	U.S. Data Encryption Standard	56	64	Ja	Ja
MCRYPT_GOST	Soviet Gosudarstvennyi Standard (»Government Standard«)	256	64	Ja	Ja

Tabelle 17-1: mcrypt-Algorithmen-Konstanten (Fortsetzung)

Algorithmen-Konstante	Beschreibung	Schlüssel-größe	Block-größe	2.2.x	2.4.x
MCRYPT_IDEA	International Data Encryption Algorithm	128	64	Ja	Ja
MCRYPT_LOKI97	LOKI97 (Lawrie Brown, Josef Pieprzyk)	128, 192, oder 256	64	Ja	Ja
MCRYPT_MARS	MARS (IBM)	128–448	128	Nein	Ja
MCRYPT_PANAMA	PANAMA (Joan Daemen, Craig Clapp)	–	Stream	Nein	Ja
MCRYPT_RC2	Rivest Cipher 2	8–1024	64	Nein	Ja
MCRYPT_RC2_1024	Rivest Cipher 2	1024	64	Ja	Nein
MCRYPT_RC2_128	Rivest Cipher 2	128	64	Ja	Nein
MCRYPT_RC2_256	Rivest Cipher 2	256	64	Ja	Nein
MCRYPT_RC4	Rivest Cipher 4	bis zu 2048	Stream	Ja	Nein
MCRYPT_ARCFOUR	Warenzeichenlos, RC4-kompatibel	bis zu 2048	Stream	Nein	Ja
MCRYPT_ARCFOUR_IV	Arcfour mit Initialization Vector	bis zu 2048	Stream	Nein	Ja
MCRYPT_RC6	Rivest Cipher 6	128, 192 oder 256	128	Nein	Ja
MCRYPT_RC6_128	Rivest Cipher 6	128	128	Ja	Nein
MCRYPT_RC6_192	Rivest Cipher 6	192	128	Ja	Nein
MCRYPT_RC6_256	Rivest Cipher 6	256	128	Ja	Nein
MCRYPT_RIJNDAEL_128	Rijndael (Joan Daemen, Vincent Rijmen)	128	128	Ja	Ja
MCRYPT_RIJNDAEL_192	Rijndael	192	192	Ja	Ja
MCRYPT_RIJNDAEL_256	Rijndael	256	256	Ja	Ja
MCRYPT_SAFERPLUS	SAFER+ (basiert auf SAFER)	128, 192 oder 256	128	Ja	Ja
MCRYPT_SAFER_128	Secure And Fast Encryption Routine mit verstärktem Schlüsselformat	128	64	Ja	Ja
MCRYPT_SAFER_64	Secure And Fast Encryption Routine mit verstärktem Schlüssel	64	64	Ja	Ja
MCRYPT_SERPENT	Serpent (Ross Anderson, Eli Biham, Lars Knudsen)	128, 192 oder 256	128	Nein	Ja
MCRYPT_SERPENT_128	Serpent	128	128	Ja	Nein
MCRYPT_SERPENT_192	Serpent	192	128	Ja	Nein
MCRYPT_SERPENT_256	Serpent	256	128	Ja	Nein
MCRYPT_SKIPJACK	U.S. NSA Clipper Escrowed Encryption Standard	80	64	Nein	Ja
MCRYPT_TWOFISH	Twofish (Counterpane Systems)	128, 192 oder 256	128	Nein	Ja

Tabelle 17-1: mcrypt-Algorithmen-Konstanten (Fortsetzung)

Algorithmen-Konstante	Beschreibung	Schlüssel-größe	Block-größe	2.2.x	2.4.x
MCRYPT_TWOFISH_128	Twofish	128	128	Ja	Nein
MCRYPT_TWOFISH_192	Twofish	192	128	Ja	Nein
MCRYPT_TWOFISH_256	Twofish	256	128	Ja	Nein
MCRYPT_WAKE	Word Auto Key Encryption (David Wheeler)	256	32	Nein	Ja
MCRYPT_XTEA	Extended Tiny Encryption Algorithm (David Wheeler, Roger Needham)	128	64	Ja	Ja

Abgesehen von den zu ver- oder entschlüsselnden Daten müssen alle Argumente beim Ver- und Entschlüsseln gleich sein. Wenn Sie einen Modus verwenden, der einen Initialisierungsvektor braucht, können Sie den Initialisierungsvektor im Klartext zusammen mit dem verschlüsselten Text übertragen.

Die verschiedenen Modi sind in unterschiedlichen Situationen angebracht. Der Cipher Block Chaining-(CBC-)Modus verschlüsselt Daten blockweise und verwendet den verschlüsselten Wert jedes Blocks (und den Schlüssel), um den verschlüsselten Wert des nächsten Blocks zu berechnen. Der Initialisierungsvektor beeinflusst den verschlüsselten Wert des ersten Blocks. Cipher Feedback (CFB) und Output Feedback (OFB) verwenden auch einen Initialisierungsvektor, aber sie verschlüsseln die Daten in Einheiten, die kleiner sind als die Blockeinheit. Beachten Sie, dass der OFB-Modus Sicherheitsprobleme aufweist, wenn Sie Daten in Einheiten verschlüsseln, die kleiner als die OFB-Blockeinheit sind. Der Electronic Code Book-(ECB-)Modus verschlüsselt Daten in diskreten und unabhängigen Blöcken und verwendet keinen Initialisierungsvektor. Für wiederholte Verschlüsselung ist er außerdem weniger sicher als andere Modi, da der gleiche Klartext mit dem gleichen Schlüssel immer den gleichen Chiffriertext ergibt. Die Konstanten zum Setzen der einzelnen Modi sind in Tabelle 17-2 aufgeführt.

Tabelle 17-2: mcrypt-Modus-Konstanten

Modus-Konstante	Beschreibung
MCRYPT_MODE_ECB	Electronic Code Book-Modus
MCRYPT_MODE_CBC	Cipher Block Chaining-Modus
MCRYPT_MODE_CFB	Cipher Feedback-Modus
MCRYPT_MODE_OFB	Output Feedback-Modus mit 8-Bit-Feedback
MCRYPT_MODE_NOFB	Output Feedback-Modus mit *n* Bits Feedback, wobei *n* die Blockgröße des verwendeten Algorithmus ist (nur *libmcrypt* 2.4 und höher)
MCRYPT_MODE_STREAM	Stream Cipher-Modus, für Algorithmen wie RC4 und WAKE (nur *libmcrypt* 2.4 und höher)

Die verschiedenen Algorithmen weisen unterschiedliche Blockgrößen auf. Mit `mcrypt_get_block_size()` können Sie die Blockgröße des angegebenen Algorithmus erfahren. Außerdem wird die Größe des Initialisierungsvektors durch den Algorithmus und den Modus bestimmt. `mcrypt_create_iv()` und `mcrypt_get_iv_size()` machen es einfach, einen passenden Zufalls-Initialisierungsvektor zu erzeugen:

```
$iv = mcrypt_create_iv(mcrypt_get_iv_size($alg,$modus),MCRYPT_DEV_URANDOM);
```

Das erste Funktionsargument von `mcrypt_create_iv()` ist die Vektorlänge, das zweite eine Zufallsquelle. Sie haben drei Zufallsquellen zur Auswahl. `MCRYPT_DEV_RANDOM` liest von dem Pseudogerät */dev/random*, `MCRYPT_DEV_URANDOM` von dem Pseudogerät */dev/urandom*, und `MCRYPT_RAND` verwendet einen internen Zufallsgenerator. Nicht alle Betriebssysteme unterstützen Pseudogeräte zur Erzeugung von Zufallszahlen. Stellen Sie sicher, dass Sie `srand()` aufrufen, bevor Sie `MCRYPT_RAND` verwenden – sonst bekommen Sie immer wieder dieselbe Folge von Zufallszahlen.

Der Code und die Beispiele in diesem Rezept sind mit *mcrypt* 2.4 kompatibel. Die *mcrypt*-Schnittstelle von PHP unterstützt sowohl *mcrypt* 2.2 als auch *mcrypt* 2.4, aber es bestehen Unterschiede zwischen den beiden. Wenn Sie *mcrypt* 2.2 verwenden, unterstützt PHP lediglich die folgenden *mcrypt*-Funktionen: `mcrypt_ecb()`, `mcrypt_cbc()`, `mcrypt_cfb()`, `mcrypt_ofb()`, `mcrypt_get_key_size()`, `mcrypt_get_block_size()`, `mcrypt_get_cipher_name()` und `mcrypt_create_iv()`. Um Daten mit *mcrypt* 2.2 zu ver- oder entschlüsseln, rufen Sie die passende `mcrypt_`*`MODUS`*`()`-Funktion auf, je nachdem, welchen Modus Sie verwenden wollen, und übergeben ihr ein Argument, das die Anweisung zur Ver- oder Entschlüsselung gibt. Der folgende Code ist die *mcrypt* 2.2-kompatible Version des Codes, der in der Lösung verwendet wird:

```
$schluessel  = 'That golden key that opes the palace of eternity.';
$daten = 'The chicken escapes at dawn. Send help with Mr. Blue.';
$alg = MCRYPT_BLOWFISH;

$iv = mcrypt_create_iv(mcrypt_get_block_size($alg),MCRYPT_DEV_URANDOM);
$verschluesselte_daten = mcrypt_cbc($alg,$schluessel,$daten,MCRYPT_ENCRYPT);
$zeichentext = base64_encode($verschluesselte_daten);

print $zeichentext."\n";

$decodiert = mcrypt_cbc($alg,$schluessel,base64_decode($zeichentext),MCRYPT_DECRYPT);

print $decodiert."\n";
```

Siehe auch

Die Dokumentation zur *mcrypt*-Erweiterung unter *http://www.php.net/mcrypt*; die *mcrypt*-Bibliothek erhalten Sie unter *http://mcrypt.sourceforge.net/*. Die Auswahl eines passenden Algorithmus und seine sichere Anwendung brauchen Sorgfalt und eine gute Planung: Weitere Informationen über *mcrypt* und den Chiffrieralgorithmus, den *mcrypt* anwendet, finden Sie im Online-PHP-Handbuch unter *mcrypt*, der *mcrypt*-Homepage

und den Man-Pages für */dev/random* und */dev/urandom*; gute Bücher über Kryptographie sind *Angewandte Kryptographie* von Bruce Schneier (Addison-Wesley), *Cryptography: Theory and Practice* von Douglas R. Stinson (Chapman & Hall) und *Cryptographic Security Architecture: Design and Verification* von Peter Gutmann (Springer).

17.8 Verschlüsselte Daten in einer Datei oder Datenbank speichern

Problem

Sie wollen verschlüsselte Daten speichern, auf die Ihr Webserver später zugreifen kann, um sie zu entschlüsseln.

Lösung

Speichern Sie die Informationen, die für die Entschlüsselung zusätzlich notwendig sind (z.B. Algorithmus, Chiffriermodus und Initialisierungsvektor), zusammen mit den verschlüsselten Informationen, aber ohne den Schlüssel:

```
// Daten verschlüsseln.
$alg  = MCRYPT_BLOWFISH;
$mode = MCRYPT_MODE_CBC;
$iv = mcrypt_create_iv(mcrypt_get_iv_size($alg,$mode),MCRYPT_DEV_URANDOM);
$ciphertext = mcrypt_encrypt($alg,$_REQUEST['key'],$_REQUEST['data'],$mode,$iv);

// Verschlüsselte Daten speichern.
$dbh->query('INSERT INTO noc_list (algorithm,mode,iv,data) values (?,?,?,?)',
            array($alg,$mode,$iv,$ciphertext));
```

Zur Entschlüsselung greifen Sie auf einen Schlüssel des Benutzers zu und verwenden ihn für die gespeicherten Daten:

```
$row = $dbh->getRow('SELECT * FROM noc_list WHERE id = 27');
$plaintext = mcrypt_decrypt($row->algorithm,$_REQUEST['key'],$row->data,
                            $row->mode,$row->iv);
```

Diskussion

Das in Beispiel 17-2 beschriebene *save-crypt.php*-Programm speichert verschlüsselte Daten in einer Datei.

Beispiel 17-2: save-crypt.php

```
function show_form() {
    print<<<_FORM_
<form method="post" action="$_SERVER[PHP_SELF]">
<textarea name="data" rows="10" cols="40">
```

Beispiel 17-2: save-crypt.php (Fortsetzung)

```
Geben Sie hier die zu verschl&uuml;sselnden Daten ein.
</textarea>
<br>
Schl&uuml;ssel: <input type="text" name="key">
<input name="submit" type="submit" value="Speichern">
</form>
_FORM_;
}

function save_form() {
    $alg  = MCRYPT_BLOWFISH;
    $mode = MCRYPT_MODE_CBC;

    // Daten verschlüsseln.
    $iv = mcrypt_create_iv(mcrypt_get_iv_size($alg,$mode),MCRYPT_DEV_URANDOM);
    $ciphertext = mcrypt_encrypt($alg, $_REQUEST['key'],
                                 $_REQUEST['data'], $mode, $iv);

    // Verschlüsselte Daten speichern.
    $filename = tempnam('/tmp','enc') or die($php_errormsg);
    $fh = fopen($filename,'w')        or die($php_errormsg);
    if (false === fwrite($fh,$iv.$ciphertext)) {
        fclose($fh);
        die($php_errormsg);
    }
    fclose($fh)                       or die($php_errormsg);

    return $filename;
}

if ($_REQUEST['submit']) {
    $file = save_form();
    print "Verschlu&uuml;sselte Daten in Datei $file gespeichert\n";
} else {
    show_form();
}
```

Beispiel 17-3 zeigt das dazugehörige Programm *get-crypt.php*, das einen Dateinamen und einen Schlüssel entgegennimmt und die entschlüsselten Daten ausgibt.

Beispiel 17-3: get-crypt.php

```
function show_form() {
    print<<<_FORM_
<form method="post" action="$_SERVER[PHP_SELF]">
Verschl&uuml;sselte Datei: <input type="text" name="file">
<br>
Schl&uuml;sel: <input type="text" name="key">
<input name="submit" type="submit" value="Anzeigen">
</form>
_FORM_;
```

Beispiel 17-3: get-crypt.php (Fortsetzung)

```
}

function display() {
    $alg  = MCRYPT_BLOWFISH;
    $mode = MCRYPT_MODE_CBC;

    $fh = fopen($_REQUEST['file'],'r') or die($php_errormsg);
    $iv = fread($fh,mcrypt_get_iv_size($alg,$mode));
    $ciphertext = fread($fh,filesize($_REQUEST['file']));
    fclose($fh);

    $plaintext = mcrypt_decrypt($alg,$_REQUEST['key'],$ciphertext,$mode,$iv);
    print "<pre>$plaintext</pre>";
}

if ($_REQUEST['submit']) {
    display();
} else {
    show_form();
}
```

In diesen beiden Programmen sind der Verschlüsselungsalgorithmus und der Modus bereits fest einprogrammiert, sodass diese Informationen nicht in der Datei gespeichert werden müssen. Die Datei besteht aus dem Initialisierungsvektor, dem direkt die verschlüsselten Daten folgen. Hinter dem Initialisierungsvektor (IV) wird kein Begrenzungszeichen benötigt, da `mcrypt_get_iv_size( )` genau die Anzahl der Bytes angibt, die das Verschlüsselungsprogramm lesen muss, um den gesamten IV zu erhalten. Sämtliche nachfolgenden Informationen in der Datei sind verschlüsselte Daten.

Wenn Sie Dateien mit der in diesem Rezept vorgestellten Methode verschlüsseln, sind diese geschützt, wenn ein Angreifer Zugriff auf den Server erhält, auf dem sie gespeichert sind. Ohne den Schlüssel oder ungeheuer viel Rechenaufwand kann der Angreifer die Dateien nicht lesen. Allerdings wird die Sicherheit, den diese verschlüsselten Daten bieten, unterhöhlt, wenn die zu verschlüsselnden Daten und der Schlüssel im Klartext zwischen Ihrem Server und dem Webbrowser Ihres Benutzers verschickt werden. Ein Angreifer, der den Netzwerkverkehr abfangen oder beobachten kann, kann Daten lesen, bevor sie überhaupt verschlüsselt werden. Um diese Art des Mitwissertums zu vermeiden, benutzen Sie SSL (siehe Rezept 17.10).

Wenn Ihr Webserver Daten nach dem hier vorgestellten Rezept verschlüsselt, besteht eine zusätzliche Gefahr, je nachdem, wie die Daten vor der Verschlüsselung und dem Schreiben in eine Datei sichtbar sind. Wenn jemand Zugang zum Server als root oder als Systemadministrator besitzt, kann er den vom Webserverprozess verwendeten Speicher einsehen und die unverschlüsselten Daten und den Schlüssel herausschnüffeln. Lagert das Betriebssystem das Speicherabbild des Webservers auf eine Festplatte aus, können die unverschlüsselten Daten auch über die Auslagerungsdatei zugänglich sein. Diese Art Attacke ist zwar schwer auszuführen, kann aber verheerende Auswirkungen haben.

Sobald die verschlüsselten Daten in einer Datei sind, sind sie unlesbar, auch für einen Angreifer, der als root Zugang zum Webserver besitzt. Wenn der Angreifer die unverschlüsselten Daten aber sehen kann, bevor sie in dieser Datei sind, bietet die Verschlüsselung nur wenig Schutz.

Siehe auch

Rezept 17.10 behandelt SSL und den Datenschutz beim Versand über das Netzwerk; die Dokumentationen zu `mcrypt_encrypt()` finden Sie unter *http://www.php.net/mcrypt-encrypt*, `mcrypt_decrypt()` unter *http://www.php.net/mcrypt-decrypt*, `mcrypt_create_iv()` unter *http://www.php.net/mcrypt-create-iv* und `mcrypt_get_iv_size()` unter *http://www.php.net/mcrypt-get-iv-size*.

17.9 Verschlüsselte Daten gemeinsam mit einer anderen Website nutzen

Problem

Sie wollen auf eine sichere Art und Weise Daten mit einer anderen Website austauschen.

Lösung

Wenn die andere Website die Daten von Ihrer Seite herunterlädt, können Sie die Daten auf eine passwortgeschützte Seite stellen. Sie können die Daten auch im verschlüsselten Format zur Verfügung stellen, und zwar mit oder ohne Passwort. Wollen Sie die Daten zu einer anderen Website hinüberschieben, schicken Sie die verschlüsselten Daten über POST an eine passwortgeschützte URL.

Diskussion

Die folgende Seite fragt nach Benutzername und Passwort. Sie verschlüsselt dann den Inhalt einer Datei, die die gestrigen Kontoaktivitäten enthält, und zeigt sie an:

```
$user = 'bank';
$password = 'fas8uj3';

if (! (($_SERVER['PHP_AUTH_USER'] == $user) &&
       ($_SERVER['PHP_AUTH_PW'] == $password))) {
    header('WWW-Authenticate: Basic realm="Secure Transfer"');
    header('HTTP/1.0 401 Unauthorized');
    echo "You must supply a valid username and password for access.";
    exit;
}

header('Content-type: text/plain');
```

```
$filename = strftime('/usr/local/account-activity.%Y-%m-%d',time() - 86400);
$data = join('',file($filename));

$alg  = MCRYPT_BLOWFISH;
$mode = MCRYPT_MODE_CBC;
$key  = "There are many ways to butter your toast.";

// Daten verschlüsseln.
$iv = $iv = mcrypt_create_iv(mcrypt_get_iv_size($alg,$mode),
                             MCRYPT_DEV_URANDOM);
$ciphertext = mcrypt_encrypt($alg, $key, $data, $mode, $iv);

print base64_encode($iv.$ciphertext);
```

Hier folgt der dazugehörige Code, mit dem Sie auf die verschlüsselte Seite wieder zugreifen und die darin enthaltenen Informationen entschlüsseln können:

```
$user = 'bank';
$password = 'fas8uj3';
$alg  = MCRYPT_BLOWFISH;
$mode = MCRYPT_MODE_CBC;
$key  = "There are many ways to butter your toast.";

$fh = fopen("http://$user:$password@bank.example.com/accounts.php",'r')
      or die($php_errormsg);
$data = '';
while (! feof($fh)) { $data .= fgets($fh,1048576); }
fclose($fh) or die($php_errormsg);
$binary_data = base64_decode($data);
$iv_size = mcrypt_get_iv_size($alg,$mode);
$iv = substr($binary_data,0,$iv_size);
$ciphertext = substr($binary_data,$iv_size,strlen($binary_data));

print mcrypt_decrypt($alg,$key,$ciphertext,$mode,$iv);
```

Das Zugriffsprogramm führt alle Schritte des Verschlüsselungsprogramms durch, aber in umgekehrter Richtung. Es ruft die Base64-kodierten verschlüsselten Daten ab, indem es einen Benutzernamen und ein Passwort angibt. Dann dekodiert es die Daten mit Base64 und trennt den Initialisierungsvektor ab. Schließlich entschlüsselt es die Daten und druckt sie aus.

In den vorhergehenden Beispielen werden der Benutzername und das Passwort immer noch als Klartext über das Netzwerk gesendet, es sei denn, die Verbindung wird über SSL hergestellt. Wenn Sie SSL verwenden, ist es allerdings wahrscheinlich nicht notwendig, den Dateiinhalt zu verschlüsseln. Wir haben in diesen Beispielen sowohl Passwortabfrage als auch Dateiverschlüsselung durchgeführt, um zu zeigen, wie es gemacht wird.

Es gibt allerdings einen Fall, in dem es sinnvoll ist, sowohl Passwortschutz als auch Dateiverschlüsselung durchzuführen: wenn die Datei beim Zugriff nicht automatisch entschlüsselt wird. Ein automatisiertes Programm kann auf die verschlüsselte Datei zugreifen und sie, noch immer verschlüsselt, an eine Stelle setzen, auf die später zugegriffen werden kann. Der Schlüssel muss also nicht im Zugriffsprogramm gespeichert sein.

Siehe auch

Rezept 10.9 für Informationen zur Anwendung von HTTP-Basic-Authentifizierung; Rezept 17.10 behandelt SSL und Datenschutz beim Senden über das Netzwerk; die Dokumentationen zu `mcrypt_encrypt()` unter *http://www.php.net/mcrypt-encrypt* und `mcrypt_decrypt()` unter *http://www.php.net/mcrypt-decrypt*.

17.10 SSL ermitteln

Problem

Sie wollen wissen, ob eine Anfrage über SSL geschickt wurde.

Lösung

Prüfen Sie den Wert von `$_SERVER['HTTPS']`:

```
if ('on' == $_SERVER['HTTPS']) {
  print "Die Geheimzutat von Coca-Cola ist Soylent Green.";
} else {
  print "Coca-Cola enth&auml;lt viele schmackhafte nat&uuml;rliche und k&uuml;nstliche
                   Geschmacksstoffe.";
}
```

Diskussion

Das SSL-Protokoll arbeitet auf einer niedrigeren Ebene als HTTP. Der Webserver und ein Browser einigen sich auf eine geeignete, ihren Fähigkeiten entsprechende sichere Verbindung, damit das HTTP-Protokoll über diese sichere Verbindung ablaufen kann. Für einen Angreifer, der den Verkehr abfängt, ist dies nur eine Abfolge von Bytes, die keinen Sinn ergeben und nicht gelesen werden können.

Unterschiedliche Webserver haben unterschiedliche Anforderungen bei der Verwendung von SSL – konsultieren Sie also Ihre Serverdokumentation, wenn es um Details geht. Um mit SSL arbeiten zu können, sind keine Änderungen an PHP notwendig.

Zusätzlich zu Code-Alternativen, die je nach `$_SERVER['HTTPS']` angesprochen werden, können Sie auch Cookies so einstellen, dass sie nur über SSL-Verbindungen gesendet werden. Wenn das letzte Argument von `setcookie()` 1 ist, schickt der Browser das Cookie nur über eine sichere Verbindung zum Server zurück:

```
/* Ein Cookie setzen, das nur für SSL bestimmt ist, den Namen "nurssl" und den Wert "ja"
   hat und das am Ende der momentanen Browser-Session abläuft. */
setcookie('nurssl','ja','','/','sklar.com',1);
```

Obwohl der Browser diese Cookies nur über eine SSL-Verbindung zum Server zurücksendet, sendet der Server (wenn Sie `setcookie()` auf Ihrer Seite aufrufen) sie immer über

die Verbindung zum Browser, über die die Cookie-setzende Seite angefordert wurde – egal ob das SSL-Protokoll aktiv ist oder nicht. Wenn Sie vertrauliche Daten in das Cookie einfließen lassen wollen, müssen Sie sicherstellen, dass Sie das Cookie auch nur bei einer SSL-Anfrage setzen. Vergessen Sie auch nicht, dass die Cookie-Daten auf dem Rechner des Benutzers unverschlüsselt sind.

Siehe auch

Rezept 10.1 bespricht das Setzen von Cookies; die Dokumentation von `setcookie()` unter *http://www.php.net/setcookie*.

17.11 E-Mail mit GPG verschlüsseln

Problem

Sie wollen verschlüsselte E-Mails versenden. Sie erhalten beispielsweise Bestellungen über Ihre Webseite und müssen eine E-Mail an Ihre Fabrik senden, um dieser Einzelheiten der Bestellung zur weiteren Bearbeitung mitzuteilen. Indem Sie E-Mail-Nachrichten verschlüsseln, verhindern Sie, dass vertrauliche Daten wie z.B. Kreditkartennummern im Klartext über das Netzwerk laufen.

Lösung

Verschlüsseln Sie den Body der Nachricht mit GNU Privacy Guard (GPG), bevor Sie die E-Mail senden:

```
$msg_body = escapeshellarg($message_body);
$gpg_path     = '/usr/local/bin/gpg';
$sender       = 'web@example.com';
$recipient    = 'bestellungen@example.com';
$home_dir     = '/home/web';
$user_env     = 'web';

$cmd = "echo $message_body | HOME=$home_dir USER=$user_env $gpg_path " .
       '--quiet --no-secmem-warning --encrypt --sign --armor ' .
       "--recipient $recipient --local-user $sender";

$message_body = `$cmd`;

mail($recipient,'Web Site Order',$message_body);
```

Die E-Mail kann mit GPG oder Pretty Good Privacy (PGP) entschlüsselt werden oder mit einem E-Mail-Client-Plugin, das eines der beiden Programme unterstützt.

Diskussion

PGP ist ein beliebtes Public-Key-Verschlüsselungsprogramm; GPG ist ein Open Source Programm, das auf PGP aufbaut. Da PGP durch verschiedene Patent- und Kontrollprobleme beeinträchtigt ist, bietet es sich an, GPG zu verwenden.

Der Code in der Lösung ruft */usr/local/bin/gpg* auf, um die Nachricht in `$msg` zu verschlüsseln. Es verwendet den privaten Schlüssel des Absenders `$sender` und den öffentlichen Schlüssel des Empfängers `$recipient`. Das bedeutet, dass nur `$recipient` die E-Mail-Nachricht entschlüsseln kann. Wenn er das tut, weiß er, dass die Nachricht von `$sender` kam.

Das Setzen der Umgebungsvariablen `HOME` und USER `teilt` GPG mit, wo es nach seinem Schlüsselring suchen muss: *$HOME/.gnupg/secring.gpg*. Die Optionen `--quiet` und `--no-secmem-warning` unterdrücken Warnungen, die GPG ansonsten ausgeben würde. Die Optionen `--encrypt` und `--sign` weisen GPG an, dass die Nachricht sowohl verschlüsselt als auch unterschrieben werden muss. Die Verschlüsselung der Nachricht macht sie unlesbar, außer für den Empfänger. Die Unterschrift stellt zusätzliche Informationen zur Verfügung, sodass der Empfänger weiß, wer die Nachricht erstellt hat und wann sie erstellt wurde. Die Option `--armor` erstellt die Ausgabe als Text statt in binärer Form, sodass die verschlüsselte Nachricht zum E-Mailen geeignet ist.

Normalerweise sind private Schlüssel mit einer Passphrase geschützt. Wenn ein Angreifer einen privaten Schlüssel, der mit einer Passphrase geschützt ist, kopiert, kann er Nachrichten nicht mit dem privaten Schlüssel verschlüsseln, wenn er nicht außerdem die Passphrase kennt. In diesem Rezept wollen wir aber nicht, dass der private Schlüssel von `$sender` eine Passphrase hat. Wenn er eine hätte, könnte die Webseite keine E-Mails mit Neubestellungen senden, ohne dass die Passphrase jedes Mal von Hand eingetippt werden müsste. Wenn Sie die Passphrase in einer Datei speichern und diese GPG für jede Verschlüsselung zur Verfügung stellen, bietet dieses keine zusätzliche Sicherheit im Vergleich zur Verschlüsselung ohne Passphrase.

Der Nachteil der Benutzung eines Schlüssels ohne Passphrase ist, dass ein Angreifer, der Ihren Geheimschlüssel herausfindet, gefälschte E-Mail-Bestellungen an Ihre Fabrik senden kann. Das ist eine Gefahr, die sich kontrollieren lässt. Da Bestellungen sowieso über eine Webseite eingereicht werden können, besteht hier schon eine Stelle, durch die gefälschte Informationen in den Bestellprozess eingeschleust werden können. Prozeduren zum Abfangen von gefälschten Bestellungen können auch durch diese möglicherweise gefälschten E-Mails in Gang gebracht werden. Ist der Schlüsseldiebstahl einmal entdeckt und das Problem, das den Diebstahl ermöglicht hat, behoben, kann der Angreifer leicht kaltgestellt werden, indem man zu einem neuen privaten Schlüssel wechselt.

Siehe auch

Die GNU Privacy Guard-Homepage unter *http://www.gnupg.org/*.

17.12 Metazeichen der Shell mit Escape-Zeichen versehen

Problem

Sie müssen externe Daten in eine Kommandozeile einfügen, aber Sie wollen Sonderzeichen durch Escape-Zeichen entschärfen, sodass nichts Unvorhergesehenes passieren kann. Beispiel: Sie wollen Benutzereingaben als Kommandozeilenparameter an ein Programm übergeben.

Lösung

Verwenden Sie dagegen `escapeshellarg()`, wenn es sich um Kommandozeilenparameter handelt:

```
system('ls -al '.escapeshellarg($directory));
```

Nehmen Sie `escapeshellcmd()`, wenn es sich um Programmnamen handelt:

```
system(escapeshellcmd($ls_program).' -al');
```

Diskussion

Die Kommandozeile ist ein gefährlicher Platz für ungefilterte Zeichen. Geben Sie nie unmodifizierte Benutzereingaben an eine Funktion von PHP durch, die ihre Parameter an die Shell (Kommandozeile) Ihres Betriebssystems weiterreicht. Versehen Sie die entsprechenden Zeichen im Befehl und in den Parametern immer mit Escape-Zeichen. Dies ist unbedingt notwendig. Es ist unüblich, Kommandozeilen mit von Webformularen stammenden Daten auszuführen, und etwas, das wir nicht leichtfertig empfehlen. Allerdings ist es manchmal unumgänglich, ein externes Programm laufen zu lassen. Dafür ist das Escapen von Befehlen und Argumenten nützlich.

`escapeshellarg()` umgibt das Argument mit einfachen Anführungszeichen (und versieht eventuell existierende einfache Anführungszeichen mit einem vorangestellten Backslash \). Um z.B. den Prozess-Status eines bestimmten Prozesses zu drucken, schreiben Sie dies:

```
system('/bin/ps '.escapeshellarg($process_id));
```

Die Verwendung von `escapeshellarg()` stellt sicher, dass der richtige Prozess angezeigt wird, auch wenn er ein unerwartetes Zeichen enthält (zum Beispiel ein Leerzeichen). Es verhindert außerdem die ungewollte Ausführung zusätzlicher Befehle. Wenn `$process_id`

```
1; rm -rf /
```

enthält, zeigt

```
system("/bin/ps $process_id")
```

nämlich nicht nur den Status von Prozess 1 an, sondern führt auch den Befehl *rm -rf /* aus. Dagegen führt

```
system('/bin/ps '.escapeshellarg($process_id))
```

den Befehl */bin/ps 1\; rm -rf* aus, was einen Fehler ergibt, da »1-Semikolon-Leerzeichen-rm-Leerzeichen-Bindestrich-rf« keine gültige Prozessnummer ist.

Der Befehl `escapeshellcmd()` soll auf ähnliche Weise die Ausführung von unbeabsichtigten Befehlen verhindern. Das gelingt aber nur begrenzt – wenn Sie Ihr Kommando unbedingt von einer Variablen steuern lassen müssen, sollten Sie mit dem Verhalten dieser Funktion im Detail vertraut sein. Der folgende Code bringt in Abhängigkeit des Werts von `$which_program` unterschiedliche Programme zum Laufen:

```
system("/usr/local/bin/formatter-$which_program");
```

Wenn `$which_program` beispielsweise `pdf 12` ist, lässt das Skript */usr/local/bin/formatter-pdf* mit dem Kommandozeilenparameter `12` laufen. Ist `$which_program` jedoch `pdf 12; rm -rf /`, lässt das Skript zunächst */usr/local/bin/formatter-pdf* mit dem Kommandozeilenparameter `12` laufen, startet anschließend aber auch Programm *rm -rf /*, was Sie höchstwahrscheinlich nicht wollen. Um hier Probleme zu verhindern, können Sie die Variable an *formatter-pdf* geben, nachdem Sie diese mit `escapeshellcmd()` bearbeitet haben:

```
system(escapeshellcmd("/usr/local/bin/formatter-$which_program"));
```

Dieser Code lässt */usr/local/bin/formatter-pdf* laufen und übergibt ihm vier Kommandozeilenparameter: `12\;` und `rm` und `-rf` und `/`.

Beachten Sie aber, dass Ihnen ein Benutzer trotz Filterung mit `escapeshellcmd()` immer noch zusätzliche Kommandozeilenparameter einschmuggeln kann, wie im vorigen Beispiel gezeigt. Sowohl bei `escapeshellcmd()` als auch bei `escapeshellarg()` sollten Sie außerdem sicherstellen, dass Ihr externes Programm nicht durch gezielt konstruierte Kommandozeilenparameter (z.B. zur Erzielung eines Pufferüberlaufs) dazu gebracht werden kann, unbeabsichtigte Funktionen auszuführen.

Siehe auch

Die Dokumentationen zu `system()` unter *http://www.php.net/system*, zu `escapeshellarg()` unter *http://www.php.net/escapeshellarg* und zu `escapeshellcmd()` unter *http://www.php.net/escapeshellcmd*.

17.13 Session-Fixierung verhindern

Problem

Sie müssen sicherstellen, dass sich kein Dritter, ein Angreifer beispielsweise, der die Session des Benutzers kapern will, die Session-ID eines Benutzers verschaffen kann.

Lösung

Regenerieren Sie die Session-ID mit `session_regenerate_id( )`, wenn es eine Änderung der Berechtigungen des Benutzers gibt, beispielsweise nachdem er sich eingeloggt hat:

```
<?php
session_regenerate_id();
$_SESSION['logged_in'] = true;
?>
```

Diskussion

Mit Sessions können Sie Variablen erstellen, die über mehrere Anfragen erhalten bleiben. Damit Sessions funktionieren, muss jede Anfrage eines Benutzers eine Session-ID einschließen, die die Session eindeutig identifiziert.

Standardmäßig akzeptiert PHP Session-IDs, die über ein Cookie oder in der URL übergeben werden. Ein Angreifer kann sein Opfer dazu verleiten, über einen Link auf Ihre Anwendung zuzugreifen, in den eine Session-ID eingebettet ist:

```
<a href="http://example.org/login.php?PHPSESSID=1234">Hier klicken!</a>
```

Einem Benutzer, der diesem Link folgt, wird damit die Session untergeschoben, die über die ID `1234` identifiziert wird. Der Angreifer kennt jetzt also die Session-ID des Benutzers und kann versuchen, die Session des Benutzers zu kapern, indem er die gleiche Session-ID verwendet.

Wenn sich der Benutzer nicht einloggt und keine anderen Aktionen durchführt, die ihn von den anderen Benutzern der Anwendung unterscheidbar machen, bringt das dem Angreifer wenig. Indem Sie dafür sorgen, dass die Session-ID neu generiert wird, wenn sich die Berechtigungsstufe ändert, können Sie die Gefahr der Session-Fixierung also effektiv ausräumen. PHP kümmert sich darum, dass der Session-Speicher aktualisiert und die neue Session weitergegeben wird. Sie müssen also an entsprechender Stelle nur diese eine Funktion aufrufen.

Siehe auch

Rezept 10.5 bietet grundlegende Informationen zur Benutzung von Sessions.

17.14 Sich gegen Formular-Spoofing schützen

Problem

Sie brauchen die Sicherheit, dass eine Formularübermittlung gültig und beabsichtigt ist.

Lösung

Nutzen Sie ein verborgenes Formularfeld mit einem eindeutigen Token und speichern Sie dieses Token in der Session des Benutzers:

```
<?php
session_start();
$_SESSION['token'] = md5(uniqid(mt_rand(), true));
?>
<form action="buy.php" method="POST">
<input type="hidden" name="token" value="<?php echo $_SESSION['token']; ?>" />
<p>Stock Symbol: <input type="text" name="symbol" /></p>
<p>Quantity: <input type="text" name="quantity" /></p>
<p><input type="submit" value="Buy Stocks" /></p>
</form>
```

Prüfen Sie dann, wenn Sie eine Anfrage erhalten, die eine Formularübermittlung darstellt, ob die Token übereinstimmen:

```
<?php
session_start();
if ($_POST['token'] != $_SESSION['token'] ||
    !isset($_SESSION['token'])) {
    /* Benutzer zur Passworteingabe auffordern. */
} else {
    /* Fortfahren. */
}
?>
```

Diskussion

Diese Technik bietet Schutz vor einer Gruppe von Angriffen, die man als Cross-Site Request Forgery (CSRF) bezeichnet. Diese Angriffe sorgen alle dafür, dass ein Opfer eine Anfrage an eine Zielseite sendet, ohne dass es sich dessen bewusst ist. Üblicherweise hat das Opfer bereits bestimmte Berechtigungen für die Zielseite ausgehandelt. Das ermöglicht dem Angreifer, Aktionen auszuführen, die er ansonsten nicht durchführen könnte.

Fügen Sie Ihren Formularen auf diese Weise ein Token hinzu, verhindert das in keiner Weise, dass ein Benutzer Anfragen für sich selbst fälscht (etwas, das Sie ohnehin nicht verhindern können und worüber Sie sich auch keine Gedanken machen müssen). Wenn Sie Eingaben, wie in Rezept 17.15 beschrieben, filtern, zwingen Sie Anfragen, Ihre Regeln zu befolgen. Die in diesem Rezept vorgestellte Technik hilft Ihnen nur dabei, sicherzustellen, dass die Anfrage beabsichtigt ist.

17.15 Sicherstellen, dass Eingaben gefiltert werden

Problem

Es muss gesichert sein, dass alle Eingaben, die Sie erhalten, vor der Verwendung gefiltert werden.

Lösung

Initialisieren Sie ein leeres Array zur Speicherung der gefilterten Daten. Speichern Sie die Daten in diesem Array, nachdem Sie ihre Zulässigkeit überprüft haben:

```
<?php
/* Das Array für die gefilterten Daten initialisieren. */
$clean = array();
/* Nur alphabetische Namen zulassen. */
if (ctype_alpha($_POST['name'])) {
    $clean['name'] = $_POST['name'];
} else {
    /* Fehler */
}
?>
```

Diskussion

Wenn Sie strenge Namenskonventionen aufstellen, können Sie besser nachhalten, welche Eingaben bereits gefiltert wurden. Dass `$clean` immer mit einem leeren Array initialisiert wird, sichert, dass in das Array keine Daten injiziert werden können. Sie müssen sie ihm explizit hinzufügen.

Sofern Sie Techniken wie den Einsatz von `$clean` übernehmen, ist es wichtig, dass Sie in Ihrer Programmlogik nur mit den Daten in diesem Array arbeiten.

Siehe auch

Rezept 11.2 befasst sich ausführlich mit der Validierung unterschiedlicher Typen von Eingabedaten.

17.16 Cross-Site-Scripting verhindern

Problem

Sie müssen zuverlässig Cross-Site-Scripting-Angriffe (XSS) auf Ihre PHP-Anwendungen verhindern.

Lösung

Maskieren Sie alle HTML-Ausgaben mit `htmlentities( )` und achten Sie dabei darauf, dass Sie die richtige Zeichenkodierung angeben:

```
<?php
/* Beachten Sie die Angaben der Zeichenkodierung. */
header('Content-Type: text/html; charset=UTF-8');
/* Array für maskierte Daten initialisieren. */
$html = array();
/* Gefilterte Daten maskieren. */
$html['username'] = htmlentities($clean['username'], ENT_QUOTES, 'UTF-8');
echo "<p>Willkommen, {$html['username']}.</p>";
?>
```

Diskussion

Die Funktion `htmlentities( )` ersetzt alle Zeichen durch das entsprechende HTML-Entity, wenn es eins gibt. `>` wird beispielsweise durch `>` ersetzt. Obwohl die unmittelbare Konsequenz ist, dass die Daten geändert werden, ist der eigentliche Zweck des Maskierens, dass die Daten in einem anderen Kontext erhalten bleiben. Wenn ein Browser `>` als HTML darstellt, erscheint es auf dem Bildschirm als `>`.

XSS-Angriffe versuchen, Situationen auszunutzen, in denen Daten, die von Dritten angegeben werden, in HTML eingeschlossen werden, ohne dass sie zuvor ordentlich maskiert wurden. Ein gerissener Angreifer kann Code angeben, der für Ihre Benutzer sehr gefährlich werden kann, wenn er von ihren Browsern ausgeführt wird. Durch den Einsatz von `htmlentities( )` können Sie sicherstellen, dass Daten aus externen Quellen korrekt angezeigt und nicht interpretiert werden.

Siehe auch

Rezept 11.8 erörtert das Verhindern von Cross-Site-Scripting in Bezug auf übermittelte Formulardaten.

17.17 SQL-Injection verhindern

Problem

Sie müssen die Gefährdung Ihrer PHP-Anwendung durch SQL-Injection-Angriffe ausräumen.

Lösung

Nutzen Sie eine Datenbankbibliothek wie PDO, die die für Ihre Datenbank erforderlichen Maskierungsvorgänge übernimmt:

```
<?php
$db = new PDO('mysql:host=localhost;dbname=users',
              $_SERVER['DB_USER'],
              $_SERVER['DB_PASSWORD']);
$statement = $db->prepare("INSERT
                           INTO   users (username, password)
                           VALUES (:username, :password)");
$statement->bindParam(':username', $clean['username']);
$statement->bindParam(':password', $clean['password']);
$statement->execute();
$db = NULL;
?>
```

Diskussion

Der Einsatz gebundener Parameter sichert, dass Ihre Daten nie in einen Kontext gelangen, in dem sie als etwas anderes betrachtet werden als reine Daten, und verhindert so, dass ein Wert jemals die Form einer SQL-Abfrage beeinflussen kann.

Können Sie nicht auf PDO zurückgreifen, haben Sie die Möglichkeit, eine in PHP geschriebene Datenbankbibliothek wie PEAR::DB einzusetzen, die ähnliche Features bietet:

```
<?php
$st = $db->query('INSERT
            INTO   users (username, password)
            VALUES (?, ?)',
            array($clean['username'], $clean['password']));
?>
```

Obwohl bei diesem Verfahren Ihre Daten immer noch mit der SQL-Abfrage gemischt werden, sichert PEAR::DB, dass die Daten sauber maskiert werden und praktisch keine SQL-Injection-Gefahr mehr besteht.

Siehe auch

Kapitel 12 liefert weitere Informationen zu PDO, insbesondere Rezept 12.5 und Rezept 12.8; die Dokumentationen zu PDO unter *http://www.php.net/pdo* und zu PEAR::DB unter *http://pear.php.net/manual/en/package.database.db.php*.

KAPITEL 18

Grafik

18.0 Einführung

Mit Hilfe der GD-Bibliothek können Sie mit PHP Anwendungen erstellen, die über dynamisch erzeugte Bilder Börsenwerte anzeigen, Wahlergebnisse veröffentlichen oder die Leistung eines Systems überwachen. Sie können sogar Spiele erfinden. Allerdings funktioniert es nicht so wie in Photoshop oder GIMP; Sie können nicht einfach durch Hin- und Herbewegen der Maus eine Linie ziehen. Stattdessen müssen Sie die Art, Größe und Position eines Grafikobjekts genau spezifizieren.

GD besitzt bereits eine Schnittstelle für Anwendungsprogrammierer. PHP versucht, deren Konventionen in Sachen Syntax und Funktionsbenennung zu folgen. Wenn Sie GD also schon aus anderen Sprachen kennen, z.B. von C oder Perl, können Sie GD mit Leichtigkeit unter PHP verwenden. Wenn GD für Sie neu ist, brauchen Sie vielleicht ein paar Minuten, um durchzusteigen, aber dann zeichnen Sie wie Picasso.

Der Funktionssatz von GD hängt stark davon ab, welche GD-Version Sie benutzen und welche Funktionen während der Konfiguration aktiviert wurden. Die GD-Versionen bis einschließlich 1.6 unterstützten das Lesen und Schreiben von GIFs, dieser Code wurde aber auf Grund von Patentproblemen entfernt und erst in 2.0.28 wieder hinzugefügt. Wenn Sie Ihre GD-Bibliothek nicht (wie z.B. in der Windows-Distribution von PHP) zusammen mit PHP 5 installiert haben, sollten Sie sie auf den neuesten Stand bringen (lassen). Außerdem unterstützen neuere GD-Versionen JPEGs, PNGs und WBMPs. Da PNGs im Allgemeinen kleiner als GIFs sind, Ihnen viel mehr Farben und eine eingebaute Gamma-Korrektur zur Verfügung stehen und sie von allen großen Webbrowsern unterstützt werden, wird das Fehlen der GIF-Unterstützung als ein Feature gesehen, nicht als ein Bug. Wenn Sie mehr über PNG wissen wollen, schauen Sie unter *http://www.libpng.org/pub/png/* nach oder lesen Sie Kapitel 31, »PNG«, in *Web Design in a Nutshell* von Jennifer Niederst (O'Reilly).

GD unterstützt nicht nur unterschiedliche Dateiformate, sondern lässt Sie auch Punkte, Linien, Vierecke, Vielecke, Bogen, Ellipsen und Kreise in jeder beliebigen Farbe zeichnen. Rezept 18.1 behandelt gerade Grafikobjekte und Rezept 18.2 die runden. Wie Sie Grafikobjekte mit einem Muster anstatt einer vollen Farbe füllen, sehen Sie in Rezept 18.3.

Außerdem können Sie Texte in unterschiedlichen Schriftarten zeichnen, darunter eingebaute Schriftarten, TrueType- und PostScript-Type-1-Fonts. Rezept 18.4 zeigt Ihnen alle Details der drei textzeichnenden Hauptfunktionen, und Rezept 18.5 beschreibt, wie Sie einen Text in einem ersten Entwurf zentrieren. Diese zwei Rezepte bilden die Grundlage für Rezept 18.6, das eine Bildvorlage mit Echtzeitdaten verbindet, um dynamische Bilder zu erstellen. Mit GD können Sie auch transparente GIFs und PNGs erstellen. Das Setzen einer transparenten Farbe und das Benutzen von Transparenzen in Mustern werden in Rezept 18.7 behandelt.

Rezept 18.8 zeigt Ihnen, wie Sie heraufgeladene hochauflösende Bilder von Digitalkameras auf Webniveau verkleinern, damit der Download nicht ewig dauert.

Rezept 18.9 verlässt das Thema GD und zeigt, wie Sie Bilder sicher zugänglich machen können, indem Sie den Benutzerzugang begrenzen. Schließlich stellen wir eine Beispielanwendung vor – wir nehmen Umfrageergebnisse und erstellen ein dynamisches Säulendiagramm, das für jede Antwort den Prozentsatz der Benutzer angibt, der diese Antwort gewählt hat.

All diese Funktionen arbeiten unter GD 2.0.33, die im Moment aktuellste stabile Version der Bibliothek. Falls Sie eine ältere Version haben, sollte das keine Probleme bereiten. Benötigt ein bestimmtes Rezept eine spezielle GD-Version, geben wir das im Rezept an.

GD kann von der offiziellen GD-Seite unter *http://www.boutell.com/gd/* heruntergeladen werden. Außerdem führt der GD-Teil des Online-PHP-Handbuchs unter *http://www.php.net/image* die Quelle der zusätzlich notwendigen Bibliotheken auf, mit denen JPEGs und Type-1-Schriftarten unterstützt werden.

Über zwei einfache Wege können Sie herausfinden, ob und, wenn ja, welche GD-Version auf Ihrem Server installiert und wie sie konfiguriert ist. Eine Möglichkeit ist, `phpinfo()` aufzurufen. Oben, unter »Configure Command«, sollten Sie `--with-gd` sehen, weiter unten auf der Seite ist ein Abschnitt mit der Überschrift »gd«, der mehr Informationen über die installierte GD-Version und die aktivierten Funktionen enthält. Die andere Möglichkeit besteht darin, den Rückgabewert von `function_exists('imagecreate')` zu prüfen. Wenn `true` zurückgegeben wird, ist GD installiert. Die Funktion `imagetypes()` gibt ein Bit-Feld aus, das die vorhandenen Grafikformate angibt (siehe *http://www.php.net/imagetypes* zur Verwendung dieser Funktion). Wenn Sie eine Funktion benutzen wollen, die nicht aktiviert ist, müssen Sie PHP selbst neu kompilieren und linken oder Ihren Internetprovider darum bitten.

Der grundlegende Ablauf beim Erstellen eines Bilds besteht aus drei Schritten: die Bildfläche erstellen, der Bildfläche Grafiken und Text hinzufügen und das Bild anzeigen oder abspeichern. Ein Beispiel:

```
$image = ImageCreate(200, 50);
$background_color = ImageColorAllocate($image, 255, 255, 255); // weiß
$gray  = ImageColorAllocate($image, 204, 204, 204); // grau

ImageFilledRectangle($image, 50, 10, 150, 40, $gray);

header('Content-type: image/png');
ImagePNG($image);
```

Die Code-Ausgabe, die ein graues Rechteck auf einem weißen Hintergrund druckt, ist in Abbildung 18-1 dargestellt.

Abbildung 18-1: Ein graues Rechteck auf weißem Hintergrund

Zuerst erstellen Sie eine Bildfläche, den so genannten *Canvas* (Leinwand). Die Funktion `ImageCreate( )` erstellt kein wirkliches Bild. Stattdessen liefert sie Ihnen ein Handle für die Bildfläche einer Grafik – eine Art leere Leinwand sozusagen. Erst wenn Sie PHP speziell aufforden, das Bild auszugeben, wird sie zu einer echten Grafik. Einen einmal mit `ImageCreate( )` erzeugten Canvas können Sie zum Erstellen mehrerer Grafiken nutzen, indem sie ihn modifizieren und erneut ausgeben.

Die Parameter, die Sie `ImageCreate( )` übergeben, stellen die Höhe und Breite der Grafik in Bildpunkten dar. In diesem Fall ist das Bild 200 Pixel breit und 50 Pixel hoch. Neben der Neuerstellung von Bildern können Sie auch bestehende Bilder bearbeiten. Um eine bestehende Grafik zu öffnen, rufen Sie `ImageCreateFromPNG( )` oder eine der ähnlich benannten Funktionen auf, um ein anderes Dateiformat zu öffnen. Der Dateiname ist das einzige Argument, und die Dateien können lokal oder auf anderen Servern gespeichert sein:

```
// Eine PNG-Grafik auf der lokalen Maschine öffnen.
$graph = ImageCreateFromPNG('/path/to/graph.png');

// Ein JPEG von einem anderen Server öffnen.
$icon  = ImageCreateFromJPEG('http://www.example.com/images/icon.jpeg');
```

Sobald Sie eine editierbare Bildfläche haben, können Sie auf Zeichenfarben zugreifen, indem Sie `ImageColorAllocate( )` aufrufen:

```
$background_color = ImageColorAllocate($image, 255, 255, 255); // weiß
$gray             = ImageColorAllocate($image, 204, 204, 204); // grau
```

Der Funktion `ImageColorAllocate( )` wird ein Bild-Handle für das Bild übergeben, dem die Farbe zugeordnet werden soll, zusammen mit drei ganzen Zahlen. Jede dieser Zahlen liegt zwischen 0 und 255. Sie geben jeweils die rote, grüne und blaue Komponente der Farbe an. Dabei handelt es sich um die gleiche RGB-Farbkombination, die unter HTML

verwendet wird, um eine Schrift- oder Hintergrundfarbe auszuwählen. Damit ist Weiß 255, 255, 255, Schwarz 0, 0, 0, und alles andere liegt irgendwo dazwischen.

Der erste Aufruf von `ImageAllocateColor( )` legt die Hintergrundfarbe fest. Weitere Aufrufe weisen Farben zum Zeichnen von Linien, Formen oder Text zu. Setzen Sie also die Hintergrundfarbe auf 255, 255, 255 und nehmen Sie dann einen grauen Stift mit `ImageAllocateColor($image, 204, 204, 204)`. Es kommt Ihnen vielleicht komisch vor, dass die Hintergrundfarbe durch die Abfolge bestimmt wird, in der `ImageAllocateColor()` aufgerufen wird, anstatt über eine zusätzliche Funktion. Aber so arbeitet GD eben, und PHP respektiert diese Gepflogenheit.

Mit einem Aufruf von `ImageFilledRectangle( )` setzen Sie einen Rahmen auf die Leinwand. `ImageFilledRectangle( )` arbeitet mit vielen Parametern: das Bild, auf dem die Zeichnung entstehen soll, die x- und y-Koordinaten der linken oberen Ecke des Rechtecks, die x- und y-Koordinaten der rechten unteren Ecke des Rechtecks und schließlich die Farbe, die zum Zeichnen benutzt werden soll. Fordern Sie `ImageFilledRectangle( )` auf, ein Rechteck auf `$image` zu zeichnen, das bei (50,10) beginnt und bis (150,40) geht, und die Farbe Grau benutzt:

```
ImageFilledRectangle($image, 50, 10, 150, 40, $gray);
```

Im Unterschied zum Kartesischen System ist (0,0) nicht die linke untere, sondern die linke obere Ecke. Entsprechend ist die vertikale Koordinate, die 10 Bildpunkte unterhalb des oberen Rands einer 50 Bildpunkte hohen Bildfläche liegt, 10, weil sie 10 Pixel unter dem oberen Rand der Bildfläche liegt. Sie ist nicht 40, da ja von oben aus gesehen gemessen wird. Ebenso ist sie nicht –10, da die Richtung von oben nach unten als die positive Richtung angesehen wird und nicht als die negative Richtung.

Nachdem das Bild jetzt so weit fertig ist, können Sie es über den Server senden. Als Erstes senden Sie einen `Content-type`-Header, um dem Browser den Bildtyp mitzuteilen, den Sie senden werden. In diesem Fall arbeiten Sie mit PNG. Als Nächstes fordern Sie PHP auf, das PNG-Bild mit `ImagePNG( )` auszugeben. Wenn das Bild gesendet ist, haben Sie Ihre Arbeit erledigt:

```
header('Content-Type: image/png');
ImagePNG($image);
```

Um das Bild auf die Festplatte zu schreiben anstatt es an den Browser zu senden, fügen Sie ein zweites Argument in `ImagePNG( )` ein, das den Pfad zum Abspeichern angibt:

```
ImagePng($image, '/path/to/your/new/image.png');
```

Da die Datei nicht zum Browser geht, braucht `header( )` nicht aufgerufen zu werden. Stellen Sie sicher, dass Sie einen Pfad und den Bildnamen angeben und dass es PHP erlaubt ist, dorthin zu schreiben.

Nach Beendigung des Skripts beseitigt PHP das Bild im Speicher. Wenn Sie den Speicher, der für das Bild verwendet wird, von Hand löschen, können Sie `ImageDestroy($image)` aufrufen, das PHP dazu zwingt, das Bild sofort zu zerstören.

18.1 Linien, Rechtecke und Vielecke zeichnen

Problem

Sie wollen eine Linie, ein Rechteck oder ein Vieleck zeichnen. Sie wollen außerdem bestimmen können, ob das Recht- oder Vieleck offen oder gefüllt sein soll. Beispiel: Sie wollen Säulendiagramme oder Kurven mit Börsenwerten erstellen.

Lösung

Mit `ImageLine( )` können Sie eine Linie zeichnen:

```
ImageLine($image, $x1, $y1, $x2, $y2, $color);
```

Mit `ImageRectangle( )` können Sie ein leeres Rechteck zeichnen:

```
ImageRectangle($image, $x1, $y1, $x2, $y2, $color);
```

Mit `ImageFilledRectangle( )` zeichnen Sie ein gefülltes Rechteck:

```
ImageFilledRectangle($image, $x1, $y1, $x2, $y2, $color);
```

Ein leeres Vieleck zeichnen Sie mit `ImagePolygon( )`:

```
$points = array($x1, $y1, $x2, $y2, $x3, $y3);
ImagePolygon($image, $points, count($points)/2, $color);
```

Ein gefülltes Vieleck erzeugen Sie mit `ImageFilledPolygon( )`:

```
$points = array($x1, $y1, $x2, $y2, $x3, $y3);
ImageFilledPolygon($image, $points, count($points)/2, $color);
```

Diskussion

Die Prototypen für alle fünf Funktionen in der Lösung ähneln sich. Der erste Parameter gibt die Bildfläche an, auf der gezeichnet werden soll. Der nächste Parametersatz gibt die x- und y-Koordinaten an und legt damit fest, wo GD die Form zeichnen soll. Bei `ImageLine( )` geben die vier Koordinaten die Endpunkte der Linie an. Bei `ImageRectangle( )` geben Sie die gegenüberliegenden Ecken des Rechtecks an. Im Beispiel erstellt `ImageLine($image, 0, 0, 100, 100, $color)` eine diagonale Linie. Werden die gleichen Parameter in `ImageRectangle( )` aufgerufen, wird ein Rechteck mit den Ecken bei (0,0), (100,0), (0,100) und (100,100) erstellt. Beide Formen sind in Abbildung 18-2 dargestellt.

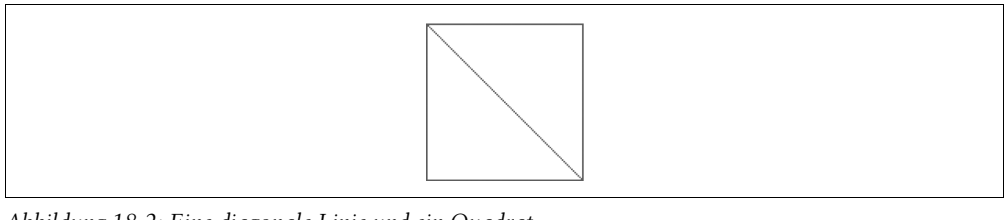


Abbildung 18-2: Eine diagonale Linie und ein Quadrat

Die Funktion ImagePolygon( ) ist etwas anders, da sie eine unbestimmte Anzahl von Eckpunkten annimmt. Deshalb ist der zweite Parameter ein Array von x- und y-Koordinaten. Die Funktion beginnt am ersten angegebenen Punkt und zeichnet Linien von Eckpunkt zu Eckpunkt, bis die Form schließlich durch Verknüpfung mit dem Ausgangspunkt geschlossen wird. Sie müssen mindestens drei Eckpunkte in Ihrem Vieleck haben (was insgesamt sechs Elementen im Array entspricht). Der dritte Parameter gibt die Anzahl der Eckpunkte des Vielecks an. Da das immer die halbe Anzahl der Elemente im Array mit den Eckpunkt-Koordinaten ist, wird dafür ein flexibler Wert count($points) / 2 verwendet, der Ihnen ermöglicht, das Array der Scheitelpunkte zu modifizieren, ohne den Aufruf von ImageLine() zu ändern.

Zum Schluss haben alle Funktionen einen letzten Parameter, der die zum Zeichnen verwendete Farbe angibt. Dies ist üblicherweise ein Wert, der von ImageColorAllocate( ) zurückgegeben wird. Es kann aber auch eine Konstante IMG_COLOR_STYLED oder IMG_COLOR_STYLEDBRUSHED sein, wenn Sie unterbrochene Linien zeichnen wollen, wie in Rezept 18.3 beschrieben.

Alle diese Funktionen zeichnen leere Formen. Um GD aufzufordern, auch die Innenfläche mit der Zeichenfarbe auszufüllen, können Sie ImageFilledRectangle( ) und ImageFilled-Polygon( ) mit genau dem gleichen Argumentensatz wie ihre unausgefüllten Verwandten verwenden.

Siehe auch

Rezept 18.2 zum Zeichnen von anderen Formen; Rezept 18.3 zum weiterführenden Zeichnen mit Styles und Brushes; die Dokumentationen zu ImageLine( ) unter *http://www.php.net/imageline*, ImageRectangle( ) unter *http://www.php.net/imagerectangle*, ImagePolygon( ) unter *http://www.php.net/imagepolygon* und ImageColorAllocate( ) unter *http://www.php.net/imagecolorallocate*.

18.2 Bogen, Ellipsen und Kreise zeichnen

Problem

Sie wollen leere oder gefüllte Kurven zeichnen. Beispielsweise wollen Sie ein Kuchendiagramm zeichnen, das das Ergebnis einer Benutzerumfrage anzeigt.

Lösung

Verwenden Sie ImageArc( ), um einen Bogen zu zeichnen:

```
ImageArc($image, $x, $y, $width, $height, $start, $end, $color);
```

Um eine Ellipse zu zeichnen, verwenden Sie ImageArc( ) und setzen $start auf 0 und $end auf 360:

```
ImageArc($image, $x, $y, $width, $height, 0, 360, $color);
```

Um einen Kreis zu zeichnen, verwenden Sie `ImageArc( )`, wobei Sie `$start` auf `0` und `$end` auf `360` setzen und den gleichen Wert für `$width` und `$height` verwenden:

```
ImageArc($image, $x, $y, $diameter, $diameter, 0, 360, $color);
```

Diskussion

Da die Funktion `ImageArc( )` sehr flexibel ist, können Sie ganz leicht übliche Kurven (wie z.B. Ellipsen und Kreise) zeichnen, indem Sie ihnen die richtigen Werte übergeben. Wie bei vielen GD-Funktionen stellt der erste Parameter die Bildfläche dar. Die nächsten beiden Parameter sind die x- und y-Koordinaten des Bogenmittelpunkts, d.h. des Mittelpunkts der Ellipse und oder des Kreises. Dann folgen Bogenbreite und -höhe. Da ein Kreis ein Bogen ist, der die gleiche Höhe wie Breite hat, setzen Sie diese beide Parameter auf den Kreisdurchmesser, wenn Sie einen Kreis zeichnen wollen.

Der sechste und siebte Parameter sind die Start- und Endwinkel in Grad. Der Wert 0 zeigt dabei auf 3 Uhr. Der Bogen wird im Uhrzeigersinn gezeichnet, so dass 90 auf 6 Uhr zeigt, 180 auf 9 Uhr und 270 auf 12 Uhr. (Hier sollten Sie vorsichtig sein, da diese Rechenart nicht für alle GD-Funktionen gilt. Wenn Sie z.B. Text rotieren lassen, drehen Sie ihn gegen den Uhrzeigersinn.) Da sich der Bogenmittelpunkt bei `($x,$y)` befindet und Sie einen Halbkreis von 0 nach 180 zeichnen, beginnt der Bogen nicht bei `($x,$y)`, sondern bei `($x+($diameter/2),$y)`.

Wie gewöhnlich stellt der letzte Parameter die Bogenfarbe dar.

Das folgende Beispiel zeichnet einen leeren schwarzen Kreis mit einem Durchmesser von 100 Bildpunkten, zentriert auf der Bildfläche, wie die linke Hälfte von Abbildung 18-3 zeigt:

```
$image = ImageCreate(100,100);
$bg = ImageColorAllocate($image, 255, 255, 255);
$black = ImageColorAllocate($image, 0, 0, 0);
ImageArc($image, 50, 50, 100, 100, 0, 360, $black);
```

Um mit Farbe gefüllte Ellipsen oder Kreise zu zeichnen, rufen Sie `ImageFillToBorder( )` auf:

```
ImageArc($image, $x, $y, $diameter, $diameter, 0, 360, $color);
ImageFillToBorder($image, $x, $y, $color, $color);
```

Die Funktion `ImageFillToBorder( )` füllt, beginnend bei `($x,$y)`, eine Bildregion mit der im letzten Parameter angegebenen Farbe aus, bis der Rand der Bildfläche oder eine Linie mit der Farbe des vierten Parameters erreicht wird.

Beim Einbau in das obere Beispiel erhalten Sie:

```
$image = ImageCreate(100,100);
$bg = ImageColorAllocate($image, 255, 255, 255);
$black = ImageColorAllocate($image, 0, 0, 0);
ImageArc($image, 50, 50, 100, 100, 0, 360, $black);
ImageFillToBorder($image, 50, 50, $black, $black);
```

Das Ergebnis wird in der rechten Hälfte von Abbildung 18-3 ausgegeben.

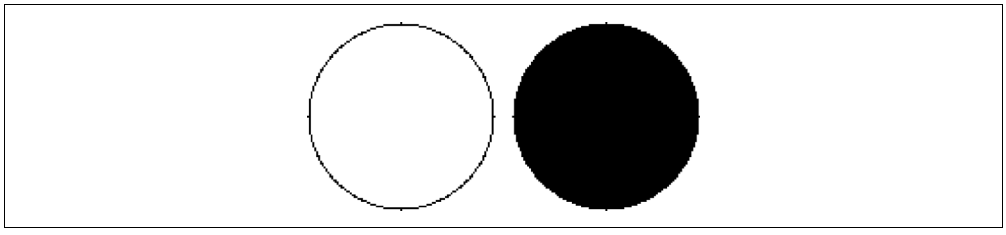


Abbildung 18-3: Ein offener schwarzer Kreis und ein gefüllter schwarzer Kreis

Wenn Sie mit GD 2.x arbeiten, können auch Sie `ImageFilledArc( )` aufrufen und einen letzten Parameter eingeben, der den Füllstil beschreibt. GD 2.x unterstützt auch spezielle Funktionen für `ImageEllipse( )` und `ImageFilledEllipse( )`.

Siehe auch

Rezept 18.1 mit mehr Informationen über das Zeichnen von anderen Formtypen; Rezept 18.3 mit weiterführenden Informationen über das Zeichnen mit Styles und Brushes; die Dokumentationen zu `ImageArc( )` unter *http://www.php.net/imagearc*, `ImageFilledArc( )` unter *http://www.php.net/imagefilledarc* und `ImageFillToBorder( )` unter *http://www.php.net/imagefilltoborder*.

18.3 Unterbrochene Linien zeichnen

Problem

Sie wollen Formen in anderen Linienarten zeichnen als mit der voreingestellten durchgezogenen Linie.

Lösung

Um Formen mit unterbrochenen Linien zu zeichnen, verwenden Sie `ImageSetStyle( )` und geben `IMG_COLOR_STYLED` als Bildfarbe an:

```
$black = ImageColorAllocate($image,   0,   0,   0);
$white = ImageColorAllocate($image, 255, 255, 255);

// Zeichne eine 2 Pixel dicke, schwarzweiß unterbrochene Linie.
$style = array($black, $black, $white, $white);
ImageSetStyle($image, $style);

ImageLine($image, 0, 0, 50, 50, IMG_COLOR_STYLED);
ImageFilledRectangle($image, 50, 50, 100, 100, IMG_COLOR_STYLED);
```

Diskussion

Die Linienart wird durch ein Array von Farben bestimmt. Jedes Element im Array stellt ein Bildpunkt im »Pinsel«, dem so genannten *Brush*, dar. Es ist oft hilfreich, die gleiche Farbe in aufeinander folgenden Elementen zu wiederholen, da dadurch die Streifen im Muster vergrößert werden können.

Zum Beispiel zeigt der Code hier ein Quadrat mit abwechselnd weißen und schwarzen Pixeln, wie im linken Teil von Abbildung 18-4 zu sehen:

```
$style = array($white, $black);
ImageSetStyle($image, $style);
ImageFilledRectangle($image, 0, 0, 49, 49, IMG_COLOR_STYLED);
```

Dies ist das gleiche Quadrat, aber mit fünf weißen Bildpunkten gefolgt von fünf schwarzen, wie im mittleren Teil von Abbildung 18-4 gezeigt:

```
$style = array($white, $white, $white, $white, $white,
               $black, $black, $black, $black, $black);
ImageSetStyle($image, $style);
ImageFilledRectangle($image, 0, 0, 49, 49, IMG_COLOR_STYLED);
```

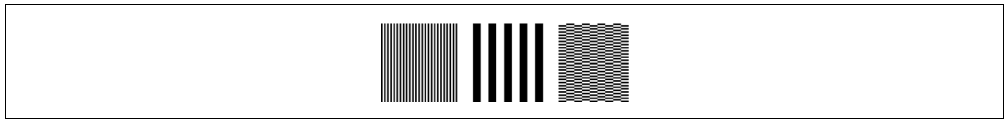


Abbildung 18-4: Drei Quadrate mit abwechselnd weißen und schwarzen Bildpunkten

Die Muster sehen komplett anders aus, obwohl beide Arten nur aus weißen und schwarzen Bildpunkten bestehen.

Wenn die Formbreite nicht eine ganzzahlige Anzahl von Brush-Kopien aufnehmen kann, wird in der nächsten Pixelzeile der Form mit dem Rest des letzten Brushs in der vorhergehenden Zeile weitergepinselt. Im vorigen Beispiel ist das Quadrat 50 Bildpunkte breit. Da der erste Pinsel 2 Pixel lang ist, passt er genau 25-mal hinein; der zweite Pinsel hat 10 Bildpunkte, also passt er 5-mal hinein. Wenn Sie das Quadrat aber 45 mal 45 groß machen und den zweiten Pinsel verwenden, erhalten Sie im Gegensatz zu obigem Beispiel keine geraden Linien. Das ist im rechten Teil von Abbildung 18-4 dargestellt:

```
ImageFilledRectangle($image, 0, 0, 44, 44, IMG_COLOR_STYLED);
```

Siehe auch

Die Rezepte 18.1 und 18.2 mit Informationen über das Zeichnen von Formen; die Dokumentation zu `ImageSetStyle( )` unter *http://www.php.net/imagesetstyle*.

18.4 Text zeichnen

Problem

Sie wollen Text als Grafik zeichnen. Damit können Sie dynamische Buttons oder Hit-Zähler erstellen.

Lösung

Für eingebaute GD-Schriftarten verwenden Sie `ImageString( )`:

```
ImageString($image, 1, $x, $y, 'I love PHP Cookbook', $text_color);
```

Für TrueType-Schriftarten verwenden Sie `ImageTTFText( )`:

```
ImageTTFText($image, $size, 0, $x, $y, $text_color, '/pfad/zu/font.ttf',
             'I love PHP Cookbook');
```

Für PostScript-Type-1-Schriftarten verwenden Sie `ImagePSLoadFont( )` und `ImagePSText( )`

```
$font = ImagePSLoadFont('/pfad/zu/font.pfb');
ImagePSText($image, 'I love PHP Cookbook', $font, $size,
            $text_color, $background_color, $x, $y);
```

Diskussion

Um Text auf der Bildfläche zu positionieren, rufen Sie `ImageString( )` auf. `ImageString( )` braucht, wie andere GD-Zeichenfunktionen auch, viele Eingaben: die Bildfläche, auf der gezeichnet wird, die Nummer der Schriftart, die x- und y-Koordinaten der linken oberen Ecke des ersten Buchstabens, den anzuzeigenden Text-String und schließlich die Farbe, mit der der Text gezeichnet wird.

