Inhalt

1 Einführung in PHP

In den vergangenen fünf Jahren ist das Internet explosionsartig angewachsen und hat eine Fülle neuer Kommunikationswege hervorgebracht. Eine Vorreiterrolle bei dieser Entwicklung hat dabei das World Wide Web (WWW) eingenommen, in dem täglich Tausende von neuen Websites veröffentlicht und den Benutzern zahlreiche und außergewöhnliche Dienste angeboten werden. Dieser rasant wachsende Markt führt nicht nur zu einem ständigen Bedarf an neuen Technologien, sondern erfordert gleichzeitig von den Entwicklern, sich mit diesen Technologien zu beschäftigen. Da Sie diesen Abschnitt lesen, kann wohl davon ausgegangen werden, dass Sie einer dieser Entwickler sind oder bald einer sein werden. Doch ob Entwickler oder nicht: Sie haben zu diesem Buch gegriffen, weil Sie von der neuen, faszinierenden Technologie namens *PHP* gehört haben. In diesem Kapitel erhalten Sie eine Einführung in die Sprache PHP, erfahren mehr über deren Entstehung und Fähigkeiten und werden mit den grundlegenden Informationen versorgt, die Sie zur Entwicklung PHP-fähiger Sites benötigen. Die zahlreichen Beispiele in diesem Buch tragen hoffentlich dazu bei, Sie für die Möglichkeiten zu begeistern, die sich Ihnen und Ihrem Unternehmen mit PHP erschließen. Sie lernen, wie Sie die PHP-Software auf Rechnern mit den Betriebssystemen Linux/UNIX und Windows installieren und konfigurieren und wie Sie PHP in HTML einbetten. Nach der Bearbeitung dieses Kapitels werden Sie über genügend Grundwissen verfügen, um sich in die vielen verschiedenen Aspekte von PHP einzuarbeiten. Legen Sie also Ihre Lieblings-CD ein und machen Sie es sich bequem: Sie sind im Begriff, in eine spannende Welt einzutauchen – die PHP-Programmierung.

1.1 Ein kurzer Abriss der Geschichte

Der Grundstock für PHP wurde 1995 von einem selbstständigen Softwareentwickler namens Rasmus Lerdorf gelegt, als dieser ein Perl/CGI-Skript entwickelte, mit dessen Hilfe er feststellen konnte, wie viele Besucher seinen ins Netz gestellten Lebenslauf lasen. Sein Skript führte zwei Aufgaben aus: Zum einen protokollierte es Besucherinformationen, zum anderen zeigte es die Anzahl der Besucher auf der Webseite an. Da das heutige WWW zum damaligen Zeitpunkt noch in den Kinderschuhen steckte, gab es Tools dieser Art noch nicht, und bei Lerdorf gingen per E-Mail zahlreiche Anfragen zu seinen Skripts ein. Daraufhin begann Lerdorf mit dem Vertrieb seines Toolsets, das die Bezeichnung *Personal Home Page* (PHP) bzw. *Hypertext Preprocessor* erhielt.

Der Rummel um das PHP-Toolset veranlasste Lerdorf zur Entwicklung von PHP-Zusätzen. Einer dieser Zusätze konvertierte im HTML-Format eingegebene Daten in symbolische Variablen, die wiederum in andere Systeme exportiert werden konnten. Um dies zu bewerkstelligen, entschloss er sich, nicht in Perl, sondern in

C weiterzuentwickeln. Der Zusatz zum bestehenden PHP-Toolset führte zu Version PHP 2.0 oder PHP-FI (Personal Home Page – Form Interpreter). Version 2.0 wartete zudem mit einer Reihe von Erweiterungen und Verbesserungen auf, die von Programmierern aus aller Welt eingebracht worden waren.

Die neue PHP-Version erfreute sich großer Beliebtheit und es bildete sich bald ein Kernteam von Entwicklern. Diese behielten das ursprüngliche Konzept der direkten Codeeinbettung in HTML bei und schrieben das gesamte Parsermodul um. Fertig war PHP 3.0. Als die Version 1997 auf den Markt kam, setzten bereits mehr als 50.000 Benutzer PHP zur Erweiterung ihrer Webseiten ein.

Anmerkung 1997 wurde auch die Bedeutung des Akronyms PHP von »Personal Home Page« in »Hypertext Preprocessor« geändert.

Im Laufe der folgenden zwei Jahre setzte sich die Entwicklung mit rasanter Geschwindigkeit fort und führte zu einer Erweiterung um Hunderte von Funktionen. Gleichzeitig stieg auch die Anzahl der Benutzer sprunghaft an. Zu Beginn des Jahres 1999 gab Netcraft *(http://www.netcraft.com)* in einer vorsichtigen Schätzung die Zahl der PHP-Benutzer mit 1.000.000 an, was PHP als eine der beliebtesten Skriptsprachen weltweit auswies.

Anfang 1999 wurde die noch in der Entwicklung befindliche Version PHP 4.0 angekündigt. Obwohl eines der stärksten Merkmale von PHP in der außerordentlichen Leistungsfähigkeit bestand, Skripts auszuführen, hatten die Entwickler die Sprache nicht für die Entwicklung groß angelegter Anwendungen konzipiert. Also machten sie sich an die Entwicklung eines noch stabileren Parsermoduls, besser bekannt als Zend *(http://www.zend.com*). Die Entwicklung wurde rasch vorangetrieben, sodass PHP 4.0 bereits am 22. Mai 2000 auf den Markt kam.

Zusätzlich zum Zend-Prozessor bietet das in Israel ansässige Unternehmen Zend Technologies Ltd. den Zend-Optimizer an, mit dem die Leistung des Zend-Parsermoduls noch weiter verbessert werden kann. Wie Benchmarktests zeigen, kann der kostenlos per Download verfügbare Optimizer die Gesamtleistung um 40 bis 100 Prozent steigern. Weitere Informationen finden Sie auf der Zend-Website.

Zum Zeitpunkt der Entstehung dieses Buchs war PHP laut Netcraft *(http://www.netcraft.com)* in mehr als 3,6 Millionen Domänen implementiert und damit eine der populärsten Skriptsprachen der Welt. Angesichts der Tatsache, dass die Anzahl der großen Websites und Privatanwender, die das Produkt einsetzen, ständig wächst, kann PHP eine glänzende Zukunft vorausgesagt werden.

PHP kann kurz gesagt als eingebettete, serverseitige Webskriptsprache beschrieben werden, die dem Entwickler das schnelle und effiziente Entwickeln dynamischer Webanwendungen ermöglicht. PHP weist sowohl hinsichtlich der Syntax als auch im

Hinblick auf die Grammatik eine große Ähnlichkeit mit der Programmiersprache C auf, obwohl die Entwickler die besten Features aus einer Vielzahl von Sprachen, darunter Perl, Java und C++ integriert haben. Viele dieser nützlichen geliehenen Features umfassen das Parsen von regulären Ausdrücken, leistungsstarke Arrayfunktionen, eine objektorientierte Methodik sowie eine umfassende Datenbankunterstützung.

PHP eignet sich auch für die Entwicklung von Anwendungen, die über die herkömmliche, statische Methode zur Webseitenentwicklung (also HTML) hinausgehen. Hier kann PHP als nützliches Tool zur Erstellung und Verwaltung von dynamischen Inhalten eingesetzt und direkt in Sprachen wie JavaScript, Stylesheets, WML (Wireless Markup Language) und viele andere eingebettet werden.

PHP stellt Hunderte vordefinierter Funktionen zur Verfügung und ist damit in der Lage, so ziemlich alles zu verarbeiten, was ein Entwickler sich ausdenkt.

PHP bietet umfangreiche Unterstützung für das Erstellen und Bearbeiten grafischer Elemente, für mathematische Berechnungen, E-Commerce und neue Technologien wie XML (Extensible Markup Language), ODBC (Open Database Connectivity) und Macromedia Shockwave. Diese riesige Featurepalette macht die mühsame und kostspielige Integration von Drittanbietermodulen überflüssig. Auch aus diesem Grund gilt PHP weltweit als das Entwicklertool schlechthin.

Eine der großen Stärken von PHP ist die, dass aufgrund der direkten Einbettung in HTML-Code die Notwendigkeit entfällt, Programme mit vielen Befehlen zu erstellen, die nur zur Ausgabe des HTML-Codes dienen. HTML und PHP können je nach Bedarf abwechselnd eingesetzt werden und arbeiten reibungslos zusammen. Mit PHP können Sie beispielsweise Folgendes tun:

```
<html>
<title><? print "Hallo Welt!"; ?></title>
</html>
```

Als Ergebnis wird Hallo Welt! in der Titelleiste der Webseite angezeigt. Interessanterweise handelt es sich bei der von den PHP-Escapezeichen (<?…?>) umschlossenen einzeiligen **print**-Anweisung um ein komplettes Programm. Sie benötigen also weder langatmigen Einleitungscode noch Bibliotheken; Sie schreiben nur den Code, den Sie tatsächlich brauchen.

Um ein PHP-Skript ausführen zu können, müssen Sie natürlich zunächst die PHP-Software auf Ihren Server installieren und anschließend konfigurieren. Dieser Prozess wird an späterer Stelle in diesem Kapitel in den Abschnitten »Download von PHP/Apache« und »Installation und Konfiguration« erläutert. Unmittelbar vor diesen Abschnitten finden Sie einige Aussagen von namhaften Benutzern, die die Leistungsfähigkeit von PHP bestätigen.

1.2 Merkmale von PHP

Wie Ihnen vielleicht schon aufgefallen ist, dreht sich bei der Sprache PHP alles um den Aspekt der Anwendbarkeit. PHP liefert dem Entwickler die Tools, die für eine schnelle und effiziente Programmierung benötigt werden. Der praxisbezogene Ansatz von PHP ergibt sich aus fünf wichtigen Merkmalen:

- Vertrautheit
- Einfachheit
- Effizienz
- Sicherheit
- Flexibilität

Und dann gibt es da noch etwas, das PHP ungemein reizvoll macht: Es kostet nichts!

1.2.1 Vertrautheit

Programmierer aus den unterschiedlichsten Bereichen werden feststellen, dass sie von Beginn an mühelos mit PHP umgehen können. Viele Konstrukte der Sprache sind C und Perl entliehen und oft ist es kaum möglich, den PHP-Code vom typischen C- oder Pascal-Code zu unterscheiden. Dies reduziert den Lernaufwand beträchtlich.

1.2.2 Einfachheit

Ein PHP-Skript besteht nur aus genau so vielen Zeilen, wie Sie für die jeweilige Aufgabe benötigen. Es müssen weder Bibliotheken noch spezielle Compilerdirektiven oder andere vergleichbare Elemente einbezogen werden. Das PHP-Modul beginnt mit der Ausführung des Codes nach der ersten Escapesequenz (<?) und setzt diese solange fort, bis die schließende Escapesequenz (?>) erreicht ist. Sofern die Codesyntax keine Fehler aufweist, wird der Code genau wie angezeigt ausgeführt.

1.2.3 Effizienz

Die Effizienz stellt bei der Arbeit in Mehrbenutzerumgebungen (z. B. dem WWW) einen ungemein wichtigen Faktor dar. Mit PHP 4.0 wurden Mechanismen zur Ressourcenzuweisung, eine ausgeprägtere Unterstützung der objektorientierten Programmierung sowie Features für die Sitzungsverwaltung eingeführt. Version 4.0 umfasst außerdem eine Verweiszählung, um eine überflüssige Speicherzuordnung auszuschließen.

1.2.4 Sicherheit

PHP bietet Entwicklern und Administratoren eine Reihe flexibler und effizienter Sicherheitsfeatures. Diese Features können grundsätzlich in zwei Kategorien unterteilt werden, die Sicherheit auf Systemebene und Sicherheit auf Anwendungsebene bereitstellen.

Sicherheit auf Systemebene

PHP ist mit einer Reihe von Sicherheitsmechanismen ausgestattet, die von Administratoren bearbeitet werden können. So gewährleistet PHP bei ordnungsgemäßer Konfiguration ein größtmögliches Maß an Freiheit bei maximaler Sicherheit. PHP kann im so genannten *Sicherheitsmodus* ausgeführt werden, über den Sie Benutzer daran hindern können, die PHP-Implementierung missbräuchlich einzusetzen. Im sicheren Modus werden u.a. die maximale Ausführungszeit und die Speichernutzung begrenzt, um einen Abfall der Serverleistung zu verhindern. Ähnlich wie beim **cgi-bin**-Ordner können Administratoren auch den Zugriff auf Bereiche beschränken, in denen Benutzer PHP-Skripts einsehen und ausführen bzw. PHP-Skripts zur Anzeige geschützter Serverinformationen wie der **passwd**-Datei verwenden können.

Sicherheit auf Anwendungsebene

Der vordefinierte Funktionssatz von PHP unterstützt verschiedene zuverlässige Datenverschlüsselungsoptionen. PHP ist außerdem mit vielen Drittanbieteranwendungen kompatibel und kann leicht in sichere E-Commerce-Technologien integriert werden. Ein weiterer Vorteil von PHP besteht darin, dass der Quellcode über den Browser nicht sichtbar ist, da das Skript vollständig analysiert wird, bevor es an den Benutzer zurückgesendet wird, der die Anforderung gestellt hat. Dieser Vorteil der serverseitigen Architektur von PHP verhindert den Verlust kreativer Skripts an Benutzer, die über ausreichend Wissen verfügen, um einen Befehl zur Quellcodeanzeige auszuführen.

Die Sicherheit ist ein bedeutender Aspekt, daher wurde diesem Thema in diesem Buch ein komplettes Kapitel gewidmet. Eine ausführliche Erläuterung der PHP-Sicherheitsfeatures finden Sie in Kapitel 16, »Sicherheit«.

1.2.5 Flexibilität

PHP ist eine eingebettete Sprache und daher im Hinblick auf die Entwickleranforderungen außerordentlich flexibel. PHP wird in der Regel zwar vor allem wegen der Einsatzmöglichkeiten mit HTML gepriesen, kann jedoch ebenso in JavaScript, WML, ASPs (Active Server Pages) und eine breite Palette anderer Implementie-

rungen integriert werden. Zusätzlich können, wie bei den meisten etablierten Sprachen, sinnvoll geplante PHP-Anwendungen nach Bedarf problemlos erweitert werden.

Das Problem der Browserabhängigkeit stellt sich nicht, da PHP-Skripts vollständig auf Serverseite analysiert werden. PHP-Skripts können an so ziemlich jedes Gerät gesendet werden, das über einen Browser verfügt, u.a. an Mobiltelefone, PDA-Geräte (Personal Digital Assistant), Pager, Laptops und natürlich an herkömmliche PCs. Zur Entwicklung shellbasierter Anwendungen kann PHP auch von der Befehlszeile ausgeführt werden.

Da PHP keinen serverspezifischen Code enthält, sind Benutzer nicht an einen bestimmten, ihnen möglicherweise ungewohnten Webserver gebunden. Apache, Microsoft IIS, Netscape Enterprise Server, Stronghold und Zeus eignen sich alle für eine PHP-Serverintegration. Aufgrund der Vielzahl an Plattformen, auf denen diese Server arbeiten, ist PHP nahezu plattformunabhängig und kann beispielsweise unter UNIX, Solaris, FreeBSD und Windows 95/98/NT eingesetzt werden.