Bei `ImageString( )` haben Sie die Auswahl zwischen fünf Schriftarten von 1 bis 5. Wie Abbildung 18-5 zeigt, ist Nummer 1 die kleinste und Nummer 5 die größte Schriftart. Bei allen Eingaben, die unter oder über diesem Bereich liegen, wird die Größe verwendet, die einer zulässigen Nummer am nächsten kommt.

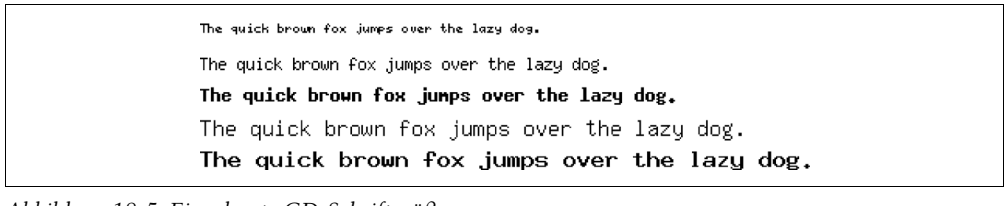


Abbildung 18-5: Eingebaute GD-Schriftgrößen

Zum vertikalen anstatt horizontalen Zeichnen von Text verwenden Sie `ImageStringUp( )` statt `ImageString( )`. Abbildung 18-6 zeigt die Ausgabe.

```
ImageStringUp($image, 1, $x, $y, 'I love PHP Cookbook', $text_color);
```

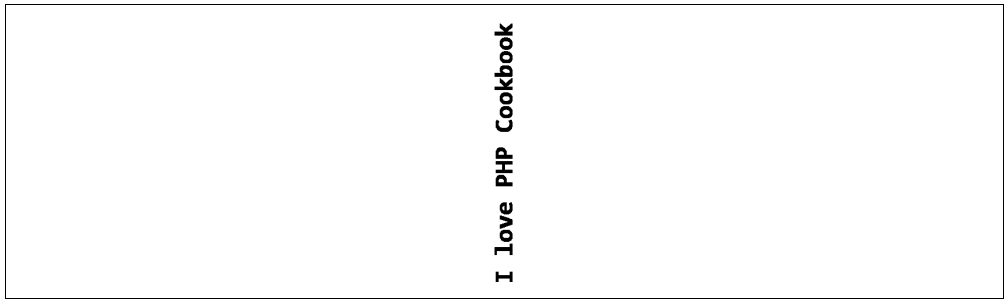


Abbildung 18-6: Vertikaler Text

Um TrueType-Schriftarten zu verwenden, müssen Sie auch die FreeType-Bibliothek installieren und PHP während der Installation so konfigurieren, dass FreeType verwendet werden kann. Die Homepage von FreeType finden Sie unter *http://www.freetype.org*. Um die Unterstützung für FreeType 1.x zu aktivieren, verwenden Sie `--with-ttf`, und für FreeType 2.x geben Sie `--with-freetype-dir=DIR` an.

Wie `ImageString( )` druckt auch `ImageTTFText( )` einen String auf eine Bildfläche, braucht dazu aber etwas andere Parameter und benötigt diese auch in einer anderen Reihenfolge:

```
ImageTTFText($image, $size, $angle, $x, $y, $text_color, '/pfad/zu/font.ttf',
             $text);
```

Das Argument `$size` ist die Schriftgröße in Pixeln; `$angle` ist ein Rotationswinkel in Grad gegen den Uhrzeigersinn, und */pfad/zu/font.ttf* ist der Pfadname der TrueType-Font-Datei. Anders als bei `ImageString( )` sind `($x,$y)` die Koordinaten links unten an der Grundlinie des ersten Buchstabens (die Grundlinie ist der untere Rand der meisten Buchstaben. Buchstaben wie z.B. »g« und »j« reichen unter die Grundlinie; »a« und »z« sitzen auf der Grundlinie).

Für PostScript-Type-1-Schriftarten müssen Sie *t1lib* installiert haben. Sie können diese Bibliothek von *ftp://sunsite.unc.edu/pub/Linux/libs/graphics/* herunterladen und mit `--with-t1lib` in PHP einbauen.

Die Syntax zur Textausgabe ist hier wieder einmal ähnlich, aber nicht genau gleich:

```
$font = ImagePSLoadFont('/pfad/zu/font.pfb');
ImagePSText($image, $text, $font, $size, $text_color, $background_color, $x, $y);
ImagePSFreeFont($font);
```

Zunächst einmal lassen sich PostScript-Schriftartnamen nicht direkt in `ImagePSText( )` eingeben. Stattdessen müssen sie mit `ImagePSLoadFont( )` geladen werden. Bei Erfolg gibt die Funktion eine Font-Ressource zurück, die mit `ImagePSText( )` verwendet werden kann. Neben der Festlegung einer Textfarbe legen Sie zusätzlich eine Hintergrundfarbe fest, die bei der Antialiasing-Rechnung verwendet wird (Vermeidung des »Lattenzauneffekts«). Die Positionierung von `($x,$y)` erfolgt wie bei der TrueType-Bibliothek. Wenn Sie Ihre Arbeit mit der Schriftart beendet haben, können Sie sie aus dem Speicher werfen, indem Sie `ImagePSFreeFont( )` aufrufen.

Neben den oben angeführten obligatorischen Argumenten nimmt `ImagePSText()` auch vier optionale an, und zwar in der folgenden Reihenfolge: Leerzeichengröße, Zeichenabstand, Winkel und Antialias-Schritte. Sie müssen entweder alle oder dürfen keinen der vier Parameter verwenden (d.h., Sie können nicht nur ein, zwei oder drei der Argumente festlegen). Das erste Argument legt die Größe eines Leerzeichens (also des Zeichens, das ausgegeben wird, wenn die Leerzeichentaste gedrückt wird) fest; das zweite beschreibt den Abstand zwischen den Buchstaben; das dritte ist der Rotationswinkel in Grad gegen den Uhrzeigersinn; und das letzte ist ein Antialiasing-Wert. Diese Zahl muss entweder 4 oder 16 sein. Wenn Sie ein besseres Aussehen erreichen möchten und rechnerisch aufwendigere Bilder nicht scheuen, sollten Sie 16 anstatt 4 verwenden.

In der Voreinstellung sind Leerschritt, Abstand und Winkel 0. Eine positive Zahl bewirkt einen größeren Leerschritt zwischen Wörtern und Buchstaben oder dreht die Grafik gegen den Uhrzeigersinn. Eine negative Zahl unterschneidet Wörter und Buchstaben oder dreht sie in die Gegenrichtung. Das folgende Beispiel zeigt die Ausgabe in Abbildung 18-7:

```
// normales Bild
ImagePSText($image, $text, $font, $size, $black, $white, $x, $y,
            0, 0, 0, 4);

// zusätzlicher Leerraum zwischen Wörtern
ImagePSText($image, $text, $font, $size, $black, $white, $x, $y + 30,
            100, 0, 0, 4);

// zusätzlicher Leerraum zwischen Buchstaben
ImagePSText($image, $text, $font, $size, $black, $white, $x, $y + 60,
            0, 100, 0, 4);
```

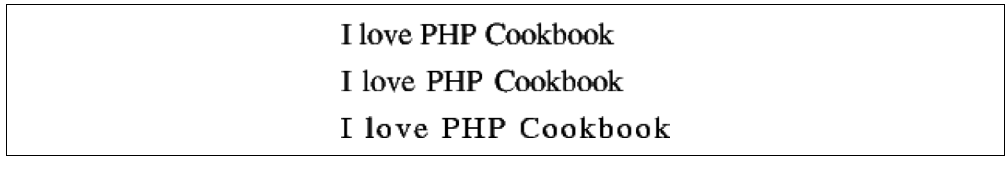


Abbildung 18-7: Wörter mit zusätzlichem Wort- und Zeichenabstand

Siehe auch

Rezept 18.5 zum Zeichnen von zentriertem Text; die Dokumentationen zu `ImageString()` unter *http://www.php.net/imagestring*, zu `ImageStringUp()` unter *http://www.php.net/imagestringup*, zu `ImageTTFText()` unter *http://www.php.net/imagettftext*, zu `ImagePSText()` unter *http://www.php.net/imagepstext* und zu `ImagePSLoadFont()` unter *http://www.php.net/imagepsloadfont*.

18.5 Zentrierten Text zeichnen

Problem

Sie wollen Text in der Mitte des Bilds zeichnen.

Lösung

Finden Sie die Größe des Bilds und des Textrahmens heraus. Mit diesen Koordinaten errechnen Sie die richtige Stelle zum Zeichnen des Texts.

Bei eingebauten GD-Schriftarten verwenden Sie die Funktion `pc_ImageStringCenter( )`, die in Beispiel 18-1 vorgestellt wird.

Beispiel 18-1: pc_ImageStringCenter()

```
function pc_ImageStringCenter($image, $text, $font) {

    // Schriftgrößen.
    $width  = array(1 => 5, 6, 7, 8, 9);
    $height = array(1 => 6, 8, 13, 15, 15);

    // Bildgröße ermitteln.
    $xi = ImageSX($image);
    $yi = ImageSY($image);

    // Textgröße ermitteln.
    $xr = $width[$font] * strlen($text);
    $yr = $height[$font];

    // Zentrierung berechnen.
    $x = intval(($xi - $xr) / 2);
    $y = intval(($yi - $yr) / 2);

    return array($x, $y);
}
```

Beispiel:

```
list($x, $y) = pc_ImageStringCenter($image, $text, $font);
ImageString($image, $font, $x, $y, $text, $fore);
```

Bei PostScript-Schriftarten verwenden Sie die Funktion `pc_ImagePSCenter( )`, die in Beispiel 18-2 vorgestellt wird.

Beispiel 18-2: pc_ImagePSCenter()

```
function pc_ImagePSCenter($image, $text, $font, $size, $space = 0,
                          $tightness = 0, $angle = 0) {

    // Bildgröße ermitteln.
    $xi = ImageSX($image);
```

Beispiel 18-2: pc_ImagePSCenter() (Fortsetzung)

```
    $yi = ImageSY($image);

    // Textgröße ermitteln.
    list($xl, $yl, $xr, $yr) = ImagePSBBox($text, $font, $size,
                                    $space, $tightness, $angle);

    // Zentrierung berechnen.
    $x = intval(($xi - $xr) / 2);
    $y = intval(($yi + $yr) / 2);

    return array($x, $y);
}
```

Zum Beispiel:

```
list($x, $y) = pc_ImagePSCenter($image, $text, $font, $size);
ImagePSText($image, $text, $font, $size, $fore, $back, $x, $y);
```

Bei TrueType-Schriftarten schließlich verwenden Sie die Funktion `pc_ImageTTFCenter( )`, die in Beispiel 18-3 vorgestellt wird

Beispiel 18-3: pc_ImageTTFCenter()

```
function pc_ImageTTFCenter($image, $text, $font, $size) {

    // Bildgröße ermitteln.
    $xi = ImageSX($image);
    $yi = ImageSY($image);

    // Textgröße ermitteln.
    $box = ImageTTFBBox($size, $angle, $font, $text);

    $xr = abs(max($box[2], $box[4]));
    $yr = abs(max($box[5], $box[7]));

    // Zentrierung berechnen.
    $x = intval(($xi - $xr) / 2);
    $y = intval(($yi + $yr) / 2);

    return array($x, $y);
}
```

Beispiel:

```
list($x, $y) = pc_ImageTTFCenter($image, $text, $font, $size);
ImageTTFText($image, $size, $angle, $x, $y, $fore, $font, $text);
```

Diskussion

Alle drei Lösungsfunktionen geben Ihnen die x- und y-Koordinaten zum Zeichnen zurück. Je nach Schriftarttyp, -größe und anderen Einstellungen ändert sich natürlich die Methode, mit der diese Koordinaten berechnet werden.

Für PostScript-Type-1-Schriftarten übergeben Sie `pc_ImagePSCenter()` ein Bild, das mit `ImageCreate()` (oder einer befreundeten Funktion) erzeugt wurde, zusammen mit einer Anzahl von Parametern, die angeben, wie der Text gezeichnet werden soll. Die ersten drei Parameter sind vorgeschrieben: der zu zeichnende Text, die Schriftart und die Schriftgröße. Die nächsten drei sind optional: die Leerzeichengröße in einem Font, der Zeichenabstand und ein Rotationswinkel in Grad.

Innerhalb der Funktion verwenden Sie `ImageSX()` und `ImageSY()`, um die Größe der Bildfläche herauszufinden; sie geben die Breite und Höhe der Grafik zurück. Dann rufen Sie `ImagePSBBox()` auf. Sie gibt vier ganze Zahlen zurück: die x- und y-Koordinaten der linken unteren Ecke des Texts und die x- und y-Koordinaten der rechten oberen Ecke. Da die Koordinaten sich auf die Grundlinie des Texts beziehen, sind sie normalerweise nicht 0. Zum Beispiel ragt ein kleines »g« unter die Unterkante der anderen Buchstaben hinaus. Deshalb ist in diesem Fall der linke untere y-Wert negativ.

Mit diesen sechs Werten gewappnet, können Sie jetzt die richtigen Zentrierungswerte berechnen. Da die Koordinaten der Bildfläche in der linken oberen Ecke (0,0) sind, die Funktion `ImagePSBText()` jedoch die linke untere Ecke will, unterscheidet sich die Berechnung von `$x` und `$y`. Für `$x` nehmen Sie die Differenz zwischen der Breite der Bildfläche und der Breite des Texts. Dadurch erhalten Sie die Größe der weißen Fläche, die den Text umgibt. Anschließend teilen Sie diese Nummer durch zwei, um die Pixelanzahl herauszufinden, die Sie auf der linken Seite des Texts lassen sollten. Für `$y` machen Sie das Gleiche, addieren aber `$yi` und `$yr` hinzu. Durch die Addition finden Sie die Koordinate der unteren Seite des Rahmens. Durch die umgekehrte Eingabeart der y-Koordinate in GD ist diese Summe der Wert, der hier gebraucht wird.

Bei diesen Berechnungen werden absichtlich die linken unteren Koordinaten ignoriert. Da sich der größte Teil des Texts über der Grundlinie befindet, wird der Code schlechter, wenn die absteigenden Pixel im Zentrierungsalgorithmus berücksichtigt werden. Für das menschliche Auge erscheint er nicht richtig zentriert.

Um einen Text zu zentrieren, setzen Sie die Code-Teile wie folgt zusammen:

```
function pc_ImagePSCenter($image, $text, $font, $size, $space = 0,
                          $tightness = 0, $angle = 0) {

    // Bildgröße ermitteln.
    $xi = ImageSX($image);
    $yi = ImageSY($image);

    // Textgröße ermitteln.
    list($xl, $yl, $xr, $yr) = ImagePSBBox($text, $font, $size,
                                   $space, $tightness, $angle);

    // Zentrierung berechnen.
    $x = intval(($xi - $xr) / 2);
    $y = intval(($yi + $yr) / 2);

    return array($x, $y);
```

```
}

$image = ImageCreate(500,500);
$text = 'Es lebe das PHP Kochbuch!';
$font = ImagePSLoadFont('/pfad/zu/font.pfb');
$size = 20;
$black = ImageColorAllocate($image, 0, 0, 0);
$white = ImageColorAllocate($image, 255, 255, 255);
list($x, $y) = pc_ImagePSCenter($image, $text, $font, $size);
ImagePSText($image, $text, $font, $size, $white, $black, $x, $y);
ImagePSFreeFont($font);

header('Content-type: image/png');
ImagePng($image);

ImageDestroy($image);
```

Leider funktioniert dieses Beispiel weder für die eingebauten Fonts von GD noch bei TrueType-Fonts. Es gibt einfach keine Funktion, die die Größe eines eingebauten Fonts angibt, und `ImageTTFBBox( )` gibt acht Werte zurück, nicht vier. Mit ein paar Modifikationen können Sie mit diesen Unterschieden aber fertig werden.

Da die eingebauten Schriftarten eine feste Breite haben, können Sie einfach die Größe eines Zeichens messen und daraus eine Funktion bauen, die die Größe des angegebenen Texts als Funktion seiner Länge berechnet. Tabelle 18-1 ist zwar nicht hundertprozentig genau, sollte aber Ergebnisse mit einer Abweichung von nicht mehr als ein oder zwei Pixeln liefern, was in den meisten Fällen ausreichen sollte.

Tabelle 18-1: Die Zeichengrößen der eingebauten GD-Fonts

Font-Nummer	Breite	Höhe
1	5	6
2	6	8
3	7	13
4	8	15
5	9	15

In `pc_ImageStringCenter( )` wird die Breite der Zeichenkette als ganzzahliges Vielfaches ihrer Länge berechnet. Die Höhe ist jeweils die Höhe eines einzelnen Zeichens. Beachten Sie, dass `ImageString( )` seine y-Koordinate als den höchsten Teil des Texts berechnet, so dass Sie beim Berechnen von `$y` das Vorzeichen wieder umdrehen müssen.

Das folgende Beispiel verwendet alle fünf Fonts und zentriert den Text horizontal:

```
$text = 'The quick brown fox jumps over the lazy dog.';
for ($font = 1, $y = 5; $font <= 5; $font++, $y += 20) {
    list($fx, $fy) = pc_ImageStringCenter($image, $text, $font);
    ImageString($image, $font, $fx, $y, $text, $color);
}
```

Die Ausgabe sehen Sie in Abbildung 18-8.

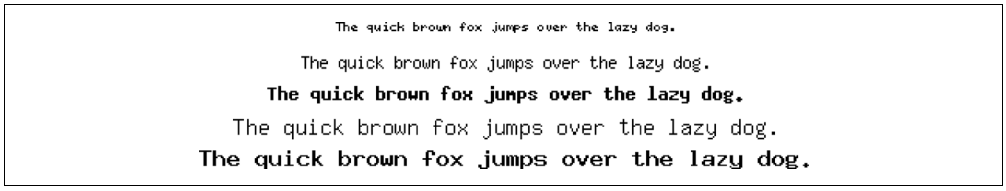


Abbildung 18-8: Zentrierte eingebaute GD-Fonts

Bei TrueType-Fonts müssen Sie `ImageTTFBBox( )` oder die etwas modernere `ImageFtBBox( )` verwenden (die Funktion mit TTF im Namen steht für FreeType-Version 1.x, die mit Ft für FreeType 2.x.). Sie gibt acht Zahlen zurück: die (x,y)-Koordinaten der vier Ecken des Texts, gegen den Uhrzeigersinn und beginnend mit der linken unteren Ecke. Damit gibt das zweite Koordinatenpaar die rechte untere Ecke an und so weiter.

Um `pc_ImageTTFCenter( )` zu erhalten, beginnen Sie mit `pc_ImagePSCenter( )` und vertauschen die Zeile:

```
// Textgröße ermitteln.
list($xl, $yl, $xr, $yr) = ImagePSBBox($text, $font, $size,
                                       $space, $tightness, $angle);
```

gegen die folgenden Zeilen aus:

```
// Textgröße ermitteln.
$box = ImageTTFBBox($size, $angle, $font, $text);

$xr = abs(max($box[2], $box[4]));
$yr = abs(max($box[5], $box[7]));
```

Hier ist ein Beispiel für die Anwendung von `pc_ImageTTFCenter()`:

```
list($x, $y) = pc_ImageTTFCenter($image, $text, $font, $size);
ImageTTFText($image, $size, $angle, $x, $y, $white, $black,
             '/pfad/zu/font.ttf', $text);
```

Siehe auch

Rezept 18.5 für weitere Informationen zum Zentrieren von Text; die Dokumentationen zu `ImageSX( )` unter *http://www.php.net/imagesx*, `ImageSY( )` unter *http://www.php.net/imagesy*, `ImagePSBBox( )` unter *http://www.php.net/imagepsbbox*, `ImageTTFBBox( )` unter *http://www.php.net/imagettfbbox* und `ImageFtBBox( )` unter *http://www.php.net/imageftbbox*.

18.6 Dynamische Bilder zusammensetzen

Problem

Sie wollen ein Bild mit dynamischen Daten (normalerweise Text) auf der Basis einer existierenden Bildvorlage erstellen. Beispiel: Sie wollen einen Hit-Zähler bauen.

Lösung

Laden Sie die Bildvorlage, ermitteln Sie die richtige Position für das Zentrieren Ihres Texts, fügen Sie der Bildfläche den Text hinzu und senden Sie das Bild zum Browser:

```
// Konfigurationseinstellungen
$image    = ImageCreateFromPNG('button.png');
$text     = $_GET['text'];
$font     = ImagePSLoadFont('Times');
$size     = 24;
$color    = ImageColorAllocate($image,   0,   0,   0); // black
$bg_color = ImageColorAllocate($image, 255, 255, 255); // white

// Zentrierten Text ausdrucken.
list($x, $y) = pc_ImagePSCenter($image, $text, $font, $size);
ImagePSText($image, $text, $font, $size, $color, $bg_color, $x, $y);

// Bild senden.
header('Content-type: image/png');
ImagePNG($image);

// Aufräumen.
ImagePSFreeFont($font);
ImageDestroy($image);
```

Diskussion

Das Bauen von dynamischen Bildern mit GD ist einfach; Sie brauchen nur ein paar Rezepte zu kombinieren. Am Anfang des Lösungs-Codes laden Sie ein Bild eines Vorlage-Buttons, den Sie vorrätig haben. Das stellt den Hintergrund dar, über den Sie den Text legen. Definieren Sie den Text so, dass er direkt aus dem String des HTTP-Requests kommt. Alternativ können Sie den String auch aus einer Datenbank entnehmen (im Fall von Zugriffszählern) oder ihn von einem ausgelagerten Server anfordern (z.B. für Börsendaten oder Icons für Wetterberichte).

Anschließend fahren Sie mit den anderen Einstellungen fort: eine Schriftart laden und ihre Größe, Farbe und Hintergrundfarbe angeben. Bevor Sie den Text ausdrucken, müssen Sie noch seine Position errechnen. Für diese Aufgabe eignet sich `pc_ImagePSCenter( )` aus Rezept 18.6 ganz hervorragend. Schließlich geben Sie das Bild an den Browser aus und geben den Speicher für Font und Bild wieder frei.

Der folgende Code erstellt beispielsweise eine HTML-Seite und Image-Tags, die dynamische Buttons verwenden, wie in Abbildung 18-9 gezeigt:

```
<?php
if (isset($_GET['button'])) {

    // Konfigurationseinstellungen
    $image    = ImageCreateFromPNG('button.png');
    $text     = $_GET['button'];      // dynamisch erstellter Text
    $font     = ImagePSLoadFont('Times');
    $size     = 24;
    $color    = ImageColorAllocate($image,   0,   0,   0); // schwarz
    $bg_color = ImageColorAllocate($image, 255, 255, 255); // weiß

    // Zentrierten Text drucken.
    list($x, $y) = pc_ImagePSCenter($image, $text, $font, $size);
    ImagePSText($image, $text, $font, $size, $color, $bg_color, $x, $y);

    // Bild senden.
    header('Content-type: image/png');
    ImagePNG($image);

    // Aufräumen.
    ImagePSFreeFont($font);
    ImageDestroy($image);

} else {
?>
<html>
<head>
    <title>Dynamic Buttons</title>
</head>
<body>
    <img src="<?php echo $_SERVER['PHP_SELF']; ?>?button=Previous"
         alt="Previous" width="132" height="46">
    <img src="<?php echo $_SERVER['PHP_SELF']; ?>?button=Next"
         alt="Next"  width="132" height="46">
</body>
</html>
<?php
}
?>
```

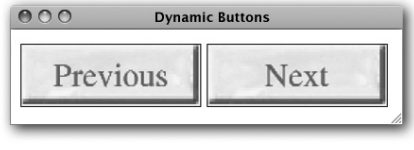


Abbildung 18-9: Beispielseite mit Buttons

Wenn in diesem Skript ein Wert für `$_GET['button']` überreicht wird, erstellen Sie einen Button und senden die PNG-Datei zurück. Wenn `$_GET['button']` nicht angegeben ist, senden Sie eine einfache HTML-Seite mit zwei eingebetteten Aufrufen des Skripts zurück, mit denen Sie Button-Bilder anfordern – eines für einen »Previous«-Button und eines für einen »Next«-Button. Eine allgemeinere Lösung ist es, eine eigene *button.php*-Seite zu erstellen, die nur Grafiken ausgibt, und die Bildquelle auf dieser Seite zeigen zu lassen.

Siehe auch

Rezept 18.5 mit weiteren Informationen zum Zeichnen von Text; Rezept 18.6 zum Zentrieren von Text; eine hervorragende Diskussion über dynamisches Bild-Caching in Kapitel 9, »Grafik«, in *Programmieren mit PHP* von Rasmus Lerdorf und Kevin Tatroe (O'Reilly Verlag).

18.7 Eine transparente Farbe ermitteln und einstellen

Problem

Sie wollen eine der Bildfarben als transparente Farbe einstellen. Wenn das Bild über einen Hintergrund gelegt wird, sieht man den Hintergrund durch den transparenten Teil des Bilds hindurch.

Lösung

Verwenden Sie `ImageColorTransparent( )`:

```
$color = ImageColorAllocate($image, $red, $green, $blue);
ImageColorTransparent($image, $color);
```

Diskussion

Sowohl GIFs als auch PNGs unterstützen transparente Farben, JPEGs jedoch nicht. Um in GD eine transparente Farbe zu beschreiben, verwenden Sie die Konstante `IMG_COLOR_TRANSPARENT`. Hier sehen Sie beispielsweise, wie man eine gestrichelte Linie erzeugen kann, die zwischen schwarz und transparent wechselt:

```
// Eine zwei Pixel breite, schwarzweiß gestrichelte Linie erzeugen.
$style = array($black, $black, IMG_COLOR_TRANSPARENT, IMG_COLOR_TRANSPARENT);
ImageSetStyle($image, $style);
```

Um die aktuelle Transparenz-Einstellung zu finden, nehmen Sie den Rückgabewert von `ImageColorTransparent( )` und reichen ihn an `ImageColorsForIndex( )` weiter:

```
$transparent = ImageColorsForIndex($image, ImageColorTransparent($image));
print_r($transparent);
```

```
Array
(
    [red] => 255
    [green] => 255
    [blue] => 255
)
```

Die Funktion `ImageColorsForIndex( )` gibt ein Array mit den Werten für Rot, Grün und Blau zurück. In diesem Fall ist die transparente Farbe Weiß.

Siehe auch

Die Dokumentationen zu `ImageColorTransparent( )` unter *http://www.php.net/imagecolortransparent* und `ImageColorsForIndex( )` unter *http://www.php.net/imagecolorsforindex*.

18.8 Programm: Heraufgeladene Digitalfotos auf Webformat verkleinern

Problem

Sie wollen, dass Ihre Benutzer Digitalfotos in beliebiger Auflösung auf Ihren Server laden können. Sie möchten die Bilder aber nicht mit voller Auflösung auf Ihre Website stellen, weil sonst der Download der Seiten zu lange dauert. Mit anderen Worten: Sie wollen die Fotos verkleinern.

Lösung

Verwenden Sie `imagecopyresized()` oder, noch besser, `imagecopyresampled()`, um Ihre Bilder auf das richtige Format zu bringen:

```
<?php

// Bildparameter - nach Bedarf konfigurieren.
$imageMaxHeight = 300; // maximale Höhe in Pixeln
$imageMaxWidth = 400; // maximale Breite in Pixeln
$destinationPath = "./images/";
$quality = 75; // JPEG-Qualitätsfaktor 1...100

if (is_uploaded_file($_FILES["cameraImage"]["tmp_name"])) {
    $uploadedClientFileName = $_FILES["cameraImage"]["name"];
    $serverTempFileName = $_FILES["cameraImage"]['tmp_name'];
    // Dateierweiterung ermitteln:
    preg_match("/(\.\w+)$/",$uploadedClientFileName,$match);
    $uploadedFileType = strtolower($match[1]);
    // Überprüfen, ob die Datei einem erlaubten Typ entspricht (Erweiterung testen).
```

```
// Dateierweiterung statt MIME-Typ verwenden – MIME kann vorgetäuscht sein –
// und nachsehen, ob es ein von uns akzeptierter Grafiktyp ist:
if (!in_array($uploadedFileType, array(".gif",".jpg",".jpeg",".png"))) {
    die("Unbekanntes Grafikformat oder keine Grafikdatei");
}

switch ($uploadedFileType) {
    case ".jpg":
    case ".jpeg":
        $image = @imagecreatefromjpeg($serverTempFileName);
        break;
    case ".png":
        $image = @imagecreatefrompng($serverTempFileName);
        break;
    case ".gif":
        if (is_callable("imagegif")) { // can resize
            $image = @imagecreatefromgif($serverTempFileName);
        }
        else die("Dieser Server kann noch kein GIF!");
        break;
    default:
        break;
}

// Bild verkleinern, wenn nötig.
list($oldWidth,$oldHeight) = getimagesize($serverTempFileName);
if (($oldWidth > $imageMaxWidth) || ($oldHeight > $imageMaxHeight)) { // zu groß!
    // Sind wir hauptsächlich zu breit oder zu hoch?
    $widthRatio = $oldWidth / $imageMaxWidth;
    $heightRatio = $oldHeight / $imageMaxHeight;
    // Neue Abmessungen berechnen.
    if ($widthRatio >= $heightRatio) { // hauptsächlich zu breit
        $newWidth = $imageMaxWidth;
        $newHeight = $oldHeight / $widthRatio;
    }
    else
    {  // hauptsächlich zu hoch
        $newHeight = $imageMaxHeight;
        $newWidth = $oldWidth / $heightRatio;
    }
    // Grafik mit den neuen Dimensionen erzeugen.
    $newImage = @imagecreatetruecolor($newWidth, $newHeight);
    // Existierende Grafik per Resample herüberkopieren ...
    if ((is_callable("imagecopyresampled")) && $cnfUseImageCopyResampled) {
        imagecopyresampled($newImage, $image,
                0, 0, 0, 0, $newWidth, $newHeight,
                $oldWidth, $oldHeight);
    }
    else
    {   // ... oder per Resize übernehmen (Qualitätsverlust).
        imagecopyresized($newImage, $image,
```

```
                    0, 0, 0, 0, $newWidth, $newHeight,
                    $oldWidth, $oldHeight);
        }
    }
    else
    {
        $newImage = $image; // Bild ist klein genug.
    }
    // Der Datei einen neuen Namen geben, über den der Benutzer keinen Einfluss hat:
    $newFileNameSuffix = uniqid("").$uploadedFileType;
    $newFileNameFullRes = $destinationPath."full".$newFileNameSuffix;
    $newFileNameSmall = $destinationPath."small".$newFileNameSuffix;

    // Hoch- und niedrigauflösende Bildversion speichern:
    switch ($uploadedFileType) {
        case ".jpg":
        case ".jpeg":
            imagejpeg($image, $newFileNameFullRes, $quality);
            imagejpeg($newImage, $newFileNameSmall, $quality);
            break;
        case ".png":
            imagepng($image, $newFileNameFullRes);
            imagepng($newImage, $newFileNameSmall);
            break;
        case ".gif":
            imagegif($image, $newFileNameFullRes);
            imagegif($newImage, $newFileNameSmall);
            break;
    }
    // Bilder über HTML ausgeben.
?>
<a href="<?php echo $newFileNameFullRes; ?>">
    <img src="<?php echo $newFileNameSmall; ?>"/>
</a>
<?php
}
    // Formular zum Bild-Upload
?>
<form action="image_resize.php" method="post" enctype="multipart/form-data">
    Upload an image:
    <input type="file" name="cameraImage">
    <input type="submit">
</form>
```

Diskussion

Beim Heraufladen von Digitalkamerafotos kann eine ganze Menge schief gehen. Viele Benutzer verstehen nicht, dass ein 1-MByte-Digitalfoto zwar auf ihrem Rechner sehr toll aussieht, aber ein Album mit 20 dieser Fotos für Benutzer mit Telefonmodems ein Albtraum ist. Die Bilder laden nicht nur extrem langsam, es wird auch nur ein geringer Teil

der Bilddaten überhaupt dargestellt. Dabei helfen auch die weit verbreiteten `width`- und `height`-Attribute der `<img>`-Tags nicht.

Außerdem müssen Sie aufpassen, dass man Ihnen keine existierenden Bilder überschreibt oder – noch übler – anstatt einer Bilddatei eine ausführbare Programmdatei unterschiebt. Dazu müssen Sie sicherstellen, dass Sie nur Dateien mit bekannten Grafik-Dateierweiterungen abspeichern und sich den Rest des Dateinamens nicht vom Benutzer vorgeben lassen.

Das oben vorgestellte Programm macht alles, was Sie brauchen: Es stellt zunächst sicher, dass die hochgeladene Datei eine der Dateierweiterungen `.gif`, `.png`, `.jpg` oder `.jpeg` hat. Damit werden anschließend das Original und die Webversion der Grafik gespeichert, und es wird verhindert, dass Ihnen jemand eine `.php`-Datei oder andere ausführbare Dateien auf den Server legt.

Im Anschluss daran erzeugt das Programm in einem `switch`-Statement eine GD-Grafik aus der heraufgeladenen Datei, wobei je nach Dateityp `imagecreatefromjpeg()`, `imagecreatefrompng()` oder `imagecreatefromgif()` aufgerufen wird. Ist die Grafik zu groß, wird sie verkleinert. Dazu prüft das Programm mit `is_callable()`, ob die installierte GD-Version Resampling unterstützt. Beim Resampling werden nicht wie beim Resizing einfach einzelne Bildpunkte kopiert, sondern ein Mittelwert aus benachbarten Pixeln gebildet. Das verhindert, dass das Bild nachher verrauscht aussieht. Bei älteren GD-Versionen ohne Resampling wird Resizing verwendet.

Danach erzeugt das Programm einen eindeutigen Dateinamen für das Original und die verkleinerte Kopie und speichert beide Versionen der Grafik ab. So können Ihre Benutzer bei Bedarf die Vollversion eines Digitalbilds sehen, indem sie auf die verkleinerte Version klicken.

Siehe auch

Die Dokumentationen zu `imagecreatefromjpeg()` unter *http://www.php.net/imagecreatefromjpeg*, zu `imagecreatefrompng()` unter *http://www.php.net/imagecreatefrompng*, zu `imagecreatefromgif()` unter *http://www.php.net/imagecreatefromgif*, zu `is_callable()` unter *http://www.php.net/is-callable*, zu `imagecopyresampled()` unter *http://www.php.net/imagecopyresampled*, zu `getimagesize()` unter *http://www.php.net/getimagesize*, zu `imagejpeg()` unter *http://www.php.net/imagejpeg*, zu `imagepng()` unter *http://www.php.net/imagepng* und zu `imagegif()` unter *http://www.php.net/imagegif*.

18.9 Grafiken geschützt ausgeben

Problem

Sie wollen bestimmen können, wer bestimmte Bilder ansehen kann und wer nicht.

Lösung

Bewahren Sie Ihre Bilder nicht in Ihrem Dokumentenpfad auf, sondern speichern Sie sie an einer anderen Stelle ab. Um eine Datei auszugeben, öffnen Sie sie von Hand und senden sie an den Browser:

```
header('Content-Type: image/png');
readfile('/pfad/zum/bild.png');
```

Diskussion

Die erste Zeile der Lösung sendet den `Content-type`-Header zum Browser, damit der Browser weiß, welche Objektart auf ihn zukommt, und diese entsprechend anzeigen kann. Die zweite Zeile öffnet eine Datei auf einer Festplatte (oder von einer ausgelagerten URL) für Lesezwecke, liest sie ein, gibt sie direkt an den Browser weiter und schließt die Datei.

Die gebräuchlichste Art, ein Bild auszugeben, ist es, ein `<img>`-Tag zu benutzen und das `src`-Attribut auf eine Datei auf Ihrer Webseite zeigen zu lassen. Wenn Sie diese Bilder schützen möchten, sollten Sie wahrscheinlich eine Passwort-Authentifizierung durchführen. Eine Möglichkeit der Prüfung ist die HTTP Basic Authentication, die in Rezept 10.9 behandelt wird.

Die gebräuchlichste Art muss aber nicht immer die beste sein. Erstens: Was passiert, wenn Sie den Zugang zu bestimmten Dateien unterbinden wollen, es aber nicht durch Benutzernamen und Passwort zusätzlich verkomplizieren möchten? Eine Möglichkeit besteht darin, einen Link nur zu bestimmten Dateien herzustellen. Wenn die Benutzer nicht auf den Link klicken können, können sie die Datei nicht einsehen. Allerdings könnte es sein, dass sie Lesezeichen auf alte Dateien setzen oder dass sie auf Grund Ihres Namensgebungssystems versuchen, andere Dateinamen zu erraten, und die URL von Hand in den Browser eingeben.

Wenn Ihre Inhalte gesperrt sein sollen, werden Sie nicht wollen, dass andere Ihr Namensgebungssystem erraten und Bilder einsehen können. Wenn Informationen gesperrt sind, erhält oft eine ausgewählte Personengruppe, beispielsweise Journalisten, eine Vorab-Veröffentlichung, so dass sie Artikel über das Thema schreiben und im Moment der Aufhebung des Embargos veröffentlichen können. Sie können das erreichen, indem Sie sicherstellen, dass nur legaler Text im Dokumentenpfad steht. Allerdings erfordert dies ein großes Hin- und Hergeschiebe von Dateien zwischen Verzeichnissen. Stattdessen können Sie alle Dateien an einem permanenten Platz aufbewahren und nur Dateien ausgeben, die eine Prüfung in ihrem Code bestehen.

Nehmen wir beispielsweise einmal an, dass Sie einen Vertrag mit einem Verlag haben, um Comics des Verlags auf Ihrer Webseite zu veröffentlichen. Allerdings will der Verlag nicht, dass Sie ein elektronisches Archiv erstellen. Deshalb einigen Sie sich darauf, dass Benutzer nur die Comics der letzten zwei Wochen sehen dürfen. Für alle anderen müssen sie zur offiziellen Site gehen. Außerdem bekommen Sie die Folgen eventuell bereits vor

dem Veröffentlichungstermin. Sie wollen den Leuten aber keine kostenlose Vorschau geben, sondern sie dazu animieren, täglich auf Ihre Seite zu schauen.

Und hier kommt die Lösung. Die Dateien kommen unter dem Namen des Datums an, so dass es einfach ist herauszufinden, welche Dateien zu welchem Tag gehören. Um die Folgen außerhalb des fortlaufenden 14-Tage-Fensters zu sperren, benutzen Sie den folgenden Code:

```
// Einen Comic anzeigen, wenn er weniger als 14 Tage alt ist
// und das Datum nicht in der Zukunft liegt.

// Das aktuelle Datum berechnen.
list($now_m,$now_d,$now_y) = explode(',',date('m,d,Y'));
$now = mktime(0,0,0,$now_m,$now_d,$now_y);

// Zweistündiger Spielraum, um die Sommerzeit mit einzubeziehen.
$min_ok = $now - 14*86400 - 7200; // vor 14 Tagen
$max_ok = $now + 7200;            // heute

// Den Datumsstempel des angeforderten Comics herausfinden.
$asked_for = mktime(0,0,0,$_REQUEST['mo'],$_REQUEST['dy'],$_REQUEST['yr']);

// Die Daten vergleichen.
if (($min_ok > $asked_for) || ($max_ok < $asked_for)) {
    echo 'Schluchz! Den Comic f&uuml;r diesen Tag d&uuml;rfen Sie
          leider nicht sehen.';
} else {
    header('Content-type: image/png');
    readfile("/www/comics/$_REQUEST['mo']$_REQUEST['dy'] $_REQUEST['yr'].png");
}
```

Siehe auch

Rezept 21.4 zu weiteren Informationen über das Lesen von Dateien.

18.10 Programm: Aus Umfrageergebnissen Balkendiagramme erstellen

Beim Anzeigen von Umfrageergebnissen kann es wirkungsvoller sein, ein buntes Balkendiagramm zu erstellen, als die Ergebnisse nur als Text auszudrucken. Die in Beispiel 18-4 dargestellte Funktion verwendet GD zur Darstellung eines Bilds, das die kumulativen Antworten auf eine Frage zeigt.

Beispiel 18-4: Grafische Balkendiagramme

```
function pc_bar_chart($question, $answers) {

    // Farben für die Balken definieren.
    $colors = array(array(255,102,0), array(0,153,0),
```

Beispiel 18-4: Grafische Balkendiagramme (Fortsetzung)

```
                    array(51,51,204), array(255,0,51),
                    array(255,255,0), array(102,255,255),
                    array(153,0,204));

    $total = array_sum($answers['votes']);

    // Leerräume und andere magische Werte definieren.
    $padding = 5;
    $line_width = 20;
    $scale = $line_width * 7.5;
    $bar_height = 10;

    $x = $y = $padding;

    // Eine große Palette zum Zeichnen definieren,
    // da Sie die Bildlänge nicht im Voraus kennen.
    $image = ImageCreate(150, 500);
    $bg_color = ImageColorAllocate($image, 224, 224, 224);
    $black = ImageColorAllocate($image, 0, 0, 0);

    // Die Fragen ausdrucken.
    $wrapped = explode("\n", wordwrap($question, $line_width));
    foreach ($wrapped as $line) {
        ImageString($image, 3, $x, $y , $line, $black);
        $y += 12;
    }

    $y += $padding;

    // Antworten ausdrucken.
    for ($i = 0; $i < count($answers['answer']); $i++) {

        // Prozentsätze formatieren.
        $percent = sprintf('%1.1f', 100*$answers['votes'][$i]/$total);
        $bar = sprintf('%d', $scale*$answers['votes'][$i]/$total);

        // Farbe wählen.
        $c = $i % count($colors); // Fälle mit mehr Balken als Farben behandeln.
        $text_color = ImageColorAllocate($image, $colors[$c][0],
                                $colors[$c][1], $colors[$c][2]);

        // Balken und Prozentzahlen zeichnen.
        ImageFilledRectangle($image, $x, $y, $x + $bar,
                             $y + $bar_height, $text_color);
        ImageString($image, 3, $x + $bar + $padding, $y,
                    "$percent%", $black);

         $y += 12;

         // Antworten drucken.
         $wrapped = explode("\n", wordwrap($answers['answer'][$i], $line_width));
```

Beispiel 18-4: Grafische Balkendiagramme (Fortsetzung)

```
        foreach ($wrapped as $line) {
            ImageString($image, 2, $x, $y, $line, $black);
            $y += 12;
        }

        $y += 7;
    }

    // Bild durch Kopieren zurechtschneiden.
    $chart = ImageCreate(150, $y);
    ImageCopy($chart, $image, 0, 0, 0, 0, 150, $y);

    // Bild ausgeben.
    header ('Content-type: image/png');
    ImagePNG($chart);

    // Aufräumen.
    ImageDestroy($image);
    ImageDestroy($chart);
}
```

Um dieses Programm aufzurufen, erstellen Sie ein Array, das zwei parallele Arrays enthält: `$answers['answer']` und `$answer['votes']`. Das Element `$i` im jeweiligen Array enthält jeweils den Antworttext bzw. die Gesamtzahl der Stimmen für die Antwort `$i`. Abbildung 18-10 zeigt die Ausgabe dieses Beispiels

```
// Im Kino
$question = 'Wie hat Ihnen der Film gefallen?';

$answers['answer'][ ] = 'Himmel und Erde bewegten sich!';
$answers['votes'][ ]  = 29;

$answers['answer'][ ] = 'Augen und Ohren bewegten sich';
$answers['votes'][ ]  = 22;

$answers['answer'][ ] = 'Nichts bewegte sich';
$answers['votes'][ ]  = 59;

$answers['answer'][ ] = 'Die Nachbarn bewegten sich';
$answers['votes'][ ]  = 45;

pc_bar_chart($question, $answers);
```

In diesem Fall werden die Antworten von Hand zugewiesen, aber bei einer echten Umfrage könnten die Daten stattdessen aus einer Datenbank entnommen werden.

Dieses Programm ist schon mal ein guter Anfang. Da es aber die eingebauten GD-Schriftarten verwendet, enthält es eine Menge magischer Zahlen, die sich auf die Höhe und Breite der Schriftart beziehen. Außerdem ist die Größe des Leerraums zwischen den Ant-

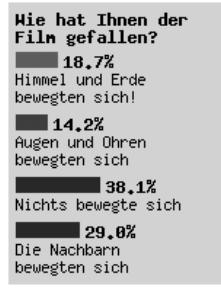


Abbildung 18-10: Grafisches Balkendiagramm von Umfrageergebnissen

worten im Code festgelegt. Wenn Sie den Code ändern, um kompliziertere Fonts wie zum Beispiel PostScript oder TrueType zu verwenden, müssen Sie den Algorithmus, der diese Zahlen bestimmt, aktualisieren.

Am Anfang der Funktion werden einige RGB-Kombinationen definiert. Sie werden als Balkenfarben verwendet. Mehrere Konstanten werden festgelegt, so zum Beispiel `$line_width`, die die maximale Anzahl von Buchstaben pro Zeile angibt. Die Variable `$bar_height` gibt die Balkenhöhe an, und `$scale` skaliert die Balkenlänge als Funktion der maximalen Zeilenlänge. `$padding` wird verwendet, um die Ergebnisse fünf Pixel vom Bildflächenrand fernzuhalten.

Dann wird eine sehr große Bildfläche erstellt, auf der das Diagramm gezeichnet wird. Da es oft schwer ist, die Gesamtgröße im Voraus zu wissen, wird die Bildfläche erst später auf ihre Endgröße zurechtgeschnitten. Die Standard-Hintergrundfarbe des Balkendiagramms ist (224, 224, 224), also ein helles Grau.

Um die Diagrammbreite auf eine vernünftige Größe zu beschränken, wird die `$question` mit `wordwrap()` verkleinert, und der Zeilenumbruch `\n` wird in der Funktion `explode()` verwendet, um die Frage wieder zu einem einzigen String zusammenzusetzen. Damit erhalten Sie ein Array von Zeilen der richtigen Größe, das Sie in einer Schleife zeilenweise ausdrucken.

Nach dem Ausdrucken der Frage geht es mit den Antworten weiter. Zuerst formatieren Sie die Ergebniszahlen mit `sprintf()`. Um den Gesamtprozentsatz der Stimmen für eine Antwort als Fließkommazahl mit einer Dezimalstelle zu formatieren, verwenden Sie `%1.1f`. Um die entsprechende Balkenlänge für diese Zahl zu finden, berechnen Sie eine ähnliche Zahl, multiplizieren aber an Stelle von 100 mit einer magischen Nummer, `$scale`, und geben eine ganze Zahl zurück.

Die Textfarbe wird dem `$colors`-Array der RGB-Triplets entnommen. Dann rufen Sie `ImageFilledRectangle()` auf, um den Balken zu zeichnen, und `ImageString()`, um die Prozentzahl rechts vom Balken zu zeichnen. Nachdem Sie etwas begrenzenden Leerraum (padding) hinzugefügt haben, drucken Sie die Antwort, indem Sie den gleichen Algorithmus wie beim Ausdrucken der Frage verwenden.

Wenn alle Antworten ausgedruckt sind, wird die Gesamtgröße des Balkendiagramms in `$y` gespeichert. Jetzt können Sie die Grafik auf ihre richtige Größe zurückschneiden. Leider gibt es aber keine Funktion `ImageCrop()`. Deshalb erzeugen Sie eine neue Bildfläche mit der richtigen Größe und kopieren mit `ImageCopy()` den Teil der ursprünglichen Bildfläche, den Sie behalten wollen, in das neue Bild. Dann senden Sie das zugeschnittene Bild mit `ImagePNG()` als PNG und räumen mit zwei Aufrufen von `ImageDestroy()` auf.

Wie am Anfang dieses Abschnitts bereits erwähnt, handelt es sich hier nur um eine schnelle und einfache Funktion zum Ausdrucken von Balkendiagrammen. Es funktioniert und löst einige Probleme, wie zum Beispiel Zeilenumbrüche, ist aber nicht 100%ig perfekt. Zum Beispiel lässt sie sich nur schwer an Benutzerwünsche anpassen. Viele Einstellungen sind direkt im Code festgelegt. Trotzdem zeigt sie, wie man mehrere GD-Funktionen kombinieren kann, um eine nützliche grafische Anwendung zu erstellen.

KAPITEL 19

Internationalisierung und Lokalisierung

19.0 Einführung

Obwohl jeder PHP-Programmierer letztendlich etwas Englisch lernen muss, um mit Funktionsnamen und Sprachkonstrukten umgehen zu können, kann man mit PHP Anwendungen in praktisch allen Sprachen erstellen. Manche Anwendungen müssen von Benutzern in vielen unterschiedlichen Sprachen verwendet werden können. Durch die PHP-Unterstützung für Internationalisierung und Lokalisierung wird es einfacher, eine Anwendung, die für Franzosen geschrieben wurde, für Deutsche nutzbar zu machen.

Bei der Internationalisierung (oft abgekürzt zu I18N[1]) wird eine Anwendung, die nur für ein Locale geschrieben wurde, so abgeändert, dass sie in vielen verschiedenen Locales verwendet werden kann. Bei der Lokalisierung (abgekürzt L10N[2]) wird einer internationalisierten Anwendung die Unterstützung für ein neues Locale hinzugefügt.

Ein Locale ist eine Gruppe von Einstellungen, die die Textformatierung und Sprachgewohnheiten in einem bestimmten Teil der Welt beschreiben. Die Einstellungen sind in sechs Kategorien eingeteilt:

`LC_COLLATE`
: Diese Einstellungen steuern die Textsortierung, also in welcher Reihenfolge die Buchstaben im Alphabet stehen.

`LC_CTYPE`
: Diese Einstellungen bestimmen die Zuordnung zu Groß- und Kleinbuchstaben und welche Zeichen in die einzelnen Zeichenklassen, zum Beispiel alphanumerische Zeichen, fallen.

1 Das Wort »internationalization« hat 18 Buchstaben zwischen dem »i« am Anfang und dem »n« am Ende.

2 Das Wort »localization« hat 10 Buchstaben zwischen dem »l« am Anfang und dem »n« am Ende.

`LC_MONETARY`
: Diese Einstellungen beschreiben das bevorzugte Format für Währungen, z.B. welches Zeichen zur Beschreibung des Dezimalkommas verwendet wird und wie negative Beträge angezeigt werden sollen.

`LC_NUMERIC`
: Diese Einstellungen beschreiben das bevorzugte Format für Zahlen, z.B. wie Zahlen gruppiert werden und welches Zeichen zum Trennen der Tausender verwendet wird.

`LC_TIME`
: Diese Einstellungen beschreiben das bevorzugte Zeit- und Datumsformat, z.B. die Namen der Monate und Tage, und ob das 12- oder 24-Stunden-Format verwendet wird.

`LC_MESSAGES`
: Diese Kategorie beinhaltet Textmeldungen von Anwendungen, die Informationen in mehreren Sprachen anzeigen müssen.

Außerdem gibt es eine Metakategorie, `LC_ALL`, die alle anderen Kategorien umfasst.

Der Name eines Locale besteht im Regelfall aus drei Teilen. Der erste Teil, eine Abkürzung, die eine Sprache kennzeichnet, muss immer vorhanden sein, zum Beispiel »en« für Englisch oder »pt« für Portugiesisch. Als Nächstes, durch einen Unterstrich getrennt, folgt eine optionale Länderangabe, um zwischen verschiedenen Ländern zu unterscheiden, die unterschiedliche Formen der gleichen Sprache sprechen, beispielsweise »en_US« für amerikanisches Englisch und »en_GB« für britisches Englisch oder »pt_BR« für brasilianisches Portugiesisch und »pt_PT« für portugiesisches Portugiesisch. Als Letztes folgt, nach einem Punkt, eine optionale Zeichensatzangabe, z.B. »zh_TW.Big5« für Taiwan-Chinesen, die den Big5-Zeichensatz verwenden. Obwohl die meisten Locale-Namen diesen Konventionen entsprechen, trifft das nicht auf alle zu. Ein Problem beim Verwenden von Locales ist, dass sie beliebig benannt sein können. Das Herausfinden und Setzen von Locales wird in den Rezepten 19.1 bis 19.3 behandelt.

Für die korrekte Lokalisierung von normalem Text, Datum/Zeit und Währung sind unterschiedliche Techniken erforderlich. Die Lokalisierung kann auch auf von Ihrem Programm verwendete externe Objekte angewendet werden, z.B. auf Bilder und eingebundene Dateien. Die Lokalisierung von derartigen Inhalten wird in den Rezepten 19.4 bis 19.8 behandelt.

Systeme zum Umgang mit großen Mengen von Lokalisierungsdaten werden in den Rezepten 19.9 und 19.10 besprochen. Rezept 19.9 stellt einige einfache Methoden zur Verwaltung der Daten vor, und Rezept 19.10 beschreibt GNU *gettext*, eine umfangreiche Werkzeugsammlung, die Lokalisierungsunterstützung zur Verfügung stellt.

PHP unterstützt außerdem in begrenztem Umfang Unicode. Die Datenkonvertierung von und nach Unicode UTF-8-Codierung wird in Rezept 19.11 erläutert. Die Rezepte 19.12 und 19.13 befassen sich mit dem Handling verschiedener Zeichen-Encodings und dem

UTF-8-Zeichensatz. Dieser deckt die Zeichen aus dem Latin-1-Zeichensatz (ISO-8859-1) und vielen weiteren Zeichensätzen ab.

PHP 6, das sich aktuell noch in der Entwicklung befindet, wird einen erheblich verbesserten Unicode-Support bieten, wie z.B. eine effizientere Verarbeitung von Multibyte-Zeichenketten und ein wesentlich verbessertes Locale-System. Andrei Zmievskis Vortrag »PHP 6 and Unicode«, der auf der Seite *www.gravitonic.com/talks/* erhältlich ist, bietet einen Überblick über die zu erwartenden Verbesserungen bezüglich Unicode in PHP 6.

19.1 Vorhandene Locales auflisten

Problem

Sie wollen wissen, welche Locales Ihr System unterstützt.

Lösung

Verwenden Sie das Programm *locale*, um verfügbare Locales aufzulisten; *locale -a* druckt die Locales aus, die Ihr System unterstützt.

Diskussion

Auf Linux- und Solaris-Systemen können Sie Locales unter */usr/bin/locale* finden. Unter Windows sind die Locales in der Sektion »Ländereinstellungen« der Systemsteuerung zu finden.

Ihre Möglichkeiten unter anderen Betriebssystemen sind unterschiedlich. BSD beispielsweise unterstützt Locales, hat aber kein *locale*-Programm, um Locales aufzulisten. BSD-Locales sind oft unter */usr/share/locale* abgespeichert, sodass ein Blick in dieses Verzeichnis eine Liste von benutzbaren Locales liefern könnte.

Während das Locale-System bei vielen Lokalisierungsaufgaben hilfreich ist, ist seine mangelnde Standardisierung frustrierend. Es gibt keine Garantie dafür, dass verschiedene Systeme die gleichen Locales haben oder auch nur die gleichen Namen für äquivalente Locales verwenden.

Siehe auch

Die Manpage *locale(1)* Ihres Systems. Eine Liste mit Kürzeln, die Windows als Locale-Bezeichner versteht, finden Sie unter *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vclib/html/_crt_language_strings.asp*. Eine Liste mit Kürzeln, die Windows als Länder-/Regionen-Bezeichner versteht, gibt es unter *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vclib/html/_crt_country_strings.asp*.

19.2 Ein bestimmtes Locale verwenden

Problem

Sie wollen PHP anweisen, die Einstellungen eines bestimmten Locales zu verwenden.

Lösung

Rufen Sie `setlocale()` mit der entsprechenden Kategorie und dem entsprechenden Locale auf. So verwenden Sie beispielsweise das Locale `es_US` (US-amerikanisches Spanisch) für alle Kategorien:

```
setlocale(LC_ALL,'es_US');
```

Und so verwenden Sie das Locale `de_AT-Locale` (österreichisches Deutsch) für die Zeit- und Datumsformatierung:

```
setlocale(LC_TIME,'de_AT');
```

Diskussion

Um das derzeitige Locale herauszufinden, ohne es zu ändern, rufen Sie `setlocale()` mit einem `NULL`-Locale auf:

```
print setlocale(LC_ALL, null);
en_US
```

Viele Systeme unterstützen auch Aliasnamen für gebräuchliche Locales, die in einer Datei wie z.B. */usr/share/locale/locale.alias* aufgelistet sind. Diese Datei enthält eine Anzahl Zeilen, u.a.:

```
russian         ru_RU.ISO-8859-5
slovak          sk_SK.ISO-8859-2
slovene         sl_SI.ISO-8859-2
slovenian       sl_SI.ISO-8859-2
spanish         es_ES.ISO-8859-1
swedish         sv_SE.ISO-8859-1
```

Die erste Spalte jeder Zeile ist ein Aliasname, die zweite Spalte gibt den Locale und den Zeichensatz an, auf den der Aliasname verweist. In Aufrufen von `setlocale()` können Sie den Aliasnamen anstatt des dazugehörigen Strings verwenden, auf den er verweist. Beispielsweise können Sie

```
setlocale(LC_ALL,'swedish');
```

anstatt

```
setlocale(LC_ALL,'sv_SE.ISO-8859-1');
```

angeben. Unter Windows müssen Sie zum Wechseln des Locale in die Systemsteuerung gehen. Im Abschnitt »Ländereinstellungen« können Sie sich dort ein neues Locale aussuchen und seine Einstellungen anpassen.

Wenn Sie sich nicht sicher sind, ob ein bestimmtes Locale auf Ihrem System vorhanden ist, können Sie in `setlocale()` auch ein Array mit Locales angeben. Dieses Array wird Element für Element ausprobiert, bis ein vorhandenes Locale gefunden wird:

```
setlocale(LC_ALL,array('de_AT','de_DE','de_CH'));
```

Wenn Sie PHP in einer Multi-Threaded-Umgebung einsetzen, kann das Ändern der Locale-Einstellung unvorhersehbare Auswirkungen haben. Durch einen Aufruf von `setlocale()` wird für alle Threads des aktuellen Prozesses die Locale-Einstellung verändert! Verwenden Sie `setlocale()`, sollten Sie PHP in einer Single-Threaded-Umgebung einsetzen oder sehr genau die Thread-Sicherheit Ihrer Skripten prüfen.

Siehe auch

Rezept 19.3 zeigt, wie ein Default-Locale gesetzt wird; die Dokumentation zu `setlocale()` unter *http://www.php.net/setlocale*.

19.3 Das Default-Locale setzen

Problem

Sie möchten ein Locale setzen, das alle Ihre PHP-Programme verwenden können.

Lösung

Am Anfang einer Datei, die über die Konfigurationsdirektive `auto_prepend_file` geladen wird, rufen Sie `setlocale()` auf, um Ihr gewünschtes Locale zu setzen:

```
setlocale(LC_ALL,'es_US');
```

Um die Locale-Einstellung wieder auf die Systemeinstellung zurückzusetzen, rufen Sie `setlocale()` mit `NULL` als zweitem Parameter auf:

```
setlocale(LC_ALL, null);
```

Diskussion

Auch wenn Sie vor dem Start Ihres Webservers oder des PHP-Programms entsprechende Umgebungsvariablen setzen, ändert PHP sein Locale nicht, bis Sie `setlocale()` aufrufen. Nachdem Sie die Umgebungsvariable `LC_ALL` auf `es_US` gesetzt haben, fährt PHP beispielsweise immer noch das voreingestellte C-Locale.

Siehe auch

Rezept 19.2 zeigt, wie Sie ein bestimmtes Locale verwenden; die Dokumentation zu `setlocale()` unter *http://www.php.net/setlocale*.

19.4 Textmeldungen lokalisieren

Problem

Sie wollen Textmeldungen in einer dem Locale entsprechenden Sprache anzeigen.

Lösung

Unterhalten Sie einen Meldungskatalog mit Wörtern und Phrasen und holen Sie sich den entsprechenden String aus dem Meldungskatalog, bevor Sie ihn ausgeben. Hier ist ein einfacher Meldungskatalog mit Lebensmitteln in amerikanischem Englisch und in Deutsch sowie einer Funktion, die die Wörter aus dem Katalog liest:

```
$messages = array ('en_US' =>
             array(
              'My favorite foods are' => 'My favorite foods are',
              'french fries' => 'french fries',
              'biscuit'      => 'biscuit',
              'candy'        => 'candy',
              'potato chips' => 'potato chips',
              'cookie'       => 'cookie',
              'corn'         => 'corn',
              'eggplant'     => 'eggplant'
             ),
            'de_DE' =>
             array(
              'My favorite foods are' => 'Meine Lieblingsspeisen sind',
              'french fries' => 'Pommes',
              'biscuit'      => 's&uuml;&szlig;e Br&ouml;tchen',
              'candy'        => 'S&uuml;&szlig;igkeiten',
              'potato chips' => 'Kartoffelchips',
              'cookie'       => 'Kekse',
              'corn'         => 'Mais',
              'eggplant'     => 'Auberginen'
             )
            );

function msg($s) {
  global $LANG;
  global $messages;
  if (isset($messages[$LANG][$s])) {
    return $messages[$LANG][$s];
  } else {
    error_log("l10n-Fehler: SPRACHE: $LANG, Meldung: '$s'");
  }
}
```

Diskussion

Dieses kurze Programm verwendet den Meldungskatalog, um eine Liste von Lebensmitteln auszugeben:

```
$LANG = 'de_DE';
print msg('My favorite foods are').":\n";
print msg('french fries')."\n";
print msg('potato chips')."\n";
print msg('corn')."\n";
print msg('candy')."\n";
Meine Lieblingsspeisen sind:
Pommes
Kartoffelchips
Mais
Süßigkeiten
```

Um das Programm amerikanisches Englisch anstelle von deutschem Deutsch ausgeben zu lassen, setzen Sie einfach `$LANG` zu `en_US`.

Sie können die Meldungsabruf-Funktion `msg( )` mit `sprintf( )` kombinieren, um Phrasen zu speichern, in die Werte eingesetzt werden müssen. Nehmen Sie beispielsweise den deutschen Satz »Ich bin 12 Jahre alt« an. In Spanisch heißt die entsprechende Phrase »Tengo 12 años«. Die spanische Phrase lässt sich aber nicht durch das Zusammenflicken von Übersetzungen von »Ich bin«, der Zahl 12 und »Jahre alt« konstruieren. Stattdessen speichern Sie sie in den Meldungskatalogen als Format-Strings ab, wie sie mit `sprintf( )` verwendet werden:

```
$messages = array ('de_DE' => array('Ich bin X Jahre alt' => 'Ich bin %d Jahre alt'),
                   'es_US' => array('Ich bin X Jahre alt' => 'Tengo %d años.')
            );
```

Sie können das Ergebnis von `msg( )` dann an `sprintf( )` als Format-String übergeben:

```
$LANG = 'es_US';
print sprintf(msg('Ich bin X Jahre alt'),12);
Tengo 12 años.
```

Bei Sätzen, in denen der ersetzte Wert in der anderen Sprache in einer anderen Reihenfolge stehen muss, unterstützt `sprintf( )` die Änderung der Reihenfolge im Argument:

```
$messages = array ('de_DE' =>
                    array('Ich bin X Jahre und Y Monate alt' =>
                          'Ich bin %d Jahre und %d Monate alt'),
                   'es_US' =>
                    array('Ich bin X Jahre und Y Monate alt' =>
                          'Tengo %2$d meses y %1$d años.')
            );
```

In beiden Sprachen können Sie `sprintf( )` mit der gleichen Reihenfolge der Argumente aufrufen (z.B. zuerst Jahre, dann Monate):

```
$LANG = 'es_US';
print sprintf(msg('Ich bin X Jahre und Y Monate alt'),12,7);
Tengo 7 meses y 12 años.
```

Im Format-String weist %2$ die Funktion sprintf( ) an, das zweite Argument zu verwenden, und %1$ weist sie an, das erste einzusetzen.

Wenn Sie PHP 5.3 oder höher einsetzen, können Sie alternativ zu sprintf() auch mit der Klasse MessageFormatter arbeiten, um Platzhalter in Ihre Meldungen einzufügen:

```
$args = array(12, 7);
print MessageFormatter::formatMessage("de_DE", "Ich bin {0,number,integer}
Jahre und {1,number,integer} Monate alt.", $args);
print MessageFormatter::formatMessage("de_DE", "Ich bin {1,number,integer}
Monate und {0,number,integer} Jahre alt.", $args);
Ich bin 12 Jahre und 7 Monate alt.
Ich bin 7 Monate und 12 Jahre alt.
```

Die Formatierungsmöglichkeiten stehen denen von sprintf() in nichts nach. Sie können sogar Zahlen, Datumswerte und auch Währungen einfach sprachabhängig formatieren. Weitere Details zu den Format-Strings stellt die Seite *http://www.icu-project.org/apiref/icu4c/classMessageFormat.html* bereit.

Anstatt eines Strings in einem Array können diese Phrasen auch als Rückgabewert einer Funktion gespeichert werden. Das Speichern der Sätze als Funktion macht die Verwendung von sprintf( ) überflüssig. Funktionen, die einen Satz zurückgeben, sehen wie folgt aus:

```
// deutsche Version
function ich_bin_X_jahre_alt($age) {
 return "Ich bin $age Jahre alt.";
}

// spanische Version
function ich_bin_X_jahre_alt($age) {
 return "Tengo $age años.";
}
```

Wenn Teile des Meldungskatalogs in ein Array gehören und andere Teile in Funktionen, ist ein Objekt eine nützliche Konstruktion zum Aufbewahren des Meldungskatalogs für eine Sprache. Ein Basisobjekt und zwei einfache Meldungskataloge sehen so aus:

```
class pc_MC_Base {
  var $messages;
  var $lang;

  function msg($s) {
    if (isset($this->messages[$s])) {
      return $this->messages[$s];
    } else {
      error_log("l10n error: LANG: $this->lang, message: '$s'");
    }
  }

}
```

```
class pc_MC_es_US extends pc_MC_Base {

  function __construct() {
    $this->lang = 'es_US';
    $this->messages = array ('chicken' => 'pollo',
                'cow'     => 'vaca',
                'horse'   => 'caballo'
                );
  }

  function i_am_X_years_old($age) {
    return "Tengo $age años";
  }
}

class pc_MC_en_US extends pc_MC_Base {

  function __construct() {
    $this->lang = 'en_US';
    $this->messages = array ('chicken' => 'chicken',
                'cow'     => 'cow',
                'horse'   => 'horse'
                );
  }

  function i_am_X_years_old($age) {
    return "I am $age years old.";
  }
}
```

Jedes Nachrichtenkatalog-Objekt erweitert die `pc_MC_Base`-Klasse um die `msg( )`-Methode und definiert dann seine eigenen Nachrichten (in seinem Konstruktor) und seine eigenen Funktionen, die Phrasen zurückgeben. Hier sehen Sie, wie Text in Spanisch gedruckt wird:

```
$MC = new pc_MC_es_US;

print $MC->msg('cow');
print $MC->i_am_X_years_old(15);
```

Um den gleichen Text auf Englisch zu drucken, muss `$MC` nur als ein `pc_MC_en_US`-Objekt anstatt als `pc_MC_es_US`-Objekt erzeugt werden. Der restliche Code bleibt unverändert.

Siehe auch

Weitere Informationen zu `MessageFormatter` finden Sie unter *http://php.net/manual/class.messageformatter.php*. Die Einleitung zu Kapitel 7 bespricht Objektvererbung; die Dokumentation zu `sprintf( )` unter *http://www.php.net/sprintf*.

19.5 Datum und Uhrzeiten lokalisieren

Problem

Sie wollen Datum und Uhrzeiten Locale-abhängig anzeigen.

Lösung

In PHP-Versionen ab 5.3 können Sie die Klasse `IntlDateFormatter` verwenden:

```
$datum = time();
$fmt = new IntlDateFormatter( "de_DE" , IntlDateFormatter::FULL,
        IntlDateFormatter::FULL, 'Europe/Berlin',
        IntlDateFormatter::GREGORIAN);
$fmt->format($datum);
```

Sollten Sie nicht über PHP 5.3 oder höher verfügen, verwenden Sie den Format-String `%c` von `strftime()`:

```
 print strftime('%c');
```

Sie können `strftime()`-Format-Strings auch als Meldungen in Ihrem Meldungskatalog speichern:

```
$MC = new pc_MC_es_US;
print strftime($MC->msg('%Y-%m-%d'));
```

Diskussion

Die *intl*-Erweiterung gehört seit PHP 5.3 zum Lieferumfang von PHP. Sie können die enthaltene Klasse `IntlDateFormatter` nutzen, um Datumswerte Ihren Bedürfnissen entsprechend anzupassen:

```
$datum = time();

$fmt = new IntlDateFormatter( "de_DE" , IntlDateFormatter::FULL,
        IntlDateFormatter::FULL, 'Europe/Berlin',
        IntlDateFormatter::GREGORIAN);
print "FULL: " . $fmt->format($datum);

$fmt = new IntlDateFormatter( "de_DE" , IntlDateFormatter::LONG,
        IntlDateFormatter::LONG, 'Europe/Berlin',
        IntlDateFormatter::GREGORIAN);
print "LONG: " . $fmt->format($datum);

$fmt = new IntlDateFormatter( "de_DE" , IntlDateFormatter::MEDIUM,
        IntlDateFormatter::MEDIUM, 'Europe/Berlin',
        IntlDateFormatter::GREGORIAN);
print "MEDIUM: " . $fmt->format($datum);

$fmt = new IntlDateFormatter( "de_DE" , IntlDateFormatter::SHORT,
        IntlDateFormatter::SHORT, 'Europe/Berlin',
```

```
        IntlDateFormatter::GREGORIAN);
print "SHORT: " . $fmt->format($datum);

$fmt = new IntlDateFormatter( "de_DE" , IntlDateFormatter::NONE,
        IntlDateFormatter::NONE, 'Europe/Berlin',
        IntlDateFormatter::GREGORIAN);
print "NONE: " . $fmt->format($datum);
FULL: Sonntag, 28. Juni 2009 15:48:54 Deutschland
LONG: 28. Juni 2009 15:48:54 MESZ
MEDIUM: 28.06.2009 15:48:54
SHORT: 28.06.09 15:48
NONE: 20090628 03:48 nachm.
```

Beim Erzeugen einer Instanz der `IntlDateFormatter`-Klasse können Sie angeben, welches Locale und welches Datums- und Zeitformat verwendet werden soll. Optional ist die Angabe der Zeitzone und der Kalendertyp. Ist die Instanz erstellt, können Sie die `format()`-Methode aufrufen und dieser das Datum z.B. als Epochen-Zeitstempel übergeben. Für das Datums- und Zeitformat können Sie eine der im Beispiel gezeigten Konstanten verwenden, um den Umfang der ausgegebenen Informationen zu steuern. Die Klasse `IntlDateFormatter` erlaubt zudem, das Muster der ausgegebenen Informationen explizit über die Methode `setPattern()` festzulegen.