Schließlich bietet PHP Zugang zu externen Komponenten wie Enterprise Java Beans und Win32 Com-Objekten. Dank dieser neuen Features wird PHP zu einem Produkt der Oberklasse, mit dem der Entwickler alle Möglichkeiten zur Skalierung von PHP-Projekten erhält.

Kostenlose Bereitstellung

Die Strategie der offenen Entwicklung wurde von der Softwareindustrie nicht immer positiv aufgenommen. Das Konzept, einer breiten Masse Quellcode zugänglich zu machen, hat jedoch bei einer Vielzahl von Projekten zu unleugbar positiven Ergebnissen geführt. Am meisten Furore machte dabei sicherlich Linux, doch auch der Erfolg des Apache-Projekts hat wesentlich dazu beigetragen, das Konzept der offenen Entwicklung zu bestätigen. Dasselbe gilt für die Entwicklungsgeschichte von PHP, denn auch hier haben Anwender aus allen Teilen der Welt dazu beigetragen, das Projekt voranzutreiben.

Die offene Entwicklungsstrategie führt dazu, dass PHP-Benutzer von verbesserten Leistungsergebnissen profitieren können. Gleichzeitig wird der Code kostenlos zur Verfügung gestellt. Darüber hinaus stellt eine außerordentlich aufnahmebereite Gemeinde aus Tausenden von Benutzern eine Art »Kundensupport« bereit, indem im Rahmen von gut besuchten Onlinediskussionsforen selbst zu den rätselhaftesten Fragen Antworten geliefert werden.

1.3 Benutzerstimmen

»Wir stehen schon seit langem persönlich mit einigen PHP-Entwicklern in Kontakt und hatten in der Vergangenheit einen regen E-Mail-Austausch. Wenn die PHP-Entwickler irgendwelche Probleme hinsichtlich MySQL hatten, waren wir immer bemüht, ihnen bei der Lösung dieser Probleme zu helfen. Bei manchen Gelegenheiten haben wir MySQL auch neue Features hinzugefügt, um die PHP-Integration zu verbessern. Dieser Einsatz hat dazu geführt, dass MySQL außerordentlich gut mit PHP arbeitet, und wir werden dafür sorgen, dass dies so bleibt.«

Michael »Monty« Widenius, MySQL-Entwickler
http://www.mysql.com

»FAST hat PHP aus verschiedenen Gründen zur Implementierung von mp3.lycos.com eingesetzt. Der wichtigste Grund bestand dabei in der Produkteinführungszeit, denn mit PHP kann die Entwicklung stark beschleunigt werden. Ein weiterer Grund war die Geschwindigkeit: Innerhalb eines Tages kamen wir von 0 auf 1,4 Millionen Seitenanzeigeoperationen, PHP hatte damit keinerlei Probleme. Das dritte Argument für PHP bestand natürlich in der Gewissheit, im Rahmen von Belastungstests aufgespürte Bugs selbst beheben zu können, da PHP offen entwickelt wurde.«

Stig Bakken, FAST Search & Transfer ASA
http://www.fast.no

»Ich benutze PHP schon seit langem, genau gesagt seit Version PHP/FI 1.x. Ich war sehr angetan von der Möglichkeit, mit einer so leicht einsetzbaren Sprache aus dem Stegreif Formulare zu verarbeiten und Seiten nach meinen Wünschen gestalten zu können. Ebenso wie die Bedürfnisse meines Unternehmens hat sich auch PHP weiterentwickelt. Heute verfügt PHP über eine außerordentlich große Auswahl an Features. Davon machen wir auch bei so ziemlich jeder von uns entwickelten Website Gebrauch, darunter 32bit.com und DevShed.com. Wir setzen PHP bei InfoWest sogar zur Kundendienstverwaltung, zur Kontoführung und zur Portüberwachung ein.

Entwicklung und Aufnahme von PHP sind ein Paradebeispiel für ein erfolgreiches Open Source-Projekt. Aufgeschlossenheit, die Mitwirkung der Benutzergemeinde und eine sinnvoll verwaltete Codebasis haben dazu beigetragen, PHP einen Erfolg zu bescheren, an den bislang die wenigsten Handelsunternehmen in dieser Form anknüpfen konnten. Ich freue mich auf das, was uns PHP in der Zukunft noch bringen wird. Ich möchte jeden Webentwickler, der noch am Anfang steht, dazu ermutigen, PHP auszuprobieren. Vielleicht geht es Ihnen so wie mir, und Sie wollen danach nie wieder ohne PHP arbeiten.«

Randy Cosby, Geschäftsführer der nGenuity, Inc., DevShed
http://www.devshed. com

1.4 Ein Beispiel zur Einführung

Aus Codebeispiel 1.1 können Sie ersehen, wie einfach die Einbettung von PHP in HTML ist.

Listing 1.1: Dynamische PHP-Seitenerstellung

```
<?
// Festlegen einiger Variablen
$site_name = "PHP-Rezepte";
$hintergrund_farbe = "white";
$benutzer_name = "Chef Luigi";
?>

<html>
<head>
<title><? print $site_name; ?></title>
</head>
<body bgcolor="<? print $hintergrund_farbe; ?>" >
<?
// Intromeldung mit Datum und Benutzername anzeigen
print "
PHP-Rezepte | ".date("d.m.Y")." <br>
Herzlich willkommen, $benutzer_name! <br>
";
?>
</body>
</html>
```

Abbildung 1.1 zeigt, wie das Skript nach Ausführung in einem Browser angezeigt wird.

Gar nicht schlecht, oder? Ich kann mir vorstellen, dass in den Köpfen vieler Leser schon unzählige Ideen entstehen. Und dabei war dies nur eine Kostprobe dessen, was PHP kann.

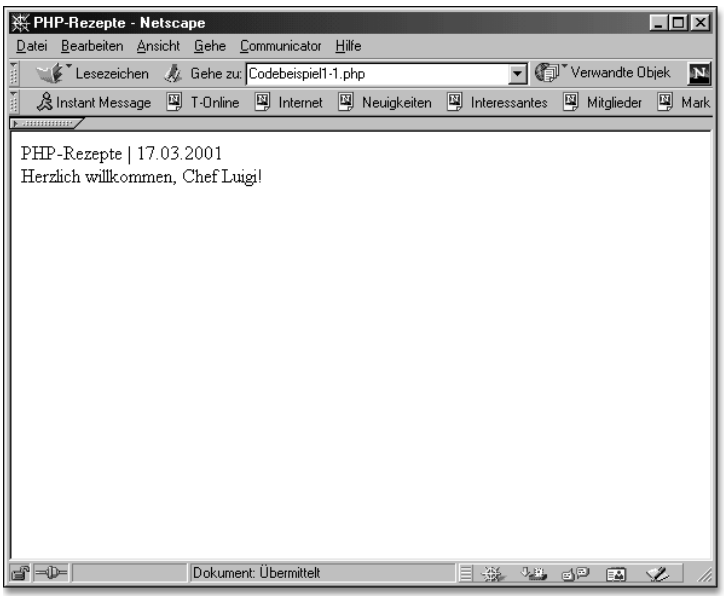

Abbildung 1.1 Skriptausführung im Browser

1.5 Download von PHP/Apache

Bevor Sie weiterlesen, sollten Sie zunächst PHP und einen Webserver auf Ihren Rechner herunterladen und anschließend installieren und konfigurieren. Obwohl PHP mit einer großen Auswahl an Webservern kompatibel ist, gehe ich davon aus, dass Sie mit Apache arbeiten werden, da es sich hierbei momentan um den beliebtesten Webserver und gleichzeitig um den Webserver handelt, der am häufigsten mit PHP eingesetzt wird. Und selbst wenn dies nicht der Fall ist: Der übliche Installationsvorgang weist bei den verschiedenen Webservern keine starken Abweichungen auf.

Sie können die PHP-Bereitstellung von der offiziellen PHP-Site oder von einer der vielen, weltweit vorhandenen Spiegelsites herunterladen. Unter *http://www.php.net* finden Sie ein aktuelles Verzeichnis der gespiegelten Sites. Von dieser Site aus können Sie PHP in einem der folgenden drei Formate herunterladen:

- Binäre TAR-
- Datei
- RPM (RedHat Package Manager)
- Quellcode

Für Plattformen mit entsprechender Unterstützung stehen neben dem Win32-Binärformat für Windows auch RPM-Dateien (RedHat Package Manager) zur Verfügung. Viele Benutzer arbeiten gerne mit dem RPM-Bereitstellungsformat, da

dieses eine problemlose Installation der Anwendungspakete ermöglicht. RPM unterstützt eine Konfiguration der Anwendung im Laufe der Installation allerdings nur begrenzt. Benutzer, die auf der Windows-Plattform arbeiten, werden dagegen feststellen, dass sich die Win32-Binärdateien optimal für Installationen eignen, solange der Quellcode nicht verändert werden muss (was die meisten Benutzer auch nicht betreffen sollte).

Sollten Sie UNIX verwenden, möchten Sie den Quellcode wahrscheinlich selbst erstellen. Auch wenn viele PHP-Einsteiger bei dem bloßen Gedanken daran vielleicht schaudern, handelt es sich, wie Sie bald feststellen werden, eigentlich um einen relativ einfachen Vorgang. Wenn Sie unter Windows arbeiten und den Code unbedingt kompilieren möchten, müssen Sie einen aktuellen VC++-Compiler installieren.

Laden Sie von der Site *http://www.php.net* die Bereitstellung herunter, die am besten auf Ihre Bedürfnisse abgestimmt ist. Die Downloadzeit hängt von der jeweiligen Verbindungsart und -geschwindigkeit ab. Zusätzlich kann die Dokumentation heruntergeladen werden. Ich empfehle Ihnen dringend, sich die aktuellste Version herunterzuladen.

Tipp Als dieses Buch in Druck ging, war PHP 4.0.3 die aktuelle stabile Version. Selbstverständlich wird sich diese Versionsnummer im Rahmen der Weiterentwicklung des PHP-Pakets ändern. Ich empfehle Ihnen, sich immer die neueste stabile Produktversion herunterzuladen.

Sofern Sie den Apache-Server – den von PHP-Benutzern am häufigsten eingesetzten Webserver – noch nicht installiert haben, sollten Sie auch von diesem die aktuellste stabile Version herunterladen. Die Apache-Pakete sind unter *http://www.apache.org/dist/binaries/* erhältlich, es stehen Verzeichnisse für eine Fülle von Betriebssystemen bereit. Laden Sie sich dasjenige herunter, das Ihren Anforderungen entspricht. Anleitungen für den genauen PHP-Konfigurationsablauf sämtlicher verfügbarer Plattformen und Webserver zu liefern, würde weit über den Rahmen dieses Buches hinausgehen. Aus diesem Grund werde ich besonders auf die Konfiguration des Apache-Servers eingehen. Unabhängig vom verwendeten Webserver empfehle ich Ihnen dringend, die Konfigurationsabschnitte in diesem Kapitel zu lesen, um einen Einblick in die allgemeinen Konfigurationsaspekte zu erhalten, mit denen Sie sich möglicherweise auseinandersetzen müssen.

Das Installieren einer neuen Software stellt einen Arbeitsschritt dar, der auf Neueinsteiger eine oft abschreckende Wirkung hat. Die Entwickler von PHP haben jedoch einige zusätzliche Anstrengungen unternommen, um die Installation von

PHP möglichst einfach zu gestalten. In den folgenden Abschnitten wird auf die Schritte eingegangen, die Sie zur Installation und Konfiguration von PHP sowohl unter Linux/UNIX als auch auf den Win32-Plattformen durchführen müssen.

Anmerkung In späteren Kapiteln wird der MySQL-Datenbankserver (*http://www.mysql.com*) zur Veranschaulichung einiger der komplexeren Konzepte aus dem Bereich der Webanwendungsentwicklung eingesetzt. Um diese Beispiele nachvollziehen zu können, müssen Sie das MySQL-Paket installieren.

Um diejenigen nicht zu benachteiligen, die sich für ein anderes Datenbankpaket zur Verwendung mit PHP entscheiden, wird der Installationsvorgang von MySQL hier ebenfalls nicht beschrieben.

Ungeachtet des verwendeten Datenbankservers kann es sein, dass der Installationsvorgang von dem hier beschriebenen leicht abweicht. Weiterführende Informationen zu MySQL finden Sie in Kapitel 11, »Datenbanken«.

1.6 Installation und Konfiguration

Ich gehe davon aus, dass Sie den Download von PHP und Apache inzwischen erfolgreich abgeschlossen haben. Als Nächstes müssen Sie sich überlegen, welche Methode Sie zum Installieren der Bereitstellung verwenden möchten. Für Rechner, die unter UNIX arbeiten, bieten sich drei verschiedene Methoden an: CGI-Version, Apache-Modul und dynamisches Apache-Modul. Aufgrund einiger Leistungsschwächen von CGI werde ich mich auf die Installationsbeschreibung der beiden letztgenannten Methoden beschränken. Für Windows-Rechner führe ich die Installation mithilfe der vorkompilierten Binärdateien aus.

Vielleicht möchten einige von Ihnen MySQL zusammen mit ihrer PHP-Bereitstellung installieren. Um die Installationsanweisungen jedoch für alle Leser einfach nachvollziehbar zu machen, habe ich mich dagegen entschieden, eine Anleitung zur Installation von MySQL in dieses Kapitel aufzunehmen. Sollten Sie sich für eine gemeinsame Nutzung von MySQL und PHP interessieren, lesen Sie hierzu die weiterführenden Informationen in Kapitel 11 und setzen Sie den Installationsvorgang erst *anschließend* fort.

Anmerkung Wie Sie bereits wissen, bietet PHP4 Unterstützung für viele Webserver, einschließlich AOL Server, Netscape Enterprise Server, Microsoft IIS, Zeus usw. Ich werde den Installationsvorgang jedoch auf die Erläuterung der Apache-Unterstützung beschränken. Ausführliche Anweisungen zur Installation von PHP mit anderen Servern finden Sie in der PHP-Dokumentation unter *http://www.php.net.*

1.6.1 UNIX

Unabhängig davon, für welche Installationsvariante Sie sich entschieden haben, besteht der erste Schritt immer in der Dekomprimierung der Bereitstellungen. Diese führen Sie in zwei einfachen Schritten aus:

1. Entzippen Sie die Pakete. Sobald Sie diesen Schritt ausgeführt haben, werden Sie feststellen, dass die Dateien nun *.**tar**-Erweiterungen aufweisen:

   ```
   gunzip apache_1.3.9.tar.gz
   gunzip php-4.0.0.tar.gz
   ```

2. Führen Sie diese Archivdateien in ihr ursprüngliches Format zurück.

   ```
   tar -zxvf apache_1.3.x.tar
   tar -zxvf php-4.0.x.tar
   ```

Von diesem Punkt an übernimmt die jeweilige Installationsprozedur die Ausführung der verbleibenden Arbeitsschritte.