Der Format-String `%c` weist `strftime()` an, die gewünschte Datums- und Zeitangabe im Format des aktuellen Locale zurückzugeben. Die schnellste Art, einen nach dem Locale formatierten Zeit-String zu erzeugen, ist:

```
print strftime('%c');
```

Dieser Code produziert verschiedene Ergebnisse:

```
Tue Aug 13 18:37:11 2002     // in dem Default-Locale C
mar 13 ago 2002 18:37:11 EDT // in dem Locale es_US
mar 13 août 2002 18:37:11 EDT // in dem Locale fr_FR
```

Obwohl der formatierte Zeit-String, den `%c` erstellt, der richtige für das jeweilige Locale ist, ist er nicht sehr flexibel. Wenn Sie zum Beispiel nur die Zeit wollen, müssen Sie einen anderen Format-String an `strftime()` übergeben. Diese Format-Strings unterscheiden sich allerdings auch wieder von Locale zu Locale. Bei einigen Locales ist wahrscheinlich die Anzeige der Stunden im 12-Stunden-Format mit A.M.-/P.M.-Angabe angebracht, bei anderen Locales dagegen eine 24-Stunden-Anzeige. Um passende Zeit-Strings für ein Locale anzuzeigen, fügen Sie für jedes gewünschte Zeitformat dem `$messages`-Array des Locales Elemente hinzu. Der Schlüssel für ein bestimmtes Zeitformat, wie zum Beispiel `%H:%M`, ist für jedes Locale die gleiche. Allerdings kann der Wert unterschiedlich sein, wie zum Beispiel `%H:%M` für eine 24-Stunden-Anzeige oder `%I:%M %P` für eine 12-Stunden-Anzeige. Dann schlagen Sie einfach den passenden Format-String nach und übergeben ihn an `strftime()`:

```
$MC = new pc_MC_es_US;

print strftime($MC->msg('%H:%M'));
```

Das Ändern des Locale hat keinen Einfluss auf die Zeitzone, sondern nur auf das Format des angezeigten Ergebnisses.

Siehe auch

Weitere Informationen zur Klasse `IntlDateFormatter` finden Sie im PHP-Manual unter *http://php.net/manual/class.intldateformatter.php*. Rezept 3.4 bespricht die Format-Strings, die von `strftime( )` angenommen werden; Rezept 3.12 behandelt das Ändern von Zeitzonen in Ihrem Programm; die Dokumentation zu `strftime( )` unter *http://www.php.net/strftime*.

19.6 Lokalisierung von Währungen

Problem

Sie wollen Geldbeträge in Währungen in einem Locale-abhängigen Format anzeigen.

Lösung

Wenn Sie PHP 5.3 oder höher einsetzen, sollten Sie die Klasse `NumberFormatter` der *intl*-Erweiterung verwenden:

```
$fmt = new NumberFormatter( 'de_DE', NumberFormatter::CURRENCY );
echo $fmt->formatCurrency(1234567.891234567890000, "EUR");
```

In früheren PHP-Versionen verwenden Sie die in Beispiel 19-1 gezeigte Funktion `pc_format_currency( )`, um einen String mit der passenden Formatierung zu erstellen, zum Beispiel:

```
setlocale(LC_ALL,'fr_CA');
print pc_format_currency(-12345678.45);
(12 345 678,45 $)
```

Diskussion

Die Klasse `NumberFormatter` kann zur sprachabhängigen Formatierung aller möglichen Zahlen verwendet werden, unter anderem natürlich auch für Währungen:

```
$fmt = new NumberFormatter('de_DE', NumberFormatter::CURRENCY);
echo $fmt->formatCurrency(1234567.891234567890000, "EUR");
echo $fmt->formatCurrency(1234567.891234567890000, "USD");
1.234.567,89 €
1.234.567,89 $

$fmt = new NumberFormatter('en_US', NumberFormatter::CURRENCY);
echo $fmt->formatCurrency(1234567.891234567890000, "EUR");
echo $fmt->formatCurrency(1234567.891234567890000, "USD");
```

```
€1,234,567.89
$1,234,567.89
```

Sie können mit dieser Klasse sogar die ausgeschriebene Form eines Betrags ausgeben:

```
$fmt = new NumberFormatter( 'de_DE', NumberFormatter::CURRENCY_CODE);
echo $fmt->formatCurrency(4726272.12, "EUR");
vier Millionen siebenhundertsechsundzwanzigtausendzweihundertzweiundsiebzig komma eins zwei
```

Die in Beispiel 19-1 vorgestellte Funktion `pc_format_currency()` bezieht die Informationen zur Währungsformatierung von `localeconv()` und benutzt dann `number_format()` und ein bisschen Logik, um den richtigen String zu konstruieren.

Beispiel 19-1: pc_format_currency

```
function pc_format_currency($amt) {
    // Locale-spezifische Informationen zur Währungsformatierung ermitteln.
    $a = localeconv();

    // Vorzeichen von $amt berechnen und dann entfernen.
    if ($amt < 0) { $sign = -1; } else { $sign = 1; }
    $amt = abs($amt);
    // $amt mit der passenden Gruppierung sowie Dezimalzeichen
    // und Nachkommastellen formatieren.
    $amt = number_format($amt,$a['frac_digits'],$a['mon_decimal_point'],
                         $a['mon_thousands_sep']);

    // Wo kommen das Währungssymbol und die Kennzeichnung positiv/negativ hin?
    $currency_symbol = $a['currency_symbol'];
    // Ist $amt >= 0?
    if (1 == $sign) {
        $sign_symbol  = 'positive_sign';
        $cs_precedes  = 'p_cs_precedes';
        $sign_posn    = 'p_sign_posn';
        $sep_by_space = 'p_sep_by_space';
    } else {
        $sign_symbol  = 'negative_sign';
        $cs_precedes  = 'n_cs_precedes';
        $sign_posn    = 'n_sign_posn';
        $sep_by_space = 'n_sep_by_space';
    }
    if ($a[$cs_precedes]) {
        if (3 == $a[$sign_posn]) {
            $currency_symbol = $a[$sign_symbol].$currency_symbol;
        } elseif (4 == $a[$sign_posn]) {
            $currency_symbol .= $a[$sign_symbol];
        }
        // Währungssymbol vorangestellt.
        if ($a[$sep_by_space]) {
            $amt = $currency_symbol.' '.$amt;
        } else {
            $amt = $currency_symbol.$amt;
        }
    } else {
        // Währungssymbol an Betrag angehängt.
        if ($a[$sep_by_space]) {
```

Beispiel 19-1: pc_format_currency (Fortsetzung)

```
            $amt .= ' '.$currency_symbol;
        } else {
            $amt .= $currency_symbol;
        }
    }
    if (0 == $a[$sign_posn]) {
        $amt = "($amt)";
    } elseif (1 == $a[$sign_posn]) {
        $amt = $a[$sign_symbol].$amt;
    } elseif (2 == $a[$sign_posn]) {
        $amt .= $a[$sign_symbol];
    }
    return $amt;
}
```

Der Code in `pc_format_currency( )`, der das Währungssymbol und das Vorzeichen an der richtigen Stelle platziert, ist für positive und negative Zahlen fast gleich. Er verwendet nur unterschiedliche Elemente des Arrays, das von `localeconv( )` zurückgegeben wird. Die verwendeten Elemente des von `localeconv( )` zurückgegebenen Arrays sind in Tabelle 19-1 aufgelistet.

Tabelle 19-1: Währungsbezogene Informationen, die von localeconv() zurückgegeben werden

Array-Element	Beschreibung
`currency_symbol`	Währungssymbol des Locale
`mon_decimal_point`	Dezimalzeichen für Geldbeträge
`mon_thousands_sep`	Zeichen zur Tausendertrennung bei Geldbeträgen
`positive_sign`	Kennzeichnung für positive Werte
`negative_sign`	Kennzeichnung für negative Werte
`frac_digits`	Anzahl der Nachkommastellen
`p_cs_precedes`	1, wenn `currency_symbol` einem positiven Wert voranzustellen ist; 0, wenn es angehängt werden soll
`p_sep_by_space`	1, wenn ein Leerzeichen zwischen dem Währungssymbol und einem positiven Betrag stehen soll; 0, wenn nicht
`n_cs_precedes`	1, wenn `currency_symbol` einem negativen Wert voranzustellen ist; 0, wenn es angehängt werden soll
`n_sep_by_space`	1, wenn ein Leerzeichen zwischen dem Währungssymbol und einem negativen Betrag stehen soll; 0, wenn nicht
`p_sign_posn`	Position der positiven Kennzeichnung: • 0, wenn Klammern den Betrag und `currency_symbol` umschließen sollen • 1, wenn der Kennzeichnungs-String vor dem Betrag und `currency_symbol` stehen soll • 2, wenn der Kennzeichnungs-String nach dem Betrag und `currency_symbol` stehen soll • 3, wenn der Kennzeichnungs-String direkt vor `currency_symbol` stehen soll • 4, wenn der Kennzeichnungs-String direkt nach `currency_symbol` stehen soll
`n_sign_posn`	Position der negativen Kennzeichnung: gleiche Werte möglich wie für `p_sign_posn`

In der C-Bibliothek gibt es eine Funktion namens `strfmon()`, die mit Währungen das Gleiche macht wie `strftime()` mit Datum und Zeit. Sie ist aber in PHP nicht implementiert. Die Funktion `pc_format_currency()` deckt die meisten dieser Fähigkeiten ab.

Siehe auch

Weitere Informationen zu `NumberFormatter` finden Sie unter *http://php.net/manual/class.numberformatter.php*. Rezept 2.9 bespricht ebenfalls `number_format()`. Dokumentationen zu `localeconv()` unter *http://www.php.net/localeconv* und `number_format()` unter *http://www.php.net/number-format*.

19.7 Bilder lokalisieren

Problem

Sie wollen Bilder mit integriertem Text darstellen, der in einer für das Locale passenden Sprache geschrieben ist.

Lösung

Erstellen Sie ein Bildverzeichnis für jedes Locale, das Sie unterstützen wollen, und ein globales Bildverzeichnis für Bilder, die keine Locale-abhängigen Informationen enthalten. Machen Sie eine Kopie von jedem Locale-abhängigen Bild und speichern Sie es in einem passenden Locale-abhängigen Verzeichnis. Achten Sie dabei darauf, dass die Bilder in den verschiedenen Verzeichnissen den gleichen Dateinamen bekommen. Anstatt die Bild-URLs direkt auszudrucken, verwenden Sie eine Wrapper-Funktion – im Prinzip ähnlich wie bei der Funktion `msg()` in Rezept 19.4 –, die Locale-abhängigen Text ausdruckt.

Diskussion

Die Wrapper-Funktion `img()` sucht zuerst nach einer Locale-abhängigen Bildversion und dann nach einer globalen Version. Wenn keine der beiden vorhanden ist, gibt sie eine Meldung im Fehler-Log aus:

```
$image_base_path = '/usr/local/www/images';
$image_base_url  = '/images';

function img($f) {
    global $LANG;
    global $image_base_path;
    global $image_base_url;

    if (is_readable("$image_base_path/$LANG/$f")) {
        print "$image_base_url/$LANG/$f";
```

```
    } elseif (is_readable("$image_base_path/global/$f")) {
        print "$image_base_url/global/$f";
    } else {
        error_log("l10n error: LANG: $lang, image: '$f'");
    }
}
```

Diese Funktion muss sowohl den Pfad zur Bilddatei im Dateisystem kennen (`$image_base_path`) als auch den Pfad zum Bild relativ zur Basis-URL Ihrer Seite (*/images*). Den ersten Pfad verwendet die Funktion, um zu prüfen, ob die Datei gelesen werden kann, den zweiten, um eine passende URL für das Bild zu konstruieren.

Ein lokalisiertes Bild muss in jedem Lokalisierungsverzeichnis den gleichen Dateinamen haben. Zum Beispiel sollte ein Bild, auf dem auf einem gelben Stern »New!« zu lesen ist, sowohl im *images/en_US*-Verzeichnis als auch im *images/es_US*-Verzeichnis *new.gif* genannt werden, selbst wenn die Datei *images/es_US/new.gif* ein Bild eines gelben Sterns mit dem Schriftzug »¡Nuevo!« ist.

Vergessen Sie nicht, dass Sie auch den Alternativtext in Ihren Image-Tags lokalisieren sollten. Ein vollständig lokalisiertes `<img>`-Tag sieht so aus:

```
printf('<img src="%s" alt="%s">',img('cancel.png'),msg('Cancel'));
```

Wenn die lokalisierten Versionen eines bestimmten Bilds unterschiedliche Abmessungen haben, speichern Sie einfach die Bildhöhe und -breite mit im Meldungskatalog:

```
printf('<img src="%s" alt="%s" height="%d" width="%d">',
       img('cancel.png'),msg('Cancel'),
       msg('img-cancel-height'),msg('img-cancel-width'));
```

Bei den lokalisierten Meldungen für `img-cancel-height` und `img-cancel-width` handelt es sich nicht um Text-Strings, sondern um ganze Zahlen, die die Dimensionen des Bilds *cancel.png* im jeweiligen Locale angeben.

Siehe auch

Rezept 19.4 behandelt Locale-abhängige Meldungskataloge.

19.8 Eingebundene Dateien lokalisieren

Problem

Sie wollen Locale-abhängige Dateien in Ihre Seiten einbinden.

Lösung

Modifizieren Sie den `include_path` dynamisch, nachdem Sie das notwendige Locale bestimmt haben:

```
$base = '/usr/local/php-include';
$LANG = 'en_US';

$include_path = ini_get('include_path');
ini_set('include_path',"$base/$LANG:$base/global:$include_path");
```

Diskussion

Die Variable `$base` enthält den Namen des Basisverzeichnisses für Ihre einzubindenden lokalisierten Dateien. Dateien, die nicht Locale-abhängig sind, werden in dem Unterverzeichnis *global* von `$base` abgelegt. Locale-abhängige Dateien werden in einem nach ihrem Locale benannten Unterverzeichnis abgelegt (z.B. *en_US*). Wenn Sie das Locale-abhängige Verzeichnis und dann das globale Verzeichnis vor den Include-Pfad schreiben, werden diese beiden Verzeichnisse zu den ersten Stellen, an denen sich PHP nach einzubeziehenden Dateien umsieht. Dadurch, dass das Locale-abhängige Verzeichnis zuerst genannt wird, ist sichergestellt, dass nicht-lokalisierte Informationen nur dann geladen werden, wenn keine lokalisierten Informationen vorhanden sind.

Diese Technik ähnelt der der `img( )`-Funktion aus Rezept 19.7. Hier können Sie allerdings den `include_path` von PHP ausnutzen, um das Verzeichnis automatisch setzen zu lassen. Am besten ist es, `include_path` so früh wie möglich zu setzen, vorzugsweise am Anfang einer Datei, die bei jedem Request über `auto_prepend_file` geladen wird.

Siehe auch

Die Dokumentation zu `include_path` unter *http://www.php.net/manual/function.set-include-path.php*.

19.9 Lokalisierungsressourcen verwalten

Problem

Sie müssen Ihre verschiedenen Meldungskataloge und Bilder verwalten.

Lösung

Zwei Techniken erleichtern Ihnen die Verwaltung Ihrer Lokalisierungsressourcen. Die erste besteht darin, das Objekt für eine neue Sprache, beispielsweise kanadisches Englisch, durch die Erweiterung eines Objekts einer bestehenden Sprache zu erzeugen, z.B. amerikanisches Englisch. Sie müssen dann im neuen Objekt nur die Wörter und Phrasen ändern, die sich von der ursprünglichen Sprache unterscheiden.

Die zweite Technik: Um einen Überblick darüber zu behalten, welche Phrasen noch in eine neue Sprache zu übersetzen sind, setzen Sie Dummy-Methoden in Ihr neues Sprach-

objekt, die den gleichen Wert haben wie in Ihrem Basisobjekt. Indem Sie herausfinden, welche Werte in der Basissprache mit Werten in der neuen Sprache übereinstimmen, können Sie sich eine Liste der noch zu übersetzenden Wörter und Phrasen erstellen.

Diskussion

Das Programm *catalog-compare.php*, das in Beispiel 19-2 gezeigt wird, druckt Meldungen aus, die in zwei Katalogen vorkommen, sowie Meldungen, die in einem Katalog fehlen, aber in einem anderen vorkommen.

Beispiel 19-2: catalog-compare.php

```
$base = 'pc_MC_'.$_SERVER['argv'][1];
$other  = 'pc_MC_'.$_SERVER['argv'][2];

require 'pc_MC_Base.php';
require "$base.php";
require "$other.php";

$base_obj = new $base;
$other_obj = new $other;

/* Auf Meldungen in der anderen Klasse prüfen, die
 * die gleichen wie in der Basisklasse sind oder die
 * in der Basisklasse vorkommen, aber in der anderen Klasse fehlen. */
foreach ($base_obj->messages as $k => $v) {
    if (isset($other_obj->messages[$k])) {
        if ($v == $other_obj->messages[$k]) {
            print "GLEICH: $k\n";
        }
    } else {
        print "FEHLEN: $k\n";
    }
}

/* Welche Medungen kommen in der anderen Klasse vor,
 * fehlen aber in der Basisklasse? */
foreach ($other_obj->messages as $k => $v) {
    if (! isset($base_obj->messages[$k])) {
        print "FEHLEN (IN BASIS): $k\n";
    }
}
```

Um dieses Programm zu verwenden, setzen Sie jedes Meldungskatalog-Objekt in eine Datei, die den gleichen Namen hat wie das Objekt (z.B. sollte die Klasse `pc_MC_en_US` in einer Datei namens *pc_MC_en_US.php* und die Klasse `pc_MC_es_US` in einer Datei namens *pc_MC_es_US.php* stehen). Sie rufen das Programm dann mit den beiden Locale-Namen als Kommandozeilenparameter auf:

```
% php catalog-compare.php en_US es_US
```

In einem Webkontext kann es nützlich sein, für jeden Request unterschiedliche Locales und Meldungskataloge zu verwenden. Das zu benutzende Locale kann vom Browser kommen (durch einen Accept-Language-Header), oder es kann explizit durch den Server gesetzt werden (man könnte beispielsweise unterschiedliche virtuelle Hosts einrichten, die die gleichen Inhalte in verschiedenen Sprachen anzeigen).

Wenn Sie einen vom Browser übergebenen Wert zum Setzen von Locales verwenden wollen, ergibt sich ein potenzielles Sicherheitsproblem. Da der Wert von Accept-Language-Headern usw. vom Browser bestimmt wird (d.h. letzten Endes von einem Hacker an einem beliebigen Computer im Internet gesetzt werden kann), sollte er nicht direkt in einem Dateinamen für eine include-Funktion oder einem require-Statement verwendet werden. Auf diese Weise ließen sich sonst beliebige Dateien »hinzuladen«, was zum Ausspionieren von Dateien oder auch zur Ausführung von PHP-Code anderer Dateien auf dem Server führen könnte. Wenn derselbe Code je nach Request einen Meldungskatalog auswählen soll, können Sie die Meldungskatalogklasse wie folgt sicher instantiieren:

```
// $locale enthält den vom Browser übergebenen Locale-Wunsch
switch ($locale) {
case "en_US":
case "en_CA":
case "de_DE":
case "fr_FR":
$safelocale = $locale;
default:           // potenziell verdächtige Locale-Werte
$safelocale = "en_US";
}

$classname = "pc_MC_$safelocale.php";

require 'pc_MC_Base.php';
require $classname.'.php';

$MC = new $classname;
```

Siehe auch

Rezept 19.4 behandelt Meldungskataloge.

19.10 gettext verwenden

Problem

Sie möchten ein zusammenhängendes System zur Erzeugung, Wartung und Verwendung von Meldungskatalogen erstellen.

Lösung

Verwenden Sie die *gettext*-Erweiterung von PHP, die Ihnen Zugriff auf die GNU-*gettext*-Utilities gibt:

```
bindtextdomain('gnumeric','/usr/share/locale');
textdomain('gnumeric');

$languages = array('en_CA','da_DK','de_AT','fr_FR');
foreach ($languages as $language) {
  setlocale(LC_ALL, $language);
  print gettext(" Unknown formula")."\n";
}
```

Diskussion

gettext ist eine Sammlung von Werkzeugen, die es Ihrer Anwendung leichter machen, vielsprachige Meldungen zu erzeugen. Wenn Sie PHP mit der Option `--with-gettext` kompilieren, stehen Ihnen Funktionen zu Verfügung, die den entsprechenden Text aus Meldungskatalogen im *gettext*-Format lesen. Zur Bearbeitung der Meldungskataloge gibt es eine Anzahl externer Programme.

Mit *gettext* lassen sich Meldungen in sogenannte Domains unterteilen. Alle Meldungen für eine bestimmte Domain werden in derselben Datei gespeichert. `bindtextdomain()` teilt *gettext* mit, wo es den Meldungskatalog für eine bestimmte Domain finden kann. Ein Aufruf

```
bindtextdomain('gnumeric','/usr/share/locale')
```

zeigt beispielsweise an, dass sich der Meldungskatalog für die Domain `gnumeric` in dem Locale `en_CA` in der Datei */usr/share/locale/en_CA/LC_MESSAGES/gnumeric.mo* befindet.

Die Funktion `textdomain('gnumeric')` setzt die Default-Domain zu `gnumeric`. Der Aufruf von `gettext()` liest eine Meldung aus der Default-Domain. Es gibt auch noch andere Funktionen, wie beispielsweise `dgettext()`, mit denen Sie Meldungen aus einer anderen Domain lesen können. Wenn Sie `gettext()` (oder `dgettext()`) aufrufen, gibt sie die entsprechende Meldung für das aktuelle Locale zurück. Wenn es für das aktuelle Locale keine Meldung gibt, die zu dem übergebenen Argument passt, gibt `gettext()` (oder `dgettext()`) einfach sein Argument zurück. Das bedeutet, dass Ihr Code Deutsch (oder was auch immer Ihre Basissprache ist) für diejenigen Meldungen ausgibt, die Sie noch nicht übersetzt haben.

Wenn Sie die Default-Domain mit `textdomain()` setzen, macht das jeden weiteren Abruf einer Meldung aus dieser Domain etwas einfacher, da Sie jetzt nur noch `gettext('Guten Morgen')` statt `dgettext('domain','Guten Morgen')` aufrufen müssen. Wenn Ihnen auch `gettext('Guten Morgen')` noch zu viel Schreibarbeit ist, können Sie den undokumentierten Funktions-Alias `_()` für `gettext()` verwenden. Statt `gettext('Guten Morgen')` schreiben Sie einfach nur `_('Guten Morgen')`.

Auf der *gettext*-Website gibt es hilfreiche und detaillierte Informationen darüber, wie Sie den Informationsfluss zwischen Programmierern und Übersetzern managen und *gettext* effizient einsetzen können. Sie enthält auch Informationen über andere Tools, die Sie zur Verwaltung Ihrer Meldungskataloge verwenden können, wie zum Beispiel einen speziellen Modus für GNU Emacs.

Ein Nachteil von *gettext* ist die Tatsache, dass diese Bibliothek nicht Thread-sicher ist. Wie in Rezept 19.2 schon bei der Funktion `setlocale()` angesprochen, wirken sich Änderungen an *gettext*-Einstellungen auf alle Threads im gleichen Prozess aus. Es ist also zu empfehlen, dass Sie *gettext* nur in einer Single-Threaded-Umgebung einsetzen. Andernfalls sollten Sie sehr genau die Thread-Sicherheit Ihrer Skripten überprüfen und sicherstellen.

Siehe auch

Die Dokumentation zu *gettext* unter *http://www.php.net/gettext*; die *gettext*-Bibliothek unter *http://www.gnu.org/software/gettext/gettext.html*.

19.11 Unicode-Zeichen lesen und ausgeben

Problem

Sie möchten Unicode-codierte Zeichen aus einer Datei, Datenbank oder aus einem Formular lesen, oder Sie möchten Unicode-codierte Zeichen ausgeben.

Lösung

Verwenden Sie `utf8_encode()`, um aus einem Byte bestehende ISO-8859-1-codierte Zeichen in UTF-8 umzuwandeln:

```
print utf8_encode('Kurt Gödel ist nett.');
```

Verwenden Sie `utf8_decode()`, um UTF-8-codierte Zeichen in aus einem Byte bestehende ISO-8859-1-codierte Zeichen umzuwandeln:

```
print utf8_decode("Kurt G\xc3\xb6del ist nett.");
```

Diskussion

Es gibt 256 mögliche ASCII-Zeichen. Die Zeichen zwischen den Codes 0 und 127 sind standardisiert: Steuerzeichen, Buchstaben, Ziffern und Interpunktionszeichen. Für die Zeichen, auf die die Codes 128 bis 255 abbilden, gibt es allerdings unterschiedliche Regeln. Eine Codierung heißt ISO-8859-1. Sie enthält Zeichen, die zum Schreiben der meisten europäischen Sprachen benötigt werden, wie zum Beispiel Umlaute oder das ñ im spanischen Wort pestaña. Viele Sprachen brauchen aber mehr als 256 Zeichen. Ein

Zeichensatz, der mehr als nur eine Sprache ausdrücken soll, kann noch viel mehr benötigen. Hier kommt Unicode zu Hilfe; mit seiner UTF-8-Codierung lassen sich mehr als eine Million Zeichen darstellen.

Für diese zusätzliche Funktionalität zahlen Sie mit zusätzlichem Speicherplatz. ASCII-Zeichen lassen sich in nur einem Byte speichern, bei UTF-8-codierten Zeichen können es dagegen bis zu vier Byte sein. Tabelle 19-2 zeigt die Byte-Darstellungen UTF-8-codierter Zeichen.

Tabelle 19-2: Byte-Darstellung von UTF-8

Zeichencode-Bereich	Verwendete Bytes	Byte 1	Byte 2	Byte 3	Byte 4
0x00000000 – 0x0000007F	1	0xxxxxxx			
0x00000080 – 0x000007FF	2	110xxxxx	10xxxxxx		
0x00000800 – 0x0000FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
0x00010000 – 0x001FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

In Tabelle 19-2 stellen die x-Positionen Bits dar, die für die eigentlichen Zeichendaten verwendet werden. Das am wenigsten signifikante Bit ist dabei das am weitesten rechts stehende Bit im am weitesten rechts stehenden Byte. In Zeichen, die aus mehreren Bytes bestehen, gibt die Zahl der anführenden 1-Bits in dem am weitesten links stehenden Byte die Anzahl der Bytes im Zeichen an.

Siehe auch

Die Dokumentationen zu `utf8_encode()` unter *http://www.php.net/utf8-encode* und `utf8_decode()` unter *http://www.php.net/utf8-decode*; mehr Informationen zu Unicode gibt's auf der Homepage des Unicode-Consortiums, *http://www.unicode.org*; die UTF-8- und die Unicode-FAQ unter *http://www.cl.cam.ac.uk/~mgk25/unicode.html* sind ebenfalls hilfreich.

19.12 Die Zeichenkodierung ausgehender Daten setzen

Problem

Sie möchten sichergehen, dass Browser den UTF-8-kodierten Text richtig behandeln, den Ihr Programm ausgibt.

Lösung

Setzen Sie die PHP-Konfigurationsdirektive `default_encoding` auf `utf-8`. Das sichert, dass der `Content-Type`-Header, den PHP in HTML-Antworten ausgibt, den Teil `charset=utf-8`

einschließt, der dem Browser sagt, dass er die Seiteninhalte als UTF-8-kodiert behandeln soll.

Diskussion

Das Setzen von `default_encoding` gibt dem Browser den Hinweis, dass Ihre Seiteninhalte als UTF-8-kodiert interpretiert werden sollten. Sie sind allerdings immer noch dafür verantwortlich, dass Ihre Seiten tatsächlich korrekt UTF-8-kodiert werden, indem Sie die entsprechenden String-Funktionen einsetzen. Rezept 19.11 sagt Ihnen, wie Sie das machen.

Wenn Sie die Konfigurationsdirektive `default_encoding` nicht ändern können, haben Sie die Möglichkeit, den entsprechenden `Content-Type`-Header manuell mit der Funktion `header( )` zu versenden, wie Sie es in Beispiel 19-3 sehen.

Beispiel 19-3: Die Zeichenkodierung setzen

```
<?php
header('Content-Type: text/html;charset=utf-8');
?>
```

Siehe auch

Rezept 19.11 erläutert, wie man UTF-8-kodierten Text generiert.

19.13 Die Zeichenkodierung eingehender Daten setzen

Problem

Sie möchten sichergehen, dass der Datenfluss in Ihr Programm konsistent eine Zeichenkodierung verwendet, damit Sie die Daten korrekt verarbeiten können. Beispielsweise wollen Sie alle eingehenden Formulardaten als UTF-8-kodierte Daten behandeln.

Lösung

Sie können nicht garantieren, dass Browser Ihre Anweisungen in Bezug auf die Zeichenkodierung befolgen, aber Sie können eine Reihe von Dingen auf den Weg bringen, die dafür sorgen, dass brave Browser den Regeln folgen.

Zunächst sollten Sie den Anweisungen in Rezept 19.12 folgen, damit Ihre Programme Browsern sagen, dass sie UTF-8-kodierten Text aussenden. Ein `Content-Type`-Header mit einer `charset`-Angabe ist ein guter Hinweis für einen Browser, dass seine Formulare in der gleichen Zeichenkodierung versendet werden sollten, wie im Header angegeben.

Schließen Sie außerdem das Attribut `accept-charset="utf-8"` in `<form/>`-Elemente, die Sie ausgeben, ein. Das wird zwar nicht von allen Browsern unterstützt, sagt unterstützenden

Browsern aber, dass die vom Benutzer eingegebenen Daten in UTF-8 kodiert werden sollen, bevor sie an den Server geschickt werden.

Diskussion

In der Regel senden Browser Formulardaten mit der Kodierung, die bei der Generierung der Seite eingesetzt wurde, die das Formular enthält. Wenn Sie konsistent UTF-8-Ausgaben erzeugen, können Sie hinreichend sicher sein, dass Sie auch immer UTF-8-Eingaben erhalten. Das `<form/>`-Attribut `accept-charset` ist Teil der HTML-4.0-Spezifikation, ist aber nicht überall implementiert.

Siehe auch

Rezept 19.12 liefert Informationen dazu, wie man UTF-8-kodierte Ausgaben erzeugt; das `<form/>`-Attribut `accept-charset` wird unter *http://www.w3.org/TR/REC-html40/interact/forms.html#adef-accept-charset* beschrieben.

KAPITEL 20

Internetdienste

20.0 Einführung

Bevor es HTTP gab, war das Internet eine Buchstabensuppe von FTP, NNTP, IMAP, POP3 und einer ganzen Menge anderer Protokolle. Viele Benutzer nahmen Webbrowser dankbar in Empfang, da sie integrierte Programme darstellen, die sie ihre E-Mails abrufen, Newsgruppen lesen, Dateien übertragen und Dokumente einsehen lassen konnten, ohne sich über die Details der darunter liegenden Kommunikationsverfahren kümmern zu müssen. PHP stellt Funktionen zur Benutzung dieser anderen Protokolle bereit, und zwar sowohl eingebaute Funktionen als auch durch PEAR. Mit ihnen können Sie mit PHP Web-Frontend-Anwendungen für alle möglichen netzwerkgebundenen Aufgaben erstellen, wie z.B. das Auflösen von Domainnamen oder das Senden von webgestützten E-Mails. Obwohl PHP diese Aufgaben einfacher macht, ist es wichtig, die Stärken und Schwächen der individuellen Protokolle zu kennen.

Die Rezepte 20.1 bis 20.3 decken das beliebteste Feature von allen ab: E-Mail. Rezept 20.1 zeigt, wie man einfache E-Mail-Nachrichten verschickt. Rezept 20.2 beschreibt MIME-codierte E-Mail, mit der Sie sowohl einfache Text- als auch HTML-formatierte Nachrichten verschicken können. Die IMAP- und POP3-Protokolle, die zum Lesen von Mailboxen verwendet werden, werden in Rezept 20.3 behandelt.

Die nächsten beiden Rezepte diskutieren, wie man Newsgruppen mit NNTP liest. Newsgruppen sind etwas Ähnliches wie Mailinglisten, wobei allerdings nicht jede teilnehmende Person eine E-Mail geschickt bekommt. Stattdessen können die Benutzer auf einen Newsserver zugreifen und sich nur die Nachrichten ansehen, an denen sie interessiert sind. Newsgruppen ermöglichen auch strukturierte Diskussionen, so dass es einfach wird, einen Dialog durch die Archive hindurch zu verfolgen. Rezept 20.4 behandelt das Posten von Nachrichten, während Rezept 20.5 sich mit dem Herunterladen von Nachrichten beschäftigt.

Rezept 20.6 deckt den Dateiaustausch mittels FTP ab. FTP, das File Transfer Protocol, ist eine Methode zum Senden und Empfangen von Dateien über das Internet. FTP-Server

können von Benutzern entweder ein Login mit Passwort verlangen oder auch die anonyme Verwendung gestatten.

Das Suchen auf LDAP-Servern ist Thema in Rezept 20.7, während Rezept 20.8 zeigt, wie man Benutzer mit Hilfe eines LDAP-Servers authentifizieren kann. LDAP-Server dienen als Adressbücher und als zentrale Ablage für Benutzerinformationen. Sie sind auf Informationsabrufe optimiert und lassen sich so konfigurieren, dass sie ihre Daten auf mehrere Kopien verteilen, was eine hohe Zuverlässigkeit und schnelle Antwortzeiten garantiert.

Das Kapitel schließt mit Rezepten zur Netzwerktechnik. Rezept 20.9 behandelt DNS-Anfragen sowohl von Domainnamen zu IP als auch andersherum. Das letzte Rezept verrät, wie man mit dem Ping-Modul von PEAR feststellen kann, ob ein Host läuft und kontaktiert werden kann.

Andere Teile dieses Buchs beschäftigen sich ebenfalls mit Netzwerkprotokollen. HTTP wird ausführlich in Kapitel 13 behandelt. Diese Rezepte beschreiben mehrere unterschiedliche Arten zum Abfragen von URLs. Protokolle, die HTTP und XML kombinieren, werden in Kapitel 14 beschrieben. In diesem Kapitel besprechen wir neben DOM und XSLT auch das zunehmend wichtiger werdende Gebiet der Web Services, in dem die Protokolle XML-RPC und SOAP verwendet werden.

20.1 E-Mails senden

Problem

Sie wollen eine E-Mail versenden. Die Nachricht kann eine direkte Reaktion auf eine Benutzerhandlung sein, wie beispielsweise die Registrierung des Benutzers als Neumitglied auf Ihrer Seite. Sie kann aber auch ein sich wiederholendes Ereignis zu einer festgelegten Zeit sein, wie z.B. ein wöchentliches Bulletin.

Lösung

Verwenden Sie die `Mail`-Klasse von PEAR:

```
require 'Mail.php';

$to = 'adam@example.com';

$headers['From'] = 'webmaster@example.com';
$headers['Subject'] = 'Neue Version von PHP freigegeben!';

$body = 'Jetzt von http://www.php.net herunterladen!';

$message =& Mail::factory('mail');
$message->send($to, $headers, $body);
```

Wenn Sie die Mail-Klasse von PEAR nicht verwenden können, benutzen Sie die eingebaute mail()-Funktion von PHP:

```
$to = 'adam@example.com';
$subject = 'Neue Version von PHP freigegeben!';
$body = 'Jetzt von http://www.php.net herunterladen!';

mail($to, $subject, $body);
```

Diskussion

Mit der Mail-Klasse aus dem PEAR-Archiv können Sie E-Mail auf drei verschiedene Arten versenden. Sie geben die zu verwendende Art an, wenn Sie ein Mail-Objekt mit Mail::factory() instantiieren.

- Um E-Mail über ein externes Programm wie beispielsweise *sendmail* oder *qmail* zu versenden, übergeben Sie sendmail.
- Um einen SMTP-Server zu verwenden, übergeben Sie smtp.
- Um die eingebaute mail()-Funktion zu verwenden, übergeben Sie mail. Das weist Mail an, die Einstellungen aus Ihrer *php.ini* zu benutzen.

Um sendmail oder smtp zu verwenden, müssen Sie einen zweiten Parameter angeben, der Ihre Einstellungen enthält. Um sendmail zu verwenden, geben Sie einen sendmail_path sowie sendmail_args an:

```
$params['sendmail_path'] = '/usr/sbin/sendmail';
$params['sendmail_args'] = '-oi -t';

$message =& Mail::factory('sendmail', $params);
```

Ein üblicher Wert für sendmail_path ist */usr/lib/sendmail*. Leider tendiert *sendmail* dazu, von System zu System in einem anderen Verzeichnis zu hausen, so dass es schwierig sein kann, das Programm zu finden. Wenn Sie es nicht finden können, versuchen Sie es mit */usr/sbin/sendmail* oder fragen Ihren Systemadministrator.

Zwei nützliche Flags zur Übergabe an *sendmail* sind -oi und -t. Das -oi-Flag weist *sendmail* an, einen einzelnen Punkt (.) in einer Zeile nicht als Nachrichtenende zu interpretieren. Das -t-Flag bewirkt, dass *sendmail* To: und andere Kopfzeilen aus der übergebenen Datei entnimmt.

Wenn Sie *qmail* bevorzugen, versuchen Sie, */var/qmail/bin/qmail-inject* oder */var/qmail/bin/sendmail* zu verwenden.

Wenn Sie unter Windows arbeiten, wollen Sie wahrscheinlich einen SMTP-Server benutzen, da die meisten Windows-Maschinen keine Kopie von *sendmail* installiert haben. Zu diesem Zweck geben Sie smtp ein:

```
$params['host'] = 'smtp.example.com';

$message =& Mail::factory('smtp', $params);
```

Im smtp-Modus können Sie fünf optionale Parameter eingeben. Der host ist der Hostname des SMTP-Servers; die Default-Einstellung ist localhost. Der port ist der Port der Verbindung; die Default-Einstellung ist 25. Zur Aktivierung der SMTP-Authentifizierung setzen Sie auth auf true, und um dem Server die Möglichkeit zu geben, Sie als gültigen Benutzer zu erkennen, setzen Sie username und password. Die SMTP-Funktionalität ist nicht auf Windows beschränkt, sie läuft auch auf Unix-Servern.

Wenn Sie nicht über die Mail-Klasse von PEAR verfügen, können Sie die eingebaute Mail( )-Funktion verwenden. Das Programm, das mail( ) zum Versenden von E-Mail verwendet, wird in der sendmail_path-Konfigurationsvariablen Ihrer *php.ini*-Datei angegeben. Wenn Sie unter Windows arbeiten, setzen Sie dort die Variable SMTP auf den Hostnamen Ihres SMTP-Servers. Ihre From-Adresse wird unter Windows der sendmail_from-Variablen entnommen. Unter Unix wird als Default der Benutzername verwendet, unter dem Ihr Webserver läuft – üblicherweise nobody. Falls Sie hier eine andere Absenderadresse verwenden wollen, müssen Sie das explizit mit einem From:-Header tun.

Das folgende Beispiel verwendet mail( ):

```
$to = 'adam@example.com';
$subject = 'Neue Version von PHP freigegeben!';
$body = 'Jetzt von http://www.php.net herunterladen!';

mail($to, $subject, $body);
```

Der erste Parameter ist die E-Mail-Adresse des Empfängers, der zweite ist der Betreff und der letzte der Hauptteil (Body) der Nachricht. Mit einem optionalen vierten Parameter können Sie zusätzliche Kopfzeilen hinzufügen. Hier sehen Sie ein Beispiel, das zeigt, wie man Reply-To and Organization-Header hinzufügt:

```
$to = 'adam@example.com';
$subject = 'Neue Version von PHP freigegeben!';
$body = 'Jetzt von http://www.php.net herunterladen!';
$header = "Reply-To: webmaster@example.com\r\n"
         ."Organization: The PHP Group";

mail($to, $subject, $body, $header);
```

Trennen Sie jede Kopfzeile mit \r\n, aber fügen Sie nach dem letzten Header kein \r\n an.

Unabhängig von der von Ihnen gewählten Methode ist es hilfreich, eine Wrapper-Funktion zu schreiben, die Ihnen beim Senden der E-Mail hilft. Indem Sie jede E-Mail durch diese Funktion zwängen, können Sie jeder gesendeten Nachricht einfach ein Logging und andere Checks hinzuzufügen.

```
function pc_mail($to, $headers, $body) {
    $message =& Mail::factory('mail');

    $message->send($to, $headers, $body);
    error_log("[MAIL][TO: $to]");
}
```

Hier wird für jede gesendete Nachricht eine Meldung ins Fehler-Log geschrieben, die den Empfänger der Nachricht festhält. Das gibt Ihnen einen Datumsstempel, mit dem Sie Beschwerden nachgehen können, wenn jemand versucht, Ihre Site zum Senden von Spam zu missbrauchen. Eine andere Option ist das Anlegen einer Liste von Adressen, die keine E-Mails mehr von Ihrer Site empfangen möchten. Sie können auch die E-Mail-Adressen Ihrer Empfänger validieren, was die Anzahl der unzustellbaren Nachrichten verringern sollte.

Siehe auch

Rezept 16.5 für einen regulären Ausdruck zum Validieren von E-Mail-Adressen; Rezept 20.2 zum Senden von MIME-E-Mail; Rezept 20.3 für mehr Informationen zum Herunterladen von E-Mail; die Dokumentation zu `mail()` unter *http://www.php.net/mail*; die PEAR-`Mail`-Klasse unter *http://pear.php.net/package/Mail*; RFC 822 unter *http://www.faqs.org/rfcs/rfc822.html*; von O'Reilly gibt es den Titel *sendmail* von Bryan Costales und Eric Allman.

20.2 MIME-Mail senden

Problem

Sie wollen MIME-E-Mail versenden. Beispielsweise möchten Sie mehrteilige Nachrichten sowohl mit Normaltext als mit auch HTML-Teilen senden. Dazu haben Sie E-Mail-Programme, die MIME verarbeiten können und automatisch den richtigen Teil anzeigen.

Lösung

Verwenden Sie die `Mail_mime`-Klasse in PEAR:

```
require 'Mail.php';
require 'Mail/mime.php';

$to = 'adam@example.com, sklar@example.com';

$headers['From'] = 'webmaster@example.com';
$headers['Subject'] = 'Neue Version von PHP freigegeben!';

$body = 'Jetzt von http://www.php.net herunterladen!';

// MIME-Objekt erzeugen.
$mime = new Mail_mime;

// Body-Teile hinzufügen.
$text = 'Textversion der E-Mail';
$mime->setTXTBody($text);
```

```
$html = '<html><body>HTML-Version der E-Mail</body></html>';
$mime->setHTMLBody($html);

$file = '/pfad/zur/datei.png';
$mime->addAttachment($file, 'image/png');

// MIME-formatierte Kopfzeilen und Body ermitteln.
$body = $mime->get();
$headers = $mime->headers($headers);

$message =& Mail::factory('mail');
$message->send($to, $headers, $body);
```

Diskussion

Die `Mail_mime`-Klasse von PEAR stellt eine objektorientierte Schnittstelle zu allen Details zur Verfügung, die hinter den Kulissen gebraucht werden, um eine E-Mail-Nachricht zu erstellen, die sowohl Text als auch HTML-Teile enthält. Die Klasse ähnelt der `Mail`-Klasse von PEAR. Anstatt den Hauptteil als einen Text-String zu definieren, erstellen Sie aber ein `Mail_mime`-Objekt und rufen dessen Methoden auf, um MIME-Teile in den Body der Nachricht einzufügen:

```
// MIME-Objekt erzeugen.
$mime = new Mail_mime;

// Body aus Einzelteilen zusammensetzen.
$text = 'Textversion der E-Mail';
$mime->setTXTBody($text);

$html = '<html><body>HTML-Version der E-Mail</body></html>';
$mime->setHTMLBody($html);

$file = '/pfad/zur/datei.txt';
$mime->addAttachment($file, 'text/plain');

// MIME-formatierte Kopfzeilen und Body ermitteln.
$headers = $mime->headers($headers);
$body = $mime->get();
```

Die Methoden `Mail_mime::setTXTBody( )` bzw. `Mail_mime::setHTMLBody( )` fügen den Normaltext bzw. den HTML-Body-Teil an. Hier übergeben Sie Variablen, Sie können aber auch einen Dateinamen übergeben, der dann von `Mail_mime` gelesen wird. Um von dieser Möglichkeit Gebrauch zu machen, übergeben Sie `true` als zweiten Parameter:

```
$text = '/pfad/zur/email.txt';
$mime->setTXTBody($text, true);
```

Um der Nachricht eine Anlage hinzuzufügen, wie z.B. eine Grafik oder ein Archiv, rufen Sie `Mail_mime::addAttachment( )` auf:

```
$file = '/pfad/zur/datei.png';
$mime->addAttachment($file,'image/png');
```

Übergeben Sie der Funktion den Pfad der Datei und ihren MIME-Typ.

Wenn die Nachricht vollständig ist, führen Sie die letzten Vorbereitungen aus und senden sie ab:

```
// MIME-formatierte Kopfzeilen und Body ermitteln.
$headers = $mime->headers($headers);
$body = $mime->get( );

$message =& Mail::factory('mail');
$message->send($to, $headers, $body);
```

Zunächst haben Sie das `Mail_mime`-Objekt, das die richtig formatierten Kopfzeilen und den Body zur Verfügung stellt. Anschließend verwenden Sie die Elternklasse `Mail`, um die Nachricht zu formatieren und mit `Mail_mime::send( )` abzuschicken.

Wenn Sie eine Grafik in den HTML-Teil Ihrer E-Mail einbetten wollen, gehen Sie folgendermaßen vor:

```
$html = '<html><body>Dieses Bild <img src="my_inline_image.jpg"> ';
$html .= 'steht mitten in der E-Mail</body></html>';
$mime->addHTMLImage("images/my_inline_image.jpg", "image/jpeg");
$mime->setHTMLBody($html);
```

Hier müssen Sie darauf achten, dass das `src`-Attribut des `<img>`-Tags im HTML-Teil Ihrer Mail den Dateinamen (ohne Pfad!) der zugehörigen Bilddatei als Wert zugewiesen bekommt. Wenn Sie dann mit `$mime->addHTMLImage()` die eigentliche Bilddatei hinzufügen und die Mail abschicken, wird der Dateiname im `src`-Attribut durch eine Content-ID ersetzt. Im Quelltext der Mail sieht das so aus:

```
--=_097463bca714fa334875e100cffac621
Content-Type: text/html; charset="ISO-8859-1"
Content-Transfer-Encoding: quoted-printable

<html><body>Dieses Bild <img src=3D"cid:0c08f298fc5931b94de11a222541ae1d"> =
steht mitten in der E-Mail</body></html>
--=_097463bca714fa334875e100cffac621
Content-Type: image/jpeg
Content-Transfer-Encoding: base64
Content-Disposition: inline; filename="my_inline_image.jpg"
Content-ID: <0c08f298fc5931b94de11a222541ae1d>

/9j/4AAQSkZJRgABAQEASABIAAD/4TTzRXhpZgAASUkqAAgAAAALAA8BAgAJAAAAkgAAABABAgAQ
...
```

Wie Sie sehen können, hat `Mail_mime` hier die `Content-ID` der zugehörigen Bilddatei im nächsten Abschnitt der MIME-Nachricht verwendet. Damit weiß der HTML-Mail-Client, woher das Bild zu holen ist.

Siehe auch

Rezept 20.1 zum Senden von normaler E-Mail; Rezept 20.3 für mehr Informationen zum Herunterladen von E-Mail; die `Mail_Mime`-Klasse von PEAR unter *http://pear.php.net/package/Mail_Mime*.

20.3 E-Mail mit IMAP oder POP3 lesen

Problem

Sie wollen E-Mail mit IMAP oder POP3 lesen, wodurch Sie ein webbasiertes E-Mail-Client-Programm erstellen können.

Lösung

Verwenden Sie die IMAP-Erweiterung von PHP, die sowohl IMAP als auch POP3 spricht:

```
// IMAP-Verbindung öffnen.
$mail = imap_open('{mail.server.com:143}',      'username', 'password');
// Oder: POP3-Verbindung öffnen.
$mail = imap_open('{mail.server.com:110/pop3}', 'username', 'password');

// Liste aller E-Mail-Header herunterladen.
$headers = imap_headers($mail);

// Header-Objekt der letzten Nachricht in der Mailbox herunterladen.
$last = imap_num_msg($mail);
$header = imap_header($mail, $last);

// Body derselben Nachricht holen.
$body = imap_body($mail, $last);

// Verbindung schließen.
imap_close($mail);
```

Diskussion

Die Bibliothek, die PHP verwendet, um IMAP und POP3 zu unterstützen, stellt eine fast unendliche Anzahl an Funktionen zur Verfügung, die es Ihnen erlauben, ein vollständiges E-Mail-Client-Programm zu schreiben. Durch die Vielzahl der Funktionen wird die Sache aber auch komplex. Zum Beispiel gibt es zurzeit 63 verschiedene PHP-Funktionen, die mit dem Wort `imap` anfangen, was aber noch nicht berücksichtigt, dass manche davon auch POP3 und NNTP sprechen.

Allerdings sind die Grundlagen der Kommunikation mit einem Mailserver ziemlich einfach. Wie bei vielen PHP-Funktionen fangen Sie damit an, die Verbindung zu öffnen und sich ein Handle zu holen:

```
$mail = imap_open('{mail.server.com:143}', 'username', 'password');
```

Diese Zeile öffnet eine IMAP-Verbindung zum Server mit dem Namen *mail.server.com* an Port 143. Sie gibt außerdem einen Benutzernamen und das Passwort als zweites und drittes Argument an.

Um stattdessen eine POP3-Verbindung zu öffnen, hängen Sie /pop3 ans Ende von Server und Port an. Da POP3 normalerweise auf Port 110 läuft, fügen Sie nach dem Servernamen :110 an:

```
$mail = imap_open('{mail.server.com:110/pop3}', 'username', 'password');
```

Zum Verschlüsseln Ihrer Verbindung mit SSL fügen Sie /ssl am Ende an, so wie Sie es auch mit pop3 gemacht haben. Sie müssen außerdem sicherstellen, dass Ihre PHP-Installation zusätzlich zu --with-imap mit der --with-imap-ssl-Konfigurationsoption kompiliert wurde. Weiterhin müssen Sie die IMAP-Bibliothek Ihres Systems selbst mit SSL-Unterstützung kompilieren. Wenn Sie ein selbst unterschriebenes Zertifikat verwenden und möchten, dass Validierungsversuche unterbunden werden, fügen Sie /novalidate-cert an. Außerdem kommunizieren die meisten SSL-Verbindungen entweder über Port 993 oder 995. Diese Optionen können in beliebiger Reihenfolge aufgeführt werden. Das Folgende ist also durchaus legal:

```
$mail = imap_open('{mail.server.com:993/novalidate-cert/pop3/ssl}',
                  'username', 'password');
```

Wenn Sie eine Variable innerhalb eines Strings mit doppelten Anführungszeichen durch geschweifte Klammern einschließen, wie z.B. {$var}, teilen Sie PHP genau mit, welche Variable evaluiert werden soll. Um evaluierte Variablen in diesem ersten Parameter mit imap_open( ) zu verwenden, entschärfen Sie daher die öffnende { durch einen vorangestellten Backslash:

```
$server = 'mail.server.com';
$port = 993;

$mail = imap_open("\{$server:$port}", 'username', 'password');
```

Sobald Sie eine Verbindung geöffnet haben, können Sie dem Mailserver verschiedene Fragen stellen. Um eine Liste mit allen Nachrichten in Ihrer Eingangs-Mailbox aufzurufen, verwenden Sie imap_headers( ):

```
$headers = imap_headers($mail);
```

Dies gibt ein Array zurück, in dem jedes Element ein formatierter String ist, der sich auf eine Nachricht bezieht:

```
A   189) 5-Aug-2002 Beth Hondl           Eine Einladung (1992 chars)
```

Um stattdessen eine bestimmte Nachricht herunterzuladen, verwenden Sie imap_header( ) und imap_body( ), um sich den Kopfzeilen- und den Body-String herüberzuziehen:

```
$header = imap_header($message_number);
$body   = imap_body($message_number);
```

Das body-Element ist einfach nur ein String. Wenn die Nachricht aber aus mehreren Teilen, z.B. einem HTML- und einem Normaltext-Teil, besteht, enthält $body beide Teile sowie die MIME-Zeilen, die diese beschreiben:

```
------=_Part_1046_3914492.1008372096119
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Normaltext-Nachricht

------=_Part_1046_3914492.1008372096119
Content-Type: text/html
Content-Transfer-Encoding: 7bit

<html>HTML-Nachricht</html>
------=_Part_1046_3914492.1008372096119--
```

Um das Auftauchen dieses Durcheinanders zu vermeiden, verwenden Sie `imap_fetchstructure()` in Kombination mit `imap_fetchbody()`, um herauszufinden, wie der Hauptteil formatiert ist, und nur diejenigen Teile herauszuholen, die Sie wollen:

```
// Normaltext für Nachricht $n extrahieren.
$st = imap_fetchstructure($mail, $n);
if (!empty($st->parts)) {
    for ($i = 0, $j = count($st->parts); $i < $j; $i++) {
        $part = $st->parts[$i];
        if ($part->subtype == 'PLAIN') {
             $body = imap_fetchbody($mail, $n, $i+1);
        }
     }
} else {
    $body = imap_body($mail, $n));
}
```

Wenn eine Nachricht aus mehreren Teilen besteht, enthält `$st->parts` ein Array mit Objekten, die die Teile beschreiben. Die Eigenschaft `part` enthält eine ganze Zahl, die den MIME-Typ des Haupt-Bodys beschreibt. Tabelle 20-1 führt auf, welche Zahlen welchen MIME-Typen zugeordnet sind. Die Eigenschaft `subtype` enthält den MIME-Subtyp und gibt an, ob der Teil `plain`, `html`, `png` oder ein anderer Typ, z.B. `octet-stream`, ist.

Tabelle 20-1: IMAP-Werte für MIME-Typen

Nummer	MIME-Typ	PHP-Konstante	Beschreibung	Beispiele
0	`text`	`TYPETEXT`	unformatierter Text	normaler Text, HTML, XML
1	`multipart`	`TYPEMULTIPART`	aus mehreren Teilen bestehende Nachricht	Mixed, Formulardaten, signiert
2	`message`	`TYPEMESSAGE`	gekapselte Nachricht	News, HTTP
3	`application`	`TYPEAPPLICATION`	Anwendungsdaten	Oktett-Sequenz, PDF, Zip
4	`audio`	`TYPEAUDIO`	Musikdateien	MP3, RealAudio
5	`image`	`TYPEIMAGE`	Grafikbilder	GIF, JPEG, PNG
6	`video`	`TYPEVIDEO`	Videoclips	MPEG, Quicktime
7	`other`	`TYPEOTHER`	alles andere	VRML-Modelle

Siehe auch

Die Rezepte 20.1 und 20.3 für mehr Informationen zum Senden von E-Mail; die Dokumentationen zu `imap_open()` unter *http://www.php.net/imap_open*, `imap_header()` unter *http://www.php.net/imap-header*, `imap-body()` unter *http://www.php.net/imap-body* und IMAP im Allgemeinen unter *http://www.php.net/imap*.

20.4 Nachrichten an Usenet-Newsgruppen senden

Problem

Sie wollen eine Nachricht an eine Usenet-Newsgruppe senden, wie z.B. *comp.lang.php*.

Lösung

Verwenden Sie `imap_mail_compose()`, um die Nachricht zu formatieren, und schicken Sie die Nachricht dann über Sockets an den Server:

```
$headers['from'] = 'adam@example.com';
$headers['subject'] =  'Neue Version von PHP freigegeben!';
$headers['custom_headers'][] = 'Newsgroups: comp.lang.php';

$body[0]['type'] = TYPETEXT;
$body[0]['subtype'] = 'plain';
$body[0]['contents.data'] = 'Jetzt von http://www.php.net herunterladen!';

$post = imap_mail_compose($headers, $body);

$server = 'nntp.example.com';
$port = 119;

$sh = fsockopen($server, $port) or die ("Verbindung zu $server konnte
                                                    nicht hergestellt werden!");
fputs($sh, "POST\r\n");
fputs($sh, $post);
fputs($sh, ".\r\n");
fclose($sh);
```

Diskussion

Keine der in PHP eingebauten Funktionen kann Nachrichten an eine Newsgruppe posten. Deshalb müssen Sie eine direkte Socket-Verbindung zum Newsserver öffnen und die Befehle zum Posten der Nachricht senden. Sie können aber `imap_mail_compose()` verwenden, um die Nachricht zu formatieren und um die Kopfzeilen und den Body der Nachricht zu erstellen. Jede Nachricht muss drei Kopfzeilen enthalten: die `From:`-Adresse, das `Subject:` der Nachricht und den Namen der Newsgruppe:

```
$headers['from'] = 'adam@example.com';
$headers['subject'] = 'Neue Version von PHP freigegeben!';
$headers['custom_headers'][ ] = 'Newsgroups: comp.lang.php';
```

Erstellen Sie ein Array `$headers`, das die Kopfzeilen der Nachricht enthalten soll. Die Werte für die Header `From:` und `Subject:` können Sie direkt zuweisen. Für den `Newsgroups:`-Header geht das aber nicht. Da `imap_mail_compose( )` vorwiegend zur Erstellung von E-Mail-Nachrichten verwendet wird, ist `Newsgroups:` kein vordefinierter Header. Sie müssen diese Kopfzeile deshalb mit dem Array-Element `custom_headers` hinzufügen.

Die Syntax für `custom_headers` unterscheidet sich von der für gewöhnliche Header. Anstatt die kleingeschriebene Kopfzeile als Elementname und den Kopfzeilenwert als Array-Wert zu verwenden, setzen Sie die gesamte Kopfzeile als einen Array-Wert. Zwischen dem Namen des Headers und dem Wert fügen Sie einen Doppelpunkt ein, gefolgt von einem Leerzeichen. Vergewissern Sie sich, dass Sie `Newsgroups:` richtig buchstabieren, mit einem großen `N` und einem `s` am Ende.

Der Hauptteil der Nachricht kann aus mehreren Teilen bestehen. Folglich ist der Body-Parameter, der `imap_mail_compose( )` übergeben wird, ein Array von Arrays. In der Lösung gab es nur einen einzigen Teil. Also weisen Sie die Werte `$body[0]` direkt zu:

```
$body[0]['type'] = TYPETEXT;
$body[0]['subtype'] = 'plain';
$body[0]['contents.data'] = 'Jetzt von http://www.php.net herunterladen!';
```

Jeder Teil des Bodys braucht je einen MIME-Typ und -Subtyp. Diese Nachricht ist in ASCII, so dass der Typ `TYPETEXT` und der Untertyp `plain` ist. Eine Auflistung von IMAP-MIME-Typen-Konstanten mit ihrer Bedeutung enthält Tabelle 20-1 in Rezept 20.3. Das Feld `contents.data` enthält den Body der Nachricht.

Um diese Arrays in einen formatierten String umzuwandeln, rufen Sie `imap_mail_compose($body, $headers)` auf. Sie gibt einen Post zurück, der wie folgt aussieht:

```
From: adam@example.com
Subject: Neue Version von PHP freigegeben!
MIME-Version: 1.0
Content-Type: TEXT/plain; CHARSET=US-ASCII
Newsgroups: comp.lang.php

Jetzt von http://www.php.net herunterladen!
```

Gewappnet mit einem Post, den der Newsserver akzeptiert, rufen Sie `fsockopen( )` auf, um eine Verbindung zu öffnen:

```
$server = 'nntp.example.com';
$port = 119;

$sh = fsockopen($server, $port) or die ("Verbindung zu $server konnte
                                                    nicht hergestellt werden!");
```

Der erste Parameter von `fsockopen( )` ist der Hostname des Servers, der zweite der zu verwendende Port. Wenn Sie den Namen Ihres Newsservers nicht kennen, versuchen Sie es

mit den Hostnamen *news*, *nntp* oder *news-server* in Ihrer Domain beispielsweise *news.example.com*, *nntp.example.com* oder *news-server.example.com*. Wenn keine dieser Namen funktionieren, sollten Sie Ihren Systemadministrator fragen. Traditionell verwenden alle Newsserver Port 119.

Wenn Sie verbunden sind, senden Sie die Nachricht:

```
fputs($sh, "POST\r\n");
fputs($sh, imap_mail_compose($headers, $body));
fputs($sh, ".\r\n");
```

Die erste Zeile teilt dem Newsserver mit, dass Sie eine Nachricht senden wollen, die zweite ist die Nachricht selbst. Um das Ende der Nachricht zu kennzeichnen, setzen Sie einen einzelnen Punkt in eine Zeile. Jede Zeile muss an ihrem Ende sowohl ein Wagenrücklauf-Zeichen (*carriage return*) als auch einen Zeilenvorschub (*newline*) enthalten. Schließen Sie die Verbindung, indem Sie `fclose($sh)`aufrufen.

Jeder Nachricht auf dem Server wird ein eindeutiger Name gegeben, nämlich die `Message-ID`. Wenn Sie auf eine Nachricht antworten möchten, nehmen Sie die `Message-ID` der Originalnachricht und verwenden sie als Wert für einen `References`-Header:

```
// beim Lesen der Originalnachricht erhaltene ID
$message_id = '<20030410020818.33915.php@news.example.com>';

$headers['custom_headers'][] = "References: $message_id";
```

Siehe auch

Rezept 20.5 zu mehr Informationen über das Lesen von Newsgruppen; die Dokumentationen zu `imap_mail_compose()` unter *http://www.php.net/imap-mail-compose*, `fsockopen()` unter *http://www.php.net/fsockopen*, zu `fputs()` unter *http://www.php.net/fputs* und zu `fclose()` unter *http://www.php.net/fclose*; zu RFC 977 unter *http://www.faqs.org/rfcs/rfc977.html*.

20.5 Usenet-Nachrichten lesen

Problem

Sie wollen Usenet News-Nachrichten lesen und über NNTP mit einem Newsserver kommunizieren.

Lösung

Verwenden Sie die IMAP-Erweiterung von PHP. Sie spricht auch NNTP:

```
// Eine Verbindung zum NNTP-Server öffnen.
$server = '{news.php.net/nntp:119}';
$group = 'php.general'; // zentrale Mailingliste für PHP
```

```
$nntp = imap_open("$server$group", '', '', OP_ANONYMOUS);

// Header abrufen.
$header = imap_header($nntp, $msg);

// Felder extrahieren.
$subj  = $header->subject;
$from  = $header->from;
$email = $from[0]->mailbox."@".$from[0]->host;
$name  = $from[0]->personal;
$date  = date('m/d/Y h:i A', $header->udate);

// Hauptteil lesen.
$body  = nl2br(htmlspecialchars(imap_fetchbody($nntp,$msg,1)));

// Verbindung schließen.
imap_close($nntp);
```

Diskussion

Wenn Sie News von einem Newsserver lesen wollen, müssen Sie eine Verbindung zum Server herstellen und eine Gruppe angeben, die Sie interessiert:

```
// Eine Verbindung zum NNTP-Server öffnen.
$server = "{news.php.net/nntp:119}";
$group = "php.general";
$nntp = imap_open("$server$group",'','',OP_ANONYMOUS);
```

Die Funktion `imap_open( )` nimmt vier Parameter an. Der erste gibt den zu verwendenden Newsserver und die zu lesende Newsgruppe an. Hier ist der Server *news.php.net* der Server, der alle PHP-Mailinglisten auflistet. Fügen Sie `/nntp` an, um der IMAP-Erweiterung mitzuteilen, dass Sie News und nicht E-Mail lesen wollen, und geben Sie `119` als Port an. Das ist normalerweise der Port, der für NNTP reserviert ist. NNTP steht für Network News Transport Protocol; es wird zur Kommunikation mit Newsservern verwendet, so wie HTTP mit Webservern kommuniziert. Die Gruppe ist *php.general*, also die Haupt-Mailingliste der PHP-Gemeinde.

Die beiden mittleren Argumente in `imap_open( )` sind der Benutzername und das Passwort, falls Sie Ihre Identität verifizieren müssen. Da *news.php.net* für alle Leser zugänglich ist, lassen Sie diese Argumente leer. Schließlich geben Sie das Flag `OP_ANONYMOUS` ein, das IMAP mitteilt, dass Sie ein anonymer Leser sind. IMAP wird dann keinen Eintrag über Sie in einer speziellen *.newsrc*-Datei anlegen.

Sobald Sie verbunden sind, wollen Sie normalerweise entweder eine allgemeine Liste mit neueren Nachrichten abrufen oder alle Details einer bestimmten Nachricht einsehen. Der folgende Code zeigt neuere Nachrichten an:

```
// Posting-Index lesen und anzeigen.
$last = imap_num_msg($nntp);
$n = 10; // letzte 10 Nachrichten anzeigen
```

```
// Tabellenkopf
print <<<EOH
<table>
<tr>
    <th align="left">Betreff</th>
    <th align="left">Absender</th>
    <th align="left">Datum</th>
</tr>
EOH;

// Nachrichten
for ($i = $last-$n+1; $i <= $last; $i++) {
    $header = imap_header($nntp, $i);

    if (! $header->Size) { continue; }

    $subj  = $header->subject;
    $from  = $header->from;
    $email = $from[0]->mailbox."@".$from[0]->host;
    $name  = $from[0]->personal ? $from[0]->personal : $email;
    $date  = date('m/d/Y h:i A', $header->udate);

print <<<EOM
<tr>
    <td><a href="$_SERVER[PHP_SELF]"?msg=$i\">$subj</a></td>
    <td><a href="mailto:$email">$name</a></td>
    <td>$date</td>
</tr>
EOM;
     }

// Tabellenfuß
echo "</table>\n";
```

Um eine Liste mit Posts durchzublättern, müssen Sie den gewünschten Post mit einer Nummer angeben. Die allererste Nachricht einer Gruppe erhält Nummer 1, die neuste Nachricht den Rückgabewert von `imap_num_msg()`. Um also die neusten `$n` Nachrichten zu erhalten, durchlaufen Sie eine Schleife von `$last-$n+1` bis `$last`.

Innerhalb der Schleife rufen Sie `imap_header()`auf, um die Kopfzeilen-Informationen der Nachricht herunterzuladen. Die Kopfzeilen enthalten die gesamten Meta-Informationen, nicht aber den eigentlichen Text der Nachricht – der ist im Hauptteil gespeichert. Da die Kopfzeilen normalerweise viel weniger umfangreich sind als der Body, können Sie dadurch Daten über viele Nachrichten ohne großen Zeitaufwand herunterladen.

Jetzt übergeben Sie `imap_header()` zwei Parameter: den Handle der Server-Verbindung und die Nachrichtennummer. `imap_header()` gibt ein Objekt mit vielen Eigenschaften zurück, die in Tabelle 20-2 aufgeführt sind.

Tabelle 20-2: imap_header()-Felder von einem NNTP-Server

Name	Beschreibung	Typ	Beispiel
`date` oder `Date`	nach RFC 822 formatiertes Datum: `date('r')`	String	Fri, 16 Aug 2002 01:52:24 -0400
`subject` oder `Subject`	Betreff der Nachricht	String	Re: PHP Cookbook Revisions
`message_id`	eine eindeutige ID, die die Nachricht identifiziert	String	<20030410020818. 33915.php@news.example.com>
`newsgroups`	der Name der Newsgruppe, an die die Nachricht geschickt wurde	String	php.general
`toaddress`	die Adresse, an die die Nachricht geschickt wurde	String	php-general@lists.php.net
`to`	zerlegte Version des `toaddress`-Felds	Objekt	Mailbox: »php-general«, host: »lists-php.net«
`fromaddress`	die Adresse, die die Nachricht geschickt hat	String	Ralph Josephs <ralph@example.net>
`from`	zerlegte Version des `fromaddress`-Felds	Objekt	Personal: »Ralph Josephs«, mailbox: »ralph«, host: »example.net«
`reply_toaddress`	die Adresse, an die Sie die Antwort schicken sollten, wenn Sie den Autor kontaktieren wollen	String	rjosephs@example.net
`reply_to`	zerlegte Version des `reply_toaddress`-Felds	Objekt	Mailbox: »rjosephs«, host: »example.net«
`senderaddress`	die Person, die die Nachricht verschickt hat, fast immer mit dem `from`-Feld identisch; sollte das `from`-Feld den Absender nicht eindeutig identifizieren, übernimmt das dieses Feld	String	Ralph Josephs <ralph@example.net>
`sender`	Zerlegte Version des `senderaddress`-Felds	Objekt	Personal: »Ralph Josephs«, mailbox: »ralph«, host: »example.net«
`Recent`	gibt an, ob die Nachricht erst kürzlich gesendet wurde bzw. ob sie neu eingetroffen ist, seitdem der Benutzer das letzte Mal auf neue Nachrichten geprüft hat	String	Y oder N
`Unseen`	Ist die Nachricht ungesehen?	String	Y oder »«
`Flagged`	Ist die Nachricht markiert?	String	Y oder »«
`Answered`	Wurde diese Nachricht beantwortet?	String	Y oder »«
`Deleted`	Wurde die Nachricht gelöscht?	String	Y oder »«
`Draft`	Ist die Nachricht ein Entwurf?	String	Y oder »«
`Size`	Nachrichtengröße in Bytes	String	1345
`udate`	Unix-Zeitstempel des Nachrichtendatums	Int	1013480645
`Mesgno`	Anzahl der Nachrichten in der Gruppe	String	34943

Einige der gebräuchlicheren Felder sind `size`, `subject`, die `from`-Liste und `udate`. Die Eigenschaft `size` stellt die Größe der Nachricht in Byte dar. Wenn sie 0 ist, wurde die

Nachricht entweder gelöscht oder auf andere Art entfernt. Das Feld `subject` ist der Betreff der Nachricht. Die `from`-Liste ist komplizierter. Sie ist ein Array von Objekten: Jedes Element im Array enthält ein Objekt mit drei Eigenschaften: `personal`, `mailbox` und `host`. Das Feld `personal` ist der Name des Senders: `Homer Simpson`. Das Feld `mailbox` ist der Teil der E-Mail-Adresse, der vor dem @-Zeichen steht: `homer`. Der `host` ist der Teil der E-Mail-Adresse, der nach dem @-Zeichen steht: `thesimpsons.com`. Normalerweise steht nur ein Element im Array der `from`-Liste, da die Nachricht meistens nur einen Absender hat.

Kopieren Sie das Objekt `$header->from` zur besseren Übersichtlichkeit nach `$from`. Dann verbinden Sie `$from[0]->mailbox` und `$from[0]->host`, um die E-Mail-Adresse des Absenders zu erhalten. Verwenden Sie den ternären Operator, um dem Feld `personal` den Namen des Absenders zuzuweisen, wenn dieser zur Verfügung steht; ansonsten geben Sie die E-Mail-Adresse an.

Das Feld `udate` gibt die Sendezeit als Unix-Zeitstempel an. Verwenden Sie `date()`, um diese von Sekunden in ein anwenderfreundlicheres Format zu konvertieren.

Außerdem können Sie eine bestimmte Nachricht wie folgt einsehen:

```
// Eine einzelne Nachricht lesen und anzeigen.
$header = imap_header($nntp, $msg);

$subj  = $header->subject;
$from  = $header->from;
$email = $from[0]->mailbox."@".$from[0]->host;
$name  = $from[0]->personal;
$date  = date('m/d/Y h:i A', $header->udate);
$body  = nl2br(htmlspecialchars(imap_fetchbody($nntp,$msg,1)));

print <<<EOM
<table>
<tr>
    <th align=left>Von:</th>
    <td>$name &lt;<a href="mailto:$email">$email</a>&gt;</td>
</tr>
<tr>
    <th align=left>Betreff:</th>
    <td>$subj</td>
</tr>
<tr>
    <th align=left>Datum:</th>
    <td>$date</td>
</tr>
<tr>
    <td colspan="2">$body</td>
</tr>
</table>
EOM;
```

Der Code zum Herunterladen einer einzelnen Nachricht ähnelt dem zum Herunterladen der Kopfzeilenliste. Der Hauptunterschied besteht darin, dass Sie hier eine Variable `$body`

definieren, die das Ergebnis von drei miteinander verketteten Funktionen ist. Ganz im Innersten rufen Sie `imap_fetchbody()`auf, um den Nachrichten-Hauptteil zu bekommen. Diese Funktion hat die gleichen Parameter wie `imap_header()`. Den Rückgabewert geben Sie an `htmlspecialchars()`weiter, um jegliches störende HTML durch eine Escape-Sequenz unschädlich zu machen. Das Ergebnis wird dann an `nl2br()` weitergereicht, das alle Carriage Returns in XHTML-`<br/>`-Tags konvertiert. Damit sollte die Nachricht auf einer Webseite korrekt aussehen.