Apache-Modul

Die Installation von PHP als Apache-Modul ist relativ einfach. Ich erläutere Ihnen im Folgenden die einzelnen Schritte:

1. Wechseln Sie in das Apache-Verzeichnis:

   ```
   cd apache_1.3.x
   ```

2. Konfigurieren Sie Apache. Sie können dafür jeden beliebigen Pfad verwenden. Beachten Sie, dass dem Pfadnamen *kein* umgekehrter Schrägstrich folgt:

   ```
   ./configure --prefix=[path]
   ```

3. Wechseln Sie ins PHP-Verzeichnis und konfigurieren, erstellen und installieren Sie die Bereitstellung. Die Option **with-config-file-path** legt das Verzeichnis fest, das die PHP-Konfigurationsdatei enthalten soll. Im Allgemeinen ist dieser Pfad als **/usr/local/lib** festgelegt, aber Sie können ihn beliebig verändern:

   ```
   ./configure -with-apache=../apache_1.3.x --with-config-file-path=
   [config-path]
   make
   make install
   ```

4. Wechseln Sie wieder in das Apache-Verzeichnis. Nun können Sie Apache neu konfigurieren, erstellen und installieren. Die Option **other-configuration-options** bezieht sich auf andere besondere Konfigurationsoptionen für den Apache-Webserver, die Sie festlegen können. Eine Beschreibung dieses Vorgangs

würde jedoch den Rahmen dieses Buches sprengen. Ich empfehle Ihnen die Lektüre der Apache-Dokumentation zwecks einer umfassenden Erläuterung folgender Optionen:

```
./configure -activate-module=src/modules/php4/libphp4.a --other-
configuration-options
make
make install
```

5. Im Laufe des letzten Arbeitsschrittes muss die Apache-Datei **httpd.conf** bearbeitet werden. Einige dieser Änderungen beziehen sich speziell auf Apache, während andere erforderlich sind, um sicherzustellen, dass PHP-Skripts erkannt und an den Webserver gesendet werden können. Suchen Sie zunächst nach folgender Zeile:

   ```
   ServerName new.host.name
   ```

   Ändern Sie die Zeile in:

   ```
   ServerName localhost
   ```

   Suchen Sie anschließend nach den zwei folgenden Zeilen:

   ```
   #AddType application/x-httpd-php .php .php4
   #AddType application/x-httpd-php-source .phps
   ```

Um PHP-fähige Dateien ordnungsgemäß auf dem Server ausführen zu können, müssen bei diesen Zeilen die Kommentarzeichen entfernt werden. Löschen Sie hierzu einfach das Rautenzeichen (#) aus dem Anfang jeder Zeile. Speichern Sie die Datei und wechseln Sie zum übergeordneten Verzeichnis. Starten Sie den Apache-Server mithilfe des folgenden Befehls:

```
./bin/apachectl start
```

Voilà! PHP und Apache sind nun betriebsbereit. Fügen Sie zu Testzwecken folgenden Code in eine Datei ein und speichern Sie die Datei als **phpinfo.php** im Dokumentenstammverzeichnis von Apache. Das Verzeichnis trägt die Bezeichnung **htdocs** und befindet sich im Apache-Installationsverzeichnis.

```
<?
   phpinfo();
?>
```

Öffnen Sie diese Datei in einem Browser auf dem Server. Es sollte eine längere Liste mit Informationen zur PHP-Konfiguration angezeigt werden. Herzlichen Glückwunsch, Sie haben PHP erfolgreich als Apache-Modul installiert.

Dynamisches Apache-Modul

Das dynamische Modul ist deshalb so praktisch, weil es Ihnen eine Aktualisierung Ihrer PHP-Bereitstellung ermöglicht, ohne dass hierzu eine erneute Kompilierung des Webservers erforderlich ist. Apache behandelt die Bereitstellung dabei einfach wie eines seiner vielen weiteren Module, z. B. ModuleRewriter oder ModuleSpelling. Dieses Konzept erweist sich vor allem dann als besonders nützlich, wenn Sie PHP zu einem späteren Zeitpunkt ein Supportfeature hinzufügen möchten, etwa eine Verschlüsselungsfunktion. Dazu müssen Sie PHP lediglich der Verschlüsselungsfunktion entsprechend neu konfigurieren/kompilieren, und anschließend können Sie die Verschlüsselung sofort in Ihren Webanwendungen verwenden. Der Installationsvorgang verläuft folgendermaßen:

1. Wechseln Sie in das Apache-Verzeichnis:

   ```
   cd apache_1.3.x
   ```

2. Konfigurieren Sie Apache. Sie können dafür jeden beliebigen Pfad verwenden. Beachten Sie dabei, dass dem Pfadnamen *kein* umgekehrter Schrägstrich folgt. Die Option **other-configuration-options** bezieht sich auf andere besondere Konfigurationsoptionen für den Apache-Webserver, die Sie festlegen können. Eine Beschreibung dieses Vorgangs würde jedoch den Rahmen dieses Buches sprengen. Ich empfehle Ihnen die Lektüre der Apache-Dokumentation zwecks einer umfassenden Erläuterung folgender Optionen:

   ```
   ./configure --prefix=[path] --enable-module=so --other-configuration-
   options
   ```

3. Erstellen Sie den Apache-Server. Nachdem Sie den Befehl **make** eingegeben haben, werden Sie einen Haufen Meldungen über den Bildschirm laufen sehen. Das ist vollkommen normal.

   ```
   make
   ```

4. Installieren Sie den Apache-Server. Nachdem Sie den Befehl **make install** eingegeben haben, werden auch hier verschiedene Meldungen auf dem Bildschirm angezeigt. Auch hier besteht kein Anlass zur Beunruhigung. Sobald dies aufhört, werden Sie in einer Meldung über die erfolgreiche Installation des Servers informiert.

   ```
   make install
   ```

5. Sofern keine Fehler aufgetreten sind, können Sie nun die Apache-Datei **httpd.conf** bearbeiten. Die Datei befindet sich im **conf**-Verzeichnis in dem von Ihnen in Schritt 4 angegebenen Pfad. Öffnen Sie die Datei in Ihrem bevorzugten Texteditor. Machen Sie folgende Zeile ausfindig:

```
ServerName new.host.name
```

Ändern Sie die Zeile in:

```
ServerName localhost
```

6. Wechseln Sie in das Verzeichnis, in das Sie PHP heruntergeladen haben. Anschließend konfigurieren, erstellen und installieren Sie PHP. Geben Sie das Pfadverzeichnis an, das auf die **apxs**-Datei verweist. Diese Datei befindet sich im **bin**-Verzeichnis des von Ihnen in Schritt 4 festgelegten Pfades.

```
./configure --with-apxs=[path/to/apxs]
make
make install
```

7. Öffnen Sie die Apache-Datei **httpd.conf** erneut, um eine weitere Änderung vorzunehmen. Damit eingehende Anforderungen PHP-fähiger Dateien ordnungsgemäß analysiert werden können, muss die Dateierweiterung mit der in **httpd.conf** (der Apache-Serverkonfigurationsdatei) festgelegten Dateierweiterung übereinstimmen. Diese Datei enthält eine Reihe von Optionen, die der Administrator nach freiem Ermessen ändern kann. Einige dieser Optionen beziehen sich unmittelbar auf PHP. Öffnen Sie die **httpd.conf**-Datei in Ihrem bevorzugten Texteditor. Gegen Ende der Datei finden Sie zwei Zeilen, die etwa folgendermaßen aussehen:

```
#AddType application/x-httpd-php .php .php4
#AddType application/x-httpd-php-source .phps
```

8. Um PHP-fähige Dateien ordnungsgemäß auf dem Server ausführen zu können, müssen Sie die Kommentierung dieser Zeilen aufheben. Dazu entfernen Sie ganz einfach das Rautenzeichen (#) aus dem Anfang jeder Zeile.
9. Speichern Sie die Datei und wechseln Sie zum übergeordneten Verzeichnis. Starten Sie den Apache-Server mit folgendem Befehl:

```
./bin/apachectl start
```

Voilà! PHP und Apache sind nun betriebsbereit.

Fügen Sie zu Testzwecken folgenden Code in eine Datei ein und speichern Sie die Datei als **phpinfo.php** im Dokumentenstammverzeichnis von Apache. Das Verzeichnis trägt die Bezeichnung **htdocs** und befindet sich im Apache-Installationsverzeichnis.

```
<?
   phpinfo();
?>
```

Öffnen Sie die Datei in einem Browser auf dem Server. Es sollte eine längere Liste mit Informationen zur PHP-Konfiguration angezeigt werden. Herzlichen Glückwunsch, Sie haben das dynamische Apache-Modul erfolgreich installiert.

1.6.2 Installation unter Windows 95/98/NT

Wenn Sie schon einmal eine Anwendung unter dem Betriebssystem Windows installiert haben, wissen Sie, wie einfach dies ist. Sie klicken auf ein paar Schaltflächen, stimmen einigen Aussagen zu, und schon ist die Anwendung installiert. Nach genau dem gleichen Schema erfolgt auch die Installation von Apache und PHP auf einem Windows-Rechner.

1. Doppelklicken Sie auf die ausführbare Apache-Datei, um die Installation zu starten. Ein Installations-Assistent unterstützt Sie bei der Ausführung der weiteren Schritte. Lesen Sie die Lizenzbestimmungen sorgfältig durch und stimmen Sie diesen zu.
2. Der Assistent schlägt ein standardmäßiges Installationsverzeichnis vor (C:\Programme\Apache Group\Apache). Falls Ihnen dieser Pfad zu lang sein sollte, können Sie Ihn auf C:\Apache\ verkürzen.
3. Anschließend werden Sie zur Eingabe eines Namens aufgefordert, unter dem Apache im Startmenü angezeigt werden soll. Sie können hier einen beliebigen Namen eingeben oder den Standardnamen akzeptieren.
4. Als Nächstes werden Sie aufgefordert, den Installationstyp auszuwählen. Wählen Sie einfach die Standardinstallation. Nach Auswahl des Installationstyps wird die Installation vorgenommen.
5. An dieser Stelle müssen Sie die im **conf**-Verzeichnis befindliche **httpd.conf**-Datei bearbeiten. Das **conf**-Verzeichnis befindet sich in dem Verzeichnis, das Sie in Schritt 2 für die Installation des Apache-Servers gewählt haben. Öffnen Sie die Datei mithilfe Ihres bevorzugten Texteditors. Sie sollten mindestens drei grundlegende Änderungen vornehmen:

   Ersetzen Sie **yourname@yoursite.com** durch die zutreffende Adresse: **ServerAdmin yourname@yoursite.com**

   Geben Sie den zutreffenden Servernamen ein. Falls Sie keinen richtigen Servernamen haben, verwenden Sie einfach **localhost**: **ServerName localhost**
6. Versuchen Sie, Apache zu starten, um sicherzustellen, dass alles einwandfrei läuft. An diesem Punkt müssen Sie je nach verwendetem Windows-Betriebssystem unterschiedlich vorgehen:

   Wenn Sie mit Windows NT arbeiten, wählen Sie aus dem Startmenü **Install Apache as Service (NT Only)**. Wechseln Sie anschließend zur **Systemsteuerung**, öffnen Sie das Fenster **Dienste**, wählen Sie **Apache** und klicken Sie auf die Schaltfläche **Start**. Apache wird jetzt und ab sofort bei jedem Hochfahren des Rechners automatisch gestartet.

Wenn Sie nicht Windows NT verwenden, wählen Sie **Start Apache** aus dem Startmenü. Ein kleines Fenster wird geöffnet. Dieses Fenster muss zur Serverausführung geöffnet bleiben.

7. Geben Sie schließlich **http://localhost/** in einen auf dem Server installierten Browser ein. Es sollte eine Standardseite angezeigt werden, in der Sie über die erfolgreiche Installation benachrichtigt werden.
8. Jetzt muss PHP installiert werden. Wechseln Sie in das Verzeichnis, in das Sie das PHP-Paket heruntergeladen haben. Extrahieren Sie es mithilfe einer Anwendung zum Entzippen in ein Verzeichnis Ihrer Wahl.
9. Wechseln Sie in das PHP-Verzeichnis und suchen Sie nach einer Datei mit der Bezeichnung **php.ini-dist**. Benennen Sie diese Datei in **php.ini** um und speichern Sie sie im Verzeichnis **C:\Windows**.
10. Wechseln Sie wieder in das PHP-Verzeichnis. Suchen Sie die Dateien **php4ts.dll** und **Mscvrt.dll**. Speichern Sie diese Dateien im Verzeichnis **C:\Windows\System**. Sie verfügen vermutlich bereits über die Datei **Mscvrt.dll** und werden gefragt, ob diese überschrieben werden soll. Sie sollten die Datei jedoch weder überschreiben noch kopieren.
11. Wechseln Sie erneut in die Apache-Datei **http.conf**, und öffnen Sie sie abermals in einem Texteditor. Sie müssen einige Änderungen vornehmen:

    Suchen Sie nach folgender Zeile:

    ```
    ScriptAlias /cgi-bin/ "C:/Apache/cgi-bin/"
    ```

    Fügen Sie unmittelbar unter dieser Zeile Folgendes ein:

    ```
    ScriptAlias /php4/ "C:/php4/"
    ```

    Suchen Sie anschließend nach **AddType**. Sie werden die zwei folgenden Kommentarzeilen sehen:

    ```
    #AddType application/x-httpd-php3 .phtml
    #AddType application/x-httpd-php3-source .phps
    ```

    Fügen Sie unmittelbar unter diesen Zeilen Folgendes ein:

    ```
    AddType application/x-httpd-php .phtml .php
    AddType application/x-httpd-php-source .phps
    ```

    Führen Sie einen Bildlauf nach unten durch. Sie werden die folgenden Kommentarzeilen sehen:

    ```
    #
    # Action lets you define media types that will execute a script whenever
    # a matching file is called. This eliminates the need for repeated URL
    ```

```
# pathnames for oft-used CGI file processors.
# Format: Action media/type /cgi-script/location
# Format: Action handler-name /cgi-script/location
#
```

Fügen Sie darunter folgende Zeile ein:

```
Action application/x-httpd-php /php4/php.exe
```

12. Voilà! PHP und Apache sind nun betriebsbereit.

Fügen Sie zu Testzwecken folgenden Code in eine Datei ein und speichern Sie die Datei als **phpinfo.php** im Dokumentenstammverzeichnis von Apache. Es handelt sich dabei um das **htdocs**-Verzeichnis, das sich im von Ihnen in Schritt 4 festgelegten Verzeichnis befindet.

```
<?
   phpinfo();
?>
```

Öffnen Sie diese Datei in einem Browser auf dem Server. Es sollte eine längere Liste mit Informationen zur PHP-Konfiguration angezeigt werden.

Achtung Nach erfolgreicher Ausführung der oben angegebenen Arbeitsschritte kann die Webserver/PHP-Konfiguration zwar für Testzwecke eingesetzt werden, das heißt aber *nicht*, dass aus dem World Wide Web auf Ihren Webserver zugegriffen werden kann. Hierzu sind weitere Schritte erforderlich, die im Rahmen dieses Buches nicht erläutert werden können. Informationen zu diesem Thema finden Sie in der offiziellen Apache-Site unter *http://www.apache.org*. Obwohl die hier beschriebenen Schritte ausreichen, um das PHP-Paket zum Laufen zu bringen, empfiehlt es sich, die PHP-Konfiguration so zu verändern, dass sie bestmöglich auf Ihre Anforderungen zugeschnitten ist. Nähere Informationen zu diesem Thema finden Sie im nächsten Abschnitt.

1.7 Konfiguration von PHP

Obwohl PHP mit der Standardkonfigurationseinstellung ordnungsgemäß ausgeführt wird, können Sie eine Vielzahl von Änderungen vornehmen, um die Installation genau auf Ihre Anforderungen abzustimmen. Die Datei **php.ini**, die im Laufe des Installationsvorgangs standardmäßig in das Verzeichnis **/usr/local/lib/** kopiert wird, enthält sämtliche Konfigurationseinstellungen.