Um die Verbindung zum IMAP-Server zu trennen und den Datenstrom zu schließen, übergeben Sie das IMAP-Verbindungs-Handle an `imap_close()`:

```
// Am Ende Verbindung schließen.
imap_close($nntp);
```

Siehe auch

Rezept 20.4 für mehr Informationen zum Senden an Newsgruppen; die Dokumentationen zu `imap_open()` unter *http://www.php.net/imap-open*, `imap_header()` unter *http://www.php.net/imap-header*, `imap_body()` unter *http://www.php.net/imap-body* und zu IMAP im Allgemeinen unter *http://www.php.net/imap*; Code zum Lesen von Newsgruppen mit PHP ohne Verwendung von IMAP unter *http://cvs.php.net/cvs.php/php-news-web*; RFC 977 unter *http://www.faqs.org/rfcs/rfc977.html*.

20.6 Dateien mit FTP herauf- und herunterladen

Problem

Sie wollen Dateien mittels FTP übertragen.

Lösung

Verwenden Sie die eingebauten FTP-Funktionen von PHP:

```
$c = ftp_connect('ftp.example.com')     or die("Verbindung fehlgeschlagen");
ftp_login($c, $username, $password)     or die("Login fehlgeschlagen");
ftp_put($c, $remote, $local, FTP_ASCII) or die("Transfer fehlgeschlagen");
ftp_close($c);                          or die("Verbindung kann nicht geschlossen
werden");
```

Sie können auch die cURL-Erweiterung verwenden:

```
$c = curl_init("ftp://$username:$password@ftp.example.com/$remote");
// $local ist der Pfad, unter der die Datei
// auf der lokalen Maschine gespeichert werden soll.
$fh = fopen($local, 'w') or die($php_errormsg);
curl_setopt($c, CURLOPT_FILE, $fh);
curl_exec($c);
curl_close($c);
```

Diskussion

FTP steht für File Transfer Protocol und bezeichnet ein Verfahren zum Austausch von Dateien zwischen zwei Computern. Anders als bei HTTP-Servern ist es einfach, einen FTP-Server sowohl zum Senden als auch zum Empfangen von Dateien einzurichten.

Die Verwendung der eingebauten FTP-Funktionen erfordert zwar keine zusätzlichen Funktionsbibliotheken, Sie müssen die Funktionen aber beim Kompilieren explizit mit `--enable-ftp` freischalten. Da diese Funktionen auf FTP spezialisiert sind, ist es einfach, mit ihnen Dateien zu übertragen.

Alle FTP-Transaktionen fangen damit an, dass Sie eine Verbindung von Ihrem Computer (dem lokalen Client) zu einem anderen Computer (dem entfernten Server) herstellen:

```
$c = ftp_connect('ftp.example.com')     or die("Verbindung fehlgeschlagen");
```

Wenn Sie verbunden sind, müssen Sie Ihren Benutzernamen und Ihr Passwort senden, damit Sie der Server am anderen Ende der Verbindung authentifizieren und Ihnen Zugang gewähren kann:

```
ftp_login($c, $username, $password)     or die("Login fehlgeschlagen");
```

Bei einigen FTP-Servern ist so genanntes anonymes FTP (anonymous FTP) möglich. Unter anonymem FTP können sich Benutzer auch ohne ein Zugangskonto in das entfernte System einloggen. Wenn Sie anonymes FTP verwenden, ist Ihr Benutzername `anonymous`, und Ihr Passwort besteht aus Ihrer E-Mail-Adresse.

Und so übertragen Sie Dateien mit `ftp_put()` und `ftp_get()`:

```
ftp_put($c, $remote, $local,  FTP_ASCII) or die("Transfer fehlgeschlagen");
ftp_get($c, $local,  $remote, FTP_ASCII) or die("Transfer fehlgeschlagen");
```

Die Funktion `ftp_put()` nimmt eine Datei auf Ihrer Maschine und kopiert sie auf den entfernten Server; `ftp_get()` kopiert eine Datei auf dem entfernten Server auf Ihren Computer. Im voranstehenden Code ist `$remote` der Pfadname der Datei auf dem Server, und `$local` zeigt auf die Datei auf Ihrem Rechner.

Außerdem wird der Funktion noch ein abschließender Parameter mit zwei möglichen Werten übergeben. Der hier verwendete Wert `FTP_ASCII` überträgt die Datei so, als bestünde sie aus ASCII-Text. Mit dieser Option werden Zeilenumbrüche automatisch konvertiert, wenn Sie die Datei zwischen verschiedenen Betriebssystemen verschieben. Der andere mögliche Wert ist `FTP_BINARY`, der für Dateien verwendet wird, die nicht aus reinem Text bestehen. Hier findet keine Konvertierung der Zeilenumbrüche statt.

Verwenden Sie `ftp_fget()`, um eine Datei in ein existierendes Datei-Handle (mit `fopen()` geöffnet) an Stelle einer angegebenen Stelle im Dateisystem zu schreiben. Mit `ftp_fput()` geht es umgekehrt: Hier lesen Sie beim Heraufladen aus einem Datei-Handle. Das Herunterladen einer Datei mit anschließendem Schreiben in einen existierenden Dateizeiger `$fp` geht beispielsweise so:

```
$fp = fopen($file, 'w');
ftp_fget($c, $fp, $remote, FTP_ASCII)   or die("Transfer fehlgeschlagen");
```

Zum Schluss rufen Sie `ftp_close()` auf, um sich aus dem entfernten Host auszuloggen und die Verbindung zu beenden:

```
ftp_close($c);   or die("Verbindung kann nicht geschlossen werden");
```

Die Anzahl der Sekunden für ein Verbindungs-Timeout stellen Sie mit `ftp_set_option()` ein:

```
// Timeout-Wert auf 2 Minuten heraufsetzen:
set_time_limit(120)
$c = ftp_connect('ftp.example.com');
ftp_set_option($c, FTP_TIMEOUT_SEC, 120);
```

Der voreingestellte Wert ist 90 Sekunden, die voreingestellte maximale Ausführungszeit eines PHP-Skripts, `max_execution_time`, liegt jedoch bei 30 Sekunden. Wenn Ihre Verbindung also einen zu frühen Zeitabbruch aufweist, sollten Sie beide Werte überprüfen.

Um die cURL-Erweiterung zu verwenden, müssen Sie sich cURL von *http://curl.haxx.se/* herunterladen und die Konfigurationsoption `--with-curl` beim Build von PHP setzen. cURL verwenden Sie, indem Sie ein cURL-Handle mit `curl_init()` erzeugen und dann mittels `curl_setopt()` angeben, was Sie machen wollen. Die Funktion `curl_setopt()` nimmt drei Parameter an: eine cURL-Ressource, den Namen einer zu modifizierenden cURL-Konstante sowie den Wert, der dem zweiten Parameter zugewiesen werden soll. In der Lösung hier wird die Konstante `CURLOPT_FILE` verwendet:

```
$c = curl_init("ftp://$username:$password@ftp.example.com/$remote");
// $local ist der Speicherort der Datei auf dem lokalen Client-Rechner.
$fh = fopen($local, 'w') or die($php_errormsg);
curl_setopt($c, CURLOPT_FILE, $fh);
curl_exec($c);
curl_close($c);
```

Die URL der herunterzuladenden Datei übergeben Sie an `curl_init()`. Da die URL mit `ftp://` anfängt, weiß cURL, dass das FTP-Protokoll zu verwenden ist. Statt eines separaten Funktionsaufrufs zum Einloggen in den entfernten Server betten Sie den Benutzernamen und das Passwort direkt in die URL ein. Als Nächstes setzen Sie den Speicherort der Datei auf Ihrem lokalen Server. Dazu öffnen Sie eine Datei namens `$local` zum Schreiben und übergeben das Datei-Handle an `curl_setopt()` als Wert für `CURLOPT_FILE`. Wenn cURL die Datei überträgt, schreibt es automatisch in das Datei-Handle. Sobald alles konfiguriert ist, rufen Sie `curl_exec()` auf, um die Transaktion zu starten, gefolgt von `curl_close()` zum Schließen der Verbindung.

Siehe auch

Dokumentation zur FTP-Erweiterung unter *http://www.php.net/ftp* und zu cURL unter *http://www.php.net/curl*; RFC 959 unter *http://www.faqs.org/rfcs/rfc969.html*.

20.7 Adressen über LDAP abfragen

Problem

Sie möchten einen LDAP-Server nach Adressinformationen abfragen.

Lösung

Verwenden Sie die LDAP-Erweiterung von PHP:

```
$ds = ldap_connect('ldap.example.com')                or die($php_errormsg);
ldap_bind($ds)                                        or die($php_errormsg);
$sr = ldap_search($ds, 'o=Example Inc., c=US', 'sn=*') or die($php_errormsg);
$e  = ldap_get_entries($ds, $sr)                      or die($php_errormsg);

for ($i=0; $i < $e['count']; $i++) {
    echo $e[$i]['cn'][0] . ' (' . $e[$i]['mail'][0] . ')<br>';
}

ldap_close($ds)                                       or die($php_errormsg);
```

Diskussion

LDAP steht für Lightweight Directory Access Protocol. Ein LDAP-Server speichert Verzeichnis-Informationen, wie beispielsweise Namen und Adressen, und lässt Sie diese abfragen. In vielerlei Hinsicht ist das wie bei einer Datenbank, außer dass es für die Speicherung von Informationen über Personen optimiert ist.

Zusätzlich zur flachen Struktur, wie sie von einer Datenbank zur Verfügung gestellt wird, ermöglicht ein LDAP-Server Ihnen, Personen in hierarchischen Strukturen unterzubringen. So lassen sich Angestellte beispielsweise in Marketing-, technische und Betriebsabteilungen unterteilen, oder Sie können sie regional in Nordamerika, Europa und Asien aufteilen. Das vereinfacht das Auffinden aller Angestellten einer bestimmten Gruppe innerhalb einer Firma.

Unter LDAP nennt man den Adressenspeicher die *data source* (Datenquelle). Jeder Eintrag in diesem Speicher hat eine global eindeutige Kennzeichnung, auch *Distinguished Name* genannt. Der Distinguished Name enthält sowohl den Namen der Person als auch zugehörige Firmeninformationen. Zum Beispiel hat John Q. Smith, der für die U.S.-Firma Example Inc. arbeitet, einen Distinguished Name wie folgt: `cn=John Q. Smith, o=Example Inc., c=US`. In LDAP steht `cn` für *common name* (gemeiner Name), `o` für *organization* und `c` für *country*.

Die Unterstützung für LDAP in PHP müssen Sie über `--with-ldap` explizit einschalten. Einen LDAP-Server können Sie von *http://www.openldap.org* herunterladen. Dieses Rezept geht davon aus, dass Sie grundlegende Kenntnisse über LDAP besitzen. Weitergehende Informationen finden Sie in den Artikeln im O'Reilly Network unter *http://www.onlamp.com/topics/apache/ldap*.

Die Kommunikation mit einem LDAP-Server erfordert vier Schritte: Verbindung herstellen, Authentifizierung, Suche nach Einträgen und Ausloggen. Außer der Suche nach Einträgen können Sie auch Einträge hinzufügen, ändern und löschen.

In der eröffnenden Transaktion müssen Sie sich mit einem spezifischen LDAP-Server verbinden und sich dann in einem Prozess namens *binding* (Bindung) bei dem Server authentifizieren:

```
$ds = ldap_connect('ldap.example.com')                or die($php_errormsg);
ldap_bind($ds)                                        or die($php_errormsg);
```

Wenn Sie nur das Verbindungs-Handle `$ds` an `ldap_bind( )` übergeben, führt die Funktion eine anonyme Bindung aus. Um sich mit einem bestimmten Benutzernamen und Passwort zu binden, geben Sie diese als zweiten und dritten Parameter an:

```
ldap_bind($ds, $username, $password)                  or die($php_errormsg);
```

Sobald Sie eingeloggt sind, können Sie Informationen anfordern. Da die Informationen in einer Hierarchie angeordnet sind, müssen Sie den zu Grunde liegenden Distinguished Name als zweiten Parameter angeben. Zum Schluss übergeben Sie die Suchkriterien. Wenn Sie beispielsweise alle Beschäftigten mit dem Nachnamen (surname) `Jones` der Firma `Example Inc.` in den `US` finden wollen, geht das so:

```
$sr = ldap_search($ds, 'o=Example Inc., c=US', 'sn=Jones') or die($php_errormsg);
$e  = ldap_get_entries($ds, $sr)                          or die($php_errormsg);
```

Sobald `ldap_search( )` Ergebnisse liefert, verwenden Sie `ldap_get_entries( )`, um die spezifischen Dateneinträge zu lesen. Dann durchlaufen Sie das Array `$e` mit den Einträgen:

```
for ($i=0; $i < $e['count']; $i++) {
    echo $e[$i]['cn'][0] . ' (' . $e[$i]['mail'][0] . ')<br>';
}
```

Statt `count($e)` können Sie hier die bereits berechnete Anzahl der Einträge verwenden, die sich in `$e['count']` befindet. Innerhalb der Schleife drucken Sie den Common Name und die E-Mail-Adresse jedes Eintrags aus, zum Beispiel:

```
David Sklar (sklar@example.com)
Adam Trachtenberg (adam@example.com)
```

Die Funktion `ldap_search( )` durchsucht den gesamten Baum bei und unter dem zu Grunde liegenden Distinguished Name. Um die Ergebnisse auf eine bestimmte Ebene zu begrenzen, können Sie `ldap_list( )` verwenden. Da die Suche über einen kleineren Satz von Einträgen ausgeführt wird, kann `ldap_list( )` bedeutend schneller als `ldap_search( )` sein.

Siehe auch

Rezept 20.7 zur Benutzer-Authentifizierung mittels LDAP; die Dokumentation zu LDAP unter *http://www.php.net/ldap*, zu RFC 2251 unter *http://www.faqs.org/rfcs/rfc2251.html*.

20.8 LDAP zur Benutzer-Authentifizierung verwenden

Problem

Sie wollen Teile Ihrer Site auf authentifizierte Benutzer beschränken. Statt Personen gegen eine Datenbank zu authentifizieren oder HTTP Basic Authentication zu verwenden, wollen Sie einen LDAP-Server verwenden. Alle Benutzerinformationen in einem LDAP-Server zu versammeln macht zentralisierte Benutzerverwaltung einfacher.

Lösung

Verwenden Sie die `Auth`-Klasse aus PEAR, die LDAP-Authentifizierung unterstützt:

```
$options = array('host'     => 'ldap.example.com',
                 'port'     => '389',
                 'base'     => 'o=Example Inc., c=US',
                 'userattr' => 'uid');

$auth = new Auth('LDAP', $options);

// Überprüfung beginnen.
// Login-Maske für anonyme Benutzer ausgeben.
$auth->start();

if ($auth->getAuth()) {
    // Inhalt für zugelassene Benutzer
} else {
    // Inhalt für anonyme Benutzer
}

// Benutzer ausloggen.
$auth->logout();
```

Diskussion

LDAP-Server sind für das Speichern, Suchen und Abrufen von Adressen ausgelegt und daher besser geeignet als Standard-Datenbanken wie MySQL oder Oracle. LDAP-Server sind sehr schnell. Sie können einfach Zugangsprivilegien für verschiedene Benutzergruppen definieren, und viele verschiedene Programme können den Server abfragen. So können beispielsweise die meisten E-Mail-Client-Programme einen LDAP-Server als Adressbuch verwenden, das heißt, wenn Sie eine Nachricht an »John Smith« adressieren, antwortet der Server mit Johns E-Mail-Adresse, *jsmith@example.com*.

Die `Auth`-Klasse von PEAR ermöglicht es Ihnen, Benutzer über Dateien, Datenbanken und LDAP-Server zu überprüfen. Der erste Parameter ist der zu verwendende Authentifizierungstyp, und der zweite besteht aus einem Array mit Informationen darüber, wie die Benutzer überprüft werden sollen. Ein Beispiel:

```
$options = array('host'     => 'ldap.example.com',
                 'port'     => '389',
                 'base'     => 'o=Example Inc., c=US',
                 'userattr' => 'uid');

$auth = new Auth('LDAP', $options);
```

Dieser Code erzeugt ein neues `Auth`-Objekt, das seine Authentifizierungsinformationen über einen LDAP-Server bezieht, der unter *ldap.example.com* firmiert und über Port `389` kommuniziert. Der Name des Basisverzeichnisses ist `o=Example Inc., c=US`, und Benutzernamen werden mit dem `uid`-Attribut verglichen. Das `uid`-Feld steht für *user identifier*. Dabei handelt es sich üblicherweise um einen Benutzernamen für eine Website oder einen Login-Namen für ein allgemeines Benutzerkonto. Wenn Ihr Server `uid`-Attribute nicht für jeden Benutzer speichert, können Sie es auch durch das `cn`-Attribut ersetzen. Das Feld für den Common Name enthält den vollen Namen eines Benutzers, wie z.B. »John Q. Smith«.

Die Methode `Auth::auth()` nimmt auch einen optionalen dritten Parameter auf – den Namen einer Funktion, die das Login-Formular anzeigt. Dieses Formular können Sie nach Belieben formatieren, die einzige Bedingung ist, dass die Eingabefelder des Formulars `username` und `password` heißen müssen. Außerdem muss das Formular die Daten mittels POST einsenden.

```
$options = array('host'     => 'ldap.example.com',
                 'port'     => '389',
                 'base'     => 'o=Example Inc., c=US',
                 'userattr' => 'uid');

function pc_auth_ldap_signin() {
    print<<<_HTML_
<form method="post" action="$_SERVER[PHP_SELF]">
Name: <input name="username" type="text"><br />
Pa&szlig;wort: <input name="password" type="password"><br />
<input type="submit" value="Einloggen">
</form>
_HTML_;
}

$auth = new Auth('LDAP', $options, 'pc_auth_ldap_signin');
```

Sobald das `Auth`-Objekt instantiiert ist, authentifizieren Sie einen Benutzer, indem Sie `Auth::start()` aufrufen:

```
$auth->start();
```

Wenn der Benutzer bereits eingeloggt ist, passiert nichts. Ist der Benutzer unbekannt, wird das Login-Formular ausgegeben. Um einen Benutzer zu verifizieren, verbindet sich `Auth::start()` mit dem LDAP-Server, führt eine anonyme Bindung durch und sucht nach einer Adresse, für die das im Konstruktor angegebene User-Attribut dem durch das Formular übergebenen Benutzernamen entspricht:

```
$options['userattr'] == $_POST['username']
```

Falls Auth::start( ) genau eine Person findet, auf die diese Kriterien zutreffen, besorgt es sich den Distinguished Name für den Benutzer und versucht eine authentifizierte Bindung unter Verwendung des Distinguished Name und des Passworts aus dem Formular. Der LDAP-Server vergleicht dann das Passwort mit dem userPassword-Attribut, das dem Distinguished Name zugeordnet ist. Wenn sie übereinstimmen, ist der Benutzer authentifiziert.

Sie können Auth::getAuth( ) aufrufen, um einen Booleschen Wert zu erhalten, der den Status des Benutzers anzeigt:

```
if ($auth->getAuth( )) {
    print 'Willkommen als Mitglied! Gut, Sie wieder bei uns zu sehen.';
} else {
    print 'Willkommen als Gast. Ist dies Ihr erster Besuch?';
}
```

Die Auth-Klasse verwendet das eingebaute Session-Modul, um Benutzer zu verfolgen. Eine einmal eingeloggte Person bleibt daher authentifiziert, bis die Session verfällt oder bis Sie sie explizit mit ausloggen:

```
$auth->logout( );
```

Siehe auch

Rezept 20.7 zum Suchen auf LDAP-Servern; PEARs Auth-Klasse unter *http://pear.php.net/package/Auth*.

20.9 DNS-Lookups ausführen

Problem

Sie möchten einen Domainnamen oder eine IP-Adresse herausfinden.

Lösung

Verwenden Sie gethostbyname( ) und gethostbyaddr( ):

```
$ip   = gethostbyname('www.example.com'); // 192.0.34.72
$host = gethostbyaddr('192.0.34.72'); // www.example.com
```

Diskussion

Dem von gethostbyaddr( ) zurückgelieferten Namen können Sie nicht unbedingt vertrauen. Ein DNS-Server, der für eine bestimmte IP-Adresse autoritativ ist, kann einen beliebigen Hostnamen zurückgeben. Normalerweise konfigurieren Systemadministratoren DNS-Server so, dass sie mit einem korrekten Hostnamen antworten. Ein böswilliger Be-

nutzer könnte seinen DNS-Server aber so konfigurieren, dass dieser mit inkorrekten Hostnamen antwortet. Eine Möglichkeit, dieses Täuschungsmanöver abzuwenden, besteht darin, gethostbyname( ) mit dem von gethostbyaddr( ) zurückgegebenen Hostnamen aufzurufen und so sicherzustellen, dass der Name der ursprünglichen IP-Adresse entspricht.

Wenn die jeweilige Funktion die IP-Adresse oder den Domainnamen nicht erfolgreich herausfinden kann, gibt sie nicht false, sondern das ihr übergebene Argument zurück. Um auf ein Fehlschlagen zu prüfen, gehen Sie so vor:

```
if ($host == ($ip = gethostbyname($host))) {
    // Lookup fehlgeschlagen.
}
```

Dieser Code speichert den Rückgabewert von gethostbyname( ) in $ip und überprüft außerdem, ob $ip gleich dem ursprünglichen $host ist.

Gelegentlich kann ein einzelner Hostname auf mehrere IP-Adressen abbilden. Um alle Host-Adressen zu finden, verwenden Sie gethostbynamel( ):

```
$hosts = gethostbynamel('www.yahoo.com');
print_r($hosts);
Array
(
    [0] => 64.58.76.176
    [1] => 64.58.76.224
    [2] => 64.58.76.177
    [3] => 64.58.76.227
    [4] => 64.58.76.179
    [5] => 64.58.76.225
    [6] => 64.58.76.178
    [7] => 64.58.76.229
    [8] => 64.58.76.223
)
```

Im Gegensatz zu gethostbyname( ) und gethostbyaddr( ) gibt gethostbynamel( ) anstatt eines Strings ein Array zurück.

Kompliziertere DNS-bezogene Operationen sind ebenfalls möglich. Zum Beispiel lassen sich die MX-Einträge mit getmxrr( )holen:

```
getmxrr('yahoo.com', $hosts, $weight);
for ($i = 0; $i < count($hosts); $i++) {
    echo "$weight[$i] $hosts[$i]\n";
}
5 mx4.mail.yahoo.com
1 mx2.mail.yahoo.com
1 mx1.mail.yahoo.com
```

Um Zonentransfers, dynamische DNS-Updates und weitere Transaktionen durchzuführen, sollten Sie sich das Net_DNS-Paket von PEAR ansehen.

Siehe auch

Die Dokumentationen zu `gethostbyname()` unter *http://www.php.net/gethostbyname*, `gethostbyaddr()` unter *http://www.php.net/gethostbyaddr*, `gethostbynamel()` unter *http://www.php.net/gethostbynamel* und `getmxrr()` unter *http://www.php.net/getmxrr*; PEARs `Net_DNS`-Paket unter *http://pear.php.net/package/Net_DNS*; *DNS und BIND* von Paul Albitz und Cricket Liu (O'Reilly Verlag).

20.10 Überprüfen, ob ein Host erreichbar ist

Problem

Sie möchten einen Host pingen, um zu sehen, ob er läuft und von Ihrem Rechner aus zu erreichen ist.

Lösung

Verwenden Sie das `Net_Ping`-Paket von PEAR:

```
require 'Net/Ping.php';

$ping = new Net_Ping;
if ($ping->checkhost('www.oreilly.de')) {
    print 'Erreichbar';
} else {
    print 'Erreichbar';
}

$data = $ping->ping('www.oreilly.de');
```

Diskussion

Das *ping*-Programm versucht, ein Datentelegramm von Ihrer Maschine an eine andere zu senden. Wenn alles funktioniert, erhalten Sie eine Reihe von Statistiken, die diese Transaktion aufbereiten. Ein Fehler bedeutet, dass *ping* den Host aus irgendwelchen Gründen nicht erreichen kann.

Im Fehlerfall gibt `Net_Ping::checkhost()` `false` zurück, und `Net_Ping::ping()` liefert die Konstante `PING_HOST_NOT_FOUND`. Falls es bei der Ausführung von *ping* ein Problem gibt (weil `Net_Ping` ja eigentlich nur eine nette Verpackung für das Programm darstellt), wird `PING_FAILED` zurückgegeben.

Wenn alles klargeht, bekommen Sie ein Array wie das folgende:

```
$results = $ping->ping('www.oreilly.com');

foreach($results as $result) { print "$result\n"; }
```

```
PING www.oreilly.com (209.204.146.22) from 192.168.123.101 :
    32(60) bytes of data.
40 bytes from www.oreilly.com (209.204.146.22): icmp_seq=0 ttl=239
    time=96.704 msec
40 bytes from www.oreilly.com (209.204.146.22): icmp_seq=1 ttl=239
    time=86.567 msec
40 bytes from www.oreilly.com (209.204.146.22): icmp_seq=2 ttl=239
    time=86.563 msec
40 bytes from www.oreilly.com (209.204.146.22): icmp_seq=3 ttl=239
    time=136.565 msec
40 bytes from www.oreilly.com (209.204.146.22): icmp_seq=4 ttl=239
    time=86.627 msec

--- www.oreilly.com ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/mdev = 86.563/98.605/136.565/19.381 ms
```

Net_Ping führt keine Datenanalyse durch, um beispielsweise an den Prozentanteil der verloren gegangenen Pakete oder die durchschnittliche Antwortzeit zu gelangen. Das können Sie aber auch selbst:

```
$results = $ping->ping('www.oreilly.com');

// Letzte Zeile des Arrays holen, äquivalent zu einem nicht-destruktiven array_pop( )
// oder $results[count($results) - 1].
$round_trip = end($results);
preg_match_all('#[ /]([.\d]+)#', $round_trip, $times);

// Daten extrahieren
list($min,$avg,$max,$mdev) = $times[1];
// oder ausdrucken.
foreach($times[1] as $time) { print "$time\n"; }
83.229
91.230
103.223
7.485
```

Dieser reguläre Ausdruck sucht entweder nach einem Leerzeichen oder nach einem Schrägstrich. Danach liest er eine Folge von einer oder mehreren Ziffern sowie einem Dezimalpunkt ein. Um / nicht mit einem Escape-Zeichen versehen zu müssen, verwenden Sie das Sonderzeichen # als Begrenzungszeichen.

Siehe auch

PEARs Net_Ping-Paket unter *http://pear.php.net/package/Net_Ping*.

20.11 Informationen über einen Domainnamen herausfinden

Problem

Sie möchten Kontaktinformationen oder andere Details über einen Domainnamen herausfinden.

Lösung

Verwenden Sie die Net_Whois-Klasse von PEAR:

```
require 'Net/Whois.php';
$server = 'whois.networksolutions.com';
$query = 'example.org';

$whois = new Net_Whois;
$data = $whois->query($server, $query);
```

Diskussion

Die Methode Net_Whois::query( ) gibt einen langen Text-String zurück, dessen Inhalt daran erinnert, wie schwer es sein kann, verschiedene Whois-Resultate zu analysieren:

```
Registrant:
Internet Assigned Numbers Authority (EXAMPLE2-DOM)
   4676 Admiralty Way, Suite 330
   Marina del Rey, CA 90292
   US

   Domain Name: EXAMPLE.ORG

   Administrative Contact, Technical Contact, Billing Contact:
      Internet Assigned Numbers Authority  (IANA)  iana@IANA.ORG
      4676 Admiralty Way, Suite 330
      Marina del Rey, CA 90292
      US
      310-823-9358
      Fax- 310-823-8649

   Record last updated on 07-Jan-2002.
   Record expires on 01-Sep-2009.
   Record created on 31-Aug-1995.
   Database last updated on 6-Apr-2002 02:56:00 EST.

   Domain servers in listed order:

   A.IANA-SERVERS.NET               192.0.34.43
   B.IANA-SERVERS.NET               193.0.0.236
```

Wenn Sie beispielsweise die Namen und IP-Adressen der Domainname Server herauspicken wollen, verwenden Sie:

```
preg_match_all('/^\s*([\S]+)\s+([\d.]+)\s*$/m', $data, $dns,
               PREG_SET_ORDER);

foreach ($dns as $server) {
    print "$server[1] : $server[2]\n";
}
```

Sie müssen $server auf den korrekten Whois-Server für die entsprechende Domain setzen, um Informationen über diese Domain zu erhalten. Wenn Sie nicht wissen, welchen Server Sie verwenden müssen, können Sie *whois.internic.net* fragen:

```
require 'Net/Whois.php';

$whois = new Net_Whois;
print $whois->query('example.org', 'whois.internic.net');
[whois.internic.net]

Whois Server Version 1.3

Domain names in the .com, .net, and .org domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.

   Domain Name: EXAMPLE.ORG
   Registrar: NETWORK SOLUTIONS, INC.
   Whois Server: whois.networksolutions.com
   Referral URL: http://www.networksolutions.com
   Name Server: A.IANA-SERVERS.NET
   Name Server: B.IANA-SERVERS.NET
   Updated Date: 19-aug-2002


>>> Last update of whois database: Wed, 21 Aug 2002 04:56:56 EDT <<<

The Registry database contains ONLY .COM, .NET, .ORG, .EDU domains and
Registrars.
```

Die Zeile mit »Whois Server:« gibt an, dass der korrekte Server zur Beantwortung von Anfragen über *example.org* der Server *whois.networksolutions.com* ist.

Siehe auch

PEARs `Net_Whois`-Klasse unter *http://pear.php.net/package/Net_Whois*.

KAPITEL 21

Dateien

21.0 Einführung

Die Ein- und Ausgaben in einer Webanwendung finden normalerweise zwischen Browser, Server und Datenbank statt; es gibt aber viele Situationen, in denen auch Dateien eine Rolle spielen. Dateien sind nützlich, um entfernte Webseiten für die Verarbeitung vor Ort abzurufen, um Daten ohne Verwendung einer Datenbank zu speichern und um Informationen zu speichern, auf die andere Programme Zugriff brauchen. Mit der Entwicklung von PHP zu mehr als einem bloßen Instrument zum Herauspumpen von Webseiten werden die Datei-I/O(Input/Output)-Funktionen noch nützlicher.

Die PHP-Schnittstelle zur Dateiein- und -ausgabe ähnelt der von C, ist aber nicht so kompliziert. Die grundlegende Identifikation einer Datei, aus der gelesen oder in die geschrieben werden soll, ist ein *Datei-Handle*. Dieses Handle beschreibt Ihre Verbindung zu einer speziellen Datei, und Sie verwenden das Handle für Operationen mit der Datei. Dieses Kapitel konzentriert sich auf das Öffnen und Schließen von Dateien, den Umgang mit Datei-Handles in PHP sowie darauf, was Sie mit dem Dateiinhalt machen können, nachdem Sie eine Datei geöffnet haben. Kapitel 8 beschäftigt sich mit Verzeichnissen und Meta-Informationen über Dateien, wie z.B. Zugriffsberechtigungen.

Das Öffnen von */tmp/cookie-data* und das Schreiben eines bestimmten Cookie-Inhalts in die Datei sieht folgendermaßen aus:

```
$fh = fopen('/tmp/cookie-data','w')      or die("Datei kann nicht geöffnet werden");
if (-1 == fwrite($fh,$_COOKIE['flavor'])) { die("Schreiben nicht möglich"); }
fclose($fh)                              or die("Datei kann nicht geschlossen
                                                 werden");
```

Die Funktion `fopen()` gibt ein Datei-Handle zurück, wenn die Datei erfolgreich geöffnet werden konnte. Wenn sich die Datei nicht öffnen lässt (z.B. aufgrund mangelnder Berechtigung), gibt die Funktion `false` zurück. Die Rezepte 21.1 und 21.3 behandeln Möglichkeiten zum Öffnen von Dateien.

Die Funktion `fwrite()` schreibt den Wert des `flavor`-Cookies in das Datei-Handle. Sie gibt die Anzahl der geschriebenen Bytes zurück. Wenn sie den String nicht schreiben kann (z.B. wegen Platzmangel auf der Disk), gibt sie –1 zurück.

Als Letztes schließt `fclose()` das Datei-Handle. Das wird am Ende eines HTTP-Requests automatisch gemacht. Es ist aber sinnvoll, alle von Ihnen geöffneten Dateien trotzdem ausdrücklich zu schließen. Dadurch werden Probleme verhindert, die durch die Verwendung von Code in einem Kommandozeilen-Kontext entstehen, und belegte Systemressourcen werden wieder freigegeben. Dies ermöglicht Ihnen auch, den zurückgegebenen Code von `fclose()` zu prüfen. Gepufferte Informationen werden eventuell nicht wirklich auf die Festplatte geschrieben, wenn `fclose()` nicht aufgerufen wird. Daher können hier gelegentlich »Festplatte voll«-Meldungen auftauchen, die sonst nicht bemerkt würden.

PHP muss, wie bei anderen Prozessen auch, die richtigen Berechtigungen haben, um aus einer Datei lesen und in eine Datei schreiben zu können. In einem Kommandozeilen-Kontext ist das normalerweise ziemlich unkompliziert, wenn Sie aber Skripten über einen Webserver fahren, kann das Verwirrung stiften. Ihr Webserver – und damit Ihre PHP-Skripte – laufen wahrscheinlich als ein bestimmter Benutzer speziell für Webserver-Zwecke (oder auch als Benutzer `nobody`). Aus guten Sicherheitsgründen hat dieser Benutzer oft eingeschränkte Zugangsrechte im Hinblick auf Dateien. Wenn Ihr Skript mit einem Dateibefehl Schwierigkeiten hat, sollten Sie sicherstellen, dass der Benutzer oder die Gruppe des Webservers – und nicht Sie selbst – die Befugnis hat, den Dateibefehl durchzuführen. Allerdings kann es sein, dass aufgrund der Einstellungen des Webservers Ihr Skript unter Ihrem eigenen Benutzernamen läuft. In diesem Fall müssen Sie sicherstellen, dass Ihre Skripten nicht versehentlich private Dateien, die nicht Teil Ihrer Webseite sind, lesen oder schreiben können.

Da die meisten Dateifunktionen bei einem Fehler nur `false` zurückgeben, müssen Sie zusätzlichen Aufwand betreiben, um Einzelheiten über diesen Fehler herauszufinden. Wenn die Konfigurationsdirektive `track_errors on` ist, wird jede Fehlermeldung in die globale Variable `$php_errormsg` aufgenommen. Wenn Sie diese Variable in Ihre Fehlerausgabe einbeziehen, wird das Debugging vereinfacht:

```
$fh = fopen('/tmp/cookie-data','w')      or die("Kann nicht öffnen: $php_errormsg");
if (-1 == fwrite($fh,$_COOKIE['flavor'])) { die("Kann nicht schreiben:
                                                 $php_errormsg") };
fclose($fh)                              or die("Kann nicht schließen:
                                                 $php_errormsg");
```

Wenn Sie für */tmp/cookie-data* keine Schreiberlaubnis haben, bricht das Beispiel mit der folgenden Fehlerausgabe ab:

```
Kann nicht öffnen: ("/tmp/cookie-data", "w") - Permission denied
```

Zwischen Windows und Unix gibt es Unterschiede in der Behandlung von Dateien. Um sicherzugehen, dass Ihr Dateizugriffs-Code unter Unix und Windows richtig funktioniert, müssen Sie darauf achten, mit den Zeilenumbruchzeichen und Pfadnamen richtig umzugehen.

Ein Zeilenumbruch in Windows besteht aus zwei Zeichen: ASCII 13 (Wagenrücklauf bzw. Carriage Return) gefolgt von ASCII 10 (Zeilenvorschub oder Newline). Unter Unix besteht er lediglich aus ASCII 10. Die Namen dieser Zeichen aus Schreibmaschinenzeit zeigen, warum Sie einen »stufenförmigen« Text erhalten, wenn Sie eine Datei mit Unix-

Begrenzungszeichen unter Windows ausdrucken. Stellen Sie sich diese Zeichennamen als Befehle für die Schreibwalze in einer Schreibmaschine oder in einem Drucker vor, der ein Zeichen nach dem anderen druckt. Ein Wagenrücklauf schickt die Schreibwalze zurück zum Anfang der Zeile, in der sie steht. Ein Zeilenvorschub schiebt das Papier um eine Zeile nach oben. Ein falsch konfigurierter Drucker, der auf eine Datei mit Unix-Begrenzungszeichen stößt, folgt gehorsam den Anweisungen und führt einen Zeilenvorschub am Ende jeder Zeile aus. Dadurch wird auf die nächste Zeile vorgerückt, ohne jedoch die horizontale Druckposition auf den linken Rand zurückzusetzen. Die nächste stufenförmige Zeile beginnt (horizontal) dort, wo die vorhergehende Zeile aufhört.

PHP-Funktionen, die Newline als Begrenzungszeichen am Ende der Zeile verwenden (z.B. `fgets()`), arbeiten sowohl unter Windows als auch unter Unix, weil Newline auf beiden Plattformen das Zeichen am Ende der Zeile ist.

Um Zeilenendzeichen zu entfernen, verwenden Sie die PHP-Funktion `rtrim()`:

```
$fh = fopen('/tmp/lines-of-data.txt','r') or die($php_errormsg);
while($s = fgets($fh,1024)) {
    $s = rtrim($s);
    // jetzt etwas mit $s machen...
}
fclose($fh)                              or die($php_errormsg);
```

Diese Funktion entfernt etwaig angehängten Leerraum am Ende der Zeile, einschließlich ASCII 13 und ASCII 10 (sowie Tabulator-Zeichen und normalen Leerzeichen). Wenn am Ende einer Zeile ein Leerzeichen ist, das Sie beibehalten möchten, obwohl Sie trotzdem die Wagenrückläufe und Zeilenvorschübe entfernen wollen, verwenden Sie einen passenden regulären Ausdruck:

```
$fh = fopen('/tmp/lines-of-data.txt','r') or die($php_errormsg);
while($s = fgets($fh,1024)) {
    $s = preg_replace('/\r?\n$/','',$s);
    // jetzt etwas mit $s machen...
}
fclose($fh)                              or die($php_errormsg);
```

Unix und Windows unterscheiden sich auch im Zeichen, das für die Trennung von Verzeichnissen in Pfadnamen verwendet wird. Unix verwendet einen normalen Schrägstrich (`/`) und Windows traditionell einen umgekehrten Strich, einen sogenannten Backslash (`\`). Unter PHP ist das jedoch kein Problem, da Windows auch `/` als Zeichen zur Trennung von Verzeichnissen versteht (wohlgemerkt: Windows, nicht nur die Win-Version von PHP). Zum Beispiel druckt der folgende Code mit Erfolg den Inhalt von *C:\Alligator\Crocodile Menu.txt*:

```
$fh = fopen('c:/alligator/crocodile menu.txt','r') or die($php_errormsg);
while($s = fgets($fh,1024)) {
    print $s;
}
fclose($fh)                                    or die($php_errormsg);
```

Dieses Code-Stück nutzt außerdem aus, dass bei Windows-Dateinamen, im Gegensatz zu Unix-Dateinamen, die Groß- und Kleinschreibung keine Rolle spielt.

Zeilenumbruchprobleme kann es nicht nur in Ihrem Code zum Lesen und Schreiben von Dateien geben, sondern auch in Ihrem Quelltext. Wenn bei Ihnen mehrere Leute an einem Projekt arbeiten, sollten Sie sicherstellen, dass alle Entwickler ihre Editoren so konfigurieren, dass diese die gleiche Art von Zeilenumbrüchen verwenden.

Wenn Sie eine Datei geöffnet haben, stellt PHP Ihnen viele Werkzeuge zur Verarbeitung der enthaltenen Daten zur Verfügung. Im Einklang mit der C-ähnlichen I/O-Schnittstelle von PHP sind die zwei Hauptfunktionen zum Lesen von Daten aus einer Datei `fread()`, die eine bestimmte Anzahl von Bytes liest, und `fgets()`, die eine Zeile nach der anderen liest (bis zu einer bestimmten Anzahl von Bytes). Dieser Code bearbeitet Zeilen mit einer Länge von bis zu 256 Bytes:

```
$fh = fopen('orders.txt','r') or die($php_errormsg);
while (! feof($fh)) {
    $s = fgets($fh,256);
    process_order($s);
}
fclose($fh) or die($php_errormsg);
```

Wenn *orders.txt* eine 300-Byte-Zeile hat, gibt `fgets()` nur die ersten 256 Bytes zurück. Der nächste `fgets()`-Befehl liefert die nächsten 44 Bytes und stoppt, wenn er eine neue Zeile findet. Der nächste Aufruf von `fgets()` springt zur nächsten Zeile der Datei. Beispiele in diesem Kapitel geben `fgets()` normalerweise ein zweites Argument mit dem Wert 1048576: 1 MByte. Das ist in den meisten Textdateien länger als eine Zeile, aber das Vorhandensein einer derart astronomischen Zahl sollte Sie daran erinnern, bei der Verwendung von `fgets()` Ihre erwartete maximale Zeilenlänge zu berücksichtigen.

Viele Operationen mit Dateiinhalten, wie z.B. das zufällige Auswählen einer Zeile (siehe Rezept 21.10), sind vom Konzept her einfacher (und benötigen weniger Code), wenn die gesamte Datei in einen String oder ein Array eingelesen wird. Rezept 21.4 stellt eine Methode zum Einlesen einer Datei in einen String vor. Die Funktion `file()` speichert eine Datei zeilenweise in einem Array ab. Allerdings bezahlen Sie diese Einfachheit mit zusätzlichem Speicherverbrauch. Das kann besonders problematisch sein, wenn Sie PHP als Server-Modul verwenden. Wenn ein Prozess (z.B. ein Webserver-Prozess, in den PHP integriert ist) Speicherkapazität anfordert (wie das PHP macht, um eine gesamte Datei in einen String oder ein Array einzulesen), kann der Prozess diese Speicherkapazität normalerweise erst wieder an das Betriebssystem zurückgeben, wenn er beendet wird. Das heißt, wenn Sie `file()` für eine 1-MByte-Datei aufrufen und PHP als Apache-Modul läuft, erhöht dies die Größe dieses Apache-Prozesses um 1 MByte bis zur Beendigung des Prozesses. Wenn das ein paarmal wiederholt wird, erniedrigt sich dadurch die Server-Effizienz. Es gibt sicherlich gute Gründe dafür, eine gesamte Datei auf einmal zu verarbeiten, aber Sie sollten sich dabei der Auswirkungen auf die Speicherkapazität bewusst sein.

Die Rezepte 21.19 bis 21.21 beschäftigen sich mit der Ausführung von anderen Programmen aus einem PHP-Programm heraus. Manche Operatoren und Funktionen zum Ausführen von Programmen bieten die Möglichkeit, entweder ein Programm laufen zu lassen und die gesamte Ausgabe auf einmal einzulesen (Backticks) oder nur die letzte Zeile der

Ausgabe zu lesen (`system()`). PHP kann Pipelines verwenden, um ein Programm auszuführen, ihm Eingabedaten zuzuleiten oder seine Ausgabe zu lesen. Da eine Pipeline mit den standardmäßigen I/O-Funktionen gelesen wird (`fgets()` und `fread()`), entscheiden Sie, wie Sie die Eingabe gern hätten. Zwischen dem Lesen von Eingabeteilen können Sie auch andere Aufgaben erledigen. Das Schreiben in eine Pipeline geschieht entsprechend mit `fputs()` und `fwrite()`, sodass Sie dem Programm Eingabedaten als Häppchen beliebiger Größe übergeben können.

Bei Pipelines entstehen die gleichen Berechtigungsfragen wie bei normalen Dateien. Der PHP-Prozess benötigt eine Erlaubnis zur Ausführung des Programms, das als Pipeline geöffnet wird. Wenn Sie mit dem Öffnen der Pipeline Schwierigkeiten haben, insbesondere dann, wenn PHP als spezieller Webserver-User läuft, müssen Sie darauf achten, dass der Benutzer die Berechtigung zur Ausführung des Programms hat, zu dem Sie die Pipeline öffnen.

21.1 Eine lokale Datei erstellen oder öffnen

Problem

Sie wollen eine lokale Datei öffnen, um Daten aus ihr zu lesen oder um Daten in sie zu schreiben.

Lösung

Verwenden Sie `fopen()`:

```
$fh = fopen('datei.txt','r') or die("Kann datei.txt nicht öffnen: $php_errormsg");
```

Diskussion

Das erste Argument bei `fopen()` ist die zu öffnende Datei. Das zweite Argument ist der Modus, mit dem die Datei geöffnet werden soll. Der Modus gibt an, welche Operationen mit der Datei durchgeführt werden können (Lesen und/oder Schreiben), wohin nach dem Öffnen der Dateizeiger gesetzt wird (an den Anfang oder ans Ende der Datei), ob die Datei nach dem Öffnen auf die Länge null gekürzt wird und ob die Datei erstellt wird, wenn sie nicht vorhanden ist. Die verschiedenen Modi sind in Tabelle 21-1 dargestellt.

Tabelle 21-1: fopen()-Dateimodi

Modus	Lesbar?	Schreibbar?	Dateizeiger	Kürzen?	Erzeugen?
`r`	Ja	Nein	Anfang	Nein	Nein
`r+`	Ja	Ja	Anfang	Nein	Nein
`w`	Nein	Ja	Anfang	Ja	Ja
`w+`	Ja	Ja	Anfang	Ja	Ja

Tabelle 21-1: fopen()-Dateimodi (Fortsetzung)

Modus	Lesbar?	Schreibbar?	Dateizeiger	Kürzen?	Erzeugen?
a	Nein	Ja	Ende	Nein	Ja
a+	Ja	Ja	Ende	Nein	Ja
x	Nein	Ja	Anfang	Nein	Ja
x+	Ja	Ja	Anfang	Nein	Ja

Die Modi `x` und `x+` liefern `false` zurück und erzeugen eine Warnung, falls die entsprechende Datei bereits existiert. Diese Modi sind seit PHP 4.3.2 verfügbar.

Auf Systemen, die nicht POSIX-kompatibel sind, wie z.B. Windows, müssen Sie beim Öffnen einer binären Datei ein `b` an den Modus anfügen. Ansonsten gibt es beim Lesen oder Schreiben von NUL-Zeichen (ASCII 0) Probleme:

```
$fh = fopen('c:/images/logo.gif','rb');
```

Um eine Operation mit einer Datei durchzuführen, geben Sie das Datei-Handle, das von `fopen()` zurückgegeben wurde, an andere I/O-Funktionen, wie z.B. `fgets()`, `fputs()` und `fclose()`, weiter.

Wenn der an `fopen()` übergebene Dateiname keinen Pfadnamen enthält, wird die Datei im Verzeichnis des laufenden Skripts (Web-Kontext) oder im aktuellen Verzeichnis (Befehlszeilen-Kontext) geöffnet.

Sie können `fopen()` auch anweisen, die zu öffnende Datei im Pfad `include_path`, der in Ihrer Datei *php.ini* festgelegt ist, zu suchen. Dazu geben Sie 1 als drittes Argument an. Dieses, zum Beispiel, sucht nach *file.inc* im `include_path`:

```
$fh = fopen('file.inc','r',1) or die("Kann file.inc nicht öffnen: $php_errormsg");
```

Siehe auch

Die Dokumentation zu `fopen()` unter *http://www.php.net/fopen*.

21.2 Eine temporäre Datei erstellen

Problem

Sie brauchen eine Datei, um vorübergehend Daten zu speichern.

Lösung

Verwenden Sie `tmpfile()`, wenn die Datei nur für die Dauer des laufenden Skripts benötigt wird:

```
$temp_fh = tmpfile();
// Irgendwelche Daten in die temporäre Datei schreiben.
fputs($temp_fh,"Die momentane Zeit ist ".strftime('%c'));
```

```
// Die Datei verschwindet, wenn das Skript beendet wird.
exit(1);
```

Wenn die Datei über einen längeren Zeitraum benötigt wird, erstellen Sie einen Dateinamen mit `tempnam()` und verwenden dann `fopen()`:

```
$tempfilename = tempnam('/tmp','data-');
$temp_fh = fopen($tempfilename,'w') or die($php_errormsg);
fputs($temp_fh,"Die momentane Zeit ist ".strftime('%c'));
fclose($temp_fh) or die($php_errormsg);
```

Diskussion

Die Funktion `tmpfile()` erstellt eine Datei mit einem eindeutigen Namen und gibt ein Datei-Handle zurück. Die Datei wird entfernt, wenn für dieses Datei-Handle `fclose()` aufgerufen oder das Skript beendet wird.

Die zweite Möglichkeit ist die Funktion `tempnam()`, die einen Dateinamen erzeugt. Sie nimmt zwei Argumente: Das erste ist ein Verzeichnis, das zweite ein Präfix für den Dateinamen. Wenn das Verzeichnis nicht vorhanden oder schreibgeschützt ist, verwendet `tempnam()` das temporäre Systemverzeichnis – die `TMPDIR`-Umgebungsvariable unter Unix oder die `TMP`-Umgebungsvariable unter Windows, zum Beispiel:

```
$tempfilename = tempnam('/tmp','data-');
print "Temporäre Daten werden in $tempfilename gespeichert";
Temporäre Daten werden in /tmp/data-GawVoL gespeichert
```

Wenn PHP temporäre Dateinamen erstellt, wird der von `tempnam()` zurückgegebene Dateiname tatsächlich als Datei erstellt und leer gelassen, selbst wenn Ihr Skript die Datei nie ausdrücklich öffnet. Dadurch wird sichergestellt, dass in der Zeit zwischen Ihrem Aufruf von `tempnam()` und Ihrem Aufruf von `fopen()` mit dem Dateinamen kein anderes Programm eine Datei mit dem gleichen Namen erstellt.

Siehe auch

Die Dokumentationen zu `tmpfile()` unter *http://www.php.net/tmpfile* und `tempnam()` unter *http://www.php.net/tempnam*.

21.3 Eine Datei auf einem entfernten Server öffnen

Problem

Sie wollen eine Datei öffnen, zu der Sie über HTTP oder FTP Zugang haben.

Lösung

Geben Sie bei `fopen()` die URL der Datei an:

```
$fh = fopen('http://www.example.com/robots.txt','r') or die($php_errormsg);
```

Um die Datei komplett einzulesen, verwenden Sie file_get_contents():

```
$page = file_get_contents('http://www.example.com/robots.txt');
```

Diskussion

Wenn fopen() oder file_get_contents() ein mit *http://* beginnender Dateiname übergeben wird, rufen diese Funktionen die angegebene Seite mit einem HTTP/1.0 GET-Request auf (obwohl auch ein Host:-Header übergeben wird, um mit virtuellen Hosts umzugehen). Mit dem von fopen() zurückgegebenen Datei-Handle kann nur auf den Body der HTTP-Response zugegriffen werden, nicht aber auf die Header. Dateien können über HTTP zwar gelesen, aber nicht geschrieben werden.

Wenn fopen() ein mit *ftp://* beginnender Dateiname übergeben wird, gibt diese Funktion einen Zeiger, der über Passive-Mode-FTP erhalten wurde, auf die angegebene Datei zurück. Sie können Dateien entweder zum Lesen oder zum Schreiben über FTP öffnen, aber nicht für beides gleichzeitig.

Um mit fopen() URLs zu öffnen, die einen Benutzernamen und ein Passwort benötigen, integrieren Sie die Authentifizierungsinformationen wie folgt in die URL :

```
$fh = fopen('ftp://benutzername:passwort@ftp.example.com/pub/Index','r');
$fh = fopen('http://benutzername:passwort@www.example.com/robots.txt','r');
```

Um eine komplette FTP-Datei mit Inhalt zu füllen, verwenden Sie file_put_contents():

```
file_put_contents('ftp://benutzername:passwort@ftp.example.com/pub/datei.txt','Das ist
der Inhalt');
```

Sie können auch etwas an eine bestehende FTP-Datei anhängen:

```
file_put_contents('ftp://benutzername:passwort@ftp.example.com/pub/datei.txt',
    'Das ist der Inhalt', FILE_APPEND);
```

Das Öffnen von entfernten Dateien mit fopen() wird über einen Stream-Wrapper, den sogenannten *URL-fopen-Wrapper*, durchgeführt. In der Standardeinstellung ist er freigegeben, lässt sich aber durch die Einstellung allow_url_fopen off in Ihrer *php.ini* oder der Webserver-Konfigurationsdatei sperren. Wenn Sie Dateien auf einem entfernten Server nicht mit fopen() öffnen können, sollten Sie Ihre Serverkonfiguration prüfen.

Siehe auch

Rezepte 13.1 bis 13.7, die das Abrufen von URLs behandeln; Rezept 21.4 zum Einlesen von Dateien in einen String; die Dokumentationen zu fopen() unter *http://www.php.net/fopen*, zu file_get_contents() unter *http://www.php.net/file-get-contents*, zu file_put_contents() unter *http://www.php.net/file-put-contents*, zum URL-*fopen*-Wrapper-Feature unter *http://www.php.net/features.remote-files* und *http://www.php.net/wrappers*.

21.4 Eine Datei in einen String einlesen

Problem

Sie möchten den gesamten Inhalt einer Datei in einer Variablen abspeichern, z.B. um herauszufinden, ob der Text in einer Datei auf einen regulären Ausdruck passt.

Lösung

Verwenden Sie `file_get_contents()`:

```
$inhalt = file_get_contents('namen.txt');
```

Diskussion

Es gibt eine einfachere Möglichkeit, den gesamten Inhalt einer Datei auszudrucken, als sie in einen String einzulesen und dann den String auszudrucken. Die Funktion `readfile($filename)` druckt den gesamten Inhalt von `$filename` aus.

Sie können `readfile()` verwenden, um Bilder in einem Wrapper zu verpacken, wenn sie nicht immer angezeigt werden sollen. Das folgende Programm stellt sicher, dass das angeforderte Bild weniger als eine Woche alt ist:

```
$image_directory = '/usr/local/images';

if (preg_match('/^[a-zA-Z0-9]+\.(gif|jpeg)$/',$image,$matches) &&
    is_readable($image_directory."/$image") &&
    (filemtime($image_directory."/$image") >= (time() - 86400 * 7))) {

  header('Content-Type: image/'.$matches[1]);
  header('Content-Length: '.filesize($image_directory."/$image"));

  readfile($image_directory."/$image");

} else {
  error_log("Kann Bild $image nicht ausliefern");
}
```

Damit der Wrapper wasserdicht ist, muss das Verzeichnis `$image_directory`, in dem die Bilder gespeichert sind, außerhalb des Dokumenten-Stammverzeichnisses des Webservers liegen. Sonst kann der Benutzer die Bilddateien (zumindest theoretisch) einfach direkt einsehen. Das Bild wird unter drei Aspekten getestet. Erstens darf der in `$image` durchgegebene Dateiname nur aus alphanumerischen Zeichen bestehen und muss entweder in *.gif* oder *.jpeg* enden. Sie müssen sichergehen, dass keine Zeichen wie z.B. `..` oder `/` im Dateinamen enthalten sind. Das hält böswillige Benutzer davon ab, Dateien außerhalb des angegebenen Verzeichnisses abzurufen. Zum zweiten verwenden Sie `is_readable()`, um sicherzustellen, dass Sie die Datei auch lesen dürfen. Drittens ermitteln Sie die Änderungszeit der Datei mit `filemtime()` und stellen sicher, dass diese Zeit mehr

als 86.400 × 7 Sekunden zurückliegt. Ein Tag hat 86.400 Sekunden, 86.400 × 7 Sekunden sind also eine Woche.[1] Wenn alle drei Bedingungen erfüllt sind, können Sie das Bild senden. Zuerst senden Sie zwei Header, um dem Browser den MIME-Typ und die Dateigröße des Bilds mitzuteilen. Anschließend verwenden Sie `readfile()`, um dem Benutzer den gesamten Inhalt der Datei zu senden.

Siehe auch

Die Dokumentationen zu `file_get_contents()` unter *http://www.php.net/file-get-contents*, zu `readfile()` unter *http://www.php.net/readfile*, zu `filesize()` unter *http://www.php.net/filesize*, zu `fread()` unter *http://www.php.net/fread* und zu `fpassthru()` unter *http://www.php.net/fpassthru*.

21.5 Einen String in eine Datei schreiben

Problem

Sie wollen einen String in eine Datei schreiben.

Lösung

Verwenden Sie `file_put_contents()`:

```
$maerchen = "Es war einmal ...";
file_put_contents("maerchen.txt", $maerchen);
```

Diskussion

Sie können den String auch an eine existierende Datei anhängen:

```
$string1 = "Es war einmal ";
file_put_contents("maerchen.txt", $string1);
$string2 = "vor langer Zeit ...";
file_put_contents("maerchen.txt", $string2, FILE_APPEND);
```

Siehe auch

Rezept 21.3 zum Schreiben in eine entfernte Datei; die Dokumentation zu `file_put_contents()` unter *http://www.php.net/file-put-contents*.

1 Bei der Umschaltung zwischen Sommer- und Winterzeit hat ein Tag mehr bzw. weniger als 86.400 Sekunden. Nähere Einzelheiten stehen in Rezept 3.11.

21.6 Die Zeilen, Absätze oder Datensätze in einer Datei zählen

Problem

Sie wollen die Zeilen, Absätze oder Datensätze in einer Datei zählen.

Lösung

Um Zeilen zu zählen, verwenden Sie `fgets()`. Da sie eine Zeile nach der anderen liest, können Sie zählen, wie oft sie aufgerufen wird, bevor das Dateiende erreicht ist:

```
$lines = 0;

if ($fh = fopen('orders.txt','r')) {
  while (! feof($fh)) {
    if (fgets($fh,1048576)) {
      $lines++;
    }
  }
}
print $lines;
```

Zum Zählen von Absätzen erhöhen Sie den Zähler nur, wenn Sie eine leere Zeile lesen:

```
$paragraphs = 0;

if ($fh = fopen('great-american-novel.txt','r')) {
  while (! feof($fh)) {
    $s = fgets($fh,1048576);
    if (("\n" == $s) || ("\r\n" == $s)) {
      $paragraphs++;
    }
  }
}
print $paragraphs;
```

Und um Datensätze zu zählen, erhöhen Sie den Zähler nur, wenn die gelesene Zeile lediglich die Datensatz-Trennzeichen und Leerzeichen enthält:

```
$records = 0;
$record_separator = '--end--';

if ($fh = fopen('great-american-novel.txt','r')) {
  while (! feof($fh)) {
    $s = rtrim(fgets($fh,1048576));
    if ($s == $record_separator) {
      $records++;
    }
  }
}
print $records;
```

Diskussion

Beim Zeilenzähler wird `$lines` nur erhöht, wenn `fgets()` einen wahren Wert zurückgibt. Während sich `fgets()` durch die Datei bewegt, gibt sie jede Zeile zurück, die sie abruft. Wenn sie die letzte Zeile erreicht, gibt sie `false` zurück, sodass `$lines` nicht fälschlicherweise erhöht wird. Da EOF in der Datei erreicht wurde, gibt `feof()` `true` zurück, und die `while`-Schleife wird beendet.

Der Absatzzähler funktioniert gut bei einfachen Texten, aber liefert eventuell unerwartete Ergebnisse, wenn ein langer String mit leeren Zeilen oder eine Datei ohne zwei aufeinander folgende Zeilenumbrüche vorliegt. Diese Probleme können mit Funktionen auf `preg_split()`-Basis behoben werden. Bei einer kleinen Datei, die in den Speicher gelesen werden kann, verwenden Sie die Funktion `pc_split_paragraphs()`, die in Beispiel 21-1 gezeigt wird. Diese Funktion gibt ein Array zurück, das jeden Absatz in der Datei enthält.

Beispiel 21-1: pc_split_paragraphs()

```
function pc_split_paragraphs($file,$rs="\r?\n") {
    $text = join('',file($file)); // alternativ: file_get_contents()
    $matches = preg_split("/(.*?$rs)(?:$rs)+/s",$text,-1,
                          PREG_SPLIT_DELIM_CAPTURE|PREG_SPLIT_NO_EMPTY);
    return $matches;
}
```

Der Inhalt der Datei wird bei zwei oder mehr aufeinander folgenden Zeilenvorschüben aufgeteilt und im Array `$matches` zurückgegeben. Der standardmäßige reguläre Ausdruck zur Teilung von Absätzen, `\r?\n`, passt auf Zeilenumbrüche sowohl unter Windows als auch unter Unix. Wenn die Datei zu groß ist, um sie auf einmal zu speichern, verwenden Sie die Funktion `pc_split_paragraphs_largefile()`, die in Beispiel 21-2 verwendet wird und die Datei in 4-KByte-Stücken liest.

Beispiel 21-2: pc_split_paragraphs_largefile()

```
function pc_split_paragraphs_largefile($file,$rs="\r?\n") {
    global $php_errormsg;

    $unmatched_text = '';
    $paragraphs = array();

    $fh = fopen($file,'r') or die($php_errormsg);

    while(! feof($fh)) {
        $s = fread($fh,4096) or die($php_errormsg);
        $text_to_split = $unmatched_text . $s;

        $matches = preg_split("/(.*?$rs)(?:$rs)+/s",$text_to_split,-1,
                              PREG_SPLIT_DELIM_CAPTURE|PREG_SPLIT_NO_EMPTY);

        // Wenn der letzte Teil nicht mit zwei Absatztrennzeichen endet, speichern Sie
        // ihn, um ihn dem nächsten Teil, der gelesen wird, voranzustellen.
```

Beispiel 21-2: pc_split_paragraphs_largefile() (Fortsetzung)

```
        $last_match = $matches[count($matches)-1];
        if (! preg_match("/$rs$rs\$/",$last_match)) {
            $unmatched_text = $last_match;
            array_pop($matches);
        } else {
            $unmatched_text = '';
        }

        $paragraphs = array_merge($paragraphs,$matches);
    }

    // Nach dem Einlesen aller Abschnitte kann es einen letzten Teil geben,
    // der nicht mit dem Absatztrennzeichen endet. Er zählt als Absatz.
    if ($unmatched_text) {
        $paragraphs[] = $unmatched_text;
    }
    return $paragraphs;
}
```

Diese Funktion verwendet den gleichen regulären Ausdruck wie `pc_split_paragraphs()`, um die Datei in Absätze zu unterteilen. Wenn sie ein Absatzende in einem von der Datei eingelesenen Teilstück findet, speichert sie den restlichen Text des Teilstücks in `$unmatched_text` und hängt ihn vor das nächste gelesene Teilstück. Das schließt den Text, auf den der reguläre Ausdruck nicht gepasst hat, als Anfang des nächsten Absatzes in der Datei ein.

Siehe auch

Die Dokumentationen zu `fgets()` unter *http://www.php.net/fgets*, `feof()` unter *http://www.php.net/feof* und `preg_split()` unter *http://www.php.net/preg-split*.

21.7 Jedes Wort einer Datei verarbeiten

Problem

Sie wollen mit jedem Wort einer Datei etwas machen.

Lösung

Lesen Sie mit `fgets()` jede Zeile ein, teilen Sie die Zeile in Wörter auf und verarbeiten Sie jedes Wort:

```
$fh = fopen('great-american-novel.txt','r') or die($php_errormsg);
while (! feof($fh)) {
    if ($s = fgets($fh,1048576)) {
        $words = preg_split('/\s+/',$s,-1,PREG_SPLIT_NO_EMPTY);
```

```
        // Wörter verarbeiten.
    }
}
fclose($fh) or die($php_errormsg);
```

Diskussion

Hier sehen Sie, wie Sie die durchschnittliche Wortlänge einer Datei berechnen:

```
$word_count = $word_length = 0;

if ($fh = fopen('great-american-novel.txt','r')) {
  while (! feof($fh)) {
    if ($s = fgets($fh,1048576)) {
      $words = preg_split('/\s+/',$s,-1,PREG_SPLIT_NO_EMPTY);
      foreach ($words as $word) {
        $word_count++;
        $word_length += strlen($word);
      }
    }
  }
}

print sprintf("Die durchschnittliche Länge der %d Wörter ist
                    %.02f Zeichen pro Wort.",
              $word_count,
              $word_length/$word_count);
```

Die Verarbeitung der einzelnen Wörter hängt davon ab, wie Sie den Begriff »Wort« definieren. Der Code dieses Rezepts verwendet das Metazeichen `\s` der Perl-kompatiblen Regular Expression Engine für Leerzeichen aller Art, das Leerzeichen, Tabulatoren, Zeilenvorschübe, Wagenrückläufe und Seitenvorschübe umfasst. Die Perl-kompatible Maschine hat auch ein Wortgrenzen-Metazeichen (`\b`), das auf jeden Übergang zwischen einem Wortzeichen (alphanumerisch) und einem anderen Zeichen (alles außer alphanumerisch) passt. Die Verwendung von `\b` anstelle von `\s` bei der Abgrenzung von Wörtern wird vor allem bei Wörtern mit integrierter Zeichensetzung sichtbar. Der Ausdruck `6 o'clock` zählt als zwei Wörter, wenn er mit einem Leerzeichen getrennt wird (`6` und `o'clock`). Wenn er mit Wortgrenzen-Metazeichen getrennt wird, zählt er als vier Wörter (`6`, `o`, `'` und `clock`).

Siehe auch

Rezept 16.2 beschreibt reguläre Ausdrücke zum Finden von Wörtern; die Dokumentationen zu `fgets()` unter *http://www.php.net/fgets*, zu `preg_split()` unter *http://www.php.net/preg-split* und zur Erweiterung für Perl-kompatible reguläre Ausdrücke unter *http://www.php.net/pcre*.

21.8 Eine bestimmte Zeile einer Datei einlesen

Problem

Sie wollen eine bestimmte Zeile einer Datei einlesen; zum Beispiel wollen Sie den neusten Gästebucheintrag einlesen, der am Ende einer Gästebuchdatei eingetragen wurde.

Lösung

Wenn die Datei in den Speicher passt, lesen Sie sie in ein Array ein und wählen dann das passende Array-Element aus:

```
$lines = file('vacation-hotspots.txt');
print $lines[2];
```

Diskussion

Da die Array-Indizes bei 0 anfangen, bezieht sich `$lines[2]` auf die dritte Zeile der Datei.

Wenn die Datei zu groß ist, um sie in ein Array einzulesen, lesen Sie sie Zeile für Zeile ein und verfolgen, bei welcher Zeile Sie gerade sind:

```
$line_counter = 0;
$gewuenschte_zeile = 29;

$fh = fopen('vacation-hotspots.txt','r') or die($php_errormsg);
while ((! feof($fh)) && ($line_counter <= $gewuenschte_zeile)) {
    if ($s = fgets($fh,1048576)) {
        $line_counter++;
    }
}
fclose($fh) or die($php_errormsg);

print $s;
```

Mit dem Setzen von `$gewuenschte_zeile = 29` wird die 30. Zeile der Datei ausgegeben und ist damit konsistent mit dem Code in der Lösung. Um die 29. Zeile der Datei zu drucken, ändern Sie Zeile mit der `while`-Schleife zu:

```
while ((! feof($fh)) && ($line_counter < $gewuenschte_zeile)) {
```

Siehe auch

Die Dokumentationen zu `fgets()` unter *http://www.php.net/fgets* und `feof()` unter *http://www.php.net/feof*.

21.9 Eine Datei zeilen- oder absatzweise in rückwärtiger Reihenfolge bearbeiten

Problem

Sie wollen mit jeder Zeile einer Datei etwas machen und dabei am Dateiende beginnen. Zum Beispiel ist es einfach, neue Gästebucheinträge ans Dateiende anzuhängen, indem Sie die Datei im Anhängemodus öffnen. Sie wollen die Einträge aber so anzeigen, dass der neuste am Anfang steht, sodass Sie das Ende der Datei zuerst verarbeiten müssen.

Lösung

Wenn die Datei in den Speicher passt, verwenden Sie `file()`, um jede Zeile der Datei in ein Array einzulesen, und kehren dann das Array um:

```
$lines = file('guestbook.txt');
$lines = array_reverse($lines);
```

Diskussion

Sie können ein Array von Zeilen, das nicht umgekehrt ist, in umgekehrter Richtung durchlaufen. Hier sehen Sie, wie Sie die letzten zehn Zeilen einer Datei ausdrucken, wobei die letzte Zeile zuerst kommt:

```
$lines = file('guestbook.txt');
for ($i = 0, $j = count($lines); $i <= 10; $i++) {
    print $lines[$j - $i];
}
```

Siehe auch

Die Dokumentationen zu `file()` unter *http://www.php.net/file* und `array_reverse()` unter *http://www.php.net/array-reverse*.

21.10 Eine Zeile per Zufall aus einer Datei auswählen

Problem

Sie wollen per Zufall eine Zeile aus einer Datei auswählen. Beispiel: Sie wollen eine Auswahl aus einer Datei mit Sprichwörtern anzeigen.

Lösung

Verwenden Sie die in Beispiel 21-3 gezeigte Funktion `pc_randomint()`, die die Auswahlwahrscheinlichkeiten gleichmäßig über alle Zeilen einer Datei verteilt.

Beispiel 21-3: pc_randomint()

```
function pc_randomint($max = 1) {
  $m = 1000000;
  return ((mt_rand(1,$m * $max)-1)/$m);
}
```

Hier ist ein Beispiel, das die Funktion `pc_randomint()` verwendet:

```
$line_number = 0;

$fh = fopen('sprichwoerter.txt','r') or die($php_errormsg);
while (! feof($fh)) {
    if ($s = fgets($fh,1048576)) {
        $line_number++;
        if (pc_randomint($line_number) < 1) {
            $line = $s;
        }
    }
}
fclose($fh) or die($php_errormsg);
```

Diskussion

Die Funktion `pc_randomint()` berechnet eine zufällige Dezimalzahl zwischen einschließlich 0 und ausschließlich `$max`. Bei jeder gelesenen Zeile wird ein Zeilenzähler inkrementiert, und `pc_randomint()` erzeugt eine Zufallszahl zwischen 0 und `$line_number`. Wenn die Zahl kleiner als 1 ist, wird die aktuelle Zeile zur momentanen Zufallszeile. Wenn alle Zeilen gelesen sind, verbleibt die letzte ausgewählte Zufallszeile in `$line`.

Dieser Algorithmus stellt auf geschickte Weise sicher, dass jede Zeile in einer *n-zeiligen* Datei mit einer Wahrscheinlichkeit von `1/`*n* ausgewählt wird, ohne dass alle *n* Zeilen im Speicher abgelegt werden müssen.

Siehe auch

Die Dokumentation zu `mt_rand()` unter *http://www.php.net/mt-rand*.

21.11 Alle Zeilen einer Datei in eine Zufallsreihenfolge bringen

Problem

Sie wollen per Zufall alle Zeilen einer Datei in eine neue Reihenfolge bringen. Zum Beispiel möchten Sie in einer Datei mit lustigen Zitaten eines per Zufall auswählen.