Unabhängig davon, welche Plattform und welcher Webserver in Verbindung mit PHP eingesetzt werden, enthält die **php.ini**-Datei einen Satz an Standardparametern, mit denen verschiedene wichtige Merkmale der PHP-Installation verwaltet

werden können. Diese Datei beinhaltet sämtliche Einstellungen, die für die Installationseigenschaften bei der Ausführung von PHP-Skripts relevant sind. Das PHP-Modul liest die **php.ini**-Datei ein, sobald PHP gestartet wird.

Anmerkung In Version 3.0 heißt die Konfigurationsdatei **php3.ini**, in Version 4.0 trägt sie die Bezeichnung **php.ini**.

1.7.1 Allgemeine Konfigurationsdirektiven

Im Rahmen dieses Buches können nicht sämtliche Konfigurationsdirektiven besprochen werden. Einige der Direktiven sollen hier jedoch Erwähnung finden, da sie dem Entwickler von besonderem Nutzen sein können. In den nachfolgenden Kapiteln werde ich weitere, im jeweiligen Zusammenhang relevante Direktiven anführen.

short_open_tag [on | off]

Zusätzlich zu den Standardtags legt die Konfigurationsdirektive **short_open_tag [on | off]** die Verwendung der PHP-Escapetags **<?...?>** vom Typ **short** fest.

asp_tags [on | off]

Zusätzlich zu den Standardtags ist die Konfigurationsdirektive **asp_tags [on | off]** für die Festlegung der Tags im ASP-Stil zuständig. Diese Tags fügen PHP-Code folgendermaßen ein:

```
<%
print "Dies ist PHP-Code.";
%>
```

precision [integer]

Die Konfigurationsdirektive **precision [integer]** setzt die Anzahl der in Gleitkommazahlen dargestellten positiven Ziffern fest.

safe_mode [on | off]

Sie sollten den Sicherheitsmodus vor allem dann aktivieren, wenn mehrere Benutzer auf Ihrem System arbeiten. Im Wesentlichen können Sie durch Aktivieren des Sicherheitsmodus die Möglichkeit ausschließen, dass ein Benutzer ein PHP-Skript für den Zugriff auf eine andere Systemdatei einsetzt, beispielsweise die **passwd**-Datei auf einem Linux-Rechner. **Safe_mode** kann nur mit der CGI-Version von PHP verwendet werden. Kapitel 19 enthält weitere Informationen zu diesem Thema.

max_execution_time [integer]

Die Konfigurationsdirektive **max_execution_time [integer]** bestimmt die maximale Sekundenzahl, die für die Ausführung eines PHP-Skripts aufgewendet werden darf. Auf diese Weise wird verhindert, dass selbstausführende Skripts wertvolle Systemressourcen belegen.

error_reporting [1-8]

Die Konfigurationsdirektive **error_reporting [1.8]** gibt an, ab welchem Schweregrad Fehler aufgezeichnet werden. Je höher der Bitwert, desto »empfindlicher« reagiert PHP auf zu meldende Fehler:

Bitwert	Fehlerprotokollierung
1	Normale Fehler
2	Normale Warnmeldungen
4	Parserfehler
8	Hinweise

display_errors [on | off]

Die Konfigurationsdirektive **display_errors [on | off]** zeigt die Fehler im Browser an.

log_errors

Die Konfigurationsdirektive **log_errors** legt fest, ob Fehler in einer Datei protokolliert werden oder nicht. Wenn **log_errors** aktiviert ist, legt die Direktive **error_log** die Protokolldatei fest.

error_log [filename]

Ist **log_errors** aktiviert, legt **error_log** die Protokolldatei zur Aufzeichnung von Fehlern fest.

magic_quotes_gpc

Wenn **magic_quotes_gpc** aktiviert ist, werden sämtliche in Benutzer- oder Datenbankdaten enthaltenen Sonderzeichen automatisch mit dem erforderlichen umgekehrten Schrägstrich versehen. Die Abkürzung »gpc« steht übrigens für »get/post/cookie«.

Meiner Ansicht nach ist es effizienter, **magic_quotes_gpc** deaktiviert zu lassen und die Sonderzeichen explizit zu kennzeichnen. Doch egal, für welche Methode Sie sich dabei entscheiden, sie muss unbedingt durchgängig verwendet werden,

da Sie andernfalls eine Datenbeschädigung riskieren. Sollte **magic_quotes_gpc** aktiviert sein, dürfen Sie Sonderzeichen niemals durch Eingabe eines umgekehrten Schrägstrichs umgehen; falls **magic_quotes_gpc** nicht aktiviert ist, stellen Sie sicher, dass Sie Sonderzeichen stets mit Escapezeichen versehen.

track_vars

Die Konfigurationsdirektive **track_vars** ermöglicht die Aufzeichnung verschiedener wichtiger Sitzungsvariablenarrays, darunter **$HTTP_GET_VARS[], $HTTP_POST_VARS[], $HTTP_POST_FILES, $HTTP_COOKIE_VARS[], $HTTP_ENV_VARS[] und $HTTP_SERVER_VARS[]**. Diese Arrays werden ausführlich in Kapitel 13, »Cookies und Sitzungsverfolgung« besprochen.

Ich möchte an dieser Stelle noch einmal betonen, dass es noch sehr viel mehr Konfigurationsdirektiven gibt. Die hier genannten gehören meines Erachtens jedoch zu den nützlichsten. Auf viele der hier nicht erwähnten Direktiven wird in den späteren Kapiteln näher eingegangen.

1.8 Grundlegende PHP-Konstrukte

Bevor wir uns in den folgenden Kapiteln dieses Buches mit den Kernthemen von PHP beschäftigen, möchte ich Ihnen nachfolgend einige einleitende PHP-Konzepte vorstellen.

1.8.1 Kennzeichnen von PHP-Code über Escapezeichen

Das PHP-Parsermodul muss in der Lage sein, PHP-Code von anderen Seitenelementen zu unterscheiden. Der für diese Zwecke eingesetzte Mechanismus wird als das *Verwenden von Escapezeichen* bzw. als *Escaping* bezeichnet. Hierzu kann eine der folgenden vier Methoden eingesetzt werden:

- Standardtags
- **short**-Tags
- Skripttags
- Tags im ASP-Stil

Standardtags

Aufgrund ihrer klaren Definition und der einfachen Verwendung werden die Standardtags von PHP-Programmierern wohl am häufigsten verwendet.

```
<?php print "Willkommen zur Welt von PHP!"; ?>
```

Die Standardtags sind auch besonders praktisch, da den Anfangsescapezeichen die Buchstabenfolge **php** folgt, der verwendete Codetyp also eindeutig gekennzeichnet wird. Da Sie auf einer Seite möglicherweise mehrere Technologien (bei-

spielsweise JavaScript, serverseitige **include**-Anweisungen und PHP) simultan einsetzen, kann sich dies als sehr nützlich erweisen. Der Anfangsescapesequenz folgt der gesamte PHP-Code, wiederum gefolgt von der schließenden Escapesequenz »**?>**«.

short-Tags

Bei Tags vom Typ **short** handelt es sich um die kürzesten Escapetags für PHP-Code.

```
<? print "Willkommen zur Welt von PHP!"; ?>
```

Tags vom Typ **short** können erst nach Aktivierung ausgeführt werden. Diese Aktivierung kann auf drei Arten erfolgen:

- Setzen Sie die Funktion **short_tags()** an den Anfang der PHP-fähigen Seite.
- Schließen Sie die Funktion **short_tags()** in ein Skript ein, das nur dann ausgeführt wird, wenn das **short_open_tag** in der Datei **php.ini** den Wert **false** aufweist. Falls Sie beabsichtigen, Tags vom Typ **short** häufig zu verwenden, empfehle ich Ihnen dringend, diese Direktive auf **true** zu setzen.
- Aktivieren Sie die **short_open_tag**-Einstellung in der **php.ini**-Konfigurationsdatei. Dazu setzen Sie die Option **short_open_tag** auf **yes**.