Lösung

Lesen Sie alle Zeilen der Datei mit `file()` in ein Array ein und mischen Sie dann die Elemente des Arrays:

```
$lines = file('quotes-of-the-day.txt');
$lines = pc_array_shuffle($lines);
```

Diskussion

Die Funktion `pc_array_shuffle()` aus Rezept 4.20 ist zu einem höheren Grad zufällig als die eingebaute PHP-Funktion `shuffle()`, da sie nach Fisher-Yates mischt, was die Elemente gleichmäßig im Array verteilt.

Siehe auch

Rezept 4.20 zu `pc_array_shuffle()`; die Dokumentation zu `shuffle()` unter *http://www.php.net/shuffle*.

21.12 Textfelder variabler Länge verarbeiten

Problem

Sie wollen unterteilte Textfelder aus einer Datei auslesen. Vielleicht haben Sie ein Datenbankprogramm, das die Datensätze in jeweils einer Zeile ausdruckt und Tabulator-Zeichen zwischen den Feldern eines Datensatzes einfügt. Sie wollen diese Informationen in ein Array kopieren.

Lösung

Lesen Sie jede Zeile ein und teilen Sie sie dann an den Begrenzungszeichen in die Felder auf:

```
$delim = '|';

$fh = fopen('books.txt','r') or die("Öffnen nicht möglich: $php_errormsg");
while (! feof($fh)) {
    $s = rtrim(fgets($fh,1024));
    $fields = explode($delim,$s);
    // ... etwas mit den Daten machen ...
}
fclose($fh) or die("Schließen nicht möglich: $php_errormsg");
```

Diskussion

Um die folgende Information in *books.txt* aufzuschlüsseln,

```
Elmer Gantry|Sinclair Lewis|1927
The Scarlatti Inheritance|Robert Ludlum|1971
The Parsifal Mosaic|Robert Ludlum|1982
Sophie's Choice|William Styron|1979
```

verarbeiten Sie jeden Datensatz wie folgt:

```
$fh = fopen('books.txt','r') or die("Öffnen nicht möglich: $php_errormsg");
while (! feof($fh)) {
    $s = rtrim(fgets($fh,1024));
    list($title,$author,$publication_year) = explode('|',$s);
    // ... etwas mit den Daten machen ...
}
fclose($fh) or die("Schließen nicht möglich: $php_errormsg");
```

Das Argument für die Zeilenlänge für `fgets()` muss mindestens so lang wie der längste Datensatz sein, ansonsten wird der Datensatz abgeschnitten.

Die Funktion `rtrim()` muss aufgerufen werden, da `fgets()` den der gelesenen Zeile anhängenden Whitespace beinhaltet. Ohne `rtrim()` würde jedes `$publication_year` in einem Zeilenvorschub enden.

Siehe auch

Rezept 1.11 behandelt Methoden zum Unterteilen von Strings; die Rezepte 1.9 und 1.10 decken das Parsen von kommaseparierten Daten bzw. Daten fester Breite ab; die Dokumentationen zu `explode()` unter *http://www.php.net/explode* und `rtrim()` unter *http://www.php.net/rtrim*.

21.13 Konfigurationsdateien einlesen

Problem

Sie wollen Konfigurationsdateien verwenden, um Einstellungen in Ihren Programmen zu initialisieren.

Lösung

Verwenden Sie `parse_ini_file()`:

```
$config = parse_ini_file('/etc/myapp.ini');
```

Diskussion

Die Funktion `parse_ini_file()` liest Konfigurationsdateien ein, die wie die Hauptkonfigurationsdatei von PHP, *php.ini*, aufgebaut sind. Allerdings werden die Einstellungen in der Konfigurationsdatei nicht auf die Konfiguration von PHP angewendet – stattdessen gibt `parse_ini_file()` die Werte aus der Datei in einem Array zurück.

Wenn parse_ini_file() zum Beispiel eine Datei mit dem folgenden Inhalt übergeben wird:

```
; physische Eigenschaften
eyes=brown
hair=brown
glasses=yes

; andere Eigenschaften
name=Susannah
likes=monkeys,ice cream,reading
```

gibt sie das folgende Array zurück:

```
Array
(
    [eyes] => brown
    [hair] => brown
    [glasses] => 1
    [name] => Susannah
    [likes] => monkeys,ice cream,reading
)
```

Leerzeilen und Zeilen, die mit ; beginnen, werden in der Konfigurationsdatei ignoriert. Zeilen mit name=wert-Paaren werden in das Array aufgenommen, mit dem Namen als Schlüssel und dem Wert, passenderweise, als Wert. Wörter mit Werten wie on und yes werden als 1 zurückgegeben, und Wörter wie off and no werden als leerer String zurückgegeben.

Um Abschnitte der Konfigurationsdatei zu parsen, übergeben Sie 1 als zweites Argument an parse_ini_file(). Abschnitte werden durch Wörter in eckigen Klammern in der Datei hervorgehoben:

```
[physical]
eyes=brown
hair=brown
glasses=yes

[other]
name=Susannah
likes=monkeys,ice cream,reading
```

Wenn diese Datei in */etc/myapp.ini* ist, speichert:

```
$conf = parse_ini_file('/etc/myapp.ini',1);
```

dieses Array in $conf:

```
Array
(
    [physical] => Array
        (
            [eyes] => brown
            [hair] => brown
            [glasses] => 1
        )
```

```
    [other] => Array
        (
            [name] => Susannah
            [likes] => monkeys,ice cream,reading
        )

)
```

Ihre Konfigurationsdatei kann auch eine gültige PHP-Datei sein, die Sie mit `require` anstatt mit `parse_ini_file()` laden. Wenn die *config.php*-Datei

```
<?php

// physische Eigenschaften
$eyes = 'brown';
$hair = 'brown';
$glasses = 'yes';

// andere Eigenschaften
$name = 'Susannah';
$likes = array('monkeys','ice cream','reading');
?>
```

enthält, können Sie die Variablen `$eyes`, `$hair`, `$glasses`, `$name` und `$likes` mit

```
require 'config.php';
```

festlegen.

Die Konfigurationsdatei, die durch `require` geladen wird, muss gültiges PHP sein, einschließlich des `<?php`-Start- und des `?>`-End-Tags. Die Variablennamen in *config.php* sind ausdrücklich nicht innerhalb eines Arrays festgelegt, wie das bei `parse_ini_file()` der Fall ist. Bei einfachen Konfigurationsdateien ist diese Technik vielleicht nicht den zusätzlichen Rechtschreibaufwand wert, aber es ist nützlich, wenn Logik in der Konfigurationsdatei integriert wird:

```
<?php

$time_of_day = (date('a') == 'am') ? 'early' : 'late';

?>
```

Die Möglichkeit, Logik in Konfigurationsdateien zu integrieren, ist es oft wert, die Dateien als PHP-Code zu speichern. Es ist aber ebenso nützlich, alle Variablen der Konfigurationsdatei innerhalb eines Arrays festzulegen. Dadurch können Sie Variablen mit gleichem Namen in unterschiedliche Gruppierungen eingliedern (Array in Array), ohne dass diese sich stören.

Siehe auch

Die Dokumentation zu `parse_ini_file()` unter *http://www.php.net/parse-ini-file*.

21.14 Von einer bestimmten Stelle einer Datei lesen oder an eine bestimmte Stelle einer Datei schreiben

Problem

Sie wollen von einer bestimmten (oder *an* eine bestimmte) Stelle in einer Datei lesen (oder schreiben). Zum Beispiel wollen Sie den dritten Dateisatz in einer Datei mit 80-Byte-Dateisätzen ersetzen und müssen daher ab dem 161. Byte schreiben.

Lösung

Verwenden Sie `fseek()`, um zu einer bestimmten Anzahl von Bytes nach dem Dateibeginn, vor dem Dateiende oder von der aktuellen Position in der Datei entfernt zu gelangen:

```
fseek($fh,26);           // 26 Bytes nach dem Dateianfang
fseek($fh,26,SEEK_SET);  // 26 Bytes nach dem Dateianfang
fseek($fh,-39,SEEK_END); // 39 Bytes vor dem Dateiende
fseek($fh,10,SEEK_CUR);  // 10 Bytes hinter der aktuellen Position
fseek($fh,0);            // Dateianfang
```

Mit der Funktion `rewind()` gelangen Sie zum Dateianfang:

```
rewind($fh);             // wie fseek($fh,0)
```

Diskussion

Die Funktion `fseek()` gibt 0 zurück, wenn sie zu der angegebenen Position gelangen kann, andernfalls gibt sie –1 zurück. Hinter dem Dateiende zu suchen ist für `fseek()` kein Fehler. Im Gegensatz dazu gibt `rewind()` 0 zurück, wenn sie einen Fehler entdeckt.

`fseek()` kann mit lokalen Dateien verwendet werden, aber nicht mit HTTP- oder FTP-Dateien, die mit `fopen()` geöffnet wurden. Wenn Sie ein Datei-Handle einer entfernten Datei an `fseek()` übergeben, wird der Fehler `E_NOTICE` ausgegeben.

Um die momentane Dateiposition zu erhalten, verwenden Sie `ftell()`:

```
if (0 === ftell($fh)) {
  print "Am Dateianfang.";
}
```

Da `ftell()` bei einem Fehler `false` zurückgibt, müssen Sie den Operator `===` verwenden, um sicherzugehen, dass sein zurückgegebener Wert wirklich die ganze Zahl 0 ist.

Siehe auch

Die Dokumentationen zu `fseek()` unter *http://www.php.net/fseek*, `ftell()` unter *http://www.php.net/ftell* und `rewind()` unter *http://www.php.net/rewind*.

21.15 Die letzte Zeile einer Datei entfernen

Problem

Sie wollen die letzte Zeile einer Datei entfernen. Beispielsweise hat jemand einen Kommentar an das Ende Ihres Gästebuchs geschrieben. Sie mögen den Kommentar nicht und wollen ihn loswerden.

Lösung

Wenn die Datei klein ist, können Sie sie mit `file()` in ein Array einlesen und dann das letzte Element des Arrays entfernen:

```
$lines = file('employees.txt');
array_pop($lines);
$file = join('',$lines);
```

Diskussion

Wenn die Datei groß ist, verbraucht das Einlesen in ein Array zu viel Speicherkapazität. Stattdessen können Sie den folgenden Code verwenden. Er sucht nach dem Ende der Datei, arbeitet von hinten nach vorn und hält an, wenn er einen Zeilenvorschub findet:

```
$fh = fopen('employees.txt','r') or die("Öffnen nicht möglich: $php_errormsg");
$linebreak = $beginning_of_file = 0;

$gap = 80;
$filesize = filesize('employees.txt');
fseek($fh,0,SEEK_END);

while (! ($linebreak || $beginning_of_file)) {
    // Dateiposition speichern.
    $pos = ftell($fh);

    /* $gap Zeichen zurückbewegen, wenn weniger als $gap Zeichen
     * davor in der Datei sind, mit rewind() an den Dateianfang zurückgehen. */
    if ($pos < $gap) {
        rewind($fh);
    } else {
        fseek($fh,-$gap,SEEK_CUR);
    }

    // Die eben übersprungenen $gap Zeichen einlesen.
    $s = fread($fh,$gap) or die($php_errormsg);

    /* Wenn bis zum Dateiende gelesen wurde, das letzte Zeichen entfernen,
     * da es ignoriert werden sollte, falls es sich um einen Zeilenvorschub
     * handelt. */
    if ($pos + $gap >= $filesize) {
```

```
            $s = substr_replace($s,'',-1);
        }

        // Zur Position zurückspringen, an der Sie vor dem Einlesen von $gap
        // Zeichen in $s waren.
        if ($pos < $gap) {
            rewind($fh);
        } else {
            fseek($fh,-$gap,SEEK_CUR);
        }

        // Enthält $s einen Zeilenvorschub?
        if (is_integer($lb = strrpos($s,"\n"))) {
            $linebreak = 1;
            // Die letzte Zeile der Datei fängt sofort nach dem Zeilenumbruch an.
            $line_end = ftell($fh) + $lb + 1;
        }

        // Aus der Schleife ausbrechen, wenn Sie am Dateianfang sind.
        if (ftell($fh) == 0) { $beginning_of_file = 1; }

    }
    if ($linebreak) {
        rewind($fh);
        $file_without_last_line = fread($fh,$line_end) or die($php_errormsg);
    }
    fclose($fh) or die("Schließen nicht möglich: $php_errormsg");
```

Dieser Code beginnt am Dateiende, bewegt sich um je `$gap` Zeichen von hinten nach vorn und sucht dabei nach einem Zeilenvorschub. Wenn er einen findet, weiß er, dass die letzte Zeile der Datei direkt nach diesem Zeilenvorschub beginnt. Diese Position wird in `$line_end` gespeichert. Wenn `$linebreak` gesetzt ist, wird nach der `while`-Schleife der Dateiinhalt von Anfang bis `$line_end` in `$file_without_last_line` eingelesen.

Das letzte Zeichen der Datei wird ignoriert, da es, sofern es ein Zeilenvorschub ist, nicht den Anfang der letzten Zeile in der Datei angibt. Betrachten wir die 10-Zeichen-Datei mit dem Inhalt `spargel\n`. Sie hat lediglich eine Zeile mit dem Wort `spargel` und einem Zeilenvorschubzeichen als Inhalt. Ohne ihre letzte Zeile ist diese Datei leer, was durch den Code oben richtig angegeben wird. Wenn der Code beim letzten Zeichen mit dem Scannen beginnen würde, würde er den Zeilenvorschub sehen, den Scanning-Loop beenden und fälschlicherweise `spargel` ohne den Zeilenvorschub ausdrucken.

Siehe auch

Rezept 21.14 bespricht `fseek()` und `rewind()` genauer; die Dokumentationen zu `array_pop()` unter *http://www.php.net/array-pop*, `fseek()` unter *http://www.php.net/fseek* und `rewind()` unter *http://www.php.net/rewind*.

21.16 Eine Datei an ihrem Platz ohne eine temporäre Datei ändern

Problem

Sie wollen eine Datei ändern, ohne die Änderungen in einer temporären Datei zwischenzuspeichern.

Lösung

Lesen Sie die Datei in den Speicher ein, führen Sie die Änderungen durch und schreiben Sie die Datei wieder. Öffnen Sie die Datei mit dem Modus `r+` (`rb+` unter Windows, falls notwendig) und korrigieren Sie nach dem Schreiben der Änderungen die Länge mit `ftruncate()`:

```
// Datei zum Lesen und Schreiben öffnen.
$fh = fopen('pickles.txt','r+')         or die($php_errormsg);

// Gesamte Datei in $s speichern.
$s = fread($fh,filesize('pickles.txt')) or die($php_errormsg);

// ... $s modifizieren ...

// An den Dateianfang zurückspringen und das neue $s schreiben.
rewind($fh);
if (-1 == fwrite($fh,$s))                { die($php_errormsg); }

// Dateilänge auf tatsächlich Geschriebenes anpassen.
ftruncate($fh,ftell($fh))               or die($php_errormsg);

// Datei schließen.
fclose($fh)                             or die($php_errormsg);
```

Diskussion

Der folgende Code ändert Text, der mit Sternchen oder Schrägstrichen hervorgehoben ist, zu Text mit HTML-`<b>`- oder -`<i>`-Tags:

```
$fh = fopen('message.txt','r+')         or die($php_errormsg);

// Ganze Datei in $s einlesen.
$s = fread($fh,filesize('message.txt')) or die($php_errormsg);

// *Wort* zu <b>Wort</b> konvertieren.
$s = preg_replace('@\*(.*?)\*@i','<b>$1</b>',$s);
// /Wort/ zu <i>Wort</i> konvertieren.
$s = preg_replace('@/(.*?)/@i','<i>$1</i>',$s);

rewind($fh);
```

```
if (-1 == fwrite($fh,$s))               { die($php_errormsg); }
ftruncate($fh,ftell($fh))              or die($php_errormsg);
fclose($fh)                            or die($php_errormsg);
```

Da die Datei durch das Hinzufügen von HTML-Tags größer wird, muss die gesamte Datei in den Speicher eingelesen und dann verarbeitet werden. Alternativ können hierfür auch die Funktionen `file_get_contents()` und `file_put_contents()` eingesetzt werden.

Wenn durch die Änderungen an der Datei jede Zeile kürzer wird (oder gleich lang bleibt), kann die Datei Zeile für Zeile verarbeitet werden, wodurch Speicherplatz gespart wird. Dieses Beispiel konvertiert Text mit `<b>`- und `<i>`-Markup in einen Text, der durch Sternchen und Schrägstriche hervorgehoben wird:

```
$fh = fopen('message.txt','r+')        or die($php_errormsg);

// Herausfinden, wie viele Bytes zu lesen sind.
$bytes_to_read = filesize('message.txt');

// Variablen zum Speichern von Dateipositionen initialisieren.
$next_read = $last_write = 0;

// Weitermachen, solange noch Bytes zu lesen sind.
while ($next_read < $bytes_to_read) {

    /* Zur nächsten Leseposition bewegen, eine Zeile einlesen und die
     * nächste Leseposition speichern. */
    fseek($fh,$next_read);
    $s = fgets($fh,1048576)            or die($php_errormsg);
    $next_read = ftell($fh);

    // <b>Wort</b> zu *Wort* konvertieren.
    $s = preg_replace('@<b[^>]*>(.*?)</b>@i','*$1*',$s);
    // <i>Wort</i> zu /Wort/ konvertieren
    $s = preg_replace('@<i[^>]*>(.*?)</i>@i','/$1/',$s);

    /* Zur Position bewegen, an der der letzte Schreibvorgang beendet wurde,
     * die konvertierte Zeile schreiben und die Position für den nächsten
     * Schreibvorgang schreiben. */
    fseek($fh,$last_write);
    if (-1 == fwrite($fh,$s))           { die($php_errormsg); }
    $last_write = ftell($fh);
}

// Dateilänge auf die bereits geschriebene Länge stutzen.
ftruncate($fh,$last_write)             or die($php_errormsg);

// Datei schließen.
fclose($fh)                            or die($php_errormsg);
```

Siehe auch

Die Rezepte 13.11 und 13.12 für zusätzliche Informationen zum Konvertieren zwischen ASCII und HTML; Rezept 21.14 behandelt `fseek()` und `rewind()` genauer; die Doku-

mentationen zu fseek() unter *http://www.php.net/fseek*, zu rewind() unter *http://www.php.net/rewind* und zu ftruncate() unter *http://www.php.net/ftruncate*; die Dokumentationen zu file_get_contents() unter *http://www.php.net/file_get_contents* und zu file_put_contents() unter *http://www.php.net/file_put_contents*.

21.17 Gepufferte Ausgabedaten in eine Datei schreiben

Problem

Sie wollen alle gepufferten Daten in ein Datei-Handle schreiben.

Lösung

Verwenden Sie fflush():

```
fwrite($fh,'There are twelve pumpkins in my house.');
fflush($fh);
```

Das stellt sicher, dass »There are twelve pumpkins in my house.« nach $fh geschrieben wird.

Diskussion

Um effizienter zu arbeiten, schreiben System-I/O-Bibliotheken Daten üblicherweise nicht gleich in eine Datei, wenn Sie es ihnen befohlen haben. Stattdessen sammeln sie mehrere zu schreibende Datenstücke in einem Puffer und speichern sie dann alle gleichzeitig auf der Festplatte. Mit fflush() wird der gesamte wartende Pufferinhalt direkt auf die Platte geschrieben.

Das »Ausspülen« (Flush) des Puffers ist besonders hilfreich, wenn Sie ein Zugangs- oder Aktivitäts-Logbuch erstellen wollen. Der Aufruf von fflush() nach jeder Nachricht an die Logbuch-Datei stellt sicher, dass die Nachricht so früh wie möglich von jeder Person bzw. jedem Programm gesehen wird, das die Logbuch-Datei überwacht.

Siehe auch

Dokumentation zu fflush() unter *http://www.php.net/fflush*.

21.18 An viele Datei-Handles gleichzeitig schreiben

Problem

Sie wollen eine Ausgabe an mehr als ein Datei-Handle schreiben; zum Beispiel wollen Sie Meldungen auf den Bildschirm und an eine Datei ausgeben.

Lösung

Packen Sie Ihre Ausgabe in eine Schleife, die Ihre Datei-Handles durchläuft, wie in Beispiel 21-4 gezeigt.

Beispiel 21-4: pc_multi_fwrite()

```
function pc_multi_fwrite($fhs,$s,$length=NULL) {
  if (is_array($fhs)) {
    if (is_null($length)) {
      foreach($fhs as $fh) {
        fwrite($fh,$s);
      }
    } else {
      foreach($fhs as $fh) {
        fwrite($fh,$s,$length);
      }
    }
  }
}
```

Hier kommt ein Beispiel:

```
$fhs['file'] = fopen('log.txt','w') or die($php_errormsg);
$fhs['screen'] = fopen('php://stdout','w') or die($php_errormsg);

pc_multi_fwrite($fhs,'Der Spaceshuttle ist gelandet.');
```

Diskussion

Wenn Sie keinen Längenparameter an `fwrite()` übergeben wollen (oder Sie das immer tun wollen), können Sie die Abfrage aus Ihrem `pc_multi_fwrite()` entfernen. Diese Version akzeptiert kein `$length`-Argument:

```
function pc_multi_fwrite($fhs,$s) {
  if (is_array($fhs)) {
    foreach($fhs as $fh) {
      fwrite($fh,$s);
    }
  }
}
```

Siehe auch

Die Dokumentation zu `fwrite()` unter *http://www.php.net/fwrite*.

21.19 Einem Programm Eingabedaten durchgeben

Problem

Sie wollen Eingabedaten an ein externes Programm durchreichen, das innerhalb Ihres PHP-Skripts läuft. Zum Beispiel verwenden Sie möglicherweise eine Datenbank, auf die Sie nur durch ein externes Programm zugreifen können. Diesem Datenbankprogramm wollen Sie Texte zum Ablegen übergeben.

Lösung

Öffnen Sie eine Pipeline zum Programm mit `popen()`, schreiben Sie in die Pipeline mit `fputs()` oder `fwrite()` und schließen Sie die Pipeline mit `pclose()`:

```
$ph = popen('programm param1 param2','w')          or die($php_errormsg);
if (-1 == fputs($ph,"erste Zeile der Eingabedaten\n"))  { die($php_errormsg); }
if (-1 == fputs($ph,"zweite Zeile der Eingabedaten\n")) { die($php_errormsg); }
pclose($ph)                                        or die($php_errormsg);
```

Diskussion

Dieses Beispiel verwendet `popen()`, um den Befehl *nsupdate* aufzurufen, der Dynamic-DNS-Updateanfragen an einen Nameserver übergibt:

```
$ph = popen('/usr/bin/nsupdate -k keyfile')              or die($php_errormsg);
if (-1 == fputs($ph,"update delete test.beispiel.com A\n")) { die($php_errormsg); }
if (-1 == fputs($ph,"update add test.beispiel.com 5 A 192.168.1.1\n"))
                                                           { die($php_errormsg); }
pclose($ph)                                              or die($php_errormsg);
```

Über `popen()` werden zwei Befehle an *nsupdate* gesendet. Der erste löscht den A-Datensatz für *test.beispiel.com*, der zweite fügt einen neuen A-Datensatz für *test.beispiel.com* mit der Adresse 192.168.1.1 ein.

Siehe auch

Die Dokumentationen zu `popen()` unter *http://www.php.net/popen* und `pclose()` unter *http://www.php.net/pclose*; Dynamic DNS wird in RFC 2136 unter *http://www.faqs.org/rfcs/rfc2136.html* beschrieben.

21.20 Die Standardausgabe eines Programms lesen

Problem

Sie wollen die Ausgabe eines Programms einlesen; zum Beispiel wollen Sie die Ausgabe eines System-Tools lesen, wie z.B. *route(8)*, das Netzwerkinformationen zur Verfügung stellt.

Lösung

Um den gesamten Inhalt einer Programmausgabe einzulesen, verwenden Sie den Backtick-Operator (`):

```
$routing_table = `/sbin/route`;
```

Um die Ausgabe stückweise auszulesen, öffnen Sie eine Pipeline mit `popen()`:

```
$ph = popen('/sbin/route','r') or die($php_errormsg);
while (! feof($ph)) {
    $s = fgets($ph,1048576)    or die($php_errormsg);
}
pclose($ph)                    or die($php_errormsg);
```

Diskussion

Der Backtick-Operator (der im Safe Mode nicht zur Verfügung steht) führt ein Programm aus und gibt dessen gesamte Ausgabe als einen einzigen String zurück. Auf einem Linux-System mit 448 MB RAM speichert dieser Befehl

```
$s = `/usr/bin/free`;
```

den folgenden mehrzeiligen String in `$s`:

```
             total       used       free     shared    buffers     cached
Mem:        448620     446384       2236          0      68568     163040
-/+ buffers/cache:     214776     233844
Swap:       136512          0     136512
```

Produziert ein Programm umfangreiche Ausgaben, ist der Speicherbedarf geringer, wenn Sie eine Zeile nach der anderen aus einer Pipeline auslesen. Wenn Sie auf Basis der Pipeline-Ausgabe formatierte Daten direkt an den Browser senden, werden diese dargestellt, sobald sie ankommen. Das folgende Beispiel druckt Informationen über die letzten Logins in einem Unix-System aus und formatiert sie als HTML-Tabelle. Es verwendet den Befehl */usr/bin/last*:

```
// Tabellenkopf ausdrucken.
print<<<_HTML_
<table>
<tr>
 <td>user</td><td>Login-Port</td><td>Login von</td><td>Login-Zeit</td>
 <td>Login-Dauer</td>
</tr>
_HTML_;

// Pipeline zu /usr/bin/last öffnen.
$ph = popen('/usr/bin/last','r') or die($php_errormsg);
while (! feof($ph)) {
    $line = fgets($ph,80) or die($php_errormsg);

    // Leerzeilen und Infozeile am Ende ignorieren.
    if (trim($line) && (! preg_match('/^wtmp begins/',$line))) {
        $user = trim(substr($line,0,8));
```

```
            $port = trim(substr($line,9,12));
            $host = trim(substr($line,22,16));
            $date = trim(substr($line,38,25));
            $elapsed = trim(substr($line,63,10),' ()');

            if ('logged in' == $elapsed) {
                $elapsed = 'noch eingeloggt';
                $date = substr_replace($date,'',-5);
            }

            print "<tr><td>$user</td><td>$port</td><td>$host</td>";
            print "<td>$date</td><td>$elapsed</td></tr>\n";
        }
    }
    pclose($ph) or die($php_errormsg);

    print '</table>';
```

Siehe auch

Die Dokumentationen zu `popen()` unter *http://www.php.net/popen*, `pclose()` unter *http://www.php.net/pclose* und zum Backtick-Operator unter *http://www.php.net/language.operators.execution*. Der Safe Mode ist unter *http://www.php.net/features.safe-mode* dokumentiert.

21.21 Den Standardfehlerkanal eines Programms einlesen

Problem

Sie wollen die Fehlerausgabe eines Programms auslesen; zum Beispiel wollen Sie die Systemaufrufe, die von *strace(1)* angezeigt werden, einsehen.

Lösung

Leiten Sie den Standardfehlerkanal zur Standardausgabe um, indem Sie `2>&1` an die Kommandozeile, die an `popen()` durchgegeben wird, anfügen. Lesen Sie die Standardausgabe aus, indem Sie die Pipeline im `r`-Modus öffnen:

```
$ph = popen('strace ls 2>&1','r') or die($php_errormsg);
while (!feof($ph)) {
    $s = fgets($ph,1048576)       or die($php_errormsg);
}
pclose($ph)                       or die($php_errormsg);
```

Diskussion

Sowohl in der *sh* unter Unix als auch in der Windows-Shell *cmd.exe* ist der Standardfehlerkanal der Datei-Deskriptor 2 und die Standardausgabe der Datei-Deskriptor 1. Indem

Sie `2>&1` an eine Kommandozeile anhängen, weisen Sie die Shell an, alles, was normalerweise an den Datei-Deskriptor 2 (Standardfehler) gesendet wird, an den Datei-Deskriptor 1 (Standardausgabe) weiterzuleiten. `fgets()` liest dann sowohl den Standardfehler als auch die Standardausgabe ein.

Mit dieser Methode wird der Standardfehlerkanal eingelesen, sie stellt aber keine Möglichkeit zur Verfügung, den Standardfehlerkanal von der Standardausgabe zu unterscheiden. Um lediglich den Standardfehler einzulesen, müssen Sie die Ausgabe der Standardausgabe durch die Pipeline verhindern. Das wird durch die Umleitung der Standardausgabe an */dev/null* unter Unix und an NUL unter Windows erreicht:

```
// Unix: nur Standardfehlerkanal lesen.
$ph = popen('strace ls 2>&1 1>/dev/null','r') or die($php_errormsg);

// Windows: nur Standardfehlerkanal lesen.
$ph = popen('ipxroute.exe 2>&1 1>NUL','r') or die($php_errormsg);
```

Siehe auch

Die Dokumentation zu `popen()` unter *http://www.php.net/popen*; Einzelheiten über die Shell, die Ihr Unix-System mit `popen()` verwendet, finden Sie unter Ihrer *popen(3)*-Manpage. Informationen über die Umleitung von Shells finden Sie bei Unix-Systemen im Abschnitt »Redirection« der *sh(1)*-Manpage, bei Windows im Eintrag über Redirection im Abschnitt »Befehlsreferenz« Ihrer Systemhilfe.

21.22 Eine Datei sperren

Problem

Sie wollen alleinigen Zugang zu einer Datei haben, um Änderungen durch Dritte zu verhindern, während Sie die Datei einlesen oder ändern. Wenn Sie zum Beispiel Informationen eines Gästebuchs in eine Datei speichern, sollten zwei Benutzer gleichzeitig die Möglichkeit haben, Gästebucheinträge durchzuführen, ohne sich gegenseitig in die Quere zu kommen.

Lösung

Verwenden Sie `flock()`, um eine unverbindliche Sperre anzuzeigen:

```
$fh = fopen('guestbook.txt','a')         or die($php_errormsg);
flock($fh,LOCK_EX)                       or die($php_errormsg);
fwrite($fh,$_REQUEST['guestbook_entry']) or die($php_errormsg);
fflush($fh)                              or die($php_errormsg);
flock($fh,LOCK_UN)                       or die($php_errormsg);
fclose($fh)                              or die($php_errormsg);
```

Diskussion

Die Art der von `flock()` zur Verfügung gestellten Dateisperre heißt *advisory*, da `flock()` das Öffnen einer Datei durch andere Prozesse nicht tatsächlich verhindert – es stellt lediglich eine Methode bereit, mit der Prozesse freiwillig beim Dateizugriff kooperieren können. Alle Programme, die auf mit `flock()` gesperrte Dateien zugreifen wollen, müssen Sperren setzen und wieder freigeben, um die Sperr-Regelung effektiv zu machen.

Mit `flock()` können Sie zwei Arten von Sperren setzen: exklusive Sperren (*exclusive locks*) und gemeinsame Sperren (*shared locks*). Eine *exklusive Sperre*, durch `LOCK_EX` als zweiter Parameter für `flock()` identifiziert, kann zu einer gegebenen Zeit immer nur zu einem Prozess und einer bestimmten Datei gehören. Eine *gemeinsame Sperre*, durch `LOCK_SH` identifiziert, kann für eine bestimmte Datei gleichzeitig zu mehreren Prozessen gehören. Bevor Sie in eine Datei schreiben, sollten Sie eine exklusive Sperre setzen. Vor dem Lesen einer Datei reicht eine gemeinsame Sperre.

Um die Sperre einer Datei zurückzusetzen, rufen Sie `flock()` mit `LOCK_UN` als zweitem Parameter auf. Es ist wichtig, alle gepufferten Daten, die in die Datei geschrieben werden sollen, vor dem Zurücksetzen mit `fflush()` in die Datei zu schreiben. Andere Prozesse sollten die Datei nicht sperren können, bis die Daten geschrieben sind.

Als Voreinstellung blockiert `flock()` die Ausführung so lange, bis sie eine Sperre setzen kann. Um die Funktion anzuweisen, nicht zu blockieren, fügen Sie dem zweiten Parameter `LOCK_NB` hinzu:

```
$fh = fopen('guestbook.txt','a')         or die($php_errormsg);
$tries = 3;
while ($tries > 0) {
    $locked = flock($fh,LOCK_EX | LOCK_NB);
    if (! $locked) {
        sleep(5);
        $tries--;
    } else {
        // Schleife nicht noch einmal durchlaufen.
        $tries = 0;
    }
}
if ($locked) {
    fwrite($fh,$_REQUEST['guestbook_entry']) or die($php_errormsg);
    fflush($fh)                              or die($php_errormsg);
    flock($fh,LOCK_UN)                       or die($php_errormsg);
    fclose($fh)                              or die($php_errormsg);
} else {
    print "Kann Sperre nicht setzen.";
}
```

Wenn nicht blockiert wird, beendet `flock()` sofort – sogar dann, wenn die Funktion keine Sperre setzen konnte. Das vorhergehende Beispiel versucht dreimal, eine Sperre für *guestbook.txt* zu setzen, und hält die Ausführung zwischen den Versuchen um jeweils fünf Sekunden an.

Das Sperren mit `flock()` funktioniert nicht unter allen Bedingungen, z.B. versagt es unter einigen NFS-Implementierungen. Windows 95, 98 und ME unterstützen `flock()` ebenfalls nicht. Um in diesen Fällen Dateisperren zu simulieren, verwenden Sie ein Verzeichnis als Indikator für eine exklusive Sperre. Dabei handelt es sich um ein separates, leeres Verzeichnis, dessen Existenz anzeigt, dass die Datei gesperrt ist. Bevor Sie eine Datei öffnen, erzeugen Sie ein Sperrverzeichnis und löschen es wieder, wenn Sie Ihre Arbeiten an der Datei beendet haben. Ansonsten ist der Dateizugriffscode der gleiche, der hier gezeigt wird:

```
$fh = fopen('guestbook.txt','a')         or die($php_errormsg);

// Schleife durchlaufen, bis Sie das Sperrverzeichnis erfolgreich erzeugen können.
$locked = 0;
while (! $locked) {
    if (@mkdir('guestbook.txt.lock',0777)) {
        $locked = 1;
    } else {
        sleep(1);
    }
}

if (-1 == fwrite($fh,$_REQUEST['guestbook_entry'])) {
    rmdir('guestbook.txt.lock');
    die($php_errormsg);
}
if (! fclose($fh)) {
    rmdir('guestbook.txt.lock');
    die($php_errormsg);
}
rmdir('guestbook.txt.lock')              or die($php_errormsg);
```

Hier wird anstatt einer Datei ein Verzeichnis zum Anzeigen der Sperre verwendet, weil die Funktion `mkdir()` einen Fehler zurückgibt, falls das zu erzeugende Verzeichnis bereits existiert. Das gibt Ihnen eine Möglichkeit, in einer Operation zu überprüfen, ob der Sperranzeiger existiert, und ihn zu erzeugen, falls nicht. Etwaige Fehlerbehandlungen nach Erzeugung des Verzeichnisses müssen allerdings aufräumen, indem sie das Verzeichnis entfernen, bevor sie das Skript abbrechen. Wenn das Verzeichnis bestehen bleibt, können zukünftige Prozesse keine Sperre durch Erzeugen des Verzeichnisses mehr setzen.

Wenn Sie eine Datei als Sperrindikator verwenden wollen, sieht der Code zu ihrer Erzeugung so aus:

```
$locked = 0;
while (! $locked) {
    if (! file_exists('guestbook.txt.lock')) {
        touch('guestbook.txt.lock');
        $locked = 1;
    } else {
        sleep(1);
    }
}
```

Unter hoher Server-Belastung könnte dieses System allerdings zusammenbrechen. Das liegt daran, dass Sie die Existenz der Sperre erst mit `file_exists()` prüfen und sie dann mit `touch()` erzeugen. Nachdem ein Prozess `file_exists()` aufruft, könnte ein anderer aber `touch()` aufrufen, bevor der erste Prozess `touch()` aufrufen kann. Beide Prozesse würden dann denken, dass sie exklusiven Zugriff auf die Datei haben, obwohl das in Wirklichkeit für keinen der beiden Prozesse der Fall wäre. Bei `mkdir()` gibt es zwischen Existenzprüfung und Erzeugung kein Zeitintervall, das ein anderer Prozess nutzen könnte. Das stellt sicher, dass der Prozess, der das Verzeichnis erzeugt, auch exklusiven Zugang hat.

Siehe auch

Die Dokumentation zu `flock()` unter *http://www.php.net/flock*.

21.23 Komprimierte Dateien lesen und schreiben

Problem

Sie wollen komprimierte Dateien lesen oder schreiben.

Lösung

Verwenden Sie die *zlib*-Erweiterung von PHP, um mit *gzip* komprimierte Dateien zu lesen oder zu schreiben. So lesen Sie eine komprimierte Datei:

```
$zh = gzopen('file.gz','r') or die("Öffnen fehlgeschlagen: $php_errormsg");
while ($line = gzgets($zh,1024)) {
    // $line ist die nächste Zeile mit unkomprimierten Daten, bis zu 1024 Bytes.
}
gzclose($zh) or die("Schließen nicht möglich: $php_errormsg");
```

Und so schreiben Sie eine komprimierte Datei:

```
$zh = gzopen('file.gz','w') or die("Öffnen fehlgeschlagen: $php_errormsg");
if (-1 == gzwrite($zh,$s))   { die("Schreiben fehlgeschlagen: $php_errormsg"); }
gzclose($zh)                or die("Schließen nicht möglich: $php_errormsg");
```

Diskussion

Die *zlib*-Erweiterung enthält Versionen vieler Dateizugriffsfunktionen wie `fopen()`, `fread()` und `fwrite()`, die hier `gzopen()`, `gzread()`, `gzwrite()` usw. heißen, transparent Daten beim Schreiben komprimieren und beim Lesen dekomprimieren. Der von *zlib* verwendete Kompressions-Algorithmus ist mit den Tools *gzip* und *gunzip* kompatibel.

So arbeitet zum Beispiel gzgets($zp,1024) wie fgets($fh,1024). Die Funktion liest bis zu 1.023 Bytes und hält vorher an, wenn sie über EOF oder einen Zeilenvorschub stolpert. Im Fall von gzgets() heißt das 1.023 unkomprimierte Bytes.

gzseek() funktioniert jedoch etwas anders als fseek(). Es unterstützt nur die Bewegung zu einer angegebenen Anzahl von Bytes vom Anfang des Datei-Streams (den SEEK_SET-Parameter von fseek()). Das Vorwärtsbewegen von der gegenwärtigen Position aus wird nur bei Dateien unterstützt, die zum Schreiben geöffnet sind (die Datei wird mit einer Folge komprimierter Nullen aufgefüllt). Rückwärtsbewegungen werden bei zum Lesen geöffneten Dateien unterstützt, sind aber sehr langsam.

Die *zlib*-Erweiterung verfügt auch über ein paar Funktionen zur Erzeugung von komprimierten Strings. Die Funktion gzencode() komprimiert einen String und gibt ihm korrekte Header und eine richtige Formatierung, um mit *gunzip* kompatibel zu sein. Hier ist ein einfaches *gzip*-Programm:

```
$in_file = $_SERVER['argv'][1];
$out_file = $_SERVER['argv'][1].'.gz';

$ifh = fopen($in_file,'rb')  or die("Kann $in_file nicht öffnen: $php_errormsg");
$ofh = fopen($out_file,'wb') or die("Kann $out_file nicht öffnen: $php_errormsg");

$encoded = gzencode(fread($ifh,filesize($in_file)))
                             or die("Kann Daten nicht komprimieren: $php_errormsg");

if (-1 == fwrite($ofh,$encoded)) { die("Kann nicht schreiben: $php_errormsg"); }
fclose($ofh)                 or die("Kann $out_file nicht schließen: $php_errormsg");
fclose($ifh)                 or die("Kann $in_file nicht schließen: $php_errormsg");
```

Die Eingeweide dieses Programms bestehen aus den Zeilen:

```
$encoded = gzencode(fread($ifh,filesize($in_file)))
                             or die("Kann Daten nicht komprimieren: $php_errormsg");
if (-1 == fwrite($ofh,$encoded)) { die("Kann nicht schreiben: $php_errormsg"); }
```

Der komprimierte Inhalt von $in_file wird in $encoded gespeichert und dann mit fwrite() in die Datei $out_file geschrieben.

Sie können gzencode() einen zweiten Parameter übergeben, der den Komprimierungsgrad angibt. Dabei bedeutet 0 »keine Komprimierung« und 9 »maximale Komprimierung«. Der voreingestellte Komprimierungsgrad ist 1. Um das einfache *gzip*-Programm auf maximale Komprimierung einzustellen, ändert sich die Codierungszeile zu:

```
$encoded = gzencode(fread($ifh,filesize($in_file)),9)
                             or die("Kann Daten nicht komprimieren: $php_errormsg");
```

Sie können Strings auch ohne die *gzip*-kompatiblen Headers komprimieren und dekomprimieren, indem Sie gzcompress() und gzuncompress() verwenden.

Siehe auch

Rezept 21.24 für ein Programm, das Dateien aus einem ZIP-Archiv extrahiert; die Dokumentation der *zlib*-Erweiterung unter *http://www.php.net/zlib* können Sie unter *http://www.gzip.org/zlib/* herunterladen; der *zlib*-Algorithmus wird in RFCs 1950 (*http://www.faqs.org/rfcs/rfc1950.html*) und 1951 (*http://www.faqs.org/rfcs/rfc1951.html*) im Detail definiert.

21.24 Programm: Unzip

Das Programm *unzip.php*, in Beispiel 21-5 gezeigt, extrahiert Dateien aus einem ZIP-Archiv. Es verwendet die Funktion `pc_mkdir_parents()`, die in Rezept 22.11 definiert ist. Das Programm benötigt außerdem die *zip*-Erweiterung von PHP. Die Dokumentation zur *zip*-Erweiterung von PHP finden Sie unter *http://www.php.net/zip*.

Dieses Programm braucht ein paar Kommandozeilen-Parameter. Der erste ist der Name des zu öffnenden ZIP-Archivs. Als Voreinstellung dekomprimiert das Programm alle Dateien im Archiv. Wenn zusätzliche Kommandozeilen-Parameter angegeben sind, werden nur diejenigen Dateien dekomprimiert, deren Namen mit einem der Parameter übereinstimmen. Dazu muss der volle Pfad der Datei innerhalb des ZIP-Archivs angegeben werden. Wenn sich *schildkroeten.html* im ZIP-Archiv innerhalb des Verzeichnisses *tiere* befindet, muss *unzip.php* zwecks Dekomprimierung der Parameter *tiere/schildkroeten.html* übergeben werden – *schildkroeten.html* allein reicht nicht.

Verzeichnisse werden in ZIP-Archiven als 0-Byte-Dateien gespeichert, deswegen versucht *unzip.php* nicht, sie zu erzeugen. Stattdessen verwendet das Programm `pc_mkdir_parents()`, um ggf. die übergeordneten Verzeichnisse einer Datei zu erzeugen, bevor diese selbst erzeugt wird. Nehmen wir einmal an, dass *unzip.php* die folgenden Einträge im ZIP-Archiv sieht:

```
tiere (0 bytes)
tiere/froesche/breitmaul.html (2123 bytes)
tiere/schildkroeten.html   (1232 bytes)
```

Das Programm ignoriert *tiere*, weil die Datei 0 Bytes lang ist. Es ruft dann `pc_mkdir_parents()` für *tiere/froesche* auf, erzeugt sowohl *tiere* als auch *tiere/froesche* und schreibt *breitmaul.html* in *tiere/froesche*. Da *tiere* bereits existiert, wenn das Programm *tiere/schildkroeten.html* erreicht, schreibt es *schildkroeten.html*, ohne zusätzliche Verzeichnisse zu erzeugen.

Beispiel 21-5: unzip.php

```
// Der erste Parameter ist die Zip-Datei.
$in_file = $_SERVER['argv'][1];

// Alle weiteren Parameter sind spezifische Dateien im Archiv, die entpackt werden sollen.
if ($_SERVER['argc'] > 2) {
```

Beispiel 21-5: unzip.php (Fortsetzung)

```
    $all_files = 0;
    for ($i = 2; $i < $_SERVER['argc']; $i++) {
        $out_files[$_SERVER['argv'][$i]] = true;
    }
} else {
    // Wenn keine weiteren Dateien angegeben sind, alle Dateien auspacken.
    $all_files = true;
}

$z = zip_open($in_file) or die("Kann $in_file nicht öffnen: $php_errormsg");
while ($entry = zip_read($z)) {

    $entry_name = zip_entry_name($entry);

    // Überprüfen, ob alle Dateien ausgepackt werden sollen oder ob der Name
    // dieser Datei auf der Liste der Dateien steht, die entzippt werden sollen.
    if ($all_files || $out_files[$entry_name]) {

        // Nur weitermachen, wenn die Datei länger als 0 Bytes ist.
        if (zip_entry_filesize($entry)) {
            $dir = dirname($entry_name);

            // Alle notwendigen Verzeichnisse im Pfad der Datei erzeugen.
            if (! is_dir($dir)) { pc_mkdir_parents($dir); }

            $file = basename($entry_name);

            if (zip_entry_open($z,$entry)) {
                if ($fh = fopen($dir.'/'.$file,'w')) {
                    // Die gesamte Datei schreiben.
                    fwrite($fh,
                           zip_entry_read($entry,zip_entry_filesize($entry)))
                        or error_log("Kann nicht schreiben: $php_errormsg");
                    fclose($fh) or error_log("Kann nicht schließen: $php_errormsg");
                } else {
                    error_log("Kann $dir/$file nicht öffnen: $php_errormsg");
                }
                zip_entry_close($entry);
            } else {
                error_log("Kann Eintrag $entry_name nicht öffnen: $php_errormsg");
            }
        }
    }
}
```

Siehe auch

Rezept 21.23 zum Lesen und Schreiben von Dateien, die mit *zlib* komprimiert sind; Rezept 22.11 zur Funktion `pc_mkdir_parents()`; die Dokumentation zur *zip*-Erweiterung unter *http://www.php.net/zip*.

KAPITEL 22

Verzeichnisse

22.0 Einführung

Abgesehen vom eigentlichen Inhalt der Dateien speichert ein Dateisystem eine Menge an zusätzlichen Dateiinformationen. Dazu gehören Angaben wie die Dateigröße, in welchem Verzeichnis die Datei liegt, sowie die Zugriffsberechtigungen für die Datei. Wenn Sie mit Dateien arbeiten, müssen Sie diese Meta-Informationen eventuell verändern. PHP stellt eine Reihe von Funktionen zur Verfügung, mit denen Sie Verzeichnisse, Verzeichniseinträge und Dateiattribute lesen und ändern können. Von einigen Vereinfachungen abgesehen, ähneln die Funktionen den C-Funktionen, die den gleichen Zweck erfüllen, so wie das auch bei den anderen dateibezogenen PHP-Funktionen der Fall ist.

Dateien werden über *inodes* organisiert. Jede Datei (und andere Teile des Dateisystems, wie z.B. Verzeichnisse, Devices und Links) hat ihren eigenen Inode. Dieser Inode enthält einen Zeiger auf die Datenblöcke der Datei sowie die gesamten Metadaten über die Datei. Die Datenblöcke für ein Verzeichnis enthalten die Namen der Dateien in diesem Verzeichnis und den Inode jeder Datei.

PHP stellt drei Methoden zur Verfügung, in ein Verzeichnis hineinzuschauen und zu sehen, welche Dateien es enthält. Die erste besteht darin, `opendir()` zu verwenden, um ein Verzeichnis-Handle zu erhalten, mit `readdir()` durch die Dateien zu iterieren und das Verzeichnis-Handle mit `closedir()` zu schließen:

```
$d = opendir('/usr/local/images') or die($php_errormsg);
while (false !== ($f = readdir($d))) {
    // Datei bearbeiten.
}
closedir($d);
```

Die zweite Methode besteht darin, die Directory-Klasse zu verwenden. Instantiieren Sie die Klasse mit `dir()`, lesen Sie die Dateinamen einzeln mithilfe von `read()` ein und schließen Sie das Verzeichnis mit `close()`:

```
$d = dir('/usr/local/images') or die($php_errormsg);
while (false !== ($f = $d->read())) {
```

```
    // Datei bearbeiten.
}
$d->close();
```

Die dritte und eleganteste Methode besteht darin, den `DirectoryIterator` einzusetzen. Er steht seit PHP 5 zur Verfügung. Der Iterator liefert für jede Datei in einem Verzeichnis eine Objektinstanz mit der Schnittstelle von `SplFileInfo`. Über diese Instanz können mittels Methoden Informationen, wie z.B. Zeiten, Dateigröße und -typ, aber auch der Pfadname abgefragt werden:

```
foreach (new DirectoryIterator('/usr/local/images') as $file) {
    print $file->getPathname() . "\n";
}
```

Rezept 22.7 zeigt Ihnen die Verwendung dieses Iterators, um jede Datei eines Verzeichnisses zu bearbeiten. Das Erstellen von neuen Verzeichnissen wird in Rezept 22.11 behandelt und das Entfernen von Verzeichnissen in Rezept 22.12.

Das Dateisystem enthält mehr als nur Dateien und Verzeichnisse. Unter Unix kann es zusätzlich symbolische Links enthalten. Das sind spezielle Dateien, deren Inhalt ein Zeiger auf eine andere Datei ist. Sie können den Link löschen, ohne die Datei zu beeinflussen, auf die er zeigt. Um einen symbolischen Link zu erstellen, verwenden Sie `symlink()`:

```
symlink('/usr/local/images','/www/docroot/images') or die($php_errormsg);
```

Dies erstellt einen symbolischen Link mit dem Namen *images* in */www/docroot*, der auf */usr/local/images* zeigt.

Um Informationen über eine Datei, ein Verzeichnis oder einen Link herauszufinden, können Sie auch dessen Inode begutachten. Die Funktion `stat()`, Inhalt von Rezept 22.2, holt Ihnen die Meta-Information eines Inode. PHP hat außerdem viele Funktionen, die `stat()` intern verwenden, um Ihnen eine spezielle Information über eine Datei direkt zu geben. Diese Funktionen sind in Tabelle 22-1 aufgeführt.

Tabelle 22-1: Dateiinformationsfunktionen

Funktionsname	Welche Dateiinformation stellt die Funktion zur Verfügung?
`file_exists()`	Existiert die Datei?
`fileatime()`	Zeitstempel des letzten Zugriffs
`filectime()`	Zeitstempel der letzten Metadaten-Änderung
`filegroup()`	Gruppe (numerisch)
`fileinode()`	Inode-Nummer
`filemtime()`	Zeitstempel der letzten inhaltlichen Änderung
`fileowner()`	Owner (Eigentümer; numerisch)
`fileperms()`	Berechtigungen (dezimal, numerisch)
`filesize()`	Größe
`filetype()`	Typ (fifo, char, dir, block, link, file, unknown)

Tabelle 22-1: Dateiinformationsfunktionen (Fortsetzung)

Funktionsname	Welche Dateiinformation stellt die Funktion zur Verfügung?
`is_dir()`	Ist die Datei ein Verzeichnis?
`is_executable()`	Ist die Datei ausführbar?
`is_file()`	Handelt es sich um eine reguläre Datei?
`is_link()`	Ist sie ein symbolischer Link?
`is_readable()`	Ist die Datei lesbar?
`is_writable()`	Kann in die Datei geschrieben werden?

Unter Unix geben die Dateiberechtigungen an, welche Dateioperationen der Dateieigentümer, ein Benutzer in der Gruppe, dem die Datei zugeordnet ist, bzw. alle Benutzer durchführen können. Die möglichen Operationen sind Lesen, Schreiben und das Ausführen der Datei. Für Programme bedeutet Ausführen die Fähigkeit, das Programm laufen zu lassen; bei Verzeichnissen bedeutet es, dass das Verzeichnis durchsucht werden kann und man Dateien darin sehen kann.

Unix-Berechtigungen können auch ein setuid-Bit, ein setgid-Bit und ein Sticky-Bit enthalten. Das setuid-Bit bedeutet, dass eine Programmdatei mit der User-ID des Dateieigentümers ausgeführt wird (statt der User-ID des aufrufenden Benutzers). Das setgid-Bit bedeutet, dass ein Programm in der Datei unter der Gruppen-ID der Gruppe läuft, der die Datei zugeordnet ist. Bei einem Verzeichnis bedeutet das setgid-Bit, dass neue Dateien im Verzeichnis standardmäßig mit der gleichen Gruppe wie das Verzeichnis selbst erstellt werden. Das Sticky-Bit ist für Verzeichnisse hilfreich, in denen sich Benutzer Dateien teilen. Es verhindert, dass Nicht-Superuser-Benutzer mit Schreibberechtigung in einem Verzeichnis Dateien in diesem Verzeichnis löschen, wenn sie nicht der Eigentümer der Datei oder des Verzeichnisses sind.

Wenn Berechtigungen mit `chmod()` gesetzt werden (siehe Rezept 22.3), müssen sie als Oktalzahl dargestellt werden. Diese Zahl hat vier Stellen. Die erste Stelle kann irgendeine Spezialeinstellung für die Datei sein (wie z.B. setuid oder setgid). Die zweite Stelle stellt die Benutzerberechtigungen des Dateieigentümers dar. Die dritte Stelle steht für die Gruppenberechtigungen – was Benutzer in der Gruppe, der die Datei zugeordnet ist, mit ihr machen können. Die vierte Stelle steht für die globalen Berechtigungen – was alle Benutzer mit der Datei machen dürfen. Um den richtigen Wert für jede Stelle zu berechnen, addieren Sie die von Ihnen für die jeweilige Stelle gewünschten Berechtigungen anhand der Werte in Tabelle 22-2. Ein Berechtigungswert von zum Beispiel `0644` bedeutet: Keine Spezialeinstellungen sind vorhanden (die `0`), der Dateieigentümer kann die Datei lesen und sie beschreiben (`6`, die Summe aus `4` = lesen + `2` = schreiben), die Benutzer in der Gruppe können die Datei lesen (die erste `4`), und alle anderen Benutzer können die Datei ebenfalls lesen (die zweite `4`). Ein Berechtigungswert von `4644` bedeutet das Gleiche, außer dass die Datei zusätzlich setuid ist.

Tabelle 22-2: Dateizugriff: Berechtigungswerte

Wert	Berechtigung	Spezialeinstellung
4	Lesen	setuid
2	Schreiben	setgid
1	Ausführen	sticky

Die Berechtigungen neu erstellter Dateien und Verzeichnisse sind von einer Einstellung namens *umask* abhängig. Dies ist ein Berechtigungswert, der von der Default-Berechtigung einer Datei (`0666` bzw. `0777` bei Verzeichnissen) entfernt bzw. ausmaskiert wird. Wenn die umask beispielsweise `0022` ist, ist 0644 die Standardberechtigung für eine neue mit `touch()` oder `fopen()` erstellte Datei und 0755 die Standardberechtigung für ein neues Verzeichnis, das mit `mkdir()` erstellt wird. Mit der Funktion `umask()` können Sie die umask abfragen und einstellen. Es gibt die aktuelle umask zurück und ändert sie auf den Wert des übergebenen Arguments, falls vorhanden. Hier sehen Sie zum Beispiel, wie Sie für eine neu erstellte Datei die Berechtigungen so einstellen, dass nur der Dateieigentümer (und der Superuser) auf diese Datei Zugang haben:

```
$old_umask = umask(0077);
touch('secret-file.txt');
umask($old_umask);
```

Der erste Aufruf von `umask()` maskiert alle Berechtigungen für die Gruppe und den Rest der Welt. Wenn die Datei erstellt ist, stellt der Aufruf `umask()` die umask auf die vorherigen Einstellungen zurück. Wenn PHP als Server-Modul läuft, stellt es die umask am Ende jedes HTTP-Requests auf seinen Standardwert zurück. Ebenso wie andere Funktionen, die mit Berechtigungen zu tun haben, funktioniert `umask()` nicht unter Windows.

22.1 Zeitstempel auslesen und setzen

Problem

Sie wollen wissen, wann eine Datei zum letzten Mal geändert wurde oder wann das letzte Mal auf sie zugegriffen wurde, oder Sie wollen die Zugangs- oder Änderungszeit einer Datei auf den neusten Stand bringen; zum Beispiel wollen Sie auf jeder Seite Ihrer Website anzeigen, wann sie zum letzten Mal geändert wurde.

Lösung

Die Funktionen `fileatime()`, `filemtime()` und `filectime()` geben an, wann zum letzten Mal auf eine Datei zugegriffen oder diese modifiziert wurde oder wann deren Metadaten geändert wurden:

```
$letzter_zugriff = fileatime('larry.php');
$letze_aenderung = filemtime('moe.php');
```

```
$letzte_metaaenderung = filectime('curly.php');
```

Die Funktion `touch()` ändert die Modifikationszeit einer Datei:

```
touch('shemp.php');          // Modifikationszeit auf 'jetzt' setzen.
touch('joe.php',$timestamp); // Modifikationszeit auf $timestamp setzen.
```

Diskussion

Die Funktion `fileatime()` gibt die Zeit zurück, zu der eine Datei zum letzten Mal zum Lesen oder Schreiben geöffnet wurde. Die Funktion `filemtime()` gibt die Zeit zurück, zu der der Inhalt einer Datei zum letzten Mal geändert wurde. Die Funktion `filectime()` schließlich gibt die Zeit zurück, zu der der Inhalt oder die Meta-Informationen (wie z.B. Eigentümer oder Berechtigungen) einer Datei zum letzten Mal geändert wurden. Jede Funktion gibt die Zeit als einen Epochen-Zeitstempel zurück (Zahl der Sekunden seit dem 1.1.1970, 00:00 GMT).

Die Modifikationszeit einer Datei kann mit `touch()` geändert werden. Wenn kein zweites Argument vorhanden ist, setzt `touch()` die Modifikationszeit auf das aktuelle Datum und die aktuelle Zeit. Um die Modifikationszeit einer Datei auf einen bestimmten Wert zu setzen, übergeben Sie diesen Wert als einen Epochen-Zeitstempel als zweites Argument an `touch()`.

Dieser Code gibt die Zeit aus, zu der eine Seite Ihrer Website zuletzt geändert wurde:

```
print "Last Modified: ".strftime('%c',filemtime($_SERVER['SCRIPT_FILENAME']));
```

Siehe auch

Die Dokumentationen zu `fileatime()` unter *http://www.php.net/fileatime*, `filemtime()` unter *http://www.php.net/filemtime* und `filectime()` unter *http://www.php.net/filectime*.

22.2 Auf Dateiinformationen zugreifen

Problem

Sie wollen die Metadaten einer Datei direkt lesen, beispielsweise Berechtigungen und Eigentümerinformationen.

Lösung

Verwenden Sie `stat()`, das ein Array mit Informationen über eine Datei zurückgibt:

```
$info = stat('harpo.php');
```

Diskussion

Die Funktion `stat()` gibt ein Array mit sowohl numerischen als auch String-Indizes zurück, das Informationen über eine Datei enthält. Die Elemente dieses Arrays sind in Tabelle 22-3 aufgeführt.

Tabelle 22-3: Von stat() zurückgegebene Informationen

Numerischer Index	String-Index	Wert
0	dev	Device
1	ino	Inode
2	mode	Berechtigungen
3	nlink	Link-Anzahl
4	uid	User-ID des Eigentümers
5	gid	Gruppen-ID der Gruppe
6	rdev	Device-Typ für Inode-Devices (-1 unter Windows)
7	size	Größe (in Bytes)
8	atime	Letzte Zugriffszeit (Epochen-Zeitstempel)
9	mtime	Letzte Änderungszeit des Inhalts (Epochen-Zeitstempel)
10	ctime	Letzte Änderungszeit des Inhalts oder der Metadaten (Epochen-Zeitstempel)
11	blksize	Blockgröße für I/O (–1 unter Windows)
12	blocks	Anzahl der Blöcke, die dieser Datei zugewiesen sind

Das Element `mode` des zurückgegebenen Arrays enthält die Berechtigungen, ausgedrückt als Ganzzahl zur Basis 10. Das ist verwirrend, da Berechtigungen normalerweise entweder symbolisch (z.B. die Ausgabe `-rw-r--r--` von *ls -l*) oder durch eine Oktalzahl (z.B. `0644`) dargestellt werden. Um die Berechtigungen in eine verständlichere Form zu bringen, konvertieren Sie sie mit `base_convert()` nach Oktal:

```
$file_info = stat('/tmp/session.txt');
$permissions = base_convert($file_info['mode'],10,8);
```

Das Ergebnis ist eine sechsstellige Oktalzahl. Wenn *ls* z.B. über */tmp/session.txt* das Folgende anzeigt:

```
-rw-rw-r--    1 sklar    sklar          12 Oct 23 17:55 /tmp/session.txt
```

ist `$file_info['mode']` `33204` und `$permissions` `100664`. Die letzten drei Stellen (`664`) sind die Dateiberechtigungen des Benutzers (Lesen und Schreiben), der Gruppe (Lesen und Schreiben) und aller anderen (Lesen). Die dritte Stelle, `0`, bedeutet, dass die Datei nicht setuid oder setgid ist. Die `10` ganz links bedeutet, dass die Datei eine reguläre Datei ist (und nicht ein Socket, ein symbolischer Link oder eine andere spezielle Datei).

Da `stat()` ein Array mit sowohl numerischen als auch String-Indizes zurückgibt, werden zwei Ausfertigungen von jedem Wert erstellt, wenn das zurückgegebene Array mit

foreach durchlaufen wird. Verwenden Sie stattdessen eine for-Schleife von Element 0 bis Element 12 des zurückgegebenen Arrays.

Der Aufruf von stat() auf einem symbolischen Link gibt Informationen über die Datei zurück, auf die der symbolische Link zeigt. Um Informationen über den symbolischen Link selbst zu erhalten, verwenden Sie lstat().

Mit stat() verwandt ist fstat(), das ein Datei-Handle (von fopen() oder popen() zurückgegeben) als Argument verwendet. Allerdings können Sie fstat() nur bei lokalen Dateien verwenden und nicht bei URLs, die an fopen() weitergereicht wurden.

Die PHP-Funktion stat() verwendet den zugrunde liegenden *stat(2)*-Systemruf, der aufwendig ist. Um den Overhead zu minimieren, speichert PHP das Ergebnis des Aufrufs an *stat(2)* zwischen. Wenn Sie also stat() bei einer Datei aufrufen, ihre Berechtigungen ändern und stat() bei der gleichen Datei noch einmal aufrufen, bekommen Sie das gleiche Ergebnis. Um PHP zu zwingen, die Metadaten der Datei erneut zu laden, rufen Sie clearstatcache() auf, das die von PHP zwischengespeicherten Informationen löscht. PHP verwendet diesen Zwischenspeicher auch für die anderen Funktionen, die Meta-Informationen zurückgeben: file_exists(), fileatime(), filectime(), filegroup(), fileinode(), filemtime(), fileowner(), fileperms(), filesize(), filetype(), fstat(), is_dir(), is_executable(), is_file(), is_link(), is_readable(), is_writable() und lstat().

Siehe auch

Die Dokumentationen zu stat() unter *http://www.php.net/stat*, lstat() unter *http://www.php.net/lstat*, fstat() unter *http://www.php.net/fstat* und clearstatcache() unter *http://www.php.net/clearstatcache*.

22.3 Dateiberechtigungen oder Dateieigentümerschaft ändern

Problem

Sie möchten die Berechtigungen oder den Eigentümer einer Datei ändern. Beispiel: Sie wollen andere Benutzer davon abhalten, eine Datei mit sensiblen Daten lesen zu können.

Lösung

Verwenden Sie chmod() zum Ändern von Dateiberechtigungen:

```
chmod('/home/user/geheimnisse.txt',0400);
```

Verwenden Sie chown(), um den Eigentümer einer Datei zu ändern, und chgrp(), um die Gruppe einer Datei zu ändern:

```
chown('/tmp/myfile.txt','sklar');           // Benutzer über Namen angeben.
chgrp('/home/sklar/schedule.txt','soccer'); // Gruppe über Namen angeben.

chown('/tmp/myfile.txt',5001);              // Benutzer über uid angeben.
chgrp('/home/sklar/schedule.txt',102);      // Gruppe über gid angeben.
```

Diskussion

Die an chmod() übergebenen Berechtigungen müssen als Oktalzahl angegeben werden.

Der Superuser kann die Berechtigungen, den Eigentümer und die Gruppe jeder beliebigen Datei verändern. Andere Benutzer sind in dieser Hinsicht eingeschränkt. Sie können nur die Berechtigungen und die Gruppe von Dateien ändern, die ihnen gehören, und können den Eigentümer überhaupt nicht ändern. Nicht-Superuser können die Gruppe einer Datei auch nur zu einer Gruppe ändern, der sie selbst angehören.

Die Funktionen chmod(), chgrp() und chown() funktionieren nicht unter Windows.

Siehe auch

Die Dokumentationen zu chmod() unter *http://www.php.net/chmod*, chown() unter *http://www.php.net/chown* und chgrp() unter *http://www.php.net/chgrp*.

22.4 Einen Dateinamen in seine Bestandteile zerlegen

Problem

Sie wollen den Pfad und den Dateinamen einer Datei herausfinden. Beispielsweise wollen Sie eine Datei im selben Verzeichnis erstellen wie eine existierende Datei.

Lösung

Verwenden Sie basename(), um den Dateinamen zu erhalten, und dirname() zum Ermitteln des Pfads:

```
$voller_name = '/usr/local/php/php.ini';
$base = basename($voller_name);  // $base ist php.ini
$dir  = dirname($voller_name);   // $dir ist /usr/local/php
```

Verwenden Sie pathinfo(), um den Verzeichnisnamen, Dateinamen und die Erweiterung in einem assoziativen Array anzuzeigen:

```
$info = pathinfo('/usr/local/php/php.ini');
// $info['dirname'] ist "/usr/local/php"
// $info['basename'] ist "php.ini"
// $info['extension'] ist "ini"
```

Diskussion

Um eine temporäre Datei im selben Verzeichnis wie eine existierende Datei zu erstellen, verwenden Sie `dirname()`, um das Verzeichnis zu finden, und übergeben dieses Verzeichnis an `tempnam()`:

```
$dir = dirname($existierende_datei);
$temp = tempnam($dir,'temp');
$temp_fh = fopen($temp,'w');
```

Die Elemente in dem assoziativen Array, das von `pathinfo()` zurückgegeben wird, sind `dirname`, `basename` und `extension`:

```
$info = pathinfo('/usr/local/php/php.ini');
print_r($info);
Array
(
    [dirname] => /usr/local/php
    [basename] => php.ini
    [extension] => ini
)
```

Sie können `basename()` auch einen optionalen Suffix übergeben, der von dem Dateinamen abgezogen wird. Der folgenden Code setzt `$base` zu *php*:

```
$base = basename('/usr/local/php/php.ini','.ini');
```

Die Funktion `dirname()` ist in Kombination mit der Konstanten `__FILE__` sehr hilfreich, wenn Sie herausbekommen wollen, in welchem Verzeichnis Ihr aktuelles PHP-Skript liegt. `__FILE__` liefert den kompletten Dateinamen der aktuellen Datei zurück. So können Sie einfach weitere PHP-Skripten, die in dem Verzeichnis Ihres Skripts liegen, einbinden:

```
$currentDir = dirname(__FILE__);
include $currentDir . '/functions.php';
include $currentDir . '/classes.php';
```

Die Verwendung von Funktionen wie `basename()`, `dirname()` und `pathinfo()` ist portabler als das bloße Trennen eines vollen Dateinamens mit Schrägstrichen `/`, weil die Funktionen ein Trennzeichen verwenden, das zum Betriebssystem passt. Unter Windows behandeln diese Funktionen sowohl `/` als auch `\` als Trennzeichen zwischen Datei- bzw. Verzeichnisnamen. Auf anderen Plattformen wird nur `/` verwendet.

Zum Zusammenfügen der von `basename()`, `dirname()` und `pathinfo()` produzierten Teile zurück zu einem voll qualifizierten Dateinamen gibt es in PHP keine Funktion. Zu diesem Zweck müssen Sie die Teile mit `.` und der Konstanten `DIRECTORY_SEPARATOR` zusammenkleben, die unter Unix `/` und unter Windows `\` ist:

```
$dirname = '/usr/local/php';
$basename = 'php';
$extension = 'ini';

$voller_name = $dirname . DIRECTORY_SEPARATOR . $basename . '.' . $extension;
```

Siehe auch

Die Dokumentationen zu `basename()` unter *http://www.php.net/basename*, `dirname()` unter *http://www.php.net/dirname* und `pathinfo()` unter *http://www.php.net/pathinfo*.

22.5 Eine Datei löschen

Problem

Sie wollen eine Datei löschen.

Lösung

Verwenden Sie `unlink()`:

```
unlink($file) or die ("Kann $file nicht löschen: $php_errormsg");
```

Diskussion

Die Funktion `unlink()` kann nur solche Dateien löschen, die der Benutzer des PHP-Prozesses löschen kann. Wenn Sie Probleme haben, `unlink()` zum Laufen zu bekommen, sollten Sie die Dateiberechtigungen prüfen. Falls auch das nichts hilft, sehen Sie sich an, als welcher Benutzer bzw. welche Gruppe Sie PHP laufen lassen.

Siehe auch

Dokumentation zu `unlink()` unter *http://www.php.net/unlink*.

22.6 Eine Datei kopieren oder verschieben bzw. umbenennen

Problem

Sie wollen eine Datei kopieren oder umbenennen bzw. verschieben.

Lösung

Verwenden Sie `copy()`, um eine Datei zu kopieren:

```
copy($alt,$neu) or die("Kann $alt nicht nach $neu kopieren: $php_errormsg");
```

Verwenden Sie `rename()`, um eine Datei zu verschieben bzw. umzubenennen:

```
rename($alt,$neu) or die("Kann $alt nicht nach $neu verschieben: $php_errormsg");
```

Diskussion

Unter Unix kann `rename()` Dateien nicht über Dateisysteme hinweg verschieben, wenn die PHP-Version kleiner oder gleich 4.3.3 ist. Um das zu schaffen, kopieren Sie die Datei an ihren neuen Platz und löschen dann die alte Datei:

```
if (copy("/tmp/code.c","/usr/local/src/code.c")) {
  unlink("/tmp/code.c");
}
```

Wenn Sie mehrere Dateien kopieren oder verschieben müssen, rufen Sie `copy()` oder `rename()` in einer Schleife auf. Sie können bei jedem Aufruf dieser Funktionen nur eine Datei bearbeiten.

Siehe auch

Die Dokumentationen zu `copy()` unter *http://www.php.net/copy* und `rename()` unter *http://www.php.net/rename*.

22.7 Alle Dateien in einem Verzeichnis verarbeiten

Problem

Sie möchten alle Dateien in einem Verzeichnis durchlaufen. Sie wollen zum Beispiel in einem Formular eine `<select/>`-Box aufbauen, die alle Dateien in einem Verzeichnis aufführt.

Lösung

Nutzen Sie, wie in Beispiel 22-1 gezeigt, einen `DirectoryIterator`, um alle Dateien in jenem Verzeichnis zu erhalten.

Beispiel 22-1: Alle Dateien in einem Verzeichnis verarbeiten

```
<?php
echo "<select name='file'>\n";
foreach (new DirectoryIterator('/usr/local/images') as $file) {
    echo '<option>' . htmlentities($file) . "</option>\n";
}
echo '</select>';
?>
```

Diskussion

Der `DirectoryIterator` liefert Ihnen ein Objekt (in der Lösung in der Variablen `$file`) für jedes Element in diesem Verzeichnis. Dieses Iterator-Objekt besitzt unter anderem die

geerbte Schnittstelle von `SplFileInfo`, mit der Sie Informationen des Verzeichniselements ermitteln können. Die String-Darstellung dieses Objekts ist der Dateiname (ohne vorangehenden Pfad) des Verzeichniselements. Enthält */usr/local/images* die Dateien *cucumber.gif* und *eggplant.png*, liefert Beispiel 22-1 beispielsweise Folgendes:

```
<select name='file'>
<option>.</option>
<option>..</option>
<option>cucumber.gif</option>
<option>eggplant.png</option>
</select>
```

Ein `DirectoryIterator` liefert jeweils ein Objekt für alle Elemente des Verzeichnisses, `.` (aktuelles Verzeichnis) und `..` (Elternverzeichnis) eingeschlossen. Glücklicherweise haben diese Objekte einige Methoden, die uns dabei unterstützen, herauszufinden, was das Objekt kapselt. Die Methode `isDot( )` liefert true, wenn wir `.` oder `..` vor uns haben. Beispiel 22-2 nutzt `isDot( )`, um zu verhindern, dass diese beiden Einträge in der Aufstellung auftauchen.

Beispiel 22-2: . und .. aus der Ausgabe entfernen

```
<?php
echo "<select name='file'>\n";
foreach (new DirectoryIterator('/usr/local/images') as $file) {
    if (! $file->isDot()) {
        echo '<option>' . htmlentities($file) . "</option>\n";
    }
}
echo '</select>';
?>
```

Tabelle 22-4 führt die wichtigsten Methoden auf, die auf den Objekten verfügbar sind, die ein `DirectoryIterator` liefert.

Tabelle 22-4: DirectoryIterator

Methodenname	Rückgabewert	Beispiel
`isDir( )`	Ist Element ein Verzeichnis?	`false`
`isDot( )`	Ist Element `.` oder `..`?	`false`
`isFile( )`	Ist Element eine gewöhnliche Datei?	`true`
`isLink( )`	Ist Element ein Link?	`false`
`isReadable( )`	Ist Element lesbar?	`true`
`isWritable( )`	Ist Element schreibbar?	`true`
`isExecutable( )`	Ist Element ausführbar?	`false`
`getATime( )`	Letzte Zugriffszeit des Elements.	`1144509622`
`getCTime( )`	Erstellungszeit des Elements.	`1144509600`
`getMTime( )`	Letzte Veränderungszeit des Elements.	`1144509620`

Tabelle 22-4: DirectoryIterator (Fortsetzung)

Methodenname	Rückgabewert	Beispiel
`getFilename()`	Der Dateiname des Elements (ohne Pfad).	`eggplant.png`
`getPathname()`	Der vollständige Pfadname des Elements.	`/usr/local/images/eggplant.php`
`getPath()`	Der Pfad des Elements.	`/usr/local/images`
`getGroup()`	Die Gruppen-ID des Elements.	`500`
`getOwner()`	Die Eigentümer-ID des Elements.	`1000`
`getPerms()`	Die Berechtigungen für das Element (oktal).	`16895`
`getSize()`	Die Größe des Elements.	`328742`
`getType()`	Der Typ des Elements (dir, file, link usw.).	`file`
`getInode()`	Die Inode-Nummer des Elements.	`28720`

Die Daten der Methoden in der Aufstellung in Tabelle 22-4 beruhen auf den gleichen zugrunde liegenden Systemaufrufen wie die Daten der Funktionen in der Aufstellung in Tabelle 22-1. Es sind also die gleichen Hinweise in Bezug auf die Unterschiede zwischen Unix und Windows zu beachten.

Siehe auch

Die Dokumentationen zu `DirectoryIterator` unter *http://www.php.net/~helly/php/ext/spl/classDirectoryIterator.html* und zu `SplFileInfo` unter *http://de.php.net/manual/en/class.splfileinfo.php*.

22.8 Alle Dateinamen finden, die einem Muster entsprechen

Problem

Sie möchten alle Dateien finden, deren Namen einem bestimmten Muster entsprechen.

Lösung

Nutzen Sie eine Instanz einer `FilterIterator`-Unterklasse mit dem `DirectoryIterator`. Die `FilterIterator`-Unterklasse benötigt eine `accept()`-Methode, die entscheidet, ob ein bestimmter Wert übernommen werden soll oder nicht. Der Code in Beispiel 22-3 akzeptiert beispielsweise nur Dateinamen, die mit einer der üblichen Dateiendungen für Grafikdateien enden.

Beispiel 22-3: Einsatz eines FilterIterator

```
<?php
class ImageFilter extends FilterIterator {
    public function accept() {
        return preg_match('@\.(gif|jpe?g|png)$@i',$this->current());
```

Beispiel 22-3: Einsatz eines FilterIterator (Fortsetzung)

```
        }
}
foreach (new ImageFilter(new DirectoryIterator('/usr/local/images')) as $img) {
    print "<img src='".htmlentities($img)."'/>\n";
}
?>
```

Diskussion

Der `FilterIterator` kapselt einen `DirectoryIterator` und lässt nur bestimmte Elemente durch. Ob ein bestimmtes Element (auf das mit `$this->current()` zugegriffen wird) übernommen werden soll, entscheidet die `accept()`-Methode, indem sie entweder `true` oder `false` liefert. In Beispiel 22-3 nutzt `accept()` dafür einen regulären Ausdruck, aber Ihr Code kann jede beliebige Logik verwenden.

Ihr Muster kann auch als einfacher Shell-Glob (z.B. `*.*`) ausgedrückt und über die Funktion `glob()` eingesetzt werden, um die passenden Dateinamen zu finden. Beispiel 22-4 findet alle Textdateien in einem bestimmten Verzeichnis.

Beispiel 22-4: Einsatz von glob()

```
<?php
foreach (glob('/usr/local/docs/*.txt') as $file) {
   $contents = file_get_contents($file);
   print "$file enthält $contents\n";
}
?>
```

Die Funktion `glob()` liefert ein Array mit allen passenden vollständigen Pfadnamen. Findet das Muster keine Dateien, liefert `glob()` `false`.

Siehe auch

Rezept 22.9 führt aus, wie man die Dateien in einem Verzeichnis rekursiv durchläuft; die Dokumentationen zu `FilterIterator` unter *http://www.php.net/~helly/php/ext/spl/classFilterIterator.html* und zu `glob()` unter *http://www.php.net/glob*; Informationen zum Shell-Globbing finden Sie unter *http://www.gnu.org/software/bash/manual/bashref.html#SEC35*.

22.9 Alle Dateien in einem Verzeichnis rekursiv verarbeiten

Problem

Sie möchten etwas mit allen Dateien in einem Verzeichnis und in all seinen Unterverzeichnissen machen. Sie möchten beispielsweise ermitteln, wie viel Speicherplatz die Dateien unterhalb eines Verzeichnisses insgesamt einnehmen.

Lösung

Nutzen Sie einen `RecursiveDirectoryIterator` und einen `RecursiveIteratorIterator`. Der `RecursiveDirectoryIterator` erweitert `DirectoryIterator` mit einer `getChildren()`-Methode, die Zugriff auf die Elemente in einem Unterverzeichnis bietet. Der `RecursiveIteratorIterator` ebnet die Hierarchie, die der `RecursiveDirectoryIterator` liefert, zu einer Liste ein. Beispiel 22-5 zählt die Gesamtgröße der Dateien in einem Verzeichnis.

Beispiel 22-5: Alle Dateien in einem Verzeichnis rekursiv verarbeiten

```
<?php
$dir = new RecursiveDirectoryIterator('/usr/local');
$totalSize = 0;
foreach (new RecursiveIteratorIterator($dir) as $file) {
    $totalSize += $file->getSize();
}
print "Die Gesamtgröße ist $totalSize.\n";
?>
```

Diskussion

Die Objekte, die der `RecursiveDirectoryIterator` ausspuckt (und die deswegen vom `RecursiveIteratorIterator` weitergereicht werden), entsprechen denen, die Ihnen ein `DirectoryIterator` liefert. Auf ihnen sind also alle in Tabelle 22-4 erwähnten Methoden verfügbar.

Siehe auch

Die Dokumentationen zu `RecursiveDirectoryIterator` unter *http://www.php.net/~helly/php/ext/spl/classRecursiveDirectoryIterator.html* und zu `RecursiveIteratorIterator` unter *http://www.php.net/~helly/php/ext/spl/classRecursiveIteratorIterator.html*.

22.10 Dateien eines Verzeichnisses filtern

Problem

Sie wollen etwas mit allen Dateien eines Verzeichnisses machen, die bestimmten Kriterien entsprechen. Beispielsweise wollen Sie nur Dateien bearbeiten, die eine bestimmte Größe haben und nicht älter als ein bestimmtes Datum sind oder nicht tiefer als in einer bestimmten Ebene in Unterverzeichnissen liegen.

Lösung

Setzen Sie einen `FilterIterator` ein, dessen `accept()`-Methode die nötigen Kriterien überprüft:

```
class ComplexFilterIterator extends FilterIterator {
    public $maxSize;
    public $maxLevels;
    public $maxAge;

    public function __construct($iterator, $maxSize, $maxLevels, $maxAge) {
        parent::__construct($iterator);
        $this->maxSize = $maxSize; // maximale Größe in Bytes
        $this->maxLevels = $maxLevels; // maximale Tiefe
        $this->maxAge = $maxAge; // maximales Alter in Sek.
    }

    public function accept() {
        return (
            // Wenn es eine Datei ist UND
            $this->current()->isFile() &&
            // kleiner als $maxSize ist UND
            ($this->current()->getSize() < $this->maxSize) &&
            // weniger als $maxLevels tief liegt UND
            (count(explode("/",$this->current()->getPathName()))
                          <= $this->maxLevels + 1) &&
            // weniger als $maxAge Sekunden alt ist, true zurückgeben.
            ($this->current()->getMTime() > time() - $this->maxAge));
    }
}

foreach (new ComplexFilterIterator(
            new RecursiveIteratorIterator (
                new RecursiveDirectoryIterator(".")), 1024, 2, 86400)
            as $filename)  {
    echo $filename->getPathname()."<br/>";
}
```

Diskussion

Ein `FilterIterator` ist eine abstrakte Klasse, die Sie erst einmal ableiten müssen, um sie einsetzen zu können. In der abgeleiteten Klasse müssen Sie dann eine Methode namens `accept()` definieren, die `true` zurückgibt, wenn das mit `$this->current()` vom Iterator zurückgegebene Element ausgegeben werden soll, und `false`, wenn nicht. Die Methode `accept()` wird automatisch von `foreach` aufgerufen, wenn sie es mit einem `FilterIterator` zu tun bekommt. Eine solche Vorgehensweise ist vor allem dann praktisch, wenn Sie dieselbe Filterstrategie mehrfach einsetzen müssen. Damit können Sie den Filtercode mit den Filterkriterien von Ihrem Iterationscode trennen.

Beachten Sie, dass Sie den `FilterIterator` hier auf der obersten Ebene einsetzen müssen, die folgende Kombination führt zu einem Parsing-Fehler:

```
foreach (new RecursiveIteratorIterator (new ComplexFilterIterator(
                new RecursiveDirectoryIterator("."), 1024, 2, 86400))
            as $filename)  {
    echo $filename->getPathname()."<br/>";
}
```

Siehe auch

Die Dokumentationen zu Iteratoren und SPL unter *http://www.php.net/spl*.

22.11 Neue Verzeichnisse erstellen

Problem

Sie wollen ein Verzeichnis erstellen.

Lösung

Verwenden Sie `mkdir()`:

```
mkdir('/tmp/aepfel',0777) or die($php_errormsg);
```

Diskussion

Das zweite Argument bei `mkdir()` ist der Berechtigungsmodus für ein neues Verzeichnis, der durch eine Oktalzahl angegeben wird. Die aktuelle umask wird von diesem Berechtigungswert abgezogen, um die Berechtigungen für das neue Verzeichnis zu erstellen. Wenn also die aktuelle umask `0002` ist, werden durch den Aufruf von `mkdir('/tmp/apples',0777)` die Berechtigungen am resultierenden Verzeichnis auf `0775` gesetzt (Eigentümer und Gruppe können lesen, schreiben und Programme ausführen, andere Benutzer können lediglich lesen und Programme ausführen).

Standardmäßig erstellt `mkdir()` ein Verzeichnis nur dann, wenn sein übergeordnetes Verzeichnis existiert. Wenn zum Beispiel */tmp/a* nicht existiert, können Sie */tmp/a/b* erst erstellen, nachdem */tmp/a* erstellt wurde. Seit PHP 5.0 kennt `mkdir()` nun einen weiteren, dritten Parameter, der optional angegeben werden kann. Wenn Sie diesem den Booleschen Wert `true` übergeben, werden ebenfalls alle übergeordneten Verzeichnisse rekursiv erstellt:

```
mkdir('/viele/Verzeichnisse/zum/Ziel',0777,true) or die($php_errormsg);
```

Siehe auch

Die Dokumentation zu `mkdir()` unter *http://www.php.net/mkdir*.

22.12 Ein Verzeichnis und seinen Inhalt entfernen

Problem

Sie möchten ein Verzeichnis und seinen gesamten Inhalt löschen, Unterverzeichnisse und deren Inhalt eingeschlossen.

Lösung

Nutzen Sie `RecursiveDirectoryIterator` und `RecursiveIteratorIterator` und geben Sie dabei an, dass Kinder (Dateien und Unterverzeichnisse) vor den Eltern aufgeführt werden sollen, wie Sie es in Beispiel 22-6 sehen.

Beispiel 22-6: Ein Verzeichnis vernichten

```
<?php
function obliterate_directory($dir) {
    $iter = new RecursiveDirectoryIterator($dir);
    foreach (new RecursiveIteratorIterator($iter, RecursiveIteratorIterator::CHILD_FIRST) as $f) {
        if ($f->isDir()) {
            rmdir($f->getPathname());
        } else {
            unlink($f->getPathname());
        }
    }
    rmdir($dir);
}
obliterate_directory('/tmp/junk');
?>
```

Diskussion

Das Löschen von Dateien kann, verständlicherweise, gefährlich sein. Da PHPs eingebaute Funktion zum Löschen von Verzeichnissen, `rmdir( )`, nur bei leeren Verzeichnissen funktioniert, und `unlink( )` keine Shell-Jokerzeichen akzeptiert, muss dem `RecursiveIteratorIterator` mit der Konstanten `CHILD_FIRST` mitgeteilt werden, dass Kinder vor Eltern verarbeitet werden sollen.

Diese Konstante ist allerdings erst seit PHP 5.1 verfügbar. Wenn Sie eine ältere PHP-Version nutzen, können Sie für den gleichen Zweck die Funktion in Beispiel 22-7 nutzen.

Beispiel 22-7: Ein Verzeichnis vernichten ohne RecursiveIteratorIterator

```
<?php
function obliterate_directory($dir) {
    foreach (new DirectoryIterator($dir) as $file) {
        if ($file->isDir()) {
            if (! $file->isDot()) {
                obliterate_directory($file->getPathname());
            }
        } else {
            unlink($file->getPathname());
        }
    }
    rmdir($dir);
}
?>
```

Siehe auch

Die Dokumentationen zu `rmdir()` unter *http://www.php.net/rmdir* und zu `RecursiveIteratorIterator` unter *http://www.php.net/~helly/php/ext/spl/classRecursiveIteratorIterator.html*.

22.13 Programm: Eine Auflistung des Webserver-Verzeichnisses erstellen

Das in Beispiel 22-8 dargestellte Programm *web-ls.php* stellt eine Auflistung der Dateien

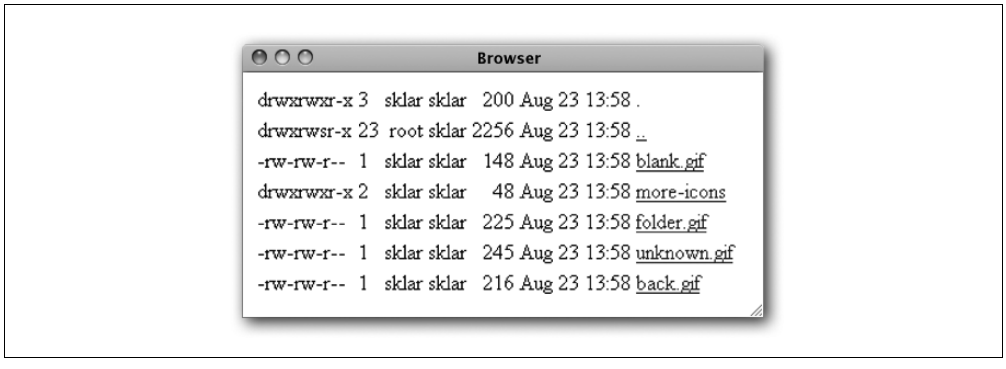


Abbildung 22-1: Auflistung des Webserver-Verzeichnisses

innerhalb des Dokumenten-Stammverzeichnisses Ihres Webservers zur Verfügung, die wie die Ausgabe des Unix-Befehls *ls* formatiert ist. Dateinamen sind so gelinkt, dass Sie jede Datei herunterladen können, und Verzeichnisnamen sind so gelinkt, dass Sie in dem jeweiligen Verzeichnis browsen können, wie in Abbildung 22-1 dargestellt.

Dieses Beispiel zeigt den Einsatz von `dir()`. Alternativ können Sie auch hier den `DirectoryIterator` einsetzen, den Sie in den Kapiteln zuvor bereits kennengelernt haben. Die meisten Zeilen in Beispiel 22-8 beschäftigen sich mit dem Bau einer leicht zu lesenden Darstellung der Dateiberechtigungen, aber der Kern des Programms ist die `while`-Schleife am Schluss. Mit der `$d->read()`-Methode wird der Name jeder Datei im Verzeichnis abgerufen. Dann holt sich `lstat()` Informationen über die jeweilige Datei, und `printf()` druckt die dazugehörigen formatierten Dateiinformationen aus.

Die Funktion `mode_string()` und die von ihr verwendeten Konstanten konvertieren die Oktal-Repräsentation des Dateimodus (z.B. `35316`) in einen einfacher zu lesenden String (z.B. `-rwsrw-r--`).

Beispiel 22-8: web-ls.php

```
/* Bit-Masken zur Ermittlung von Dateiberechtigungen und -typ. Die unten aufgeführten
 * Namen und Werte entsprechen dem POSIX-Standard, individuelle Systeme können
 * ihre eigenen Erweiterungen haben.
 */

define('S_IFMT',0170000);   // Maske für alle Typen
define('S_IFSOCK',0140000); // Typ: Socket
define('S_IFLNK',0120000);  // Typ: symbolischer Link
define('S_IFREG',0100000);  // Typ: reguläre Datei
define('S_IFBLK',0060000);  // Typ: Block-Device
define('S_IFDIR',0040000);  // Typ: Verzeichnis
define('S_IFCHR',0020000);  // Typ: Character-Device
define('S_IFIFO',0010000);  // Typ: fifo
define('S_ISUID',0004000);  // set-uid-Bit
define('S_ISGID',0002000);  // set-gid-Bit
define('S_ISVTX',0001000);  // Sticky-Bit
define('S_IRWXU',00700);    // Maske für Eigentümerberechtigungen
define('S_IRUSR',00400);    // Eigentümer: Leseberechtigung
define('S_IWUSR',00200);    // Eigentümer: Schreibberechtigung
define('S_IXUSR',00100);    // Eigentümer: Ausführungsberechtigung
define('S_IRWXG',00070);    // Maske für Gruppenberechtigungen
define('S_IRGRP',00040);    // Gruppe: Leseberechtigung
define('S_IWGRP',00020);    // Gruppe: Schreibberechtigung
define('S_IXGRP',00010);    // Gruppe: Ausführungsberechtigung
define('S_IRWXO',00007);    // Maske für Berechtigungen anderer
define('S_IROTH',00004);    // Andere: Leseberechtigung
define('S_IWOTH',00002);    // Andere: Schreibberechtigung
define('S_IXOTH',00001);    // Andere: Ausführungsberechtigung

/* mode_string() ist eine Hilfsfunktion, die einen Oktal-Modus nimmt und einen
 * 10-Zeichen-String zurückgibt, der den Dateityp und die Berechtigungen repräsentiert,
 * die dem Oktal-Modus entsprechen. Dabei handelt es sich um eine PHP-Version der
 *  mode_string()-Funktion des GNU fileutils-Pakets.
 */
function mode_string($mode) {
  $s = array();

 // Typ-Buchstaben setzen.
  if (($mode & S_IFMT) == S_IFBLK) {
    $s[0] = 'b';
  } elseif (($mode & S_IFMT) == S_IFCHR) {
    $s[0] = 'c';
  } elseif (($mode & S_IFMT) == S_IFDIR) {
    $s[0] = 'd';
  } elseif (($mode & S_IFMT) ==  S_IFREG) {
    $s[0] = '-';
  } elseif (($mode & S_IFMT) ==  S_IFIFO) {
    $s[0] = 'p';
  } elseif (($mode & S_IFMT) == S_IFLNK) {
    $s[0] = 'l';
  } elseif (($mode & S_IFMT) == S_IFSOCK) {
    $s[0] = 's';
```

Beispiel 22-8: web-ls.php (Fortsetzung)

```
  }

  //Eigentümerberechtigungen setzen.
  $s[1] = $mode & S_IRUSR ? 'r' : '-';
  $s[2] = $mode & S_IWUSR ? 'w' : '-';
  $s[3] = $mode & S_IXUSR ? 'x' : '-';

  // Gruppenberechtigungen setzen.
  $s[4] = $mode & S_IRGRP ? 'r' : '-';
  $s[5] = $mode & S_IWGRP ? 'w' : '-';
  $s[6] = $mode & S_IXGRP ? 'x' : '-';

  // Berechtigungen für andere setzen.
  $s[7] = $mode & S_IROTH ? 'r' : '-';
  $s[8] = $mode & S_IWOTH ? 'w' : '-';
  $s[9] = $mode & S_IXOTH ? 'x' : '-';

  // Ausführungsbuchstaben für set-uid, set-gid und sticky anpassen.
  if ($mode & S_ISUID) {
    if ($s[3] != 'x') {
      // set-uid, aber vom Eigentümer nicht ausführbar.
      $s[3] = 'S';
    } else {
      $s[3] = 's';
    }
  }

  if ($mode & S_ISGID) {
    if ($s[6] != 'x') {
      // set-gid, aber von der Gruppe nicht ausführbar.
      $s[6] = 'S';
    } else {
      $s[6] = 's';
    }
  }

  if ($mode & S_ISVTX) {
    if ($s[9] != 'x') {
      // sticky, aber von Dritten nicht ausführbar
      $s[9] = 'T';
    } else {
      $s[9] = 't';
    }
  }

  // Formatierten String zurückgeben.
  return join('',$s);

}

// Wenn nicht anders angegeben, im Dokumenten-Stammverzeichnis anfangen.
if (isset($_REQUEST['dir'])) {
```

Beispiel 22-8: web-ls.php (Fortsetzung)

```
    $dir = $_REQUEST['dir'];
} else {
    $dir = '';
}

// $dir im Dateisystem finden.
$real_dir = realpath($_SERVER['DOCUMENT_ROOT'].$dir);

// Sicherstellen, dass sich $real_dir innerhalb des Dokumenten-Stammverzeichnisses
// befindet.
if (! preg_match('/^'.preg_quote($_SERVER['DOCUMENT_ROOT'],'/').'/',
                 $real_dir)) {
    die("$dir ist nicht im Dokumenten-Stammverzeichnis");
}

// $dir kanonisieren, indem Sie das Dokumenten-Stammverzeichnis am Anfang entfernen.
$dir = substr_replace($real_dir,'',0,strlen($_SERVER['DOCUMENT_ROOT']));

// Wird ein Verzeichnis geöffnet?
if (! is_dir($real_dir)) {
    die("$real_dir ist kein Verzeichnis");
}

// Das angegebene Verzeichnis öffnen.
$d = dir($real_dir) or die("Kann $real_dir nicht öffnen: $php_errormsg");

print '<table>';

// Jeden Verzeichniseintrag lesen (Alternativstrategie zu DirectoryIterator).
while (false !== ($f = $d->read())) {

    // Informationen über diese Datei einholen.
    $s = lstat($d->path.'/'.$f);

    // uid in Benutzernamen übersetzen.
    $user_info = posix_getpwuid($s['uid']);

    // gid in Gruppennamen übersetzen.
    $group_info = posix_getgrgid($s['gid']);

    // Datum formatieren, damit es lesbar wird.
    $date = strftime('%b %e %H:%M',$s['mtime']);

    // Oktal-Modus in einen lesbaren String übersetzen.
    $mode = mode_string($s['mode']);

    $mode_type = substr($mode,0,1);
    if (($mode_type == 'c') || ($mode_type == 'b')) {
        /* Bei Block- oder Character-Devices, Major- und
         * Minor-Device-Typ ausgeben, anstatt der Dateigröße. */
        $major = ($s['rdev'] >> 8) & 0xff;
```

Beispiel 22-8: web-ls.php (Fortsetzung)

```
        $minor = $s['rdev'] & 0xff;
        $size = sprintf('%3u, %3u',$major,$minor);
    } else {
        $size = $s['size'];
    }

    // Den <a href="">-Link um den Dateinamen formatieren,
    // keinen Link für das aktuelle Verzeichnis.
    if ('.' == $f) {
        $href = $f;
    } else {
        // Das ".." aus dem Link für das übergeordnete Verzeichnis heraushalten.
        if ('..' == $f) {
            $href = urlencode(dirname($dir));
        } else {
            $href = urlencode($dir) . '/' . urlencode($f);
        }

        /* Alles außer "/" sollte URL-codiert werden. */
        $href = str_replace('%2F','/',$href);

        // Andere Verzeichnisse mit web-ls auflisten.
        if (is_dir(realpath($d->path . '/' . $f))) {
            $href = sprintf('<a href="%s?dir=%s">%s</a>',
                            $_SERVER['PHP_SELF'],$href,$f);
        } else {
            // Link zu Dateien zwecks Download.
            $href= sprintf('<a href="%s">%s</a>',$href,$f);
        }

        // Wenn es sich um einen Link handelt, Linkziel ebenfalls anzeigen.
        if ('l' == $mode_type) {
            $href .= ' -&gt; ' . readlink($d->path.'/'.$f);
        }
    }

    // Die entsprechenden Informationen für diese Datei ausgeben.
    printf('<tr><td>%s</td><td>%3u</td><td align="right">%s</td>
            <td align="right">%s</td><td align="right">%s</td>
            <td align="right">%s</td><td>%s</td></tr>',
           $mode,               // formatierter Modus-String
           $s['nlink'],         // Anzahl der Links auf diese Datei
           $user_info['name'],  // Benutzername des Eigentümers
           $group_info['name'], // Gruppenname
           $size,               // Dateigröße (oder Device-Nummern)
           $date,               // Datum und Zeit der letzten Änderung
           $href);              // Link zum Auflisten oder zum Download
}

print '</table>';
```

22.14 Programm: Site-Suche

Sie können, wie in Beispiel 22-9 dargestellt, *site-search.php* als Suchmaschine für eine auf Dateien basierende Website von kleiner bis mittlerer Größe verwenden.

Das Programm sucht in allen Dateien innerhalb einer ausgewählten Gruppe von Verzeichnissen unter dem Dokumenten-Stammverzeichnis nach einem Suchbegriff (in `$_REQUEST['term']`). Diese Verzeichnisse liegen in `$search_dirs`. Es rekursiert ebenfalls in Unterverzeichnisse hinein und folgt symbolischen Links, aber merkt sich, welche Dateien und Verzeichnisse es bereits gesehen hat, und kann damit nicht in eine Endlosschleife geraten.

Wenn Seiten gefunden werden, die den Suchbegriff enthalten, druckt es eine Liste mit Links zu diesen Seiten aus, und zwar alphabetisch nach dem Titel jeder Seite geordnet. Wenn eine Seite keinen Titel hat (zwischen den Tags `<title>` und `</title>`), wird der URI der Seite relativ zum Dokumenten-Stammverzeichnis verwendet.

Das Programm sucht in jeder Datei nach dem Suchbegriff zwischen den Tags `<body>` und `</body>`. Wenn Sie auf Ihrer Seite viel Text zwischen den `<body>`-Tags haben, den sie von der Suche ausschließen wollen, können Sie den Text, der durchsucht werden soll, mit speziellen HTML-Kommentaren umgeben und dann `$body_regex` so ändern, dass das Programm stattdessen nach diesen Kommentar-Tags sucht. Nehmen wir zum Beispiel an, Ihre Seite sieht wie folgt aus:

```
<body>

// Ein bisschen HTML für Menüs, Kopfzeilen usw.

<!-- starte-suche -->

<h1>Per Anhalter durch die Galaxis</h1>

<h3>von Douglas Adams</h3>

<p>Die Poesie der Vogonen ist im ganzen Universum gefürchtet.</p>

// Mehr von dieser Geschichte

<!-- suche-beenden -->

// Ein bisschen HTML für Fußzeilen usw.

</body>
```

Um die Suche auf den Titel, den Autor und die Story innerhalb der HTML-Kommentare zu beschränken, ändern Sie `$body_regex` zu:

```
$body_regex = '#<!-- starte-suche -->(.*' . preg_quote($_REQUEST['term'],'#').
              '.*)<!-- suche-beenden -->#Sis';
```

Wenn Sie die Suche auf Text außerhalb von HTML- oder PHP-Tags in Ihrer Seite beschränken wollen, fügen Sie einen Aufruf an `strip_tags()` an den Code an, der den Dateiinhalt für die Suche lädt:

```
// Den Dateiinhalt in $file laden.
$file = strip_tags(join('',file($path)));
```

Beispiel 22-9: site-search.php

```
function pc_search_dir($dir) {
    global $body_regex,$title_regex,$seen;

    // Array zum Speichern von passenden Seiten.
    $pages = array();

    // Array zum Speichern von Verzeichnissen, die zu durchsuchen sind.
    $dirs = array();

    // Dieses Verzeichnis als gesehen markieren, sodass nicht noch mal
    // hineingeschaut wird.
    $seen[realpath($dir)] = true;

    // Falls Sie ein Verzeichnis-Handle für dieses Verzeichnis bekommen können,
    if (is_readable($dir) && ($d = dir($dir))) {
        // jeden Dateinamen im Verzeichnis herunterladen.
        while (false !== ($f = $d->read())) {
            // den gesamten Pfad der Datei bauen.
            $path = $d->path.'/'.$f;
            // falls es eine reguläre Datei ist und wir sie lesen können.
            if (is_file($path) && is_readable($path)) {

                $realpath = realpath($path);
                // Falls Sie diese Datei schon mal gesehen haben,
                if ($seen[$realpath]) {
                    // überspringen Sie sie,
                    continue;
                } else {
                    // ansonsten markieren Sie sie als gesehen, sodass sie
                    // übersprungen werden kann, wenn Sie wieder auf sie treffen.
                    $seen[$realpath] = true;
                }

                // Den Inhalt der Datei nach $file laden.
                $file = join('',file($path));

                // Wenn der Suchbefehl sich innerhalb der Body-Begrenzungszeichen
                // befindet,
                if (preg_match($body_regex,$file)) {

                    // konstruiere den relativen URI der Datei durch
                    // Löschen des Stammverzeichnisses aus dem vollen Pfad.
                    $uri = substr_replace($path,'',0,strlen($_SERVER['DOCUMENT_ROOT']));
```

Beispiel 22-9: site-search.php (Fortsetzung)

```
                    // Wenn die Seite einen Titel hat, finde sie
                    if (preg_match('#<title>(.*?)</title>#Sis',$file,$match)) {
                        // und füge Titel und URI an $pages an,
                        array_push($pages,array($uri,$match[1]));
                    } else {
                        // ansonsten den URI als Titel verwenden.
                        array_push($pages,array($uri,$uri));
                    }
                }
            } else {
                // Wenn die Verzeichniseingabe ein gültiges Unterverzeichnis ist,
                if (is_dir($path) && ('.' != $f) && ('..' != $f)) {
                    // zur Liste der zu durchsuchenden Verzeichnisse hinzufügen.
                    array_push($dirs,$path);
                }
            }
        }
        $d->close();
    }

    /* Jede Datei in jedem Unterverzeichnis dieses (Unter-)Verzeichnisses durchsuchen
       und die passenden Seiten in diesen Verzeichnissen zu $pages hinzufügen. Ein
       Unterverzeichnis nur durchsuchen, wenn es noch nicht gesehen wurde.
    */
    foreach ($dirs as $subdir) {
        $realdir = realpath($subdir);
        if (! $seen[$realdir]) {
            $seen[$realdir] = true;
            $pages = array_merge($pages,pc_search_dir($subdir));
        }
    }

    return $pages;
}

// Helferfunktion, die passende Seiten alphabetisch nach Titel sortiert.
function pc_page_sort($a,$b) {
    if ($a[1] == $b[1]) {
        return strcmp($a[0],$b[0]);
    } else {
        return ($a[1] > $b[1]);
    }
}

// Array zum Speichern der Seiten, die den Suchbegriff enthalten.
$matching_pages = array();
// Array zum Speichern der Seiten, die während des Scannens
// nach dem Suchbegriff gefunden wurden.
$seen = array();
// Verzeichnisse unter dem Stammverzeichnis, die durchsucht werden sollen.
$search_dirs = array('sport','kino','essen');
```

Beispiel 22-9: site-search.php (Fortsetzung)

```
// Regulärer Ausdruck zur Suche in Dateien. Der "S"-Muster-Modifier
// weist die PCRE-Engine an, den regulären Ausdruck zu "studieren",
// um effizienter zu werden.
$body_regex = '#<body>(.*' . preg_quote($_REQUEST['term'],'#').
              '.*)</body>#Sis';

// Die Dateien, die im jeweiligen Verzeichnis gefunden wurden, zu
// $matching_pages hinzufügen.
foreach ($search_dirs as $dir) {
    $matching_pages = array_merge($matching_pages,
                                  pc_search_dir($_SERVER['DOCUMENT_ROOT'].'/'.$dir));
}

if (count($matching_pages)) {
    // Die passenden Seiten nach Titel sortieren.
    usort($matching_pages,'pc_page_sort');
    print '<ul>';
    // Jeden Titel mit einem Link auf die Seite ausdrucken.
    foreach ($matching_pages as $k => $v) {
        print sprintf('<li> <a href="%s">%s</a>',$v[0],$v[1]);
    }
    print '</ul>';
} else {
    print 'Keine Seiten gefunden';
}
```

KAPITEL 23

PHP auf der Kommandozeile

23.0 Einführung

PHP wurde für die Webprogrammierung entwickelt und wird noch immer hauptsächlich für diesen Zweck verwendet. Allerdings sind neuere PHP-Versionen immer mehr dazu im Stande, als generelle Skriptsprache verwendet zu werden. Der Gebrauch von PHP für Skripten, die Sie über die Kommandozeile laufen lassen, ist besonders hilfreich, wenn diese Code verwenden, den sie mit Ihrer Webanwendung teilen. Wenn Sie ein Diskussionsforum auf Ihrer Website haben, möchten Sie vielleicht alle paar Minuten oder Stunden ein Programm laufen lassen, das neue Postings scannt und Sie auf neue Nachrichten hinweist, die bestimmte Schlüsselwörter enthalten. Wenn Sie dieses Scannerprogramm in PHP schreiben, können Sie den zugehörigen Diskussionsforums-Code zusammen mit der eigentlichen entsprechenden Anwendung erstellen. Das spart Ihnen nicht nur Zeit, sondern hilft auch, den späteren Instandhaltungsaufwand klein zu halten.

Standardmäßig wird bei der Installation von PHP 5 nicht nur ein Modul für den Webserver installiert, sondern ein Binary, um Shell-Skripten auszuführen, das so genannte PHP-CLI (*Command Line Interface*). Um ein Skript laufen zu lassen, übergeben Sie den Dateinamen des Skripts als Kommandozeilen-Parameter:

```
% php scan-forum.php
```

Unter Unix können Sie auch die »hash-bang«-Syntax in der ersten Zeile Ihrer Skripten verwenden, um den PHP-Interpreter automatisch zu starten. Steht das PHP-Binary unter */usr/local/bin*, schreiben Sie als erste Zeile Ihres Skripts:

```
#!/usr/local/bin/php
```

Sie können dann das Skript einfach durch Eintippen seines Namens in der Kommandozeile ausführen, vorausgesetzt, Sie haben Ausführungsberechtigung für die Datei.

Möchten Sie Windows so konfigurieren, dass PHP-Skripten automatisch über einen Doppelklick ausgeführt werden, gehen Sie folgendermaßen vor:

1. Öffnen Sie den Windows-Explorer
2. Wählen Sie im Menü *Extras* den Punkt *Ordneroptionen*.
3. Wechseln Sie zum Register *Dateitypen*.
4. Suchen Sie in der Liste nach dem Dateityp *PHP* oder erstellen Sie einen neuen Dateityp durch Klick auf *Neu* und Eingabe der Dateierweiterung **`.php`**.
5. Haben Sie diesen Eintrag ausgewählt, klicken Sie auf die Schaltfläche *Erweitert*.
6. Wählen Sie bei den Aktionen *open* aus und klicken Sie auf *Bearbeiten*.
7. Geben Sie bei *Anwendung* den Text **`"C:\php5\cli\php.exe "%1" %*"`** ein und ersetzen Sie dabei gegebenenfalls den Pfad durch den auf Ihrem System gültigen Pfad.
8. Klicken Sie auf die Schaltfläche *OK*.

Sie können nun PHP-Skripten durch einen Doppelklick auf das Symbol des Skripts ausführen.

Das CLI-Binary hat Ähnlichkeiten mit dem CGI-Binary, weist aber einige wichtige Unterschiede auf, durch die es Shell-freundlicher wird. Einige Konfigurationsanweisungen haben unter CLI fest codierte Werte. So ist zum Beispiel die Direktive `html_errors` auf `false` gesetzt und `implicit_flush` auf `true`. Die Direktive `max_execution_time` ist auf 0 gesetzt und erlaubt damit eine unbegrenzte Laufzeit. Schließlich ist `register_argc_argv` auf `true` gesetzt. Damit können Sie Kommandozeilen-Informationen unter `$argv` und `$argc` suchen, anstatt auf `$_SERVER['argv']` und `$_SERVER['argc']` zurückgreifen zu müssen. Die Verarbeitung von Parametern wird in den Rezepten 23.1 und 23.2 behandelt.

Das CLI-Binary akzeptiert etwas andere Argumente als das CGI-Binary. Es akzeptiert keine `-q`- oder `-C`-Flags, da es die Funktion der Flags sowieso standardmäßig implementiert: Während `-q` das CGI-Binary anweist, keine Kopfzeilen auszudrucken, druckt die CLI-Binary nie Kopfzeilen aus. Sogar die Funktion `header( )` hat unter CLI keinen Effekt. Weiterhin weist das `-C`-Flag das CGI-Binary an, nicht in das Verzeichnis des laufenden Skripts zu wechseln. Das CLI-Binary wechselt nie ins Skriptverzeichnis.

Außerdem nimmt das CLI-Binary einige weitere Argumente an, mit denen PHP-Code direkt ausgeführt werden kann, ohne dass dieser zuerst in einer Datei abgespeichert werden muss.

Schließlich definiert das CLI-Binary noch Handles für die standardmäßigen I/O-Streams als die Konstanten `STDIN`, `STDOUT` und `STDERR`. Sie können sie verwenden, statt Ihre eigenen Datei-Handles mit `fopen( )` zu erzeugen:

```
// Von der Standardeingabe einlesen.
$input = fgets(STDIN,1024);

// Zur Standardausgabe schreiben.
fwrite(STDOUT,$jokebook);

// Zum Standardfehlerkanal schreiben.
fwrite(STDERR,$error_code);
```

Wenn Sie das CLI-Binary verwenden, verwenden Sie `php_sapi_name()`, um zu testen, ob ein Skript im Kontext einer Webanwendung oder einer Kommandozeile läuft:

```
if ('cli' == php_sapi_name()) {
  print "Datenbankfehler: ".mysql_error()."\n";
} else {
  print "Datenbankfehler.<br>";
  error_log(mysql_error());
}
```

Dadurch können Sie z.B. Fehlermeldungen anpassen oder unterschiedliche Logdateien schreiben.

Ein weiterer Vorteil ist, dass das CLI-Binary automatisch nach einer Konfigurationsdatei mit dem Namen *php-cli.ini* sucht, wodurch Sie für Ihren Webserver eine andere Konfiguration verwenden können als für die Skripten, die Sie auf der Kommandozeile ausführen.

Einige Erweiterungen werden standardmäßig nur in der CLI-Version installiert, wie z.B. die PCNTL-Erweiterung, die Thema der Rezepte 23.7 und 23.8 ist.

23.1 Programmparameter parsen

Problem

Sie wollen Parameter verarbeiten, die in der Kommandozeile übergeben wurden.

Lösung

`$_SERVER['argc']` gibt Ihnen die Anzahl der Argumente und `$_SERVER['argv']` ihre Werte. Das erste Argument, `$_SERVER['argv'][0]`, ist der Name des Skripts, das gerade läuft:

```
if ($_SERVER['argc'] != 2) {
    die("Falsche Parameteranzahl: Ich erwarte nur einen Parameter.");
}

$size = filesize($_SERVER['argv'][1]);

print "Ich bin $_SERVER[argv][0] und kann berichten, dass die Größe von ";
print "$_SERVER[argv][1] $size Bytes beträgt.";
```

Diskussion

Um Optionen anhand von durch die Kommandozeile übergebenen Flags auszuwählen, durchlaufen Sie `$_SERVER['argv']` in einer Schleife von `1` bis `$_SERVER['argc']`:

```
for ($i = 1; $i < $_SERVER['argc']; $i++) {
    switch ($_SERVER['argv'][$i]) {
    case '-v':
```

```
            // Ein Flag setzen.
            $verbose = 1;
            break;
        case '-c':
            // Zum nächsten Parameter weitergehen.
            $i++;
            // Wenn er festgelegt ist, den Wert speichern.
            if (isset($_SERVER['argv'][$i])) {
                $config_file = $_SERVER['argv'][$i];
            } else {
                // Beenden, wenn kein Dateiname angegeben ist.
                die("Nach -c muss ein Dateiname angegeben sein.");
            }
            break;
        case '-q':
            $quiet = 1;
            break;
        default:
            die('Unbekannter Parameter: '.$_SERVER['argv'][$i]);
            break;
        }
    }
```

In diesem Beispiel sind die `-v`- und `-q`-Parameter Flags, die `$verbose` und `$quiet` festlegen, aber es wird erwartet, dass auf das `-c`-Argument ein String folgt. Dieser String wird `$config_file` zugewiesen.

Siehe auch

Rezept 23.2 für weiteres Argument-Parsing mit `Console_Getopt` und `Console_Getargs`; die Dokumentationen zu `$_SERVER['argc']` und `$_SERVER['argv']` unter *http://www.php.net/reserved.variables*.

23.2 Programmparameter mit Console_Getopt oder Console_Getargs parsen

Problem

Sie wollen Programmoptionen parsen, die vielleicht als kurze oder lange Optionen angegeben oder die gruppiert sind.

Lösung

Verwenden Sie die Klasse `Console_Getopt` von PEAR. Deren `getopt()`-Methode kann sowohl Optionen im kurzen Stil parsen, wie z.B. `-a` oder `-b`, als auch im langen Stil, wie z.B. `--alice` oder `--bob`:

```
$o = new Console_Getopt;

// Akzeptiert -a, -b und -c.
$opts = $o->getopt($_SERVER['argv'],'abc');

// Akzeptiert --alice und --bob.
$opts = $o->getopt($_SERVER['argv'],'',array('alice','bob'));
```

PEAR bietet außerdem die Methode `Console_Getargs`, die mehr Funktionen bereitstellt, um dasselbe Problem zu lösen.

Diskussion

Um Optionen im kurzen Stil zu parsen, übergeben Sie das Array von Kommandozeilen-Argumenten und einen String, der die gültigen Optionen angibt, an `Console_Getopt::getopt()`. Das folgende Beispiel akzeptiert `-a`, `-b` oder `-c` als Argument, und zwar einzeln oder in Gruppen:

```
$o = new Console_Getopt;
$opts = $o->getopt($_SERVER['argv'],'abc');
```

Für den angegebenen Options-String `abc` können zum Beispiel die folgenden gültigen Optionsgruppen weitergegeben werden:

```
% programm.php -a -b -c
% programm.php -abc
% programm.php -ab -c
```

Die Methode `getopt()` gibt ein Array zurück. Das erste Element des Arrays ist eine Auflistung aller geparsten Optionen, die in der Kommandozeile angegeben wurden, zusammen mit ihren Werten. Das zweite Element enthält alle angegebenen Optionen in der Kommandozeile, die nicht in der Argumentenliste waren, die an `getopt()` übergeben wurde. Wird das vorhergehende Programm zum Beispiel so aufgerufen:

```
% program.php -a -b sneeze
```

dann ist `$opts`:

```
Array
(
    [0] => Array
        (
            [0] => Array
                (
                    [0] => a
                    [1] =>
                )
            [1] => Array
                (
                    [0] => b
                    [1] =>
                )
```

```
        )
    [1] => Array
        (
            [0] => program.php
            [1] => sneeze
        )
)
```

Wenn Sie nach einer Option im Spezifikations-String einen Doppelpunkt setzen, zeigen Sie an, dass die Option einen Wert benötigt. Zwei Doppelpunkte bedeuten, dass der Wert optional ist. Also bedeutet `ab:c::`, dass `a` keinen Wert haben darf, `b` einen Wert haben muss und `c` einen Wert haben kann, wenn dieser angegeben ist. Läuft das Programm mit diesem Spezifikations-String als:

```
% program.php -a -b sneeze
```

dann ist `$opts`:

```
Array
(
    [0] => Array
        (
            [0] => Array
                (
                    [0] => a
                    [1] =>
                )
            [1] => Array
                (
                    [0] => b
                    [1] => sneeze
                )
        )
    [1] => Array
        (
            [0] => program.php
        )
)
```

Da `sneeze` jetzt als Wert von `b` festgelegt ist, ist es nicht mehr im Array der ungeparsten Optionen. Beachten Sie, dass das Array der ungeparsten Optionen immer den Programmnamen enthält.

Um Parameter im langen Stil zu parsen, geben Sie `getopt()` ein Array, das Ihre gewünschten Argumente beschreibt. Fügen Sie jedes Argument in ein Array-Element ein (ohne `--` am Anfang) und hängen Sie `=` an, um anzuzeigen, dass es einen Wert benötigt, oder `==`, um anzuzeigen, dass es einen optionalen Wert annimmt. Dieses Array ist das dritte Argument für `getopt()`. Das zweite Argument (der String für die Argumente im kurzen Stil) kann leer gelassen werden oder auch nicht, je nachdem, ob Sie zusätzlich Argumente im kurzen Stil parsen wollen. Dieses Beispiel braucht `debug` als Argument ohne Wert, `name` zwingend mit einem Wert und `size` optional mit einem Wert:

```
require 'Console/Getopt.php';
$o = new Console_Getopt;
$opts = $o->getopt($_SERVER['argv'],'',array('debug','name=','size=='));
```

Die folgenden Programmaufrufe sind somit gültig:

```
% program.php --debug
% program.php --name=Susannah
% program.php --name Susannah
% program.php --debug --size
% program.php --size=56 --name=Susannah
% program.php --name --debug
```

Das letzte Beispiel ist gültig (wenn auch kontraproduktiv), da es `--debug` als Wert des `name`-Parameters behandelt und somit den `debug`-Parameter als nicht gesetzt betrachtet. Werte werden von den Parametern entweder durch ein `=` oder ein Leerzeichen getrennt.

Bei Argumenten im langen Stil schließt `getopt()` die anführenden `--` im Array der geparsten Parameter ein. Wenn Sie beispielsweise

```
% program.php --debug --name=Susannah
```

aufrufen, wird `$opts` wie folgt gesetzt:

```
Array
(
    [0] => Array
        (
            [0] => Array
                (
                    [0] => --debug
                    [1] =>
                )
            [1] => Array
                (
                    [0] => --name
                    [1] => Susannah
                )
        )
    [1] => Array
        (
            [0] => program.php
        )
)
```

Bisher wurde `$_SERVER['argv']` als das Array der Kommandozeilen-Argumente verwendet, was als Standardeinstellung okay ist. `Console_Getopt` stellt eine Methode namens `readPHPArgv()` zur Verfügung, mit der Sie auch in `$argv` und `$HTTP_SERVER_VARS['argv']` nach Kommandozeilen-Argumenten suchen können. Sie verwenden sie, indem Sie ihre Ergebnisse an `getopt()` übergeben:

```
require 'Console/Getopt.php';
$o = new Console_Getopt;
$opts = $o->getopt($o->readPHPArgv(),'',array('debug','name=','size=='));
```

Sowohl `getopt()` als auch `readPHPArgv()` geben ein `Getopt_Error`-Objekt zurück, wenn sie einen Fehler entdecken: wenn beispielsweise für eine Option, die einen Wert braucht, keiner angegeben ist. `Getopt_Error` erweitert die Basisklasse `PEAR_Error`, so dass Sie bereits bekannte Methoden zur Fehlerbehebung einsetzen können:

```
require 'Console/Getopt.php';
$o = new Console_Getopt;
$opts = $o->getopt($o->readPHPArgv(),'',array('debug','name=','size=='));

if (PEAR::isError($opts)) {
    print $opts->getMessage();
} else {
    // Optionen verarbeiten.
}
```

Neben `Console_Getopt` bietet PEAR auch das Paket `Console_Getargs`, das dasselbe Problem löst, jedoch komplett objektorientiert arbeitet und weitaus mehr Funktionen zur Verfügung stellt. Das gleiche Beispiel sieht unter Verwendung von `Console_Getargs` folgendermaßen aus:

```
require_once 'Console/Getargs.php';
$config = array(
            'debug' => array(
                            'short'   => 'd',
                            'max'     => 0,
                            'desc'    => 'Debug-Modus aktivieren.'
                            ),
            'name'  => array(
                            'short'   => 'n',
                            'min'     => 1,
                            'max'     => 1,
                            'desc'    => 'Name festlegen.'
                            ),
            'size' => array(
                            'short'   => 's',
                            'min'     => 0,
                            'max'     => 1,
                            'desc'    => 'Groesse festlegen',
                            'default' => 'small'
                            )
            );

$args = Console_Getargs::factory($config);
if (PEAR::isError($args)) {
    $header = "Beispiel für Parameterübergabe\n".
              'Mögliche Optionen:\n";

    if ($args->getCode() === CONSOLE_GETARGS_ERROR_USER) {
        print Console_Getargs::getHelp($config, $header, $args->getMessage())."\n";
    } else if ($args->getCode() === CONSOLE_GETARGS_HELP) {
        print Console_Getargs::getHelp($config, $header)."\n";
```

```
    }
    exit();
}

// Debug-Modus
if ($args->isDefined('debug')) {
    print "Debug-Modus ist aktiviert.\n";
}

// Name
if ($args->isDefined('name')) {
    printf("Name ist %s\n", $args->getValue('name'));
}

// Size
if ($args->isDefined('size)) {
    printf("Größe ist %s\n", $args->getValue('size'));
}
```

Zuerst definieren Sie ein Array, das die möglichen Optionen der Applikation definiert. Im Gegensatz zu `Console_Getopt` bieten sich hier mehr Möglichkeiten. Jede Option wird über einen eindeutigen Schlüssel (z.B. *debug*, *name*, *size*) im Array identifiziert. Unter diesem Namen wird die Option dann später auch von der Kommandozeile gelesen. Für jede dieser Optionen wird nun ein Array mit den folgenden Werten definiert:

- `short` wird verwendet, um die Abkürzung der Option zu definieren.
- `min` gibt an, wie viele Parameter dieser Option mindestens übergeben werden müssen.
- `max` gibt an, wie viele Parameter dieser Option maximal übergeben werden dürfen.
- `default` gibt einen Default-Wert an, wenn der Option kein Parameter übergeben wurde (falls `min` auf `0` gesetzt wurde).
- `desc` beinhaltet eine kurze Beschreibung der Option.

Um nun die Parameter zu parsen, wird diese Konfiguration der Fabrik-Methode `Console_Getargs::factory()` übergeben:

```
$args = Console_Getargs::factory($config);
```

Sind alle Parameter korrekt übergeben, liefert die Methode ein `Console_Getargs_Options`-Objekt zurück, ansonsten ein `PEAR_Error`-Objekt. Im Fehlerfall wird über die Methode `getCode()` des Fehler-Objekts zwischen zwei Fällen unterschieden:

```
if ($args->getCode() === CONSOLE_GETARGS_ERROR_USER) {
    print Console_Getargs::getHelp($config, $header, $args->getMessage())."\n";
} else if ($args->getCode() === CONSOLE_GETARGS_HELP) {
    print Console_Getargs::getHelp($config, $header)."\n";
}
```

Ist der Fehlercode `CONSOLE_GETARGS_ERROR_USER`, hat der Benutzer die Optionen nicht wie gefordert übergeben. Ist er jedoch `CONSOLE_GETARGS_HELP`, hat der Benutzer die Pro-

grammhilfe angefordert. Dies ist möglich, da Console_Getargs automatisch eine weitere Option --help zu den definierten Optionen hinzufügt. Console_Getargs ist weiterhin in der Lage, eine kurze Hilfe zu den Optionen auszugeben. Dazu bietet die Klasse die statische Methode getHelp(), der lediglich die gleiche Konfiguration übergeben werden muss wie der Fabrik-Methode. Als zweiter Parameter kann noch ein String übergeben werden, der über der Hilfe angezeigt wird. Rufen Sie das Skript also mit der Option --help oder -h auf, erhalten Sie die folgende Rückgabe, die Sie nur noch ausgeben müssen:

```
Beispiel für Parameterübergabe
Mögliche Optionen:
-d --debug                 Debug-Modus aktivieren.
-n --name=<value>          Name festlegen.
-s --size (optional)value  Groesse festlegen (small).
```

Aus diesem Grund wurde zuvor auch für jede Option eine kurze Beschreibung festgelegt.

Hat der Benutzer einen tatsächlichen Fehler bei der Eingabe gemacht, können Sie eine Beschreibung des Fehlers mit der Methode getMessage() des Fehler-Objekts erhalten. Übergeben Sie diese Meldung der Methode Console_Getargs::getHelp() als dritten Parameter, wird diese unterhalb der Hilfe dargestellt. Nach der Ausgabe des Hilfetexts wird das Skript durch exit() beendet.

Tritt kein Fehler auf, können Sie über die Methode isDefined() überprüfen, ob eine Option durch den Benutzer gesetzt wurde, Sie sparen sich also das Durchsuchen eines Arrays, wie es bei Console_Getopt nötig ist:

```
if ($args->isDefined('debug')) {
    print "Debug-Modus ist aktiviert.\n";
}
```

Handelt es sich um eine Option, der Parameter übergeben werden können, ermitteln Sie den Wert mit der Methode getValue():

```
if ($args->isDefined('name')) {
    printf("Name ist %s\n", $args->getValue('name'));
}
```

Hat der Benutzer mehr als einen Parameter übergeben, erhalten Sie automatisch ein indiziertes Array mit allen Parametern zurück. Wenn der Benutzer jedoch keinen Parameter übergeben hat, erhalten Sie den Default-Wert zurück, sofern dieser in der Konfiguration der Option definiert wurde.

Console_Getargs besteht zwar aus fünfmal so vielen Codezeilen wie Console_Getopt und ist dementsprechend langsamer beim Einbinden in Ihr Skript, jedoch lohnt sich der Einsatz in nahezu allen Fällen, da der Code der eigentlichen Applikation stark verringert werden kann. Man muss sich als Entwickler weder um das Erzeugen einer Programmhilfe noch um das Analysieren des zurückgegebenen Arrays kümmern.

Siehe auch

Rezept 23.1 zum Parsen von Programmoptionen ohne `Console_Getopt` oder `Console_Getargs`; die Dokumentation zu `Console_Getopt` unter *http://pear.php.net/manual/de/package.console.console-getopt.php*; die `Console_Getargs`-Homepage unter *http://pear.php.net/package/Console_Getargs*.

23.3 Von der Tastatur lesen

Problem

Sie müssen eine getippte Benutzereingabe einlesen.

Lösung

Verwenden Sie `fopen( )` mit dem Spezialdateinamen *php://stdin*:

```
print "Tippen Sie Ihre Nachricht. Schreiben Sie '.' in einer neuen Zeile,
                                          wenn Sie fertig sind.\n";

$fh = fopen('php://stdin','r') or die($php_errormsg);
$last_line = false;  $message = '';
while (! $last_line) {
    $next_line = fgets($fh,1024);
    if (".\n" == $next_line) {
      $last_line = true;
    } else {
      $message .= $next_line;
    }
}

print "\nIhre Nachricht ist:\n$message\n";
```

Wenn die Readline-Erweiterung installiert ist, verwenden Sie `readline( )`:

```
$last_line = false; $message = '';
while (! $last_line) {
    $next_line = readline();
    if ('.' == $next_line) {
        $last_line = true;
    } else {
        $message .= $next_line."\n";
    }
}

print "\nIhre Nachricht ist:\n$message\n";
```

Diskussion

Sobald Sie mit `fopen()` ein Datei-Handle haben, das auf *stdin* zeigt, können Sie alle standardmäßigen Funktionen zum Lesen von Dateien verwenden, um Eingaben zu verarbeiten (`fread()`, `fgets()` usw.). Die Lösung verwendet `fgets()`, die die Eingabe zeilenweise zurückgibt. Wenn Sie `fread()` verwenden, muss die Eingabe trotzdem mit einer neuen Zeile beendet werden, da `fread()` sonst nicht beendet wird. Führen Sie beispielsweise den folgenden Code aus:

```
$fh = fopen('php://stdin','r') or die($php_errormsg);
$msg = fread($fh,4);
print "[$msg]";
```

und tippen `Tomate` ein, gefolgt von einem Zeilenumbruch, dann ist die Ausgabe `[Toma]`. `fread()` nimmt wie angewiesen nur vier Zeichen aus *stdin*, braucht aber trotzdem den Zeilenumbruch als Signal, dass es nicht länger auf eine weitere Tastatureingabe zu warten braucht.

Wenn Sie das CLI-Binary verwenden, ist es nicht nötig, ein Datei-Handle mit `fopen()` zu erzeugen, Sie können stattdessen die Konstante `STDIN` verwenden:

```
$msg = fread(STDIN,4);
print "[$msg]";
```

Die Readline-Erweiterung stellt eine Schnittstelle zur GNU-Readline-Bibliothek zur Verfügung. Die Funktion `readline()` gibt Zeile für Zeile zurück, ohne den Zeilenumbruch am Ende. Mit Readline können Benutzer Zeilen so wie in Emacs und *vi* bearbeiten. Sie können mit der Funktion auch eine History-Liste mit bisher eingegebenen Befehlen führen:

```
$command_count = 1;
while (true) {
    $line = readline("[$command_count]--> ");
    readline_add_history($line);
    if (is_readable($line)) {
        print "$line ist eine lesbare Datei.\n";
    }
    $command_count++;
}
```

Dieses Beispiel zeigt einen Prompt mit einer ansteigenden Zahl vor jeder Zeile. Da jede Zeile mit `readline_add_history()` zur Readline-History hinzugefügt wird, können Sie sich mit Hilfe der Pfeiltasten nach oben und unten durch die vorher eingegebenen Zeilen bewegen.

Siehe auch

Die Dokumentationen zu `fopen()` unter *http://www.php.net/fopen*, `fgets()` unter *http://www.php.net/fgets*, `fread()` unter *http://www.php.net/fread*, PHP auf der Kommandozeile unter *http://www.php.net/features.commandline* und zur Readline-Erweiterung unter *http://*

www.php.net/readline; die Readline-Bibliothek finden Sie unter *http://cswww.cns.cwru.edu/php/chet/readline/rltop.html*.

23.4 Passwörter einlesen

Problem

Sie müssen einen String von der Kommandozeile lesen, so dass er beim Eingeben nicht wieder ausgegeben wird, beispielsweise beim Eingeben von Passwörtern.

Lösung

Unter Unix verwenden Sie */bin/stty*, um das Echo von eingetippten Zeichen ein- oder auszuschalten:

```
// Das Echo abschalten.
`/bin/stty -echo`;

// Passwort lesen.
$password = readline();

// Das Echo wieder einschalten.
`/bin/stty echo`;
```

Unter Windows verwenden Sie `w32api_register_function( )`, um `_getch( )` von *msvcrt.dll* zu importieren:

```
// Die W32api-Erweiterung laden und _getch() registrieren.
dl('php_w32api.dll');
w32api_register_function('msvcrt.dll','_getch','int');

while(true) {
    // Ein Zeichen von der Tastatur holen.
    $c = chr(_getch());
    if ( "\r" == $c ||  "\n" == $c ) {
        // Wenn es ein Newline-Zeichen ist, Schleife verlassen,
        // Sie haben das Passwort.
        break;
    } elseif ("\x08" == $c) {
        /* Wenn es ein Backspace ist, das vorherige Zeichen von $password löschen. */
        $password = substr_replace($password,'',-1,1);
    } elseif ("\x03" == $c) {
        // Wenn es sich um Control-C handelt, $password löschen und
        // aus der Schleife ausbrechen,
        $password = NULL;
        break;
    } else {
        // ansonsten das Zeichen an das Passwort anfügen.
        $password .= $c;
    }
}
```

Diskussion

Unter Unix verwenden Sie */bin/stty* zur Steuerung der Tastatureigenschaften in der Form, dass getippte Zeichen nicht wieder auf dem Bildschirm angezeigt werden, während Sie ein Passwort einlesen. Windows hat */bin/stty* nicht. Daher verwenden Sie die W32api-Erweiterung, um Zugang zu `_getch( )` in der Microsoft C-Runtime-Bibliothek *msvcrt.dll* zu erhalten. Die Funktion `_getch( )` liest ein Zeichen ein, ohne ein Echo auf dem Bildschirm anzuzeigen. Sie gibt den ASCII-Code des gelesenen Zeichens zurück, den Sie mit `chr( )` zu einem Zeichen konvertieren. Das eingetippte Zeichen entscheidet, was danach passiert. Wenn es ein Newline oder ein Carriage Return (Wagenrücklauf) ist, verlassen Sie die Schleife, da das Passwort vollständig eingegeben wurde. Ist es ein Backspace, löschen Sie das Zeichen am Ende das Passworts. Bei einer Strg-C-Unterbrechung setzen Sie das Passwort auf `NULL` und verlassen die Schleife. Wenn keine dieser Möglichkeiten zutrifft, wird das Zeichen an `$password` angehängt. Verlassen Sie dann die Schleife, enthält `$password` das eingegebene Passwort.

Der folgende Code zeigt die Prompts `Login:` und `Password:` und vergleicht das eingegebene Passwort mit dem entsprechenden verschlüsselten Passwort, das in */etc/passwd* gespeichert ist. Dies setzt voraus, dass das System keine Schattenpasswörter verwendet.

```
print "Login: ";
$fh = fopen('php://stdin','r')   or die($php_errormsg);
$username = rtrim(fgets($fh,64)) or die($php_errormsg);

preg_match('/^[a-zA-Z0-9]+$/',$username)
    or die("Ungültiger Benutzername: Nur Buchstaben und Ziffern sind erlaubt.");


print 'Password: ';
`/bin/stty -echo`;
$password = rtrim(fgets($fh,64)) or die($php_errormsg);
`/bin/stty echo`;
print "\n";

// Nichts mehr von der Tastatur einzulesen.
fclose($fh);

// Die entsprechende Zeile in /etc/passwd finden.
$fh = fopen('/etc/passwd','r')   or die($php_errormsg);
$found_user = 0;
while (! ($found_user || feof($fh))) {
    $passwd_line = fgets($fh,256);
    if (preg_match("/^$username:/",$passwd_line)) {
        $found_user = 1;
    }
}
fclose($fh);

$found_user or die ("Kann Benutzer \"$username\" nicht finden.");
```

```
// Die richtige Zeile von /etc/passwd parsen.
$passwd_parts = split(':',$passwd_line);

/* Das eingegebene Passwort verschlüsseln und mit dem Passwort
  in /etc/passwd vergleichen. */
$encrypted_password = crypt($password,
                            substr($passwd_parts[1],0,CRYPT_SALT_LENGTH));

if ($encrypted_password == $passwd_parts[1]) {
    print "Login erfolgreich";
} else {
    print "Login fehlgeschlagen";
}
```

Siehe auch

Die Dokumentationen zu `readline( )` unter *http://www.php.net/readline*, `chr( )` unter *http://www.php.net/chr*, `w32api_register_function( )` unter *http://www.php.net/w32api-register-function* und `_getch( )` unter *http://msdn.microsoft.com/de-de/library/078sfkak(VS.80).aspx*; unter Unix schauen Sie in der Manpage *stty(1)* Ihres Systems nach.

23.5 Die Ausgabe eines Kommandozeilen-Befehls weiterverarbeiten

Problem

Sie wollen die Ausgabe eines Befehls mit PHP weiterverarbeiten.

Lösung

Nutzen Sie eine Pipe (|) und die PHP-Konstanten `STDIN` und `STDOUT`.

```
#!/usr/bin/php
<?php
$zeilen = array();
while (false !== $zeile = fgets(STDIN, 1024)) {
    $zeile = trim($zeile);
    if ($zeile === '') {
        continue;
    }
    array_push($zeilen, $zeile);
}
sort($zeilen);
foreach ($zeilen as $zeile) {
    fputs(STDOUT, $zeile."\n");
}
```

Dieses Skript können Sie jetzt verwenden, um die Ausgabe eines beliebigen Befehls zu sortieren:

```
% ls -as | sort.php
```

Diskussion

Zuerst schreiben Sie ein Skript, das die Ausgabe eines Kommandozeilen-Befehls weiterverarbeiten soll. Um auf die Daten zuzugreifen, verwenden Sie die Konstante `STDIN`, genau so, als handle es sich um Benutzereingaben. Der Unterschied ist hierbei nur, dass die Eingabe nicht durch den Benutzer, sondern durch eine beliebige Anwendung erledigt wird.

Das Skript liest nun die Daten Zeile für Zeile mit der Funktion `fgets()` ein. Beim Vergleich mit `false` werden zwei Gleichheitszeichen verwendet, damit nicht schon bei einer leeren Zeile abgebrochen wird. Mit der Funktion `trim()` werden führende und abschließende Leerzeichen entfernt, und falls die Zeile dann noch Daten enthält, wird sie an das Ende des Arrays `$zeilen` angehängt.

Nachdem alle Zeilen in ein Array eingelesen wurden, wird dieses mit der Funktion `sort()` alphabetisch sortiert. Danach werden die Zeilen mit der Funktion `fputs()` wieder an den Standard-Ausgabestrom geschickt und könnten von der nächsten Anwendung weiterverarbeitet werden.

Um nun die Ausgabe eines Befehls oder Skripts zu sortieren, verwenden Sie den Pipe-Operator (|) der Shell, um die Ausgabe in das Skript umzuleiten:

```
% ls -as | sort.php
```

Sie können beliebig viele Pipes verwenden, um die sortierte Ausgabe durch weitere Befehle verarbeiten zu lassen:

```
% ls -as | sort.php | grep ".txt"
```

Dieser Befehl gibt alle Dateien im aktuellen Verzeichnis aus, sortiert die Ausgabe und filtert sie danach so, dass nur noch Zeilen ausgegeben werden, die die Zeichenkette `".txt"` beinhalten. Wenn Sie mehrere Befehle aneinander hängen, sollten Sie stets darauf achten, in welcher Reihenfolge die Befehle ausgeführt werden. So führt der folgende Aufruf zum selben Ergebnis, ist aber bei vielen Dateien wesentlich schneller:

```
% ls -as | grep ".txt" | sort.php
```

Natürlich müssen Sie darauf achten, dass Sie Ausführrechte für das Skript haben und dass es sich in dem Pfad befindet, in dem die Shell nach Befehlen sucht. Andernfalls müssen Sie den kompletten Pfad zum Skript angeben:

```
% ls -as | /pfad/zu/sort.php
```

Liegt das Skript in dem Verzeichnis, in dem Sie sich gerade befinden, genügt natürlich auch:

```
% ls -as | ./sort.php
```

Soll Ihr Skript einen Fehler signalisieren, verwenden Sie die Konstante STDERR anstatt STDOUT:

```
if (empty($zeilen)) {
    fputs(STDERR, "Keine Zeilen zum Sortieren gefunden.\n");
}
```

Ist das Array $zeilen nach dem Einlesen aller Zeilen leer, gibt das Skript eine Fehlermeldung aus:

```
% ls -al | grep "EsGibtKeineZeilen" | ./sort.php
```

Dies führt zur Ausgabe:

```
Keine Zeilen zum Sortieren gefunden.
```

Oft möchten Sie Fehlermeldungen allerdings nicht direkt ausgeben. Da Sie zur Ausgabe STDERR anstatt STDIN verwendet haben, können Sie diese Ausgaben sehr einfach umleiten:

```
% ls -al | grep "EsGibtKeineZeilen" | ./sort.php 2> fehler.txt
```

Alles, was von PHP an STDERR geschrieben wird, wird automatisch in die Datei *fehler.txt* geschrieben. Wollen Sie die Fehler einfach ignorieren, genügt es, den Aufruf folgendermaßen anzupassen:

```
% ls -al | grep "EsGibtKeineZeilen" | ./sort.php 2> /dev/null
```

Durch die Verwendung der STDERR-Konstante überlassen Sie die Fehlerbehandlung also dem Benutzer, der Ihr Skript aufruft, und geben ihm eine Möglichkeit an die Hand, Fehlermeldungen in ein Logfile zu schreiben oder einfach zu ignorieren.

Siehe auch

Rezept 23.3 zum Einlesen von der Tastatur.

23.6 Dateien zeilenweise verarbeiten

Problem

Sie wollen schnell eine Liste aller Hostnamen aus einem Apache-Access-Log ermitteln, die auf Ihre Seite zugegriffen haben.

Lösung

Verwenden Sie die neuen Schalter des PHP 5-CLI-Binary, um das Access-Log zeilenweise auszulesen, ohne dazu erst ein Skript schreiben zu müssen:

```
% cat access.log | head -200 | php5 -B '$hosts = array();' -R 'list($ip) = explode(" ",
$argn, 2); $host = gethostbyaddr($ip); if (!in_array($host, $hosts)) array_push($hosts,
$host);' -E 'echo implode("\n", $hosts);'
```

Diskussion

PHP 5 bietet drei neue Optionen, wenn man es auf der Kommandozeile ausführt, die es einem ermöglichen, die Ausgabe eines Skripts oder auch eine Datei zeilenweise zu analysieren. Dazu werden drei Codeschnipsel angegeben: eines, das bei Beginn ausgeführt wird, eines, das für jede Zeile der Ausgabe ausgeführt wird, und eines, das nach Beendigung aller Zeilen ausgeführt wird. Diese drei Codefragmente werden mit den Schaltern *-B* (Beginn), *-R* (jede Zeile) und *-E* (Ende) angegeben.

Der Code wird dabei in einfache Anführungsstriche eingeschlossen. Wollen Sie doppelte verwenden, müssen Sie darauf achten, dass Ihre Shell alle Variablen ersetzen möchte, und deshalb die Dollarzeichen escapen, also beispielsweise `\$variable` statt `$variable` schreiben.

Damit der Inhalt der Datei *access.log* von PHP zeilenweise verarbeitet werden kann, geben Sie ihn mit dem Befehl `cat` zuerst an die Standardausgabe aus und leiten diese dann über eine Pipe an PHP weiter:

```
cat access.log | php
```

Um alle Hostnamen zu speichern, wird zuerst ein leeres Array `$hosts` angelegt; da dies nur am Anfang der Datei nötig ist, wird dieser Code mit dem Schalter *-B* angegeben.

```
cat access.log | php -B '$hosts = array();'
```

Danach geben Sie den Code an, der für jede Zeile des Logfiles ausgeführt werden soll. Dazu muss natürlich das Format des Logfiles bekannt sein. Eine Zeile eines Access-Logs hat den folgenden Aufbau:

```
192.33.108.188 - - [03/Oct/2004:14:54:53 +0200] "GET /styles/main.css HTTP/1.1" 200 2354
"http://schst.net/" "X11; U; Linux i686; rv:1.7.3) Gecko/20040926 Firefox/0.10"
```

Die Zeile beginnt mit der IP-Adresse, von der die Anfrage gestartet wurde, gefolgt vom Datum und der Uhrzeit der Anfrage, der angeforderten Datei und dem Typ der Anfrage (GET), dem Response-Code des Servers (200), der Größe der ausgelieferten Datei (2.354 Bytes) und dem Hostnamen, der verwendet werden soll (für virtuelle Hosts), sowie dem verwendeten Browser des Benutzers. Um auszuwerten, welche Hosts auf Ihre Seite zugreifen, ist lediglich die IP-Adresse wichtig, von der die Anfrage gestellt wurde. Diese steht im ersten Feld und kann auf verschiedene Arten ermittelt werden. Eine einfache Lösung ist es, die Zeile beim ersten Vorkommen eines Leerzeichens zu trennen, wozu die Funktion `explode()` verwendet werden kann. Diese zerteilt einen String mit Hilfe eines Trennzeichens in ein Array. Der Funktion wird ein Trennzeichen sowie der zu zerteilende String übergeben. Optional kann der Funktion noch mitgeteilt werden, in wie viele Einzelteile der String maximal zerlegt werden soll. Da nur die IP-Adresse benötigt wird, reicht es hier, jede Zeile in zwei Teile zu zerlegen.

Bei der zeilenweisen Verarbeitung einer Datei stellt PHP die aktuelle Zeile immer in der Variablen `$argn` zur Verfügung.:

```
list($ip) = explode(" ", $argn, 2);
```

Mit der Funktion `list()` wird der erste Eintrag im entstandenen Array in der Variablen `$ip` gespeichert.

Als Nächstes wird mit der Funktion `gethostbyaddr()` der zugehörige Hostname für die IP-Adresse ermittelt. Dadurch können Sie sehr leicht ablesen, ob die Anfrage aus einem Netz der T-Com oder auch aus einem Firmennetz vorgenommen wurde:

```
$host = gethostbyaddr($ip);
```

Da anzunehmen ist, dass von jedem Host mehr als ein Request gemacht wird, überprüfen Sie nun mit der Funktion `in_array()`, ob Sie den Hostnamen bereits durch eine vorherige Anfrage im Array `$hosts` gespeichert haben; wenn nicht, fügen Sie diesen mit `array_push()` an das Ende des Arrays an:

```
if (!in_array($host, $hosts))
  array_push($hosts, $host);
```

Dadurch enthält das Array `$hosts` dann jeden Hostnamen, von dem eine Abfrage auf Ihre Website gemacht wurde, genau einmal.

Nach der kompletten Analyse der ganzen Datei bleibt nur noch, das Ergebnis an die Standardausgabe zu schicken, wozu der Schalter *-E* verwendet wird. Um jeden Eintrag im Array in eine eigene Zeile zu schreiben, wird die Funktion `implode()` genutzt, die genau umgekehrt zu `explode()` arbeitet:

```
echo implode("\n", $hosts);
```

Die Ausgabe des Befehls sieht nun in etwa so aus, abhängig davon, von wo die Besucher Ihrer Seite kamen:

```
pD9E0D676.dip.t-dialin.net
pD9E94EC1.dip0.t-ipconnect.de
phuture.xs4all.nl
crawl-66-249-64-189.googlebot.com
D203a.d.pppool.de
xdslk154.osnanet.de
www.whois.sc
dsl-80-42-182-124.access.as9105.com
p4d23e3d4.np.schlund.de
crawl-66-249-64-167.googlebot.com
```

Da der PHP-Code, der für jede Zeile ausgeführt wird, sehr schnell umfangreich wird und fehleranfällig und schwer pflegbar ist, wurde auch ein Schalter eingeführt, der es ermöglicht, den Code, anstatt ihn direkt auf der Kommandozeile einzugeben, in eine Datei zu speichern und diese für jede Zeile auszuführen. Als Erstes legen Sie also eine neue PHP-Datei mit dem Namen *hosts.php* an und kopieren den entsprechenden Code in diese Datei:

```
<?php
list($ip) = explode(" ", $argn, 2);
$host     = gethostbyaddr($ip);
if (!in_array($host, $hosts)) {
```

```
    array_push($hosts, $host);
}
?>
```

Danach müssen Sie lediglich den Aufruf Ihrer Auswertung etwas anpassen und erhalten die gleiche Ausgabe:

```
cat access.log | php -B '$hosts = array();' -F hosts.php -E 'echo implode("\n", $hosts);'
```

Gerade bei Skripten, die Sie häufiger einsetzen, lohnt es sich, diese in einer Datei für eine spätere Verwendung zu speichern.

Siehe auch

Rezept 23.5 zur Weiterverarbeitung von Ausgaben anderer Skripten.

23.7 Prozesse forken

Problem

Sie wollen einen Prozess zur Laufzeit kopieren.

Lösung

Verwenden Sie die Prozess-Kontrollfunktionen von PHP, um einen Prozess zu duplizieren und somit mehrere Aufgaben auf einmal zu erledigen:

```
<?php
$kinder = 5;
$pids     = array();
for ($i=1; $i<=$kinder; $i++) {
 $pid = pcntl_fork();
 if ($pid === -1) {
     exit("Der Prozess konnte nicht kopiert werden.\n");
 } elseif ($pid) {
     $pids[$i] = $pid;
     continue;
 } else {
     erledigeAufgabe($i);
     break;
 }
}
while(true) {
    $prozesse = 0;
    foreach ($pids as $nummer => $pid) {
        $status = null;
        $result = pcntl_waitpid($pid, $status, WNOHANG);
        if ($result === 0) {
            $prozesse++;
        }
```

```
    }
    if ($prozesse == 0) {
        exit("Alle Prozesse beendet.\n");
    }
    echo "$prozesse Prozesse arbeiten noch.\n";
    sleep(2);
}

function erledigeAufgabe($nummer)
{
    echo "Kind-Prozess $nummer beginnt Arbeit.\n";
    $zeit = rand(10, 20);
    sleep($zeit);
    echo "Kind-Prozess $nummer beendet Arbeit.\n";
    exit();
}
?>
```

Diskussion

Die nur unter Unix zur Verfügung stehenden Funktionen zur Prozess-Kontrolle erlauben es Ihnen, das aktuell laufende Skript zu kopieren und somit beliebig viele Instanzen parallel laufen zu lassen. Dies ist besonders hilfreich, wenn Sie Skripten schreiben, die mit externen Ressourcen arbeiten und auf diese warten müssen. Dadurch wartet auch Ihr PHP-Skript unnötig lange, was durch parallele Ausführung des Skripts vermieden werden kann.

Um den aktuellen Prozess zu kopieren, verwenden Sie die Funktion `pcntl_fork()`. Diese hat drei verschiedene Rückgabewerte:

Die Rückgabe `-1` bedeutet, dass kein neuer Prozess erzeugt werden konnte, `0` bedeutet, dass der Prozess kopiert wurde und das laufende Skript diese Kopie ist, und jede positive Zahl wird nur an das Original-Skript zurückgeliefert und steht dann für die Prozess-Id der neuen Kopie. Anhand dieser Rückgabe kann also entschieden werden, ob man sich in der neu erzeugten Kopie oder im Original-Skript befindet. Beide Prozesse setzen die Skriptabarbeitung an derselben Stelle, direkt nach dem Aufruf von `pcntl_fork()`, fort.

Im obigen Beispiel werden durch eine `for`-Schleife fünf neue Prozesse erzeugt. Im Eltern-Prozess werden die Prozess-Ids der erzeugten Kinder in einem Array gespeichert:

```
$kinder = 5;
$pids     = array();
for ($i=1; $i<=$kinder; $i++) {
   $pid = pcntl_fork();
   if ($pid === -1) {
       exit("Der Prozess konnte nicht kopiert werden.\n");
   } elseif ($pid) {
       $pids[$i] = $pid;
       continue;
   } else {
       erledigeAufgabe($i);
```

```
            break;
        }
    }
```

In den neu erzeugten Prozessen wird nach dem Kopieren eine Funktion mit dem Namen `erledigeAufgabe()` aufgerufen und die Iterationsnummer der Schleife übergeben. Die Überprüfung, ob der Prozess ein Kind-Prozess ist, erfolgt durch die Rückgabe von `pcntl_fork()`. In diesem Beispiel erledigt die Funktion `erledigeAufgabe()` keine wirkliche Arbeit, sondern gibt lediglich eine Meldung darüber aus, dass sie gestartet und beendet wurde, und wartet dazwischen eine zufällige Anzahl von Sekunden, um zu simulieren, dass die einzelnen Kind-Prozesse unterschiedlich lange aktiv sind. In Ihrer Anwendung würden Sie hier die eigentlichen Arbeiten erledigen.

```
function erledigeAufgabe($nummer)
{
    echo "Kind-Prozess $nummer beginnt Arbeit.\n";
    $zeit = rand(10, 20);
    sleep($zeit);
    echo "Kind-Prozess $nummer beendet Arbeit.\n";
    exit();
}
```

Nach Erledigung der Aufgabe wird das Skript mit der Funktion `exit()` beendet, wodurch der Kind-Prozess beendet wird. Der Eltern-Prozess läuft allerdings weiter.

Sind alle Kind-Prozesse mit `pcntl_fork()` erzeugt worden, soll der Eltern-Prozess nur noch warten, bis alle Kinder mit der Abarbeitung der Aufgaben fertig sind. Dazu wird einfach eine Endlosschleife mit `while(true)` verwendet. In dieser Schleife wird regelmäßig überprüft, ob alle Kind-Prozesse die Arbeit abgeschlossen haben; ist dies der Fall, wird auch der Eltern-Prozess beendet.

```
while(true) {
    $prozesse = 0;
    foreach ($pids as $nummer => $pid) {
        $status = null;
        $result = pcntl_waitpid($pid, $status, WNOHANG);
        if ($result === 0) {
            $prozesse++;
        }
    }
    if ($prozesse == 0) {
        exit("Alle Prozesse beendet.\n");
    }
    echo "$prozesse Prozesse arbeiten noch.\n";
    sleep(2);
}
```

Mit der Funktion `pcntl_waitpid()` können Sie überprüfen, ob ein Kind-Prozess noch läuft oder schon beendet wurde. Dazu übergeben Sie der Funktion einfach die Prozess-Id, die Sie von `pcntl_fork()` erhalten haben. Als zweiten Parameter übergeben Sie eine Variable per Referenz, in die die Funktion `pcntl_waitpid()` den Statuscode des Prozesses

schreibt. Als dritten Parameter können Sie noch Optionen übergeben, die das Verhalten der Funktion steuern. Im Beispiel wird `WNOHANG` übergeben, wodurch die Funktion sofort einen Wert zurückliefert. Ansonsten wartet `pcntl_waitpid()`, bis der Prozess mit der angegebenen Prozess-Id beendet wird, und das Skript kann dann erst fortgeführt werden. Die Funktion liefert `0` zurück, sofern der Prozess noch läuft. In einer `foreach`-Schleife werden somit also alle zuvor erzeugten Prozesse überprüft, und es wird gezählt, wie viele der Kind-Prozesse noch arbeiten.

Wurden alle Kind-Prozesse beendet, wird auch der Eltern-Prozess nach einer kurzen Statusmeldung gestoppt, indem das Skript mit `exit()` verlassen wird. Ansonsten gibt das Skript aus, wie viele Prozesse noch arbeiten, und wartet zwei Sekunden, bis die Kind-Prozesse erneut überprüft werden.

In realen Anwendungssituationen werden Sie sicher vorziehen, das Skript anzuhalten, bis alle Kind-Prozesse beendet sind, zu Demonstrationszwecken ist es allerdings klarer, wenn in regelmäßigen Abständen ausgegeben wird, wie viele Prozesse noch aktiv sind.

Siehe auch

Rezept 23.8 zum Erstellen einer eigenen Server-Anwendung; die Dokumentation zu den Funktionen zur Prozess-Kontrolle unter *http://www.php.net/pcntl*.

23.8 Einen Server programmieren

Problem

Sie wollen eine PHP-Anwendung programmieren, die auf eingehende Netzwerkverbindungen wartet und mit Clients kommunizieren kann.

Lösung

Verwenden Sie das PEAR-Paket Net_Server, das alle benötigten Funktionen kapselt. Dieses Paket, das einfach über den PEAR-Paket-Manager installiert werden kann, ermöglicht es Ihnen, ohne Wissen über den Aufbau von Netzwerkverbindungen eine eigene Server-Anwendung, z.B. für die Kommunikation mit Flash-Clients, zu erstellen.

Es kümmert sich um das Aufnehmen der Verbindungen sowie das Empfangen und Senden der Daten, und Sie müssen lediglich Callbacks in einem Objekt bereitstellen, die auf diese Aktionen reagieren.

```
#!/usr/bin/php
<?php
require_once 'Net/Server.php';
require_once 'Net/Server/Handler.php';
```

```
class MeinServer extends Net_Server_Handler
{
    public function onConnect($client)
    {
      $this->_server->sendData($client, "Willkommen bei meinem Server.\n");
      return true;
    }

    public function onReceiveData($client, $daten)
    {
        $daten = trim($daten);
        switch ($daten) {
              case 'uhrzeit':
                  $zeit = date('H:i:s', time());
                  $this->_server->sendData($client, "Es ist jetzt $zeit.\n");
                  break;
            case 'ende':
                  $this->_server->sendData($client, "Bis bald.\n");
                  $this->_server->closeConnection($client);
                  break;
            default:
                  $this->_server->sendData($client, "Unbekannter Befehl: $daten.\n");
                  break;
        }
        return true;
    }
}

$server = Net_Server::create('Fork', 'localhost', 9090);
$meinServer = new MeinServer();
$server->setCallbackObject($meinServer);

$server->start();
?>
```

Rufen Sie dieses Skript danach einfach auf der Kommandozeile auf:

```
% ./server.php
```

Um nun eine Verbindung zu Ihrem Server herzustellen, verwenden Sie das Kommandozeilen-Tool *telnet*:

```
% telnet localhost 9090
```

Und Sie erhalten die folgende Meldung:

```
% telnet localhost 9090
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Willkommen bei meinem Server.
```

Um nun die aktuelle Serverzeit ausgegeben zu bekommen, tippen Sie **`uhrzeit`** und drücken Sie die Return-Taste:

```
uhrzeit
Es ist jetzt 18:45:59.
```

Beenden können Sie die Verbindung durch den Befehl ende:

```
ende
Bis bald.
Connection closed by foreign host.
```

Diskussion

Das PEAR-Paket Net_Server stellt eine sehr einfache Schnittstelle zum Erstellen von Server-Anwendungen in PHP zur Verfügung, in das eine zusätzliche Abstraktionsschicht implementiert wurde. Diese Schicht kümmert sich um den Aufbau der Verbindungen mit den Clients und um das Senden und Empfangen der Daten. Damit Ihre Anwendung darauf reagieren kann, wenn ein neuer Client eine Verbindung aufbaut oder Daten schickt, müssen Sie ein so genanntes Handler-Objekt zur Verfügung stellen, das Callbacks für die verschiedenen Ereignisse bereitstellt. Die möglichen Callbacks entnehmen Sie Tabelle 23-1:

Tabelle 23-1: Ereignisse und deren Callback-Methoden im Paket Net_Server

Ereignis	Methode
Server wird gestartet	onStart()
Neuer Client baut Verbindung auf	onConnect($clientId)
Client schickt Daten	onReceiveData($clientId, $daten)
Client beendet die Verbindung	onClose($clientId)
Server wird gestoppt	onShutdown()

Um nun einen Server zu erstellen, wird eine neue Klasse erstellt und von der Basisklasse `Net_Server_Handler` abgeleitet. Dadurch werden bereits alle Callbacks als leere Methoden implementiert und weiterhin einige Hilfsmethoden zur Verfügung gestellt.

Als Erstes wird danach die Methode implementiert, die beim Aufbau einer neuen Verbindung aufgerufen wird:

```
public function onConnect($client)
{
    $this->_server->sendData($client, "Willkommen bei meinem Server.\n");
    return true;
}
```

Diese Methode muss nur einen Parameter akzeptieren, eine eindeutige Id zur Identifikation des Clients, der eine neue Verbindung aufgebaut hat. Im obigen Beispiel soll diese Meldung lediglich eine textuelle Willkommensmeldung an den Client zurückschicken, wozu die Methode `sendData()` der Objekt-Eigenschaft `$_server` verwendet werden kann.

Diese benötigt neben der eindeutigen Client-Id lediglich einen weiteren Parameter, der die Daten beinhaltet, die an den Client gesendet werden sollen.

Als zweite Methode wird noch ein Callback implementiert, der eingehende Daten verarbeitet, die Methode `onReceiveData()`:

```
public function onReceiveData($client, $daten)
{
    $daten = trim($daten);
    switch ($daten) {
        case 'uhrzeit':
            $zeit = date('H:i:s', time());
            $this->_server->sendData($client, "Es ist jetzt $zeit.\n");
            break;
        case 'ende':
            $this->_server->sendData($client, "Bis bald.\n");
            $this->_server->closeConnection($client);
            break;
        default:
            $this->_server->sendData($client, "Unbekannter Befehl: $daten.\n");
            break;
      }
      return true;
}
```

Diese Methode muss neben der Client-Id einen zweiten Parameter akzeptieren, der die vom Client geschickten Daten als String enthält. Nachdem alle unnötigen Whitespace-Zeichen entfernt wurden, wird mit einer `switch`-Anweisung überprüft, ob der Server den gesendeten Befehl kennt. Lautet der Befehl `uhrzeit`, wird die aktuelle Uhrzeit an den Client zurückgeschickt, lautet er `ende`, wird zuerst eine Abschiedsmeldung gesendet und danach die Verbindung geschlossen. Dies geschieht über den Aufruf der Methode `closeConnection()` der Objekt-Eigenschaft `$_server`. Wird eine andere Zeichenkette empfangen, wird eine Meldung an den Client geschickt, die besagt, dass der Befehl nicht bekannt ist.

Um nun den Server zu starten, muss zuerst ein neues Server-Objekt erzeugt und das Callback-Objekt übergeben werden:

```
$server = Net_Server::create('Fork', 'localhost', 9090);
$meinServer = new MeinServer();
$server->setCallbackObject($meinServer);
```

Das Paket Net_Server arbeitet treiberbasiert, d.h., es gibt nicht nur eine Server-Klasse, die über den `new`-Operator erzeugt wird, sondern verschiedene Klassen, von denen durch eine abstrakte Fabrik-Methode beliebige Klassen erzeugt werden können. Neben dem Namen des Treibers erwartet die Methode `Net_Server::create()` noch den Hostnamen und den Port, auf dem der Server eingesetzt werden soll. In diesem Beispiel wird der *Fork*-Treiber verwendet, der für jede neue Verbindung einen neuen Prozess des Servers erzeugt, ähnlich wie das bei einem Webserver der Fall ist. Dadurch kann ein Client den anderen nicht beeinflussen, was meist das gewünschte Verhalten ist. Allerdings steht die-

ser Treiber nur unter Unix-Systemen zur Verfügung, da er auf der Funktion zur Prozess-Kontrolle basiert. Ein weiterer Treiber wäre der *Sequential*-Treiber, bei dem zwar mehrere Clients eine Verbindung aufbauen, jedoch nicht gleichzeitig Daten senden können. Dafür kann dieser Treiber auch unter Windows eingesetzt werden.

Nach Erzeugen des Server-Objekts wird ein Objekt des Handlers instantiiert und dem Server-Objekt mit der Methode `setCallbackObject()` übergeben. Danach muss nur noch der Server durch `$server->start()` gestartet werden, wodurch dieser nun automatisch nach neuen Verbindungen auf dem festgelegten Port lauscht.

Siehe auch

Rezept 23.7 zum Kopieren von Prozessen; Kapitel 24 für weitere Informationen zu PEAR; die Dokumentation zu Net_Server unter *http://pear.php.net/manual/en/package.networking.net-server.php*; die Dokumentation zu den Sockets-Funktionen unter *http://www.php.net/sockets*.

KAPITEL 24

PEAR und PHAR

24.0 Einführung

PEAR ist das PHP Extension and Application Repository, eine Sammlung von zusammenarbeitenden Open Source-Klassen. Entwickler können PEAR-Klassen verwenden, um HTML zu erstellen, SOAP-Requests auszuführen, MIME-E-Mails zu senden – und zum Erledigen einiger anderer gebräuchlicher Aufgaben.

Allgemeine Informationen zu PEAR finden Sie im PEAR-Manual. Die neusten PEAR-Pakete können Sie unter *http://pear.php.net* entdecken.

Mit der Hauptdistribution von PHP werden lediglich einige wenige Kernpakete von PEAR herausgegeben. Allerdings enthält PEAR ein Programm mit dem passenden Namen *pear*, das es Ihnen erleichtert, weitere PEAR-Pakete herunterzuladen und zu installieren. Dieses Programm ist auch unter dem Namen PEAR Package Manager bekannt. Rezept 24.2 zeigt, wie der PEAR Package Manager verwendet wird. Wenn Sie PEAR in der Version 1.4.0 oder neuer installiert haben, wird noch ein zweites Programm *pecl* installiert, mit dem Sie unter Unix PHP-Erweiterungen installieren können.

Der PEAR Package Manager kann zwei Gruppen von Paketen installieren. Die eine Gruppe sind PHP-Pakete, in PHP geschriebener, objektorientierter Code von hoher Qualität, der in Produktionsumgebungen mit jeder Plattform und jedem Webserver verwendet werden kann. Die andere ist PECL (PHP Extension Code Library), »Pickel« ausgesprochen. Dabei handelt es sich um eine in C geschriebene Reihe von Erweiterungen zu PHP. Diese Erweiterungen entsprechen denen, die der PHP-Distribution beigefügt sind. Immer mehr Erweiterungen, die früher mit PHP ausgeliefert wurden, haben inzwischen den Weg zu PECL gefunden. Dadurch ergeben sich zwei Vorteile:

1. Die Erweiterungen haben einen eigenen Release-Zyklus; somit können Bugs schneller behoben und neue Versionen regelmäßiger veröffentlicht werden.
2. Die Kerndistribution von PHP wird kleiner, und beim Veröffentlichen neuer Versionen müssen weniger Erweiterungen berücksichtigt werden.

Zusätzlich können Sie mit dem PEAR Package Manager die Infrastruktur des PEAR-Klassen-Managements für Ihre persönlichen Projekte verwenden. Wenn Sie beim Erstellen Ihrer Pakete das PEAR-Format verwenden, können Ihre Benutzer *pear* dazu benutzen, die Dateien von der Website Ihres Projekts herunterzuladen und zu installieren. Wie Sie PEAR-Pakete selbst erstellen können, wird in Rezept 24.8 behandelt. Seit der Version 1.4.0 können Sie einen sogenannten Channel betreiben, um Ihre Pakete Benutzern zur Installation anzubieten. Die Rezepte 24.6 und 24.9 beschäftigen sich mit den Channel-Funktionen von PEAR.

Dieses Kapitel zeigt Ihnen, wie Sie PEAR installieren können, sofern es noch nicht mit PHP mitinstalliert wurde (Rezept 24.1) oder wenn Sie PEAR bei einem Shared-Hosting-Anbieter nutzen möchten, der Ihnen nur FTP-Zugang gewährt (Rezept 24.7). Wie Sie weitere PEAR-Pakete installieren und upgraden, ist Thema der Rezepte 24.3 und 24.4, weitere Funktionen des Package Manager werden in Rezept 24.2 behandelt.

Sobald ein PEAR-Paket installiert ist, verwenden Sie es in Ihren PHP-Skripten, indem Sie `require` aufrufen. Hier sehen Sie, wie Sie beispielsweise das Net_Dig-Paket einbeziehen:

```
require 'Net/Dig.php';
```

Wenn ein Paketname einen Unterstrich enthält, ersetzen Sie ihn durch einen Slash (/) und fügen *.php* am Ende an.

Manche Pakete verlangen, mehrere Klassen einzubeziehen, so z.B. SOAP. Statt *SOAP.php*, nehmen Sie also *SOAP/Client.php* oder *SOAP/Server.php*. Aus der Dokumentation erfahren Sie, ob ein bestimmtes Paket unübliche Datei-Includes benötigt.

Da PEAR-Pakete als normale PHP-Dateien einbezogen werden, sollten Sie sicherstellen, dass sich das Verzeichnis mit den PEAR-Klassen in Ihrem `include_path` befindet. Ist das nicht der Fall, können `include` und `require` keine PEAR-Klassen finden.

Anleitungen und Beispiele zur Verwendung einer bestimmten PEAR-Klasse finden Sie im PEAR-Handbuch unter *http://pear.php.net/manual/en/packages.php* oder im oberen Teil der PHP-Dateien des Pakets. Beispiele zur Arbeit mit verschiedenen PEAR-Paketen finden Sie in Kapitel 14 (XML-Verarbeitung) und in Kapitel 14 (Webservices).

Zum Zeitpunkt der Drucklegung dieser Auflage des PHP-Kochbuchs ist PHP 5.3 gerade veröffentlicht worden. Zu den Neuigkeiten dieser PHP-Version gehört auch die PHAR-Erweiterung. *PHAR* ist das Kürzel für PHP Archive. Dabei handelt es sich um ein Paketformat ähnlich dem von Java bekannten JAR-Format. Eine PHAR-Datei kann PHP-Quellcode und Ressourcen (z.B. Bilder, Konfigurationsdateien usw.) enthalten. Damit ist es möglich, komplette Bibliotheken oder PHP-Programme in einem solchen Archiv als einzelne Datei auszuliefern, was den Deployment-Prozess sehr erleichtert. Wie Sie PHAR-Dateien erstellen und verwenden können, zeigen die Rezepte 24.10 bis 24.12.

PEAR hat sich ebenfalls weiterentwickelt und wird unter dem Namen *PEAR2* weitergeführt. Jede Menge Code wurde neu geschrieben, und Bibliotheken werden verwendet, die mit PHP ausgeliefert werden. PEAR2 macht unter anderem Gebrauch von Namens-

räumen und setzt sehr stark auf die zuvor angesprochene PHAR-Erweiterung. Da sich dieses Projekt zum Zeitpunkt der Drucklegung dieses Buchs jedoch noch in einer sehr frühen Alpha-Phase befand, wird in diesem Kapitel nicht darauf eingegangen. Es lohnt sich jedoch, diesbezüglich auf dem Laufenden zu bleiben und den Entwicklungsstand zu verfolgen. Die Projekt-Website *http://pear2.php.net* bietet Ihnen die Möglichkeit dazu.

24.1 PEAR installieren

Problem

Sie möchten PEAR nutzen, allerdings ist es auf Ihrem Server nicht installiert.

Lösung

Verwenden Sie das unter *http://pear.php.net/go-pear* zur Verfügung gestellte Bootstrap-Skript, um PEAR auf Ihrem Server zu installieren. Unter Unix geben Sie dazu folgenden Befehl ein:

```
% lynx -source http://pear.php.net/go-pear | php
```

Die Installation unter Windows erfolgt in zwei Schritten:

```
C:\> php-cli -r 'readfile("http://pear.php.net/go-pear");' > go-pear.php
C:\> php-cli go-pear.php
```

Diskussion

Standardmäßig wird PEAR bei der Installation von PHP 5 automatisch mitinstalliert, jedoch gibt es zwei Gründe, weswegen PEAR auf Ihrem System nicht vorhanden sein könnte:

- PHP wurde mit der Option `--without-pear` kompiliert, die explizit angibt, dass PEAR nicht installiert werden soll.
- PHP wurde mit der Option `--disable-cli` kompiliert, die das Erzeugen der CLI-Binaries unterbindet. Wenn diese Option aktiviert ist, wird automatisch auch `--without-pear` aktiviert.

Sofern Sie Zugriff auf die Kommandozeile haben, ist es allerdings sehr einfach, PEAR nachträglich zu installieren.

Unter Unix führen Sie dazu einfach das folgende Kommando aus:

```
% lynx -source http://pear.php.net/go-pear | php -q
Welcome to go-pear!

Go-pear will install the 'pear' command and all the files needed by
it.  This command is your tool for PEAR installation and maintenance.
```

```
Go-pear also lets you download and install the PEAR packages bundled
with PHP: DB, Net_Socket, Net_SMTP, Mail, XML_Parser, PHPUnit.

If you wish to abort, press Control-C now, or press Enter to continue:
```

Das lädt ein PHP-Skript von der PEAR-Website herunter und gibt es zur Ausführung an PHP weiter. Das Programm lädt danach alle Dateien herunter, die zur Ausführung von *pear* gebraucht werden, und hilft Ihnen auf die Beine.

Auf einigen Unix-Systemen müssen Sie eventuell *links* anstatt *lynx* laufen lassen. Wenn Sie die Kommandozeilen-Version von PHP installiert haben, entfernen Sie das `-q`-Flag für PHP; die CLI-Version unterdrückt HTTP-Kopfzeilen automatisch. Wenn Sie glauben, dass *go-pear* sich aufgehängt hat, setzen Sie `output_buffering` in Ihrer *php.ini*-Konfigurationsdatei auf `off`.

Die Installation unter Windows erfolgt in zwei Schritten:

```
C:\> php-cli -r 'readfile("http://pear.php.net/go-pear");' > go-pear.php
C:\> php-cli go-pear.php
```

Die erste Anweisung lädt die Bootstrap-Datei von *http://*pear.php.net/go-pear herunter und speichert sie unter dem Namen *go-pear.php* ab. Diesen Schritt können Sie auch mit einem Webbrowser erledigen, indem Sie einfach dieselbe URL eingeben und dann *Datei/Seite speichern* auswählen, um die Datei auf Ihrer Festplatte zu speichern. Mit der zweiten Anweisung führen Sie die Datei schließlich aus. Wenn Sie Windows so konfiguriert haben, dass PHP-Dateien per Doppelklick ausgeführt werden können, können Sie die Installation mit einem Doppelklick auf das Symbol der Datei starten. Unter der Windows-Installation bezeichnet *php-cli* die Kommandozeilenversion von PHP.

Nachdem Sie die Installation gestartet haben, können Sie einen Proxy angeben, über den die PEAR-Dateien heruntergeladen werden sollen, und schließlich auswählen, in welchem Verzeichnis die PEAR-Installation erfolgen soll:

```
Below is a suggested file layout for your new PEAR installation.  To
change individual locations, type the number in front of the
directory.  Type 'all' to change all of them or simply press Enter to
accept these locations.

 1. Installation prefix           : /usr
 2. Binaries directory            : $prefix/bin
 3. PHP code directory ($php_dir) : $prefix/lib/php
 4. Documentation base directory  : $php_dir/docs
 5. Data base directory           : $php_dir/data
 6. Tests base directory          : $php_dir/tests

1-6, 'all' or Enter to continue:
```

In nahezu allen Fällen ist es ausreichend, die Installationspfade so zu belassen, wie *go-pear* dies vorschlägt. Danach werden automatisch alle nötigen Dateien von der PEAR-Webseite heruntergeladen und installiert:

```
The following PEAR packages are bundled with PHP: DB, Net_Socket, Net_SMTP,
```

```
Mail, XML_Parser, PHPUnit.
Would you like to install these as well? [Y/n] :

Loading zlib: ok
Downloading package: PEAR-stable......ok
Downloading package: Archive_Tar-stable....ok
Downloading package: Console_Getopt-stable....ok
Downloading package: XML_RPC-stable....ok
Bootstrapping: PEAR...................(remote) ok
Bootstrapping: Archive_Tar............(remote) ok
Bootstrapping: Console_Getopt.........(remote) ok
Downloading package: DB..............ok
Downloading package: Net_Socket.......ok
Downloading package: Net_SMTP.........ok
Downloading package: Mail.............ok
Downloading package: XML_Parser.......ok
Downloading package: PHPUnit..........ok
Extracting installer..................ok
install ok: PEAR 1.3.5
install ok: Archive_Tar 1.3.1
install ok: Console_Getopt 1.2
install ok: XML_RPC 1.2.2
install ok: DB 1.7.5
install ok: Net_Socket 1.0.6
install ok: Net_SMTP 1.2.6
install ok: Mail 1.1.4
install ok: XML_Parser 1.2.6
install ok: PHPUnit 1.2.2
```

Danach überprüft *go-pear*, ob der Pfad, unter dem Sie PEAR installiert haben, bereits in Ihrem `include_path` liegt, sodass PEAR-Pakete einfach über `include` oder `require` eingebunden werden können. Sollte dies nicht der Fall sein, wird eine Warnung ausgegeben:

```
WARNING!  The include_path defined in the currently used php.ini does not
contain the PEAR PHP directory you just specified:
</home/schst/pear/test/lib/php>
If the specified directory is also not in the include_path used by
your scripts, you will have problems getting any PEAR packages working.

Current include path           : .:/usr/local/php5/lib/php
Configured directory           : /usr/share/php
Currently used php.ini (guess) :
```

Die Installation wird dann mit einigen Hinweisen zur Verwendung von PEAR abgeschlossen. Falls noch eine weitere Konfiguration Ihres Systems vonnöten ist, wird *go-pear* Sie darauf aufmerksam machen:

```
The 'pear' command is now at your service at /usr/bin/pear

Run it without parameters to see the available actions, try 'pear list'
to see what packages are installed, or 'pear help' for help.

For more information about PEAR, see:
```

```
  http://pear.php.net/faq.php
  http://cvs.php.net/co.php/pearweb/doc/pear_package_manager.txt?p=1
  http://pear.php.net/manual/

Thanks for using go-pear!
```

Sie können nun den PEAR Package Manager verwenden, um die benötigten PEAR-Pakete zu installieren.

Siehe auch

Rezepte 24.2 und 24.3 zur Verwendung des PEAR Package Manager; Rezept 24.7 zur Installation von PEAR in einer Shared-Hosting-Umgebung; das Kapitel zur Installation im PEAR-Manual unter *http://pear.php.net/manual/de/installation.php*.

24.2 Den PEAR Package Manager verwenden

Problem

Sie wollen den PEAR Package Manager *pear* verwenden. Mit ihm können Sie neue Pakete installieren und ihre bereits vorhandenen PEAR-Pakete auf den neusten Stand bringen sowie Informationen darüber einsehen.

Lösung

Um einen Befehl mit dem PEAR Package Manager auszuführen, tippen Sie den Befehlsnamen als erstes Argument auf der Kommandozeile ein:

```
% pear befehl
```

Diskussion

Und so listen Sie alle installierten PEAR-Pakete mit dem Befehl `list` auf:[1]

```
% pear list
Installed packages, channel pear.php.net:
=========================================
Package                 Version             State
Archive_Tar             1.3.1               stable
Console_Getargs         1.2.1               stable
DB                      1.7.5               stable
Date                    1.4.3               stable
Date_Holidays           0.13.0              alpha
HTTP_Client             1.0.0               stable
HTTP_SessionServer      0.4.0               alpha
```

1 In früheren Versionen von *pear* hieß dieses Kommando `list-installed`.

```
Log                     1.8.7           stable
PEAR                    1.4.0a9         alpha
PEAR_Frontend_Gtk       0.4.0           beta
PEAR_PackageFileManager 1.5.0           stable
Services_Ebay           0.11.0          alpha
XML_Beautifier          1.1             stable
XML_Parser              1.2.6           stable
XML_RPC                 1.2.2           stable
XML_Serializer          0.15.0          beta
XML_Util                1.1.1           stable
```

Um eine Liste aller gültigen PEAR-Befehle zu erhalten, verwenden Sie `list-commands`. Viele Befehle haben auch Abkürzungen: Für `list` kann beispielsweise auch nur `l` eingegeben werden. Diese Abkürzungen sind normalerweise die ersten Buchstaben des Befehlnamens. Tabelle 24-1 führt häufig verwendete Befehle auf.

Tabelle 24-1: Befehle des PEAR Package Manager

Befehlsname	Abkürzung	Beschreibung
`install`	`i`	Pakete herunterladen und installieren.
`upgrade`	`up`	Installierte Pakete auf den neusten Stand bringen.
`uninstall`	`un`	Installierte Pakete entfernen.
`list`	`l`	Installierte Pakate auflisten.
`list-upgrades`	`lu`	Alle verfügbaren Upgrades für die installierten Pakete auflisten.
`search`	`sp`	Pakete suchen.

pear hat Befehle sowohl für den Gebrauch als auch das Entwickeln von PEAR-Klassen; folglich benötigen Sie eventuell nicht alle Befehle. Der Befehl `package` zum Beispiel erstellt ein neues PEAR-Paket. Wenn Sie lediglich Pakete installieren und benutzen möchten, können Sie diesen Befehl getrost links liegen lassen.

Wie bei allen Programmen müssen Sie Ausführungsberechtigung haben, wenn Sie *pear* ausführen wollen. Wenn Sie *pear* laufen lassen können, während Sie als root eingeloggt sind, nicht jedoch als normaler Benutzer, achten Sie darauf, dass das `group-` bzw. `world-execute`-Bit gesetzt sind. Für manche Vorgänge erstellt *pear* auch eine Lock-Datei im Verzeichnis, das die PEAR-Dateien enthält. Sie müssen eine Schreibberechtigung zum Zugriff auf die Datei mit dem Namen *.lock* in diesem Verzeichnis haben.

Um herauszufinden, wo Ihre PEAR-Pakete abgelegt sind, starten Sie den Befehl `config-get php_dir`. Sie können den Wert von `include_path` feststellen, indem Sie `ini_get('include_path')` von PHP aus aufrufen oder indem Sie in Ihrer *php.ini*-Datei nachsehen. Wenn Sie *php.ini* nicht ändern können, weil Sie sich Ihre Hosting-Umgebung mit anderen teilen, fügen Sie das Verzeichnis oben in Ihrem Skript dem `include_path` hinzu, bevor Sie die Datei einfügen. Rezept 10.18 enthält weitere Informationen zum Setzen von Konfigurationsvariablen von PHP aus.

Wenn Sie hinter einem HTTP-Proxyserver sitzen, konfigurieren Sie PEAR zum Verwenden des Servers mit dem Befehl:

```
% pear config-set http_proxy proxy.beispiel.com:8080
```

PEAR Package Manager-Einstellungen können Sie mit

```
% pear config-set parameter wert
```

konfigurieren.

Hier ist *parameter* der Name des zu ändernden Parameters und *wert* sein neuer Wert. Um alle Ihre aktuellen Einstellungen zu sehen, geben Sie den Befehl `config-show` ein:

```
% pear config-show
Configuration (channel pear.php.net):
=====================================
Auto-discover new Channels     auto_discover    <not set>
Default Channel                default_channel  pear.php.net
HTTP Proxy Server Address      http_proxy       <not set>
PEAR server [DEPRECATED]       master_server    pear.php.net
Default Channel Mirror         preferred_mirror pear.php.net
Remote Configuration File      remote_config    <not set>
PEAR executables directory     bin_dir          /usr/bin
PEAR documentation directory   doc_dir          /usr/share/php/doc
PHP extension directory        ext_dir          /usr/lib/php/extensions
PEAR directory                 php_dir          /usr/share/php
PEAR Installer cache directory cache_dir        /tmp/pear/cache
PEAR data directory            data_dir         /usr/share/php/data
PHP CLI/CGI binary             php_bin          /usr/bin/php
PEAR test directory            test_dir         /usr/share/php/test
Cache TimeToLive               cache_ttl        3600
Preferred Package State        preferred_state  alpha
Unix file mask                 umask            22
Debug Log Level                verbose          1
PEAR password (for             password         <not set>
maintainers)
Signature Handling Program     sig_bin          /usr/local/bin/gpg
Signature Key Directory        sig_keydir       /etc/pearkeys
Signature Key Id               sig_keyid        <not set>
Package Signature Type         sig_type         gpg
PEAR username (for             username         <not set>
maintainers)
User Configuration File        Filename         /home/schst/.pearrc
System Configuration File      Filename         /etc/pear.conf
```

Eine kurze Beschreibung der einzelnen Konfigurationsoptionen erhalten Sie über das Kommando `config-help`. Hilfe zu den generellen Optionen des Package Manager erhalten Sie über das Kommando `help options`, Hilfe zu einem speziellen Befehl gibt die Anweisung `help` *`befehl`* aus.

Um festzustellen, welche Pakete auf dem PEAR-Server verfügbar sind, können Sie das Kommando `list` verwenden. Möchten Sie nach einem speziellen Paket suchen, geben Sie das Kommando `search`:

```
% pear search ebay
Matched packages, channel pear.php.net:
=======================================
Package       Stable/(Latest)    Local
Services_Ebay none/(0.11.0alpha) 0.11.0 Interface to eBay's XML-API.
```

24.3 PEAR-Pakete installieren und deinstallieren

Problem

Sie wollen ein PEAR-Paket installieren oder deinstallieren.

Lösung

Laden Sie das Paket von Ihrem PEAR-Server herunter und installieren Sie es mithilfe des PEAR Package Managers:

```
% pear install Paket_Name
```

Möchten Sie eine spezielle Version installieren, können Sie diese einfach anhängen:

```
% pear install Paket_Name-Version
```

Sie können es auch von einem beliebigen Ort im Internet aus installieren:

```
% pear install http://pear.beispiel.com/Paket_Name-1.0.tgz
```

Und so installieren Sie, wenn Sie eine lokale Kopie eines Pakets haben:

```
% pear install Paket_Name-1.0.tgz
```

Benötigen Sie das Paket nicht mehr, können Sie es genauso einfach deinstallieren:

```
% pear uninstall Paket_Name
```

Diskussion

Um PEAR-Pakete zu installieren, brauchen Sie dort eine Schreibberechtigung, wo die Pakete gespeichert werden. Standardmäßig ist das */usr/local/lib/php/*.

Sie können auch mehrere Pakete gleichzeitig anfordern:

```
% pear install HTML_Common HTML_Javascript
downloading HTML_Common-1.0.tgz ...
...done: 2,959 bytes
install ok: HTML_Common 1.0
downloading HTML_Javascript-1.0.0.tgz ...
...done: 4,141 bytes
install ok: HTML_Javascript 1.0.0
```

Beim Installieren eines Pakets prüft PEAR, ob Sie alle notwendigen PHP-Funktionen und PEAR-Pakete haben, auf die das neue Paket angewiesen ist. Wenn die Prüfung Lücken aufweist, berichtet PEAR über die Abhängigkeiten:

```
% pear install HTML_Table
downloading HTML_Table-1.1.tgz ...
...done: 5,168 bytes

requires package `HTML_Common' >= 1.0
HTML_Table: dependencies failed
```

Zur Lösung dieses Problems laden Sie zuerst das fehlende Paket herunter und installieren es. Wenn Sie die Abhängigkeiten ignorieren wollen, können Sie die Installation mit `-n` oder `--nodeps` erzwingen. Das benötigte Paket können Sie später noch installieren.

Es ist ebenfalls möglich, direkt beim Installieren anzugeben, dass Sie die zur Installation benötigten PEAR-Pakete automatisch installieren möchten. Dazu verwenden Sie die Option `-o` oder `--onlyreqdeps`, wodurch automatisch die benötigten Pakete mitinstalliert werden:

```
% pear install --onlyreqdeps HTTP_Download
downloading HTTP_Download-1.0.0.tgz ...
Starting to download HTTP_Download-1.0.0.tgz (11,159 bytes)
.....done: 11,159 bytes
downloading HTTP_Header-1.1.2RC1.tgz ...
Starting to download HTTP_Header-1.1.2RC1.tgz (9,568 bytes)
...done: 9,568 bytes
install ok: HTTP_Header-1.1.2RC1
install ok: HTTP_Download-1.0.0
```

Manche PEAR-Pakete können auch optionale Abhängigkeiten haben, die nur bei der Verwendung mancher Funktionen benötigt werden. Wollen Sie auch diese optionalen Pakete mitinstallieren, benutzen Sie die Option `-a` oder `--alldeps`.

Brauchen Sie ein Paket nicht mehr, können Sie es mit dem Befehl

```
% pear uninstall HTTP_Header
```

deinstallieren. Sollte das Paket noch von einem anderen Paket benötigt werden, weigert sich der PEAR Package Manager, das Paket zu deinstallieren, und gibt eine dementsprechende Meldung aus:

```
HTTP_Header is required by installed package "HTTP_Download"
pear/HTTP_Header cannot be uninstalled, other installed packages depend on this package
```

In Ihrer PEAR-Konfiguration können Sie über die Option `preferred_state` festlegen, ob Sie nur stabile Pakete (stable) oder auch noch in der Entwicklung befindliche Pakete installieren möchten.

Siehe auch

Rezept 24.5 für Informationen zum Installieren von PECL-Paketen; Rezept 24.4 für weitere Informationen zum Upgraden von bereits vorhandenen Paketen.

24.4 PEAR-Pakete upgraden

Problem

Sie wollen ein Paket in Ihrem System auf die neuste Version aufrüsten, um zusätzliche Funktionen nutzen und Bugs beheben zu können.

Lösung

Stellen Sie fest, ob Upgrades zur Verfügung stehen, und weisen Sie *pear* an, die von Ihnen gewünschten Pakete auf den neusten Stand zu bringen:

```
% pear list-upgrades
% pear upgrade Paket_Name
```

Diskussion

Das Upgraden auf eine neue Version eines Pakets ist für den PEAR Package Manager eine einfache Aufgabe. Wenn Sie wissen, dass ein bestimmtes Paket veraltet ist, können Sie es direkt auf den neusten Stand bringen. Aber Sie möchten vielleicht die Pakete nur regelmäßig überprüfen, um zu sehen, ob neue Releases vorhanden sind.

Dazu verwenden Sie den Befehl `list-upgrades`, der eine Tabelle mit den Paketnamen, der neuen Versionsnummer und dem Umfang des Downloads angibt:

```
% pear list-upgrades
Available Upgrades (stable):
= == == == == == == == == == == == == == == ==
+-------------+---------+--------+
| Package     | Version | Size   |
| Archive_Tar | 0.9     | 8.9kB  |
| Auth        | 1.0.2   | 8.8kB  |
| Auth_HTTP   | 1.0.1   | 1.7kB  |
| DB          | 1.3     | 58kB   |
| HTTP        | 1.1     | 2.9kB  |
| Mail        | 1.0.1   | 11.6kB |
| Mail_Mime   | 1.2.1   | 15.0kB |
| Net_Ping    | 1.0.1   | 2.1kB  |
| Net_SMTP    | 1.0     | 2.8kB  |
| Net_Socket  | 1.0.1   | 3.5kB  |
| PEAR        | 0.9     | 40kB   |
| XML_Parser  | 1.0     | 4.8kB  |
| XML_RPC     | 1.0.3   | 11.9kB |
| XML_RSS     | 0.9.1   | 3.1kB  |
| XML_Tree    | 1.1     | 4.7kB  |
+-------------+---------+--------+
```

Wenn Sie auf dem neusten Stand sind, gibt *pear*

```
No upgrades available
```

aus. Um ein bestimmtes Paket aufzurüsten, verwenden Sie den Befehl `upgrade`, zum Beispiel:

```
% pear upgrade DB
downloading DB-1.3.tgz ...
...done: 59,332 bytes
```

Die Abkürzung für den Befehl `list-upgrades` ist `lu`; für `upgrade` heißt sie `up`.

Möchten Sie alle Pakete upgraden, die Sie installiert haben, können Sie den Befehl `upgrade-all` verwenden:

```
% pear upgrade-all
Will upgrade date_holidays
Will upgrade xml_parser
Will upgrade xml_xul
downloading date_holidays-0.13.0.tgz ...
Starting to download date_holidays-0.13.0.tgz (28,813 bytes)
...done: 28,813 bytes
downloading xml_parser-1.2.6.tgz ...
Starting to download xml_parser-1.2.6.tgz (12,944 bytes)
...done: 12,944 bytes
downloading xml_xul-0.8.2.tgz ...
Starting to download xml_xul-0.8.2.tgz (28,311 bytes)
...done: 28,311 bytes
upgrade-all ok: XML_Parser-1.2.6
upgrade-all ok: XML_XUL-0.8.2
upgrade-all ok: Date_Holidays-0.13.0
```

PEAR hat außerdem einen RSS-Feed, der neue Pakete auflistet, die unter *http://pear.php.net/rss.php* zur Verfügung stehen.

Siehe auch

Die Rezepte 24.3 und 24.5 für Informationen zum Installieren von PEAR- und PECL-Paketen; Rezept 15.7 für weitere Informationen zum Parsen von RSS-Feeds.

24.5 PECL-Pakete installieren

Problem

Sie wollen ein PECL-Paket installieren; dafür wird eine in C geschriebene PHP-Erweiterung gebaut, die innerhalb von PHP verwendet wird.

Lösung

Vergewissern Sie sich, dass Sie alle notwendigen Erweiterungsbibliotheken haben, und verwenden Sie dann den `install`-Befehl des PEAR Package Manager:

```
% pear install cvsclient
```

Verwenden Sie bereits PEAR 1.4.0 oder höher, können Sie den mitgelieferten `pecl`-Befehl nutzen:

```
% pecl install cvsclient
```

Um die Erweiterung von PHP aus zu verwenden, laden Sie sie mithilfe von `dl()`:

```
dl('cvsclient.so');
```

Diskussion

Der Frontend-Prozess zum Installieren von PECL-Paketen gleicht der Installation von PEAR-Paketen für Code, der in PHP geschrieben ist. Allerdings sind die Aufgaben hinter den Kulissen sehr unterschiedlich. Da die PECL-Erweiterungen in C geschrieben sind, muss der Package Manager die Erweiterung kompilieren und sie so konfigurieren, dass sie mit der installierten Version von PHP arbeiten kann. Deshalb können Sie zurzeit PECL-Pakete auf Unix-Maschinen installieren sowie auf Windows-Maschinen, wenn Sie MSDev verwenden.

Beim Installieren einer PECL-Erweiterung lädt der PEAR Package Manager die Datei herunter, entpackt sie, führt *phpize* aus, um die Erweiterung für die auf der Maschine installierte PHP-Version zu konfigurieren, und erstellt und installiert dann die Erweiterung.

```
% pear install cvsclient
downloading cvsclient-0.2.tgz ...
Starting to download cvsclient-0.2.tgz (8,710 bytes)
.....done: 8,710 bytes
3 source files, building
running: phpize
Configuring for:
PHP Api Version:         20031224
Zend Module Api No:      20041030
Zend Extension Api No:   220040412
building in /var/tmp/pear-build-root/cvsclient-0.2
running: /tmp/tmpipGQNp/cvsclient-0.2/configure
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking for gcc... gcc
...weitere configure checks...
configure: creating ./config.status
config.status: creating config.h
running: make
running: make INSTALL_ROOT="/var/tmp/pear-build-root/install-cvsclient-0.2" install
Installing shared extensions:     /var/tmp/pear-build-root/install-cvsclient-0.2/usr/lib/
php/extensions/no-debug-non-zts-20041030/
running: find "/var/tmp/pear-build-root/install-cvsclient-0.2" -ls
.....
install ok: cvsclient 0.2
```

Im Gegensatz zur Praxis bei PHP-basierten PEAR-Paketen werden Sie von den PECL-Erweiterungen nicht automatisch informiert, wenn Ihnen eine Bibliothek fehlt, die für

das Kompilieren der Erweiterung notwendig ist. Viel mehr sind Sie dafür verantwortlich, diese Dateien vorher richtig zu installieren. Wenn Sie Schwierigkeiten haben, eine PECL-Erweiterung zu installieren, sollten Sie in der *README*-Datei und in der weiteren Dokumentation, die Teil des Pakets ist, nachlesen. Der Package Manager installiert diese Dateien im Verzeichnis *docs* unter Ihrer PEAR-Hierarchie. Während der Installation fragt Sie der Package Manager nach dem Pfad der benötigten Bibliothek.

Wenn sich diese Bibliotheken an einem Standardplatz befinden, wird beim Drücken der Return-Taste die Option `autodetect` gewählt. PHP sucht dann nach den Bibliotheken und wählt sie aus. Sie brauchen dabei keinen ausdrücklichen Pfadnamen einzugeben.

PECL-Erweiterungen werden an anderen Plätzen gespeichert als Nicht-PECL-Pakete. Wenn Sie *pear* ausführen wollen, müssen Sie in das PHP-Verzeichnis *extensions* schreiben können. Manche PECL-Pakete, wie z.B. *xmms*, installieren Dateien im gleichen Verzeichnis wie die Binärdatei mit dem PHP-Programm. Daher werden Sie diese Pakete wahrscheinlich unter dem gleichen Benutzernamen installieren wollen, unter dem Sie schon PHP installiert haben. Sie sollten außerdem die Ausführungsberechtigungen dieser Dateien prüfen: Da die meisten PEAR-Dateien nicht ausführbar sind, stellt Ihre *umask* die ausführbaren Dateien vielleicht nicht mit den richtigen Berechtigungen zur Verfügung.

PECL-Erweiterungen können Sie für Windows unter der URL *http://windows.php.net/snapshots* auch direkt als DLL-Dateien herunterladen. Allerdings handelt es sich hierbei nicht um Release-Versionen, sondern um nächtliche Snapshots, also immer den neuesten Stand der Entwicklung. Es ist also gut möglich, dass diese Versionen Fehler enthalten.

Siehe auch

Rezept 24.3 für Informationen zum Installieren von PEAR-Paketen; Rezept 24.4 für weitere Informationen zum Upgraden eines vorhandenen Pakets.

24.6 Pakete aus anderen Channels installieren

Problem

Sie möchten ein PEAR-kompatibles Paket aus einem anderen Channel installieren.

Lösung

Verwenden Sie Version 1.4.0 oder höher des PEAR Package Manager:

```
% pear clear-cache
% pear upgrade PEAR
```

Fügen Sie den Channel zu Ihrem PEAR Package Manager hinzu:

```
% pear channel-discover pear.php-tools.net
Adding Channel "pear.php-tools.net" succeeded
Discovery of channel "pear.php-tools.net" succeeded
```

Installieren Sie danach das gewünschte Paket:

```
% pear install pat/patError
downloading patError-1.1.0.tgz ...
Starting to download patError-1.1.0.tgz (17,579 bytes)
......done: 17,579 bytes
install ok: channel://pear.php-tools.net/patError-1.1.0
```

Diskussion

Eine der größten Neuerungen von PEAR 1.4.0 ist die Unterstützung von Channels. Zwar war es mit früheren Versionen auch schon möglich, eigene Pakete über den PEAR Package Manager zu vertreiben, jedoch mussten Kunden dabei immer die komplette URL des Pakets angeben, um es zu installieren. Dadurch konnten auch keine erweiterten Funktionen, wie z.B. eine Liste aller angebotenen Pakete oder das Überprüfen auf Upgrades, genutzt werden.

Bei diesen Funktionen kommuniziert der PEAR Package Manager mit dem PEAR-Server über das XML-RPC-Protokoll und kann so verschiedene Funktionen auf dem PEAR-Server aufrufen, die dessen Datenbank nutzen. Mit Channels ist es nun möglich, mehr als einen Server anzugeben, mit dem der Package Manager kommunizieren soll. Diese Channel-Server müssen lediglich einen XML-RPC-Server bereitstellen, der die gleiche API bietet wie der PEAR-Server. Kennen Sie den Hostnamen eines solchen Servers, geben Sie diesen einfach dem PEAR Package Manager bekannt. Dazu führen Sie das Kommando `channel-discover` aus und übergeben den Hostnamen des Servers:

```
% pear channel-discover pear.php-tools.net
Adding Channel "pear.php-tools.net" succeeded
Discovery of channel "pear.php-tools.net" succeeded
```

Möchten Sie sehen, welche Channels Ihre Package Manager-Installation bereits kennt, verwenden Sie `channel-list`:

```
% pear list-channels
Registered Channels:
====================
Channel                Summary
pear.php-tools.net     PHP Application Tools Channel
pear.php.net           PHP Extension and Application Repository
pecl.php.net           PHP Extension Community Library
__uri                  Pseudo-channel for static packages
```

Die Channels *pear.php.net*, *pecl.php.net* und *__uri* sind bereits standardmäßig vorhanden, den Channel *pear.php-tools.net* haben Sie selbst dazugegeben. Sie können dem Package Manager beliebig viele Channels hinzufügen, Tabelle 24-2 gibt Ihnen einen

Überblick über die bereits existierenden Channels. Möchten Sie Informationen zu diesem Channel erhalten, nutzen Sie das Kommando `channel-info`:

```
% pear channel-info pear.php-tools.net
Channel pear.php-tools.net Information:
=======================================
Name and Server         pear.php-tools.net
Alias                   pat
Summary                 PHP Application Tools Channel
Validation Package Name PEAR_Validate
Validation Package      default
Version
Server Capabilities
===================
Type   Version Function Name              URI
xmlrpc 1.0     logintest
xmlrpc 1.0     package.listLatestReleases
xmlrpc 1.0     package.listAll
xmlrpc 1.0     package.info
xmlrpc 1.0     package.getDownloadURL
xmlrpc 1.1     package.getDownloadURL
xmlrpc 1.0     package.getDepDownloadURL
xmlrpc 1.1     package.getDepDownloadURL
xmlrpc 1.0     package.search
xmlrpc 1.0     channel.listAll
```

Im ersten Teil sehen Sie den Namen des Channels sowie einen Alias, den Sie statt des Hostnamens bei zukünftigen Aufrufen verwenden können. Der zweite Teil der Ausgabe beschreibt, welche Methodenaufrufe der Channel unterstützt; dieser ist im Moment nicht relevant.

Tabelle 24-2: Verfügbare PEAR-Channels

Channel-Name	URL	Beschreibung/angebotene Pakete
Chiara	*pear.chiaraquartet.net*	Server von Greg Beaver, Autor der Channel-Funktionen von PEAR 1.4.0 Verfügbare Pakete: XML-RPC-Implementierung für PHP 5, PEAR-Channel-Server
PHP Application Tools	*pear.php-tools.net*	Server des Projekts PHP Application Tools Verfügbare Pakete: Template-Engine, Abstraktionsklasse für Konfigurationsdateien, Formular-Generierung und -Validierung und weitere Pakete
Pearified	*pearified.com*	PEAR-kompatible Portierungen beliebter PHP-Projekte Verfügbare Pakete: Smarty
Cerebral Cortex	*pear.crtx.org*	Framework eines PEAR-Entwicklers Verfügbare Pakete: HTML-Frontend zum Channel-Server von Greg Beaver

Um nun ein Paket von diesem Channel zu installieren, müssen Sie natürlich zuerst wissen, welche Pakete dieser Channel bereitstellt. Dazu können Sie ein Ihnen bereits bekanntes Kommando verwenden: `list-all` zeigt alle verfügbaren Pakete an, mit dem Schalter `-c` können Sie den Namen des Channels angeben, dessen Pakete Sie auflisten möchten:

```
% pear list-all -c pear.php-tools.net
All packages:
=============
Package              Latest    Local
pat/patBBCode        1.0.0beta           Versatile BBCode parser.
pat/patConfiguration 2.0.0b1             Interface for configuration files.
pat/patError         1.1.0               Simple and powerful error management package.
pat/patPortal                            MVC-driven framework
pat/patTemplate      3.0.1               Powerful templating engine.
```

Um nun eines dieser Pakete zu installieren, verwenden Sie das Kommando `install` und übergeben den Namen des Channels und des gewünschten Pakets:

```
% pear install pear.php-tools.net/patError
downloading patError-1.1.0.tgz ...
Starting to download patError-1.1.0.tgz (17,579 bytes)
......done: 17,579 bytes
install ok: channel://pear.php-tools.net/patError-1.1.0
```

Auch Pakete aus anderen Channels können Abhängigkeiten besitzen, diese werden beim Installieren direkt mitinstalliert, sofern Sie sie nicht zuvor schon installiert hatten. Hätten Sie also soeben nicht das Paket patError installiert, sondern direkt das Paket patTemplate installieren wollen, stellt der Installer fest, dass dafür das Paket patError benötigt wird, und installiert dieses automatisch:

```
% sudo pear install pear.php-tools.net/patTemplate
Did not download optional dependencies: pat/patBBCode, use --alldeps to download
automatically
pat/patTemplate can optionally use package "pat/patBBCode"
downloading patTemplate-3.0.1.tgz ...
Starting to download patTemplate-3.0.1.tgz (84,855 bytes)
...................done: 84,855 bytes
downloading patError-1.1.0.tgz ...
Starting to download patError-1.1.0.tgz (17,579 bytes)
...done: 17,579 bytes
install ok: channel://pear.php-tools.net/patError-1.1.0
install ok: channel://pear.php-tools.net/patTemplate-3.0.1
```

Wie Sie sehen, ist es auch hier möglich, optionale Abhängigkeiten zu definieren, die dann nicht automatisch heruntergeladen werden. Dabei können Abhängigkeiten problemlos auch Channel-übergreifend definiert werden: Beispielsweise ist es möglich, dass Pakete aus dem Channel *PHP Application Tools* Pakete von PEAR benötigen.

Möchten Sie sich Informationen zu einem Paket anzeigen lassen, müssen Sie auch hier zusätzlich zum Paketnamen den Namen des Channels übergeben:

```
% pear info pear.php-tools.net/patTemplate
About pear.php-tools.net/patTemplate-3.0.1
==========================================
Release Type           PEAR-style PHP-based Package
Name                   patTemplate
Channel                pear.php-tools.net
Summary                Powerful templating engine.
```

```
Description            patTemplate is a powerful, non-compiling
                       templating engine, that uses XML tags to divide
                       a document into different parts.
                       It provides different template types to emulate
                       if/else and switch/case constructs, variable
                       modifiers, input- and output filters and several more
                       useful features.
Maintainers            Stephan Schmidt <schst@php-tools.net> (lead)
                       Sebastian Mordziol <argh@php-tools.net>
                       (contributor)
Release Date           2005-03-07 21:20:42
Release Version        3.0.1 (stable)
API Version            3.0.1 (stable)
License                LGPL (http://www.gnu.org/copyleft/lesser.html)
Release Notes          Changes since 3.0.0:
                       - readTemplatesFromInput() now checks, whether
                       any input has been passed (reported by denne)
                       - fixed notice in addVars()
                       - improved Dump() for simplecondition templates
                       - fixed notice in GZip output filter
Required Dependencies  PHP version 4.2.0-6.0.0
                       PEAR installer version 1.4.0a1 or newer
                       Package pear.php-tools.net/patError
Optional Dependencies  Package pear.php-tools.net/patBBCode
package.xml version    2.0
Last Modified          2005-04-03 17:04
Last Installed Version - None -
```

Viele der existierenden Channels bieten bereits eine Website an, die weitere Informationen zum Channel und zu den verfügbaren Paketen bietet, es lohnt sich also, einfach mal die URL des Channels in einen Browser einzugeben, wie Abbildung 24-1 zeigt.

Siehe auch

Rezepte 24.3 und 24.5 zur Installation von PEAR- und PECL-Paketen; Rezept 24.9 zum Anbieten eines eigenen Channel-Servers; den PHP Application Tools-Channel unter *http://pear.php-tools.net*.

24.7 PEAR in Shared-Hosting-Umgebungen installieren

Problem

Sie möchten PEAR bei einem Shared-Hosting-Provider wie z.B. 1&1 verwenden.

Lösung

Nutzen Sie das Bootstrap-Skript von *http://pear.php.net/go-pear* und bedienen Sie es über das Web-Frontend. Wenn Sie bereits PEAR 1.4.0 oder höher installiert haben, können Sie die Remote-Install-Funktionalität nutzen.

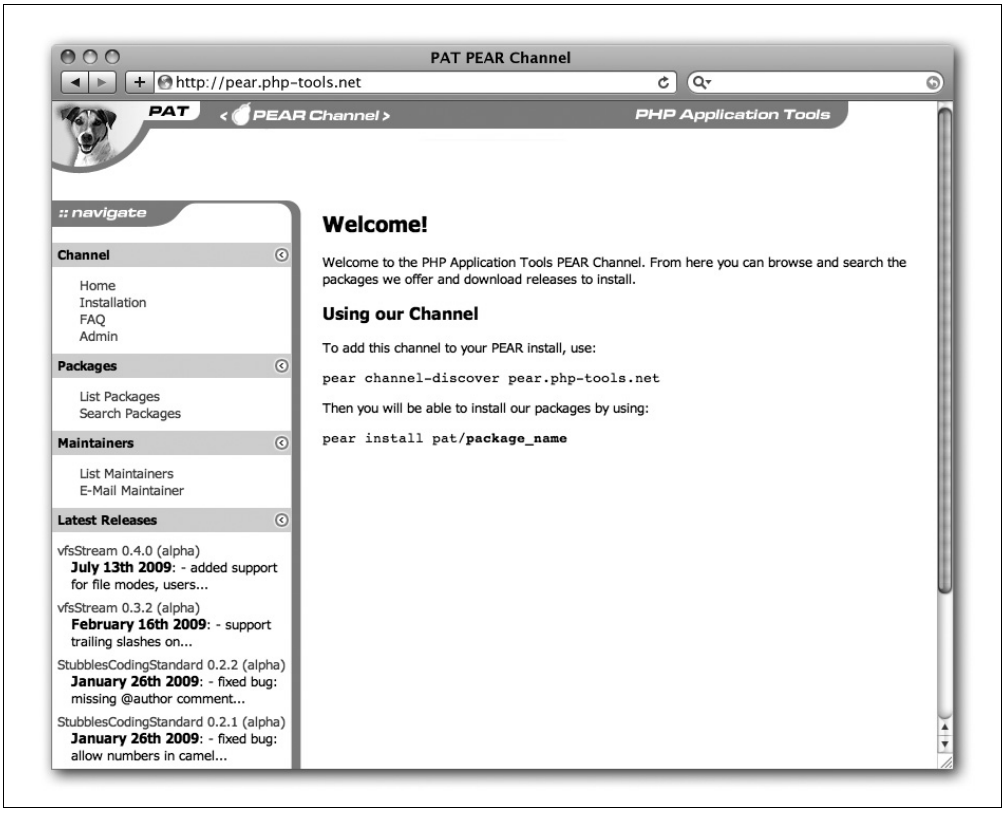


Abbildung 24-1: Das Web-Frontend des PAT-Channel-Servers

Diskussion

In Rezept 24.1 haben Sie gesehen, wie Sie PEAR sehr einfach über die Kommandozeile installieren können. Bei vielen Shared-Hosting-Anbietern haben Sie jedoch keinen Zugriff über SSH oder telnet, sondern lediglich per FTP. Aber auch hier können Sie *go-pear.org* verwenden. Öffnen Sie dazu die URL *http://pear.php.net/go-pear* in Ihrem Browser und speichern Sie die Datei danach auf Ihrem Rechner unter dem Namen *go-pear.php* ab. Diese Datei kopieren Sie nun per FTP in Ihren angemieteten Webspace und öffnen die URL, an die Sie die Datei kopiert haben, in Ihrem Browser. Abbildung 24-2 zeigt Ihnen den Willkommensbildschirm des Installers.

Klicken Sie hier einfach auf den mit *Next >>* beschrifteten Link, wodurch Sie zum nächsten Schritt der Installation gelangen, in dem Sie die Konfiguration Ihrer neuen PEAR-Installation vornehmen können. In nahezu allen Fällen reicht es auch hier, wenn Sie lediglich die Option *Installation Prefix* anpassen, da alle anderen Pfade relativ dazu angegeben werden. Per Default wird dieser Wert mit dem Verzeichnis vorbelegt, in dem sich die Datei *go-pear.php* befindet. Beachten Sie, dass Ihr Webserver Schreibrechte in dem

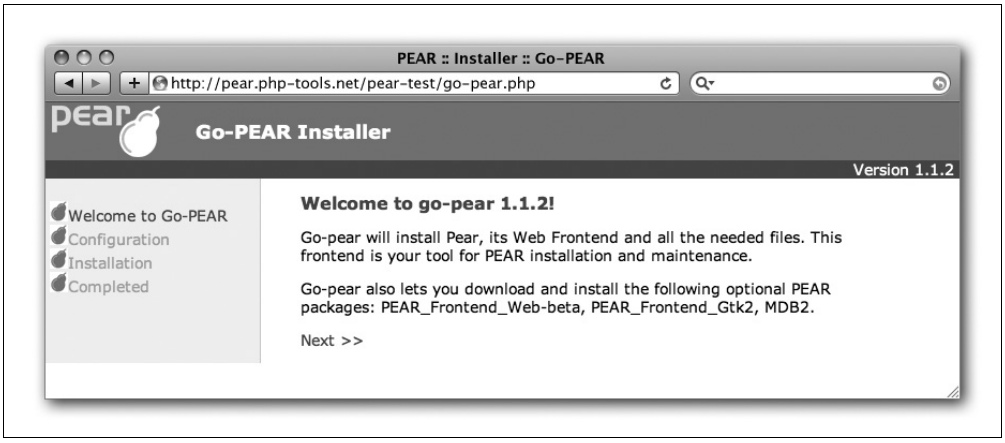


Abbildung 24-2: Das Web-Interface von Go-PEAR

Verzeichnis haben muss, in dem Sie PEAR installieren möchten. Abbildung 24-3 zeigt Ihnen den kompletten Konfigurationsbildschirm.

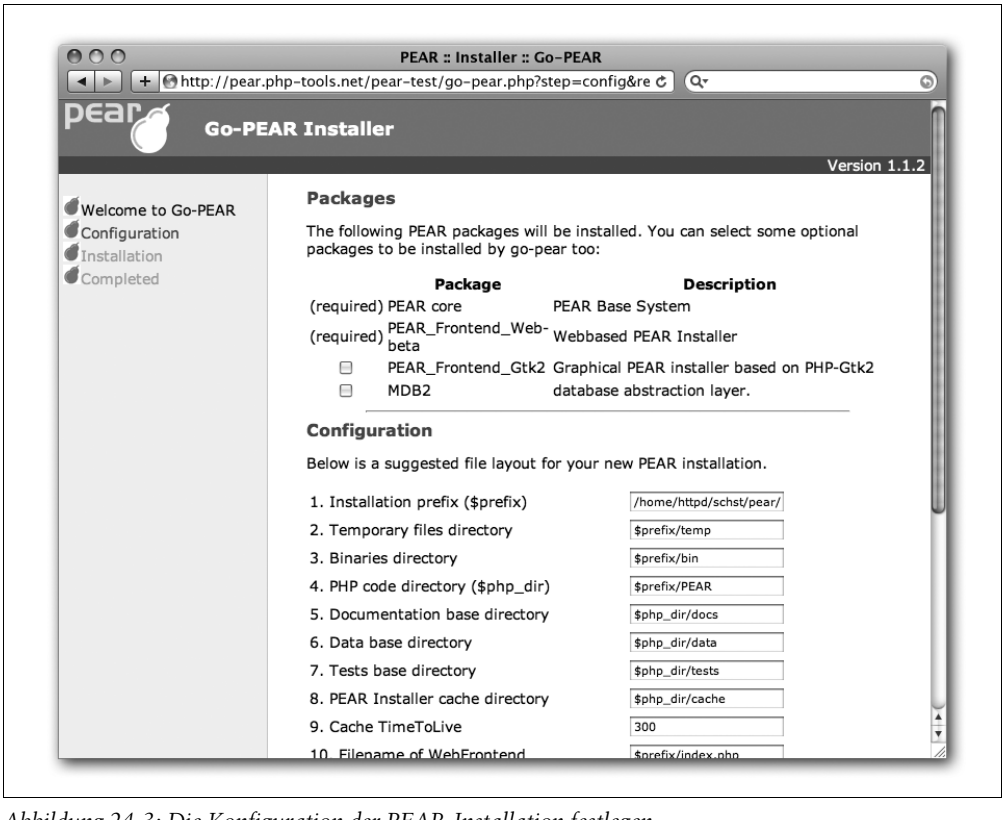


Abbildung 24-3: Die Konfiguration der PEAR-Installation festlegen

Um die Installation zu beginnen, reicht ein Klick auf den Button *Install*. Das Skript wird nun alle benötigten Pakete von der PEAR-Website downloaden und im angegebenen Verzeichnis installieren. Nach Beendigung dieses Vorgangs (siehe Abbildung 24-4) können Sie weitere Pakete über das PEAR-Installer-Web-Frontend installieren. Dazu öffnen Sie in Ihrem Browser einfach die URL *http://www.ihre-domain.de/pfad/zu/pear/index.php*.

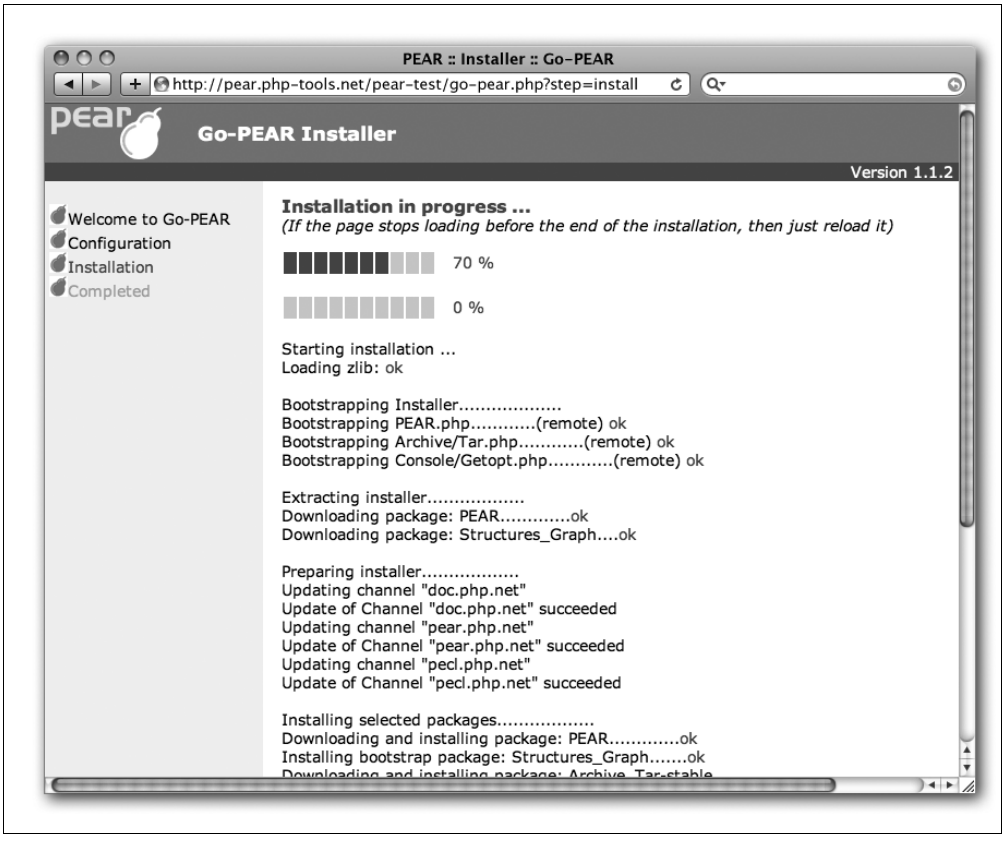


Abbildung 24-4: Download und Installation der Basispakete

Das PEAR-Web-Frontend erlaubt Ihnen, durch die verfügbaren Pakete zu blättern, Ihre PEAR-Konfiguration zu verändern sowie Pakete zu installieren und zu deinstallieren, Abbildung 24-5 zeigt einen Screenshot des Installers. Die einzelnen Funktionen sind selbsterklärend, so reicht z.B. ein Klick auf das Pluszeichen neben einem Paketnamen aus, um dieses zu installieren, durch einen Klick auf das Mülleimer-Symbol wird das Paket aus Ihrer Installation entfernt.

Damit Sie die installierten PEAR-Pakete auch in Ihren PHP-Anwendungen nutzen können, müssen Sie PHP zunächst noch den Pfad zur PEAR-Installation mitteilen. Die Pakete liegen unterhalb des bei der Installation angegebenen Pfades im Verzeichnis

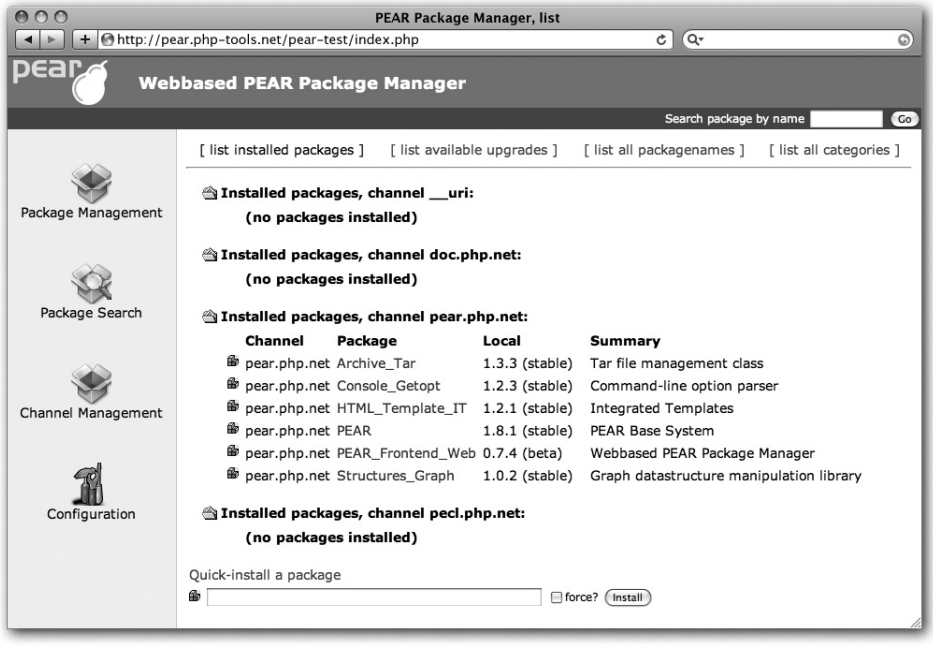


Abbildung 24-5: Das Web-Frontend des PEAR-Installers

PEAR, also z.B. in */homepages/12/d75103051/htdocs/pear-install/PEAR*. Um PHP diesen Pfad mitteilen zu können, gibt es zwei Wege:

1. Haben Sie Zugriff auf die *php.ini*-Konfigurationsdatei, können Sie den Pfad in der Option `include_path` eintragen.
2. Sollte das nicht möglich sein, können Sie die Option auch am Anfang Ihres Skripts mit der PHP-Funktion `ini_set()` setzen.

Bei den meisten Shared-Hosting-Anbietern haben Sie keine Möglichkeit, die PHP-Konfiguration zu ändern. Fügen Sie also einfach den folgenden Code in alle Ihre PHP-Skripten ein:

```
ini_set('include_path', './:/homepages/12/d75103051/htdocs/pear/PEAR');
```

Am einfachsten ist dies natürlich, wenn Sie eine Konfigurationsdatei nutzen, die auf jeder Seite eingebunden wird und in die Sie diese Anweisung eintragen können.

Mit dem Erscheinen von PEAR 1.4.0 gibt es noch eine zweite Möglichkeit, mit der Sie PEAR auf einem Webserver installieren können, zu dem Sie nur einen FTP-Zugang haben. Die sogenannte Remote-Install-Funktionalität erlaubt Ihnen, den Kommandozeilen-Installer auf Ihrem lokalen Rechner zu verwenden, um PEAR-Pakete auf dem eigentlichen Webserver zu installieren. Dazu werden diese einfach per FTP auf den Zielrechner

kopiert. Parallel dazu haben Sie die gleichen Dateien auf Ihrem lokalen Rechner, um Abhängigkeiten zu überprüfen.

Als Erstes müssen Sie den Kommandozeilen-Installer in der Version 1.4.0 oder höher auf Ihrem lokalen Rechner installiert haben. Sollten Sie noch keine lokale PEAR-Installation haben, zeigt Ihnen Rezept 24.1, wie Sie dazu kommen.

Danach verwenden Sie ein FTP-Programm, um sich auf dem Webserver anzumelden, und finden das Verzeichnis heraus, auf das Sie Zugriff haben (also Ihr Home-Verzeichnis). Eine einfache Möglichkeit, dies herauszufinden, ist z.B. das folgende PHP-Skript:

```
<?php
echo dirname(__FILE__);
?>
```

Bei vielen Anbietern ist dieses Verzeichnis z.B. etwas in der Art wie */kunden/homepages/14/d75103006* oder */home/benutzername*.

Nun müssen Sie die Konfigurationsdateien auf Ihrem lokalen Rechner vorbereiten. Wechseln Sie dazu in ein Verzeichnis, in dem Sie die lokale Kopie speichern wollen (z.B. */home/benutzername*), und führen Sie die folgenden Befehle aus:

```
% mkdir remote
% cd remote
% mkdir pear
% cd pear
% pear config-create /home/lokalernutzer remote.conf
% pear config-create /home/benutzername/pear .pearrc
```

Der Pfad */home/lokalernutzer* bezieht sich auf den Pfad zu Ihrem Home-Verzeichnis auf dem lokalen Rechner, der Pfad */home/benutzername/pear* ist der Pfad, unter dem PEAR auf dem Webserver installiert werden soll. Verwenden Sie Windows als lokalen Rechner, verwenden Sie die folgenden Befehle, um die Konfigurationsdateien zu erstellen:

```
C:\> mkdir remote
C:\> cd remote
C:\remote\> mkdir pear
C:\remote\> cd pear
C:\remote\pear> pear config-create -w C:\remote\pear remote.ini
C:\remote\pear> pear config-create /home/benutzername/pear .pearrc
```

Kopieren Sie nun die erzeugte *.pearrc*-Datei per FTP auf den Webserver in Ihr Home-Verzeichnis, also z.B. */home/benutzername/* oder */kunden/homepages/14/d75103006/*.

Nun müssen Sie in der lokalen Konfiguration *remote.ini* nur noch die Option `remote_config` auf den Pfad der Konfiguration auf dem Webserver setzen:

```
% pear -c remote.conf config-set remote_config ftp://user:pass@webserver.com/.pearrc
```

bzw. unter Windows:

```
C:\remote\pear\> pear -c remote.ini config-set remote_config ftp://user:pass@webserver.
com/.pearrc
```

Die Option -c wird verwendet, um dem PEAR Package Manager den Pfad zu einer anderen Konfigurationsdatei zu übergeben, schließlich möchten Sie ja keine Änderung an Ihrer lokalen PEAR-Konfiguration vornehmen.

Nun können Sie von Ihrem lokalen Rechner aus Pakete auf Ihrem Webserver installieren, als würden Sie den PEAR Package Manager dort verwenden. Sie müssen lediglich darauf achten, immer mit der Option -c die Konfiguration anzugeben, die Sie erstellt haben:

```
% pear -c remote.conf install Services_Ebay
```

Der PEAR Package Manager wird dabei auch alle Abhängigkeiten berücksichtigen und kopiert die installierten Dateien dann einfach per FTP auf den eigentlichen Webserver. Sie können diese Funktion also nutzen, sobald Sie einen FTP-Zugriff auf den Webserver haben.

Siehe auch

Rezept 24.1 zur Installation von PEAR.

24.8 Eigene PEAR-Pakete erstellen

Problem

Sie möchten selbst Pakete erstellen, die durch den PEAR Package Manager installiert werden können.

Lösung

Nutzen Sie das Paket PEAR_PackageFileManager, um eine *package.xml*-Datei zu erstellen, und verwenden Sie dann den PEAR Package Manager-Befehl `pear package`, um das Paket zu erstellen.

```
// PEAR_PackageFileManager einbinden.
require_once 'PEAR/PackageFileManager.php';

// Version
$version = '1.0.0';

// Release-State
$state = 'stable';

// Changelog der neuen Version
$notes = <<<EOT
Erste PEAR-Version.
EOT;

// Beschreibung des Pakets
$description = <<<EOT
```

```
Paket zum Erzeugen von RSS-Dateien aus PHP-Arrays.
EOT;

// Neuen PackageFileManager erstellen.
$package = new PEAR_PackageFileManager();

$result = $package->setOptions(array(
    'package'           => 'XML_RSS_Writer',
    'summary'           => 'Erzeugt RSS-Dateien.',
    'description'       => $description,
    'version'           => $version,
    'state'             => $state,
    'license'           => 'PHP License',
    'filelistgenerator' => 'file',
    'ignore'            => array( 'package.php', 'package.xml'),
    'notes'             => $notes,
    'simpleoutput'      => true,
    'baseinstalldir'    => 'XML',
    'packagedirectory'  => './',
    'dir_roles'         => array(
                                        'docs' => 'doc',
                                        'examples' => 'doc',
                                       'tests' => 'test',
                                    )
    )
);

if (PEAR::isError($result)) {
    echo $result->getMessage();
    die();
}

$package->addMaintainer('schst', 'lead', 'Stephan Schmidt', 'schst@php.net');
$package->addMaintainer('amt', 'contributor', 'Adam Trachtenberg', 'amt@php.net');

$package->addDependency('XML_Serializer', '', 'has', 'pkg', false);
$package->addDependency('php', '5.0.0', 'ge', 'php', false);

if (isset($_SERVER['argv'][1]) && $_SERVER['argv'][1] == 'make') {
    $result = $package->writePackageFile();
} else {
    $result = $package->debugPackageFile();
}

if (PEAR::isError($result)) {
    echo $result->getMessage();
    die();
}
```

Speichern Sie diese Datei unter dem Namen *package.php* ab und rufen Sie sie folgendermaßen auf:

```
% php package.php
```

Der PEAR_PackageFileManager erzeugt nun den XML-Code, der Ihr PEAR-Paket beschreibt. Sind Sie mit dem Ergebnis zufrieden, können Sie durch den Aufruf von

```
% php package.php make
```

die *package.xml*-Datei schreiben lassen. Um nun ein installierbares Paket zu erzeugen, verwenden Sie den Befehl `make` des PEAR Package Manager:

```
% pear package
Analyzing RSS/Writer.php
Package XML_RSS_Writer-1.0.0.tgz done
```

Dieses Paket können Sie nun auf Ihrer Webseite zum Download anbieten, und Ihre Kunden können es über den Befehl `install` installieren:

```
% pear install XML_RSS_Writer-1.0.0.tgz
install ok: XML_RSS_Writer-1.0.0
```

Diskussion

Der PEAR Package Manager kann nicht nur verwendet werden, um PEAR-Pakete zu installieren. Sie haben ebenfalls die Möglichkeit, selbst PEAR-kompatible Pakete zu erstellen, die dann von anderen Entwicklern einfach über den PEAR Package Manager installiert werden können. Als Erstes ist es dabei wichtig, dass das Verzeichnis-Layout Ihrer Pakete stimmt.

Nehmen Sie an, Sie haben ein Paket entwickelt, mit dem sehr einfach RSS-Dateien erstellt werden können, um eigene Inhalte als RSS-Stream anzubieten. Ein passender Name für das Paket wäre z.B. *XML_RSS_Writer*. Das Paket enthält nur die Klasse `XML_RSS_Writer`, die nach den PEAR-Regeln in der Datei *XML/RSS/Writer.php* zu finden sein sollte. Weiterhin enthält Ihr Paket noch zwei Beispiele, zwei Unit-Tests und eine *README*-Datei.

Damit Ihr Paket PEAR-konform ist, sollten Sie die Verzeichnisstruktur wie folgt aufbauen:

```
docs/
docs/README
examples
examples/example1.php
examples/example2.php
tests/test1.phpt
tests/test2.phpt
XML/
XML/RSS
XML/RSS/Writer.php
```

Um diese Dateien nun in ein PEAR-Paket zu verwandeln, genügt es nicht, sie in ein TAR-Archiv zu packen, da dadurch keine Informationen über die Dateien und das Paket mitgeliefert würden. Diese Informationen stehen in einem PEAR-Paket in der Datei *package.xml*. Das Erstellen dieser Datei ist äußerst komplex, deshalb bietet PEAR das Paket PEAR_PackageFileManager, das beim Erzeugen der Datei hilft.

Nach dem Einbinden der Klasse und dem Instantiieren eines Objekts setzen Sie mit der Methode `setOptions()` verschiedene Optionen für die *package.xml*-Datei:

```
$result = $package->setOptions(array(
    'package'           => 'XML_RSS_Writer',
    'summary'           => 'Erzeugt RSS-Dateien.',
    'description'       => $description,
    'version'           => $version,
    'state'             => $state,
    'license'           => 'PHP License',
    'filelistgenerator' => 'file',
    'packagedirectory'  => './',
    'ignore'            => array( 'package.php', 'package.xml'),
    'notes'             => $notes,
    'simpleoutput'      => true,
    'baseinstalldir'    => 'XML',
    'dir_roles'         => array(
                                        'docs' => 'doc',
                                        'examples' => 'doc',
                                        'tests' => 'test',
                                    )
    )
);
```

`package` ist der Name des Paketes, `summary` und `description` enthalten eine kurze und lange Beschreibung, `version`, `state` und `license` sind selbsterklärend. Mit der Option `filelistgenerator` können Sie wählen, wie der PackageFileManager die Liste der zum Paket gehörenden Dateien ermittelt. Wird diese Option auf `file` gesetzt, werden alle Dateien, die sich im Dateisystem befinden, hinzugefügt. Wird sie auf `cvs` gesetzt, werden CVS-Metadaten ignoriert. Die Option `packagedirectory` gibt das Verzeichnis an, in dem sich die Dateien befinden. Mit der Option `ignore` können Sie eine Liste mit Dateinamen übergeben, die nicht in das Paket übernommen werden sollen.

Die Option `simpleoutput` definiert, dass Sie eine einfache Version der *package.xml*-Datei erzeugen möchten. Verwenden Sie hier den Wert `true`, erzeugt der PackageFileManager auch MD5-Summen der einzelnen Dateien und fügt weitere Meta-Informationen zum Paket hinzu. `baseinstalldir` gibt an, wo das Paket später beim Nutzer installiert werden soll. Da das Paket XML_RSS_Writer heißt, soll es natürlich im *XML*-Verzeichnis installiert werden. Die letzte Option, `dir_roles`, legt fest, wie die einzelnen Dateien behandelt werden sollen: Alles was in den Verzeichnissen `docs` und `examples` gefunden wird, wird als Dokumentation behandelt, alles im Ordner `tests` ist ein Unit-Test. Alle anderen Dateien werden als normaler, zum Paket gehörender Quellcode betrachtet.

Nachdem Sie auf Fehler beim Setzen der Optionen geprüft haben, werden die einzelnen Entwickler hinzugefügt:

```
$package->addMaintainer('schst', 'lead', 'Stephan Schmidt', 'schst@php.net');
$package->addMaintainer('amt', 'contributor', 'Adam Trachtenberg', 'amt@php.net');
```

Die Methode `addMaintainer()` erwartet vier Werte:

- Den Usernamen des Entwicklers.
- Die Funktion des Entwicklers (`lead`, `developer`, `contributor` oder `helper`).
- Den vollständigen Namen des Entwicklers.
- Die E-Mail-Adresse des Entwicklers.

Einem Paket können unbegrenzt Entwickler zugeordnet werden. Nach den Entwicklern fügen Sie schließlich noch Paket-Abhängigkeiten hinzu:

```
$package->addDependency('XML_Serializer', '', 'has', 'pkg', false);
$package->addDependency('php', '5.0.0', 'ge', 'php', false);
```

Dabei wird für Abhängigkeiten von PHP-Versionen, -Erweiterungen oder anderen PEAR-Paketen immer dieselbe Methode `addDependency()` verwendet, die die folgenden Parameter erwartet:

- Den Namen des PEAR-Pakets oder der PHP-Erweiterung. Handelt es sich um eine Abhängigkeit zu einer PHP-Version, wird hier `php` übergeben.
- Die Version der Abhängigkeit.
- Die Relation zur Versionsnummer, z.B. `"has"`, wenn das Paket einfach nur vorhanden sein muss, `"ge"`, wenn eine Version größer oder gleich der angegebenen Version installiert sein muss.
- Der Typ der Abhängigkeit, z.B. `"pkg"` für ein PEAR-Paket, `"ext"` für eine PHP-Erweiterung oder `"php"` für eine PHP-Version.
- Ein Flag, das angibt, ob die Abhängigkeit optional (`true`) oder nicht (`false`) ist.

Wenn Sie das Paket PHP_CompatInfo installiert haben, können Sie auch dieses Paket automatisch die benötigte PHP-Versionsnummer und die Erweiterungen ermitteln lassen:

```
$package->detectDependencies();
```

Haben Sie alle Abhängigkeiten hinzugefügt, lassen Sie sich das erzeugte XML einfach mit der Methode `debugPackageFile()` ausgeben:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE package SYSTEM "http://pear.php.net/dtd/package-1.0">
<package version="1.0" packagerversion="1.4.0a10">
 <name>XML_RSS_Writer</name>
 <summary>Erzeugt RSS-Dateien.</summary>
 <description>Paket zum Erzeugen von RSS-Dateien aus PHP-Arrays.
 </description>
 <maintainers>
  <maintainer>
   <user>schst</user>
   <name>Stephan Schmidt</name>
   <email>schst@php-tools.net</email>
   <role>lead</role>
  </maintainer>
  <maintainer>
   <user>amt</user>
```

```
  <name>Adam Trachtenberg</name>
  <email>amt@php.net</email>
  <role>contributor</role>
 </maintainer>
 </maintainers>
<release>
 <version>1.0.0</version>
 <date>2005-04-10</date>
 <license>PHP License</license>
 <state>stable</state>
 <notes>Erste PEAR-Version.
 </notes>
 <deps>
  <dep type="pkg" rel="has" optional="no">XML_Serializer</dep>
  <dep type="php" rel="ge" version="5.0.0" optional="no"/>
 </deps>
 <filelist>
  <dir name="/" baseinstalldir="XML">
   <dir name="docs">
    <file role="doc" name="README"/>
   </dir> <!-- /docs -->
   <dir name="examples">
    <file role="doc" name="example1.php"/>
    <file role="doc" name="example2.php"/>
   </dir> <!-- /examples -->
   <dir name="RSS">
    <file role="php" name="Writer.php"/>
   </dir> <!-- /RSS -->
   <dir name="tests">
    <file role="test" name="test1.phpt"/>
    <file role="test" name="test2.phpt"/>
   </dir> <!-- /tests -->
   <file role="data" name="package.php~"/>
  </dir> <!-- / -->
 </filelist>
</release>
<changelog>
  <release>
   <version>1.0.0</version>
   <date>2005-04-10</date>
   <license>PHP License</license>
   <state>stable</state>
   <notes>Erste PEAR-Version.
   </notes>
  </release>
 </changelog>
</package>
```

Dieses XML-Dokument enthält nun alle Informationen zum Paket (wie Beschreibung, Version usw.), zu den enthaltenen Dateien, den Abhängigkeiten zu anderen Paketen und Erweiterungen sowie zu den beteiligten Entwicklern. Nachdem Sie das XML-Dokument mit der Methode `writePackageFile()` gespeichert haben, sollte Ihr Arbeitsverzeichnis die Datei *package.xml* enthalten. Um nun daraus ein PEAR-Paket zu machen, verwenden Sie den PEAR Package Manager:

```
% pear package
Analyzing RSS/Writer.php
Package XML_RSS_Writer-1.0.0.tgz done
```

Möchten Sie ein Paket erstellen, für dessen Installation keine `zlib`-Erweiterung benötigt wird, verwenden Sie stattdessen:

```
% pear package --nocompress
Analyzing RSS/Writer.php
Package XML_RSS_Writer-1.0.0.tar done
```

Sie haben nun ein PEAR-Paket erzeugt, das Sie weitergeben können und das andere Benutzer sehr einfach über den PEAR Package Manager installieren können:

```
% pear install XML_RSS_Writer-1.0.0.tgz
install ok: XML_RSS_Writer-1.0.0
```

Siehe auch

Rezept 24.9 zum Aufsetzen eines eigenen Channel-Servers; die Dokumentation des PEAR_PackageFileManager unter *http://pear.php.net/manual/en/package.pear.pear-package-filemanager.php*; die Dokumentation des *package.xml*-Formats unter *http://pear.php.net/manual/de/developers.packagedef.php*.

24.9 Eigene Pakete über einen Channel-Server vertreiben

Problem

Sie möchten Ihre eigenen Pakete PEAR-Benutzern zur Installation anbieten und dabei auch Funktionen wie das automatische Überprüfen von Updates unterstützen.

Lösung

Nutzen Sie das Paket Chiara/Chiara_PEAR_Server von Greg Beaver, um einen eigenen Channel-Server zu betreiben. Mit dem Paket Crxt/Crtx_PEAR_Server_Frontend können Sie Ihren Benutzern automatisch eine Website mit Download-Möglichkeit und Changelog anbieten.

Als Erstes muss Ihre PEAR-Installation den Channel kennen, der den Server anbietet:

```
% pear channel-discover pear.chiaraquartet.net
Adding Channel "pear.chiaraquartet.net" succeeded
Discovery of channel "pear.chiaraquartet.net" succeeded
```

Danach installieren Sie das Paket:

```
% pear install -a chiara/Chiara_PEAR_Server
downloading Chiara_PEAR_Server-0.17.0.tgz ...
Starting to download Chiara_PEAR_Server-0.17.0.tgz (31,406 bytes)
.........done: 31,406 bytes
```

```
downloading Chiara_XML_RPC5-0.3.0.tgz ...
Starting to download Chiara_XML_RPC5-0.3.0.tgz (8,202 bytes)
...done: 8,202 bytes
install ok: channel://pear.chiaraquartet.net/Chiara_XML_RPC5-0.3.0
install ok: channel://pear.chiaraquartet.net/Chiara_PEAR_Server-0.17.0
Chiara_PEAR_Server: Optional feature pearweb available (Public frontend for users to
browse channel packages)
To install use "pear install Chiara_PEAR_Server#featurename"
chiara/Chiara_PEAR_Server has post-install scripts:
/usr/share/pear/Chiara/PEAR/Server/mysqlinstall.php
Use "pear run-scripts chiara/Chiara_PEAR_Server" to run
DO NOT RUN SCRIPTS FROM UNTRUSTED SOURCES
```

Das Paket liefert ein sogenanntes Post-Install-Skript mit, das die Datenbank einrichtet und alle benötigten Dateien erzeugt:

```
% pear run-scripts chiara/Chiara_PEAR_Server
Including external post-installation script "/usr/share/pear/Chiara/PEAR/Server/
mysqlinstall.php" - any errors are in this script
Inclusion succeeded
running post-install script "Server_mysqlinstall_postinstall->init()"
init succeeded
Create/Upgrade database for PEAR_Server? All previous installations must run the script
[yes] :
Your choices:
Create/Upgrade database for PEAR_Server? All previous installations must run the script:
yes
These Choices OK? (use "abort" to halt) [yes] :
Mysql database to create [pear] : pearchannel
Mysql Username (must have create permision) [pear] : root
Mysql password [pear] : ******
Your choices:
Mysql database to create: pearchannel
Mysql Username (must have create permision): root
Mysql password: ******
These Choices OK? (use "abort" to halt) [yes] :
Creating tables succeeded
Channel Name (server uri like pear.php.net) : ihre-domain.de
Suggested Channel Alias (like pear) : cookbook
Channel summary : Test-Channel des PHP Kochbuchs
Your choices:
Channel Name (server uri like pear.php.net): ihre-domain.de
Suggested Channel Alias (like pear): cookbook
Channel summary: Test-Channel des PHP Kochbuchs
These Choices OK? (use "abort" to halt) [yes] :
adding channel to local registry
Channel Administrator Real Name : Stephan Schmidt
Channel Administrator Handle (package.xml user tag) : schst
Channel Administrator Email : schst@php-tools.net
Channel Administrator Password : ****
Your choices:
Channel Administrator Real Name: Stephan Schmidt
Channel Administrator Handle (package.xml user tag): schst
Channel Administrator Email: schst@php-tools.net
```

```
Channel Administrator Password: ****
These Choices OK? (use "abort" to halt) [yes] :
Path to document root (htdocs or public_html) : /pfad/zum/webspace
name of frontend.php HTML admin frontend file : admin.php
temporary path to save uploaded releases in : /tmp
Your choices:
Path to document root (htdocs or public_html): /pfad/zum/webspace
name of frontend.php HTML admin frontend file: admin.php
temporary path to save uploaded releases in: /tmp
These Choices OK? (use "abort" to halt) [yes] :
Successfully created /pfad/zum/webspace/xmlrpc.php
Successfully created /pfad/zum/webspace/admin.php
Successfully created /pfad/zum/webspace/pear_server.css
Install scripts complete
```

Der Server ist nun einsatzbereit. Möchten Sie auch noch ein Web-Frontend für Ihre Benutzer installieren, sind weitere Schritte nötig. Zunächst müssen Sie einen weiteren Channel zu Ihrem Package Manager hinzufügen:

```
% pear channel-discover pear.crtx.org
Adding Channel "pear.crtx.org" succeeded
Discovery of channel "pear.crtx.org" succeeded
```

Danach installieren Sie das Paket *Crtx_PEAR_Server_Frontend* von dem gerade hinzugefügten Channel:

```
% pear install crtx/Crtx_PEAR_Channel_Frontend
downloading Crtx_PEAR_Channel_Frontend-0.1.2.tgz ...
Starting to download Crtx_PEAR_Channel_Frontend-0.1.2.tgz (11,250 bytes)
.....done: 11,250 bytes
install ok: channel://pear.crtx.org/Crtx_PEAR_Channel_Frontend-0.1.2
```

Schließlich müssen Sie lediglich noch eine Datei *index.php* erstellen, die den folgenden Code enthält:

```
<?php
require_once 'Crtx/PEAR/Channel/Frontend.php';

$frontend = new Crtx_PEAR_Channel_Frontend(
                          'ihre-domain.de',
                          array(
                                'database' => 'mysql://root:******@localhost/
                                              pearchannel',
                                'index' => 'index.php',
                                'admin' => 'admin.php'));
?>
<html>
    <head>
        <title>PHP Kochbuch Testchannel</title>
        <link rel="stylesheet" type="text/css" href="pear_server.css" />
        <?php
            $frontend->showLinks();
        ?>
    </head>
    <body>
```

```
        <div id="top">
            <h1><a href="index">PHP Kochbuch Testchannel</a></h1>
        </div>
        <div id="menu">
            <?php
            $frontend->showMenu();
            ?>
            <div id="releases">
                <?php
                    $frontend->showLatestReleases();
                ?>
            </div>
        </div>
        <div id="content">
            <?php
            if (!$frontend->run()) {
                $frontend->welcome();
              }
            ?>
        </div>
    </body>
</html>
```

Diskussion

Seit der Version 1.4.0 bietet der PEAR Package Manager Unterstützung für Channels. Bei den vorherigen PEAR-Versionen konnte der Package Manager immer nur mit dem PEAR-Server auf *pear.php.net* kommunizieren und auch nur von dort Pakete installieren. Channels erlauben es nun, dass der Package Manager mit beliebig vielen Servern kommuniziert und auch von beliebig vielen Servern neue Pakete installiert. Wie der Package Manager mit verschiedenen Channels arbeiten kann, ist Thema des Rezepts 24.6.

Möchten Sie selbst einen Channel anbieten, können Sie dazu verschiedene Pakete nutzen, die bereits von anderen Channels zur Verfügung gestellt werden und die Ihnen die passende Infrastruktur bereitstellen. Als Erstes benötigen Sie dazu das Paket Chiara_PEAR_Server, das Sie vom Channel *Chiara* bekommen können. Dieses Paket entstammt der Feder von Greg Beaver, dem Entwickler, der für die neuen Features von PEAR 1.4.0 verantwortlich ist. Das Paket kann einfach über den PEAR Package Manager installiert werden:

```
% pear install -a chiara/Chiara_PEAR_Server
downloading Chiara_PEAR_Server-0.17.0.tgz ...
Starting to download Chiara_PEAR_Server-0.17.0.tgz (31,406 bytes)
.........done: 31,406 bytes
downloading Chiara_XML_RPC5-0.3.0.tgz ...
Starting to download Chiara_XML_RPC5-0.3.0.tgz (8,202 bytes)
...done: 8,202 bytes
install ok: channel://pear.chiaraquartet.net/Chiara_XML_RPC5-0.3.0
install ok: channel://pear.chiaraquartet.net/Chiara_PEAR_Server-0.17.0
Chiara_PEAR_Server: Optional feature pearweb available (Public frontend for users to
browse channel packages)
```

```
To install use "pear install Chiara_PEAR_Server#featurename"
chiara/Chiara_PEAR_Server has post-install scripts:
/usr/share/pear/Chiara/PEAR/Server/mysqlinstall.php
Use "pear run-scripts chiara/Chiara_PEAR_Server" to run
DO NOT RUN SCRIPTS FROM UNTRUSTED SOURCES
```

Der Package Manager macht Sie nach der Installation darauf aufmerksam, dass das Paket ein Post-Install-Skript zur Verfügung stellt. Post-Install-Skripten sind auch ein neues Feature von PEAR 1.4.0, die es ermöglichen, komplette Applikationen, die durch Interaktion mit dem Benutzer konfiguriert werden müssen, durch den PEAR Package Manager zu installieren. Dazu können PEAR-Pakete PHP-Skripten enthalten, die dann einfach über den PEAR-Befehl ausgeführt werden können. Dazu müssen Sie lediglich den Befehl

```
% pear run-scripts chiara/Chiara_PEAR_Server
```

ausführen. Der PEAR Package Manager wird Sie nun nach Konfigurationsoptionen fragen, die benötigt werden, um einen neuen PEAR-Server zu installieren. Die Installation ist in einzelne Gruppen aufgeteilt, die der Reihe nach abgearbeitet werden. Am Ende einer Gruppe können Sie noch einmal alle zuvor gemachten Angaben bestätigen oder ändern. Die folgenden Gruppen sind für die Installation notwendig:

1. *Zugangsdaten für die MySQL-Datenbank:* Hier müssen Sie den Namen der zu verwendenden Datenbank sowie Zugangsdaten für den passenden Benutzer angeben. Achten Sie darauf, dass der Benutzer Tabellen erstellen kann. Wegen eines Bugs in Chiara_PEAR_Server sollte der Datenbankname keinen Bindestrich (–) enthalten.
2. *Channel-Informationen:* Hier müssen Sie den Servernamen sowie eine Beschreibung des neuen Channels angeben.
3. *Channel-Administrator:* Hier werden Informationen wie Name, Passwort und E-Mail-Adresse des Channel-Server-Administrators benötigt.
4. *Webserver-Konfiguration:* Hier müssen Sie Pfade zum Document-Root des Webservers sowie eines temporären Verzeichnisses angeben.

Haben Sie alle Konfigurationsoptionen bestätigt, reagiert der PEAR Package Manager mit einer Erfolgsmeldung der folgenden Art:

```
Successfully created /pfad/zum/webspace/xmlrpc.php
Successfully created /pfad/zum/webspace/admin.php
Successfully created /pfad/zum/webspace/pear_server.css
Install scripts complete
```

Sie können die Administration des Channel-Servers nun über das Web-Frontend fortsetzen, indem Sie die URL *http://www.ihre-domain.de/admin.php* in einem Webbrowser öffnen. Es erscheint nun ein Login-Bildschirm, auf dem Sie sich mit den zuvor bei der Installation angegebenen Daten anmelden können. Abbildung 24-6 zeigt Ihnen, wie die Administration nach erfolgreichem Login aussehen sollte.

Um nun das in Rezept 24.8 erstellte Paket XML_RSS_Writer über diesen Channel-Server zu veröffentlichen, sind die folgenden Schritte nötig.

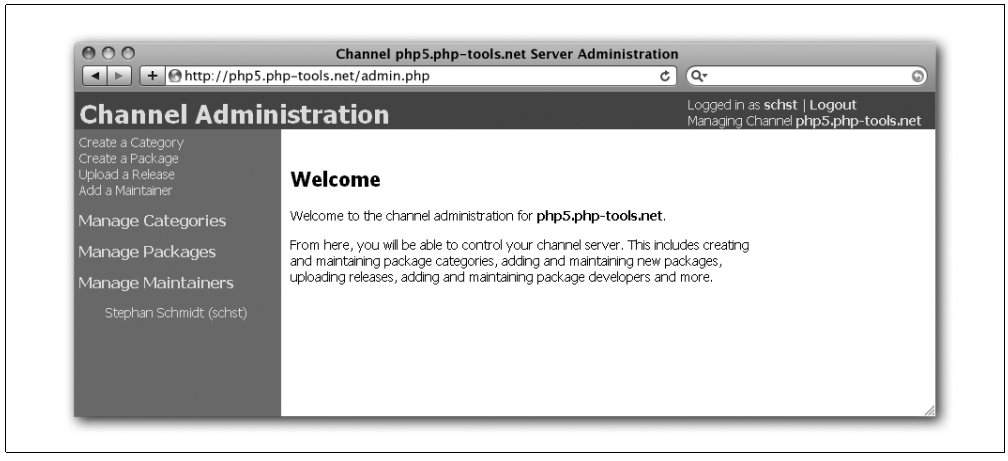


Abbildung 24-6: Die Administration Ihres Channel-Servers

Als Erstes müssen Sie das Paket anlegen. Dazu klicken Sie in der linken Navigation auf den Link *Create a package* und geben die Daten des Pakets ein. Zwingend erforderlich sind nur die Felder *Name*, *License*, *Summary* und *Description*. Nachdem Sie das neue Paket durch Klicken auf den Button *Save Changes* angelegt haben, sollte es in der linken Navigation unter dem Punkt *Manage Packages* auftauchen. Da bisher noch keine Entwickler für das Paket registriert sind, klicken Sie auf den Link *Maintainers* direkt unter dem Namen des Pakets. Hier können Sie nun Entwickler für das Paket hinzufügen. Lediglich Entwickler, die als *Lead* hinzugefügt wurden, haben auch die Berechtigung, neue Versionen zu veröffentlichen. Damit Sie also eine erste Version veröffentlichen können, müssen Sie sich hier als aktiver Lead-Entwickler eintragen und dies durch einen Klick auf *Add Maintainer* bestätigen.

Um nun zu testen, ob Ihr Channel-Server bereits korrekt arbeitet, können Sie ihn Ihrem lokalen PEAR Package Manager bekannt machen:

```
% pear channel-discover ihre-domain.de
Adding Channel "ihre-domain.de" succeeded
Discovery of channel "ihre-domain.de" succeeded
```

Lassen Sie sich nun auch Informationen zu diesem Channel anzeigen:

```
% pear channel-info cookbook
CHANNEL IHRE-DOMAIN.de INFORMATION:
===================================
Name and Server         ihre-domain.de
Alias                   cookbook
Summary                 Test-Channel des PHP Kochbuchs
Validation Package Name PEAR_Validate
Validation Package      default
Version
SERVER CAPABILITIES
===================
```

```
TYPE   VERSION FUNCTION NAME              URI
xmlrpc 1.0     logintest
xmlrpc 1.0     package.listLatestReleases
xmlrpc 1.0     package.listAll
xmlrpc 1.0     package.info
xmlrpc 1.0     package.getDownloadURL
xmlrpc 1.1     package.getDownloadURL
xmlrpc 1.0     package.getDepDownloadURL
xmlrpc 1.1     package.getDepDownloadURL
xmlrpc 1.0     package.search
xmlrpc 1.0     channel.listAll
```

Möchten Sie wissen, ob der Channel auch schon Pakete anbietet, nutzen Sie das Kommando `list-all`:

```
% pear list-all -c cookbook
ALL PACKAGES:
=============
PACKAGE              LATEST LOCAL
cookbook/XML_RSS_Writer             Erzeugt RSS-Dateien.
```

Natürlich kann das Paket noch nicht installiert werden, da Sie bisher nur die Paket-Daten eingegeben, aber noch keine Version veröffentlicht haben. Dazu müssen Sie das Paket erst wie in Rezept 24.8 beschrieben vorbereiten. Bevor Sie jedoch mit dem Kommando `package` das Archiv erzeugen, ist noch ein weiterer Schritt nötig.

Mit der Einführung von PEAR 1.4.0 und der Channel-Funktionalität waren die Informationen, die in der Datei *package.xml* gespeichert wurden, nicht mehr ausreichend. Aus diesem Grund wurde eine neue Datei, *package2.xml*, eingeführt, die unter anderem auch Informationen zum Channel speichern kann. Um den Aufwand zum Erzeugen der zweiten Datei so gering wie möglich zu halten, kann das PEAR-Kommando `convert` verwendet werden, das die Datei *package.xml* einliest und auf Basis der darin enthaltenen Informationen eine *package2.xml*-Datei erstellt.

```
% pear convert
Wrote new version 2.0 package.xml to "./package2.xml"

<?xml version="1.0"?>
<package packagerversion="1.4.0a10" version="2.0" xmlns="http://pear.php.net/dtd/package-
2.0" xmlns:tasks="http://pear.php.net/dtd/tasks-1.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://pear.php.net/dtd/tasks-1.0
http://pear.php.net/dtd/tasks-1.0.xsd
http://pear.php.net/dtd/package-2.0
http://pear.php.net/dtd/package-2.0.xsd">
 <name>XML_RSS_Writer</name>
 <channel>pear.php.net</channel>
 <summary>Erzeugt RSS-Dateien.</summary>
 <description>Paket zum Erzeugen von RSS-Dateien aus PHP-Arrays.</description>
 <lead>
  <name>Stephan Schmidt</name>
  <user>schst</user>
  <email>schst@php-tools.net</email>
```

```
 <active>yes</active>
</lead>
<contributor>
 <name>Adam Trachtenberg</name>
 <user>amt</user>
 <email>amt@php.net</email>
 <active>yes</active>
</contributor>
<date>2005-04-10</date>
<time>17:53:58</time>
<version>
 <release>1.0.0</release>
 <api>1.0.0</api>
</version>
<stability>
 <release>stable</release>
 <api>stable</api>
</stability>
<license uri="http://www.php.net/license">PHP License</license>
<notes>Erste PEAR-Version.</notes>
<contents>
 <dir name="/">
  <dir name="docs">
   <file baseinstalldir="XML" name="README" role="doc" />
  </dir> <!-- /docs -->
  <dir name="examples">
   <file baseinstalldir="XML" name="example1.php" role="doc" />
   <file baseinstalldir="XML" name="example2.php" role="doc" />
  </dir> <!-- /examples -->
  <dir name="RSS">
   <file baseinstalldir="XML" name="Writer.php" role="php" />
  </dir> <!-- /RSS -->
  <dir name="tests">
   <file baseinstalldir="XML" name="test1.phpt" role="test" />
   <file baseinstalldir="XML" name="test2.phpt" role="test" />
  </dir> <!-- /tests -->
 </dir> <!-- / -->
</contents>
<dependencies>
 <required>
  <php>
   <min>5.0.0</min>
   <max>6.0.0</max>
  </php>
  <pearinstaller>
   <min>1.4.0a1</min>
  </pearinstaller>
  <package>
   <name>XML_Serializer</name>
   <channel>pear.php.net</channel>
  </package>
 </required>
```

```
 </dependencies>
 <phprelease />
 <changelog>
  <release>
   <version>
    <release>1.0.0</release>
    <api>1.0.0</api>
   </version>
   <stability>
    <release>stable</release>
    <api>stable</api>
   </stability>
   <date>2005-04-10</date>
   <license uri="http://www.php.net/license">PHP License</license>
   <notes>Erste PEAR Version.</notes>
  </release>
 </changelog>
</package>
```

Diese Datei enthält zwar weitaus mehr Informationen und Tags als die Ihnen bekannte *package.xml*-Datei, jedoch ist für die aktuelle Aufgabe nur das Tag `<channel>pear.php.net</channel>` von Bedeutung. Dieses Tag beinhaltet den Namen des Channels, über den das Paket vertrieben wird. Da das Paket XML_RSS_Writer kein offizielles PEAR-Paket ist, sondern über Ihren Channel vertrieben werden soll, müssen Sie dieses Tag noch von Hand in `<channel>ihre-domain.de</channel>` ändern.

Danach können Sie das PEAR-Paket erzeugen, jedoch ist der Aufruf etwas anders, da Sie beide XML-Dateien übergeben müssen:

```
% pear package package.xml package2.xml
Attempting to process the second package file
Analyzing RSS/Writer.php
Package XML_RSS_Writer-1.0.0.tgz done
```

Um das erstellte Paket nun zu veröffentlichen, klicken Sie in der webbasierten Administration des Channel-Servers auf den Link *Upload a release* und wählen auf der Folgeseite das erstellte PEAR-Paket aus. Nach dem Datei-Upload sollten Sie eine Erfolgsmeldung erhalten, und das Paket ist nun für die Öffentlichkeit erreichbar. Dadurch kann das Paket mit nur einer Anweisung heruntergeladen und installiert werden:

```
% pear install cookbook/XML_RSS_Writer
downloading XML_RSS_Writer-1.0.0.tgz ...
Starting to download XML_RSS_Writer-1.0.0.tgz (2,357 bytes)
............done: 2,357 bytes
install ok: channel://ihre-domain.de/XML_RSS_Writer-1.0.0
```

Möchten Sie Ihren Benutzern auch ein Web-Frontend bieten, über das sie sich über Ihren Channel und die aktuellen Veröffentlichungen informieren können, verwenden Sie einfach das Paket Crtx_PEAR_Server_Frontend:

```
% pear install crtx/Crtx_PEAR_Channel_Frontend
downloading Crtx_PEAR_Channel_Frontend-0.1.2.tgz ...
Starting to download Crtx_PEAR_Channel_Frontend-0.1.2.tgz (11,250 bytes)
.....done: 11,250 bytes
install ok: channel://pear.crtx.org/Crtx_PEAR_Channel_Frontend-0.1.2
```

Dieses Paket liefert eine Klasse `Crtx_PEAR_Channel_Frontend`, die Informationen über den Channel, die Entwickler und die einzelnen Pakete liefern kann und diese auch in HTML konvertiert. Um damit ein HTML-Web-Frontend für Ihren Server zu erzeugen, genügt es, eine Datei *index.php* mit dem folgenden Inhalt anzulegen:

```
<?php
require_once 'Crtx/PEAR/Channel/Frontend.php';

$frontend = new Crtx_PEAR_Channel_Frontend(
                            'ihre-domain.de',
                            array(
                                  'database' => 'mysql://root:******@localhost/
                                                pearchannel',
                                  'index' => 'index.php',
                                  'admin' => 'admin.php'));
?>
<html>
    <head>
        <title>PHP Kochbuch Testchannel</title>
        <link rel="stylesheet" type="text/css" href="pear_server.css" />
        <?php
            $frontend->showLinks();
        ?>
    </head>
    <body>
        <div id="top">
            <h1><a href="index">PHP Kochbuch Testchannel</a></h1>
        </div>
        <div id="menu">
            <?php
            $frontend->showMenu();
            ?>
            <div id="releases">
                <?php
                    $frontend->showLatestReleases();
                ?>
            </div>
        </div>
        <div id="content">
            <?php
            if (!$frontend->run()) {
                $frontend->welcome();
           }
            ?>
        </div>
    </body>
</html>
```

Damit Ihre Benutzer sich nun über den Channel informieren können, muss lediglich die URL *http://www.ihre-domain.de/* in einen Webbrowser eingegeben werden, Abbildung 24-7 zeigt Ihnen, wie das Frontend aussieht.

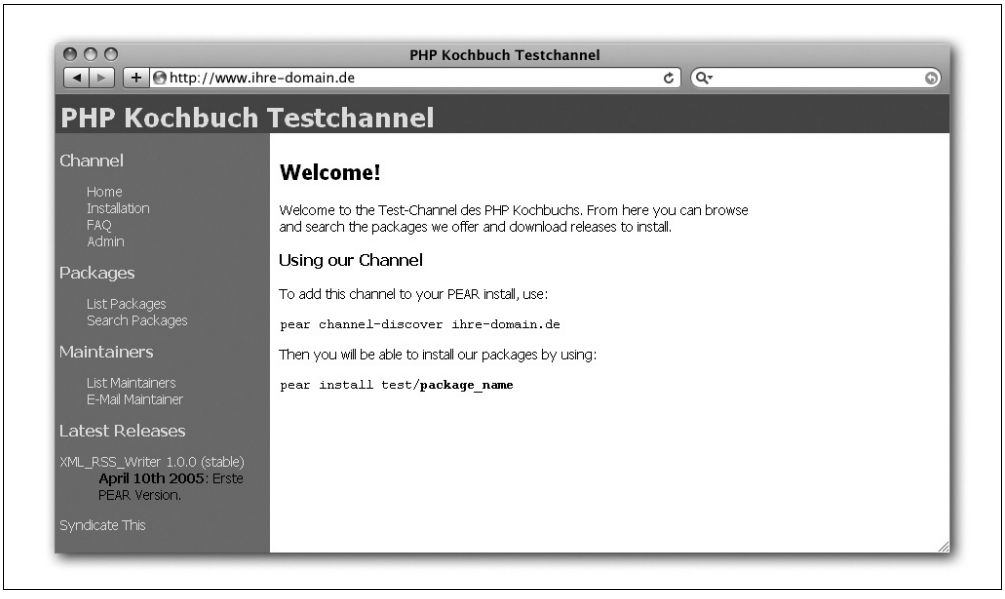


Abbildung 24-7: Das Web-Frontend Ihres Channel-Servers

Über dieses Frontend können sich andere Benutzer nun über aktuell veröffentlichte Versionen und deren Abhängigkeiten informieren, das Changelog der einzelnen Pakete lesen oder sogar ein E-Mail-Formular nutzen, um mit einem Entwickler in Kontakt zu treten.

Mit den Paketen Chiara_PEAR_Server und Crtx_PEAR_Server_Frontend ist es also möglich, innerhalb kürzester Zeit Pakete über einen eigenen PEAR-Channel-Server zu vertreiben. Die Ausgabe von Frontend und Administration kann sehr einfach über CSS-Angaben an Ihre Bedürfnisse angepasst werden. Selbst größere Umstrukturierungen sind kein Problem, da das Frontend-Objekt Methoden wie `showMenu()` oder `showLatestReleases()` bietet, die Sie an beliebiger Stelle im HTML-Code platzieren können.

Siehe auch

Rezept 24.6 zur Installation von Paketen aus anderen Channels; Rezept 24.8 zum Erstellen eigener PEAR-Pakete; die Dokumentation des *package2.xml*-Formats unter *http://pear.php.net/manual/de/guide.developers.package2.php*; den Chiara-Channel-Server unter *http://pear.chiaraquartet.net/*; den Crtx-Channel-Server *unter http://pear.crtx.org*.

24.10 Ein PHP-Archiv (PHAR) erstellen

Problem

Sie wollen ein PHP-Archiv erstellen, das Ihre Anwendung, Ihr Framework oder Ihre Bibliothek enthält. Sie möchten das Archiv so weitergeben, dass der Anwender es direkt einbinden oder ausführen kann, ohne die Inhalte vorher zu entpacken.

Lösung

Schreiben Sie das Programm, das Sie weitergeben wollen, beispielsweise eine Datei mit dem Namen *index.php* und dem folgenden Inhalt:

```
<?php
    echo "Hallo, ich bin in einem PHP-Archiv!";
?>
```

Dann schreiben Sie ein kleines Installationsprogramm, das für Sie das obige Skript in ein PHP-Archiv packt:

```
<?php
    $phar = new Phar('beispiel1.phar');
    $phar->setStub($phar->createDefaultStub('index.php'));
    $phar->addFile('index.php');
?>
```

Diskussion

PHAR ist das Kürzel für »PHP Archive«. Ein PHP-Archiv bündelt eine beliebige Anzahl Skripten und Ressourcen in einer Datei und kann direkt in PHP eingesetzt werden. Es unterstützt die Kompression der Inhalte sowie die Signierung des gesamten Archivs, um dessen Integrität sicherzustellen. PHAR eignet sich somit hervorragend zur Distribution größerer Softwarepakete.

Für den ausführenden oder lesenden Einsatz eines vorhandenen PHP-Archivs benötigen Sie nicht zwingend die PHAR-Erweiterung. Wollen Sie allerdings ein Archiv erstellen, kommen Sie nicht um diese Erweiterung herum. Seit PHP 5.3 wird PHAR nun mitgeliefert. Für ältere PHP-Versionen ab 5.2 müssen Sie das PECL-Paket für PHAR installieren. Um ein PHP-Archiv erstellen oder modifizieren zu können, müssen Sie entsprechende Schreibrechte durch den Eintrag `phar.readonly = Off` in der *php.ini* aktivieren. Aus Sicherheitsgründen können PHAR-Dateien standardmäßig nicht modifiziert werden. Wollen Sie Ihr Installationsskript verbessern, können Sie zu Beginn mittels der Methode `Phar::canWrite()` überprüfen, ob Sie diese Schreibrechte besitzen. Existieren sie nicht, wirft die PHAR-Erweiterung beim Instantiieren des PHAR-Objekts die `UnexpectedValueException`.

Ein PHP-Archiv setzt sich aus drei bis vier Datenbereichen zusammen. Am Anfang befindet sich der sogenannte Stub. Dies ist ein PHP-Skript, das die Inhalte des Archivs wieder zur Verfügung stellt. Im Regelfall kommt ein Stub ohne die PHAR-Erweiterung aus. Er öffnet sich selbst, und somit die eigene Archivdatei, zum Auslesen der Archivinhalte. Deshalb müssen Sie nach dem Erstellen manuell entsprechende Leserechte auf die PHAR-Datei setzen. Der Stub endet mit dem Befehl `__HALT_COMPILER()`, der dem PHP-Compiler das vorläufige Ende des PHP-Codes anzeigt. Dadurch wird es möglich, beliebige Byte-Daten in derselben Datei direkt an ein PHP-Skript anzuhängen. Im Fall von PHAR sind dies die Archivinhalte, die durch den Stub eingelesen und zur Verfügung gestellt werden. Am besten definieren Sie ein Default-Stub mittels der Methode `Phar::createDefaultStub()` und verzichten allgemein auf eine Archivkompression, damit Sie nicht in unnötige Kompatibilitätsprobleme zwischen unterschiedlichen PHP-Versionen und -Installationen geraten. Der Methode `Phar::createDefaultStub()` übergeben Sie den Namen des Startskripts, das ausgeführt wird, sobald Sie das Archiv einbinden bzw. direkt starten. Der zweite Datenbereich enthält Manifest-Informationen und gehört somit schon zum Archivinhalt. Danach folgen im dritten Datenbereich die eigentlichen Dateien des Archivs. Optional kann eine Signatur im letzten Datenbereich der PHAR-Datei liegen. Über diese kann die Integrität eines Archivs sichergestellt werden.

Die Funktionsweise eines PHP-Archivs kann mit der eines externen Laufwerks verglichen werden. Innerhalb eines Archivs finden sich die PHP-Skripten gegenseitig in gewohnter Weise. Sie können eingebunden werden und Ressourcen referenzieren. Das folgende Beispiel zeigt ein Archiv mit einer PHP-Datei (*index.php*), die ein Bild mittels HTML-Tags einbindet:

```
<?php
    echo "Hallo, ich bin in einem PHP-Archiv mit Bild!";
?>
<br /> <img src="daten/bild.jpg" alt="Bild" />
```

Sie benötigen zusätzlich eine Grafik mit dem Namen *bild.jpg*, die im Unterverzeichnis *daten* liegen muss. Das Installationsprogramm kann dann wie folgt aussehen:

```
<?php
    if (Phar::canWrite()) {
        $phar = new Phar('beispiel2.phar');
        $phar->setStub($phar->createDefaultStub('index.php'));
        $phar->addFile('index.php');
        $phar->addFile('daten/bild.jpg');
    } else {
        echo "Schreibrechte in der php.ini aktivieren.";
    }
?>
```

Sie können dieses Archiv direkt im Browser öffnen und sehen dann dort den Text und Ihr Bild. Welche Server-Einstellungen hierfür notwendig sind, können Sie in Rezept 24.12 nachlesen.

Siehe auch

Die PHP-Seite zu PHAR unter *http://www.php.net/phar*; Einstellung für PHAR-Schreibrechte in der *php.ini* unter *http://www.php.net/manual/phar.configuration.php#ini.phar.readonly*; Rezept 24.11, um PHP-Archive einzubinden; Rezept 24.12 für die Anzeige eines PHP-Archivs mit dem Apache-Webserver.

24.11 Auf Inhalte in einem PHP-Archiv (PHAR) zugreifen

Problem

Sie wollen archivierte PHP-Skripten in Ihre Anwendung einsetzen oder direkt auf Ressourcen innerhalb eines Archivs zugreifen.

Lösung

Sie können das PHP-Archiv in Ihr Programm einbinden und dadurch automatisch dessen Startskript ausführen:

```
include 'bibliothek.phar';
```

Der Inhalt einer bestimmten Datei kann mittels des PHAR-Stream-Wrappers innerhalb des Archivs direkt referenziert werden:

```
echo file_get_contents ("phar://bibliothek.phar/text.txt");
```

Falls Sie die PHAR-Erweiterung nicht installiert haben, können Sie die Dateien nach einem `include` direkt referenzieren:

```
include 'bibliothek.phar';
echo file_get_contents ("text.txt");
```

Diskussion

Die Funktionsweise eines PHP-Archivs ähnelt der eines externen Laufwerks. Dateien können hieraus direkt referenziert werden. Dabei gibt es zwei unterschiedliche Herangehensweisen, abhängig davon, ob Sie mit oder ohne die PHAR-Erweiterung arbeiten wollen. Seit PHP 5.3. wird diese Erweiterung nun mitgeliefert. Für ältere PHP-Versionen bis 5.2 müssen Sie hierfür das entsprechende PECL-Paket installieren.

Durch die PHAR-Erweiterung kennt PHP den PHAR-Stream-Wrapper und somit URI-Angaben, die mit `phar://` beginnen. Über diesen Wrapper können Sie auf Dateien innerhalb des Archivs komfortabel und direkt zugreifen. Wurde das Schreibrecht auf PHAR-Dateien in der *php.ini* gesetzt (`phar.readonly = Off`) und besitzt PHP ebenfalls das Schreibrecht auf die Archivdatei, haben Sie sogar die Möglichkeit, Dateien innerhalb des Archivs

zu modifizieren. Somit funktionieren beispielsweise Befehle wie `fopen()`, `opendir()` und auch `mkdir()` mit URI-Angaben auf Inhalte direkt im Archiv.

Soll Ihr Archiv auch auf Systemen ohne PHAR-Erweiterung laufen, müssen Sie auf den Einsatz des PHAR-Stream-Wrappers verzichten. Der Stub, den Sie – wie in Rezept 24.10 gezeigt – mit der Methode `Phar::createDefaultStub()` für Ihr Archiv erzeugen, stellt Ihnen in diesem Fall alle Archivdateien in einem temporären Verzeichnis zur Verfügung und bindet dieses in Ihr Include-Verzeichnis ein. Somit können Sie direkt auf diese Dateien zugreifen, als lägen sie im selben Verzeichnis wie Ihr PHP-Skript.

Das folgende Beispiel zeigt, wie Sie ein Archiv mit den Dateien *index.php* und *text.txt* erstellen und dann in Ihr Skript integrieren, indem Sie eine Funktion des Archivs ausführen und den Inhalt einer Archivdatei auslesen. Ihre Archivfunktion könnte in der Skriptdatei mit dem Namen *index.php* liegen und wie folgt aussehen:

```
<?php
    function archive_function () {
        echo "Ich bin eine Funktion im Archiv!";
    }
?>
```

Zusätzlich legen Sie eine Datei mit dem Namen *text.txt* und folgendem Inhalt in dasselbe Verzeichnis:

```
Dies ist der Inhalt einer Textdatei im Archiv.
```

Ihr Archiv erstellen Sie mit folgendem kleinen Programm:

```
<?php
    if (Phar::canWrite()) {
        $phar = new Phar('bibliothek.phar');
        $phar->setStub($phar->createDefaultStub('index.php'));
        $phar->addFile('index.php');
        $phar->addFile('text.txt');
    } else {
        echo "Schreibrechte in der php.ini aktivieren.";
    }
?>
```

Überprüfen Sie die so erstellte PHAR-Datei auf Leserechte durch PHP, ansonsten erhalten Sie mit dem folgenden Skript eine Warnmeldung, die darauf hinweist, dass das Archiv nicht durch den Stub geöffnet werden kann. In Ihrem Skript können Sie überprüfen, ob der PHAR-Stream-Wrapper installiert ist, um gegebenenfalls darauf zu reagieren und so mehr Kompatibilität zu älteren Systemen sicherzustellen:

```
<?php
    include 'bibliothek.phar';
    archive_function();
    echo "<br/>";
    if (in_array('phar', stream_get_wrappers())) {
        echo file_get_contents ("phar://bibliothek.phar/text.txt");
    } else {
```

```
        echo file_get_contents ("text.txt");
    }
?>
```

Siehe auch

PHP-Seite zu PHAR unter *http://www.php.net/phar*; Einstellung für PHAR-Schreibrechte in der *php.ini* unter *http://www.php.net/manual/phar.configuration.php#ini.phar.readonly*; Rezept 24.10, um PHP-Archive zu erstellen; Rezept 24.12 für das direkte Ausführen eines Programms im Archiv.

24.12 Ein PHP-Archiv (PHAR) direkt ausführen

Problem

Sie möchten eine komplette Webanwendung in ein PHP-Archiv integrieren und sie mit dem Browser direkt aufrufen.

Lösung

Registrieren Sie den Dateityp bzw. die Dateiendung *.phar* mit dem folgenden Eintrag in der Konfigurationsdatei *httpd.conf* des Apache-Webservers oder in der optionalen Datei *.htaccess* in Ihrem Webverzeichnis:

```
AddType application/x-httpd-php .phar
```

Danach können Sie das PHP-Archiv direkt im Webbrowser aufrufen.

Diskussion

Wenn Sie ein PHP-Archiv mittels `include` in Ihre Anwendung einbinden oder wenn Sie ein Archiv direkt aufrufen, wird das Skript ausgeführt, das Sie der Methode `Phar::createDefaultStub()` angegeben haben. Dieses Skript könnte beispielsweise weitere PHP-Dateien des Archivs einbinden, Initialisierungen durchführen oder eben der Startpunkt Ihrer Applikation sein. Sie können über die Methode `Phar::createDefaultStub()` optional noch ein weiteres Startskript angeben. Dadurch ist es Ihnen möglich, eine Unterscheidung zwischen einem Aufruf von der Kommandozeile und einem Aufruf vom Webserver zu erreichen. In dem folgenden Beispiel behandelt die Datei *cli.php* den Aufruf von der Kommandozeile, während die Datei *web/index.php* dann aufgerufen wird, wenn Sie die PHAR-Datei mit dem Browser ansteuern:

```
<?php
    try {
        $phar = new Phar('meinMultiProgramm.phar');
        $phar->setStub($phar->createDefaultStub('cli.php', 'web/index.php'));
```

```
            $phar->addFile('cli.php');
            $phar->addFile('web/index.php');
        } catch (Exception $e) {
            // Fehler behandeln.
        }
    ?>
```

Damit der Aufruf mit dem Webbrowser funktioniert, muss der Apache-Webserver zuerst den Dateityp *.phar* kennenlernen. Dies erreichen Sie – wie in der Lösung gezeigt – mit der Direktive `AddType` in der Apache-Konfiguration *httpd.conf*. Dadurch wird dem Webserver mitgeteilt, eine Datei mit der Endung *.phar* wie ein PHP-Skript zu behandeln. Somit werden der Stub und über ihn das entsprechende Startskript aufgerufen. Falls Sie keine Möglichkeit haben, Ihren Webserver über diesen Weg zu konfigurieren, können Sie diesen Eintrag auch in der Datei *.htaccess* vornehmen. Diese erstellen Sie in Ihrem Webverzeichnis, in dem sich Ihr direkt aufgerufenes Archiv befindet.

Siehe auch

PHP-Seite zu PHAR unter *http://www.php.net/phar*; Rezept 24.10 zum Erstellen eines PHP-Archivs; zur Apache-Direktive `AddType` unter *http://httpd.apache.org/docs/2.2/mod/mod_mime.html#addtype*.

Index

Symbole

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Über die amerikanischen Autoren

Adam Trachtenberg ist Senior Manager of Platform Evangelism bei eBay und predigt Entwicklern und Geschäftsleuten das Evangelium der eBay-Plattform rund um den Globus. Bevor er zu eBay ging, war er Mitbegründer und Vice President for Development der beiden Firmen Student.Com und TVGrid.Com. In beiden Unternehmen leitete er den Bereich Website-Design und -Entwicklung für Frontend und mittlere Ebene und war an der Firmenplanung und -strategie beteiligt. PHP verwendet Adam Trachtenberg seit 1997. Er ist Autor des O'Reilly-Titels *Umsteigen auf PHP 5*.

David Sklar ist Software-Architekt bei Ning, einer Plattform für die Gründung von Social Networks. Er war Mitbegründer und Chief Technology Officer von Student.Com und TVGrid.Com. Bei beiden Firmen war er für die Architektur von Systemen verantwortlich, die Benutzern auf der ganzen Welt personalisierte dynamische Inhalte liefern. Nachdem er PHP im Jahr 1996 für sich entdeckt hatte, gründete er PX (*http://px.sklar.com*) und ermöglichte damit PHP-Benutzern den Austausch von Programmen. Er ist Autor des Buchs *Essential PHP Tools* (Apress) und des O'Reilly-Titels *Einführung in PHP 5*.

Über die deutschen Autoren

Carsten Lucke ist Diplom-Informatiker und hat sein Studium der Fächer Software-Engineering und Network Computing an der FH Brandenburg mit Auszeichnung absolviert. Nach mehrjähriger Projekterfahrung in Großprojekten der Capgemini sd&m AG, unter anderem als technischer Chefdesigner, ist er aktuell wissenschaftlicher Mitarbeiter an der Professur für Wirtschaftsinformatik der Universität der Bundeswehr München.

Matthias Brusdeylins ist Diplom-Informatiker, lebt in München und arbeitet aktuell als Senior-Software-Ingenieur bei der Capgemini sd&m AG. Nach mehrjähriger Tätigkeit als Berater und Architekt in Software-Großprojekten übernimmt er aktuell Führungsaufgaben in der firmeninternen Forschungsabteilung und fungiert hier auch als Knowledge- & Quality-Consultant.

Stephan Schmidt ist Head of Webdevelopment bei der 1&1 Internet AG in Karlsruhe und verantwortet dort die Implementierung von Webanwendungen und Bestellsystemen in PHP und Java. Seit 2001 ist er fest in der PHP Open Source-Szene verwurzelt und Gründungsmitglied der Frameworks PHP Application Tools (*www.php-tools.de*) und Stubbles (*www.stubbles.net*). 2003 trat er dem PEAR-Projekt bei und betreut dort über 15 Pakete sowie eine PECL-Extension.

Ulrich Speidel lebt in Neuseeland, wo er an der University of Auckland in den Bereichen Datenkommunikation, Anwendungs- und Internetprogrammierung lehrt und auf dem Gebiet der Informationstheorie forscht. Nach einem Studium in Erlangen und Auckland erhielt er in Neuseeland seinen MSc in Physik und einen PhD in Informatik. Zusammen mit seiner Frau betreibt er eine kleine Firma, Technology Transfer Consulting Ltd., die technische Übersetzungen und Webprogrammierung anbietet.

Kolophon

Das Tier auf dem Cover des *PHP 5 Kochbuchs* ist ein Galapagos-Landleguan (*Conolophus cristatus*). Früher waren diese Reptilien auf den Galapagos-Inseln sehr zahlreich, doch leider erwiesen sich die Tiere als ausgesprochen schmackhaft für die Siedler des frühen 18. Jahrhunderts. Die Konkurrenz eingeführter Haustiere um Lebensraum und Nahrung tat dann ein Übriges, so dass der Landleguan heute auf der Insel Santiago ausgestorben ist und auf den anderen Inseln nur noch in wenigen Exemplaren vorkommt.

Galapagos-Landleguane können etwa einen Meter lang werden, und die Männchen wiegen bis zu 15 Kilogramm. Ihre harte, schuppige Haut ist gelb mit einem Muster aus weißen, schwarzen und braunen Flecken. Sie gleichen geheimnisvollen Wesen der Vergangenheit, Drachen mit langen Schwänzen, klauenbewehrten Füßen und Rückenkamm. In Wirklichkeit sind sie aber vollkommen harmlos.

Je nach Größe werden Landleguane im Alter von acht bis 15 Jahren geschlechtsreif. Sie kommen zusammen und paaren sich während bestimmter Zeiträume, die von Insel zu Insel verschieden sein können. Die Weibchen wandern dann in geeignete Regionen zur Eiablage. Nachdem das Weibchen ein Loch gegraben hat, legt es zwischen zwei und 20 Eier in dieses Nest. Sie verteidigt das abgedeckte Nest gegen andere Weibchen, damit diese ihre Eier nicht an derselben Stelle ablegen.

Die Leguan-Babys schlüpfen nach 85 bis 110 Tagen und brauchen etwa eine Woche, um sich aus dem Nest auszugraben. Die Tiere können bis zu 60 Jahre alt werden. Das erste Jahr jedoch ist kritisch. Neben Nahrungsknappheit bedeuten natürliche Feinde wie Habichte, Reiher oder Schlangen eine große Gefahr. Noch schlimmer sind verwilderte Katzen, denn die Kleinen müssen mindestens drei Jahre alt sein, bevor sie so groß sind, dass die Katzen ihnen nichts mehr anhaben können.

Der Umschlagsentwurf dieses Buches basiert auf dem Reihenlayout von Edie Freedman und stammt von Emma Colby, die hierfür einen Stich des *Dover Pictorial Archive* aus dem 19. Jahrhundert verwendet hat. Das Coverlayout der deutschen Ausgabe wurde von Michael Oreal unter Verwendung der Schriftart ITC Garamond von Adobe erstellt. Als Textschrift verwenden wir die Linotype Birka, die Überschriftenschrift ist die Adobe Myriad Condensed und die Nichtproportionalschrift für Codes ist LucasFont's TheSans Mono Condensed. Die in diesem Buch enthaltenen Abbildungen stammen von Robert Romano, Jessamyn Read und Michael Oreal. Mary Anne Weeks Mayo und Joachim Kurtz haben das Kolophon geschrieben.