Führen Sie die PHP-Kompilierung mit der Option **--enable-short-tags** aus. Diese Option erfüllt in etwa den gleichen Zweck wie **--enable-track-vars** im Installations-/Konfigurationsprozess.

Skripttags

Einige Texteditoren interpretieren PHP-Code fälschlicherweise als HTML-Code (also als anzeigbaren Code), was sich störend auf die Webseitenentwicklung auswirkt. Dieses Problem kann durch den Einsatz folgender Escapetags vermieden werden:

```
<script language="php">
print "Willkommen zur Welt von PHP!";
</script>
```

Tags im ASP-Stil

Eine vierte und letzte Methode zur Einbettung von PHP-Code besteht in der Verwendung von Tags im ASP-Stil (Active Server Pages). Hier wird ähnlich wie bei den oben beschriebenen **short**-Tags verfahren, allerdings wird anstelle des Fragezeichens ein Prozentzeichen (%) eingesetzt.

```
<% print "Willkommen zur Welt von PHP!"; %>
```

Eine Variante dieser ASP-Tags kann zu einer Codevereinfachung beitragen. Bei dieser Variante ist der Einschluss einer **print**-Anweisung in den PHP-Code nicht erforderlich. Das unmittelbar auf das öffnende ASP-Tag folgende Gleichheitszeichen **(=)** signalisiert dem PHP-Parser, dass der Variablenwert ausgegeben werden soll.

```
<%= $variable %>
```

Unter Verwendung dieses komfortablen Tagstils könnten wir folgenden Code ausführen:

```
<%
// Für $rezept-Variable irgendetwas eingeben…
$rezept = "Lasagne";
%>
Luigis Lieblingsrezept ist <%=$rezept;%>
```

Eigentlich enthält dieses Beispiel sogar zwei separate PHP-Skripts. Das erste Skript weist der Variablen **$rezept** den Wert **Lasagne** zu. Wenn an späterer Stelle der Wert der **$rezept**-Variablen angezeigt werden muss, können Sie die Tagvariante im ASP-Stil einzig zu diesem Zweck einsetzen.

1.8.2 Einbetten von HTML in PHP-Code

Das vielleicht leistungsstärkste Merkmal der Sprache PHP liegt in ihrer Fähigkeit, in anderen Sprachen wie HTML oder JavaScript sowohl ausgegeben als auch eingebettet werden zu können. In Codebeispiel 1.2 wird dieses Konzept verdeutlicht.

Listing 1.2: Anzeige von HTML unter Verwendung von PHP-Code

```
<html>
<head>
<title>Grundlegende PHP/HTML-Integration</title>
</head>
<body>
<?
// Wie Sie sehen, werden die HTML-Tags in die print-Anweisung eingeschlossen
print "<h3>PHP/HTML-Integration ist cool.</h3>";
?>
</body>
</html>
```

Codebeispiel 1.2 veranschaulicht, wie PHP den HTML-Code direkt in **print**-Anweisungen einschließen kann. Überschriftentags der Ebene 3 (<h3>…</h3>) können direkt im PHP-Code platziert werden. Im endgültigen Dokument erscheinen diese Tags als reguläre HTML-Ausgabe.

Aus Codebeispiel 1.3 können Sie ersehen, wie PHP Informationen dynamisch in eine Webseite einfügen kann. In Abbildung 1.2 wird das aktuelle Datum in die Titelzeile eingefügt.

Abbildung 1.2 Eine einfache PHP-Funktion zur dynamischen Datumseinfügung

Mit der einfachen PHP-Funktion **date()** kann das Datum in der Browsertitelleiste angezeigt werden

Listing 1.3: Dynamische Datumseinfügung

```
<html>
<head>
<title>PHP-Rezepte | <? print (date("d.m.Y")); ?></title>
</head>
<body>
Willkommen zu PHP-Rezepte, der Site für PHP-Enthusiasten mit einem Hang zum
Kochen!
</body>
</html>
```

Die einfache PHP-Funktion **date()** kann das aktuelle Datum auf mehrere Arten formatieren. Der formatierte Datumswert kann anschließend im Titel ausgegeben werden.

PHP kann sogar das Format von HTML selbst verändern, indem es Tagmerkmale zuweist und anschließend in die Datei einfügt. Codebeispiel 1.4 zeigt, wie dies ermöglicht wird. Zunächst wird einer Variablen ($grosse_schriftart) ein Schriftmerkmal (h3) zugewiesen und anschließend bei Bedarf in den anzuzeigenden Text eingefügt.

Listing 1.4: Dynamische HTML-Tags

```
<html>
<head>
<title>PHP-Rezepte | <? print (date("d.m.Y")); ?></title>
</head>
<?
$grosse_schriftart = "h3";
?>
<body>
<? print "<$grosse_schriftart>PHP-Rezepte</$grosse_schriftart>"; ?>
</body>
</html>
```

Codebeispiel 1.4 ist eine leicht veränderte Variante von Codebeispiel 1.3. Hier werden einer Variablen zunächst Überschriftentags der Ebene 3 (<h3>…</h3>) zugewiesen, anschließend wird diese Variable in einer **print**-Anweisung eingesetzt. Im endgültigen Dokument erscheinen diese Tags als reguläre HTML-Ausgabe.

1.8.3 Einbetten mehrerer PHP-Skripts

Um bei der Erstellung dynamischer Webanwendungen flexibel zu bleiben, können Sie mehrere separate PHP-Skripts in eine Seite einbetten. Codebeispiel 1.5 verdeutlicht diese Vorgehensweise.

Listing 1.5: Einbetten mehrerer PHP-Skripts in ein Dokument

```
<html>
<head>
<title>
<?
   print "Eine weitere PHP-fähige Seite";
   $variable = "Hallo Welt!";
?>
</title></head>
<body>
<? print $variable; ?>
</body>
</html>
```

Codebeispiel 1.5 weist zunächst den Aufbau einer typischen (wenn auch einfachen) HTML-Seite auf. Die Flexibilität, die Sie durch dieses Feature erhalten, liegt darin, dass Variablen in einem Codeabschnitt zugewiesen und in einem anderen Codeabschnitt auf derselben Seite wieder verwendet werden können.

1.8.4 Kommentieren von PHP-Code

Sie sollten Code selbst bei relativ kurzen und unkomplizierten Skripts ausreichend kommentieren. PHP verwendet zwei Kommentarformate:

- *Einzeilige Kommentare* werden im Allgemeinen für kurze Erklärungen oder Anmerkungen verwendet, die für den Code von Belang sind.
- *Mehrzeilige Kommentare* werden für gewöhnlich eingesetzt, um Pseudocodealgorithmen und ggf. erforderliche genauere Erklärungen bereitzustellen.

Beide Methoden führen letztlich zum gleichen Ergebnis und haben keinerlei Auswirkungen auf die Gesamtleistung des Skripts. Es bleibt Ihnen überlassen, für welche der beiden Methoden Sie sich entscheiden.

Einzeilige Kommentare

Für einzeilige Kommentare können zwei unterschiedliche Kommentarstile verwendet werden. Beide sind vollkommen identisch aufgebaut, verwenden jedoch unterschiedliche Escapezeichen. Der eine Stil setzt einen doppelten Schrägstrich (//) an den Anfang eines Kommentars, der andere verwendet dazu ein Rautenzeichen (#). Nachfolgend sehen Sie jeweils ein Beispiel:

```
<?
// Farbe der Rosen festlegen
$rosen_farbe = "rot";

# Farbe der Veilchen festlegen
$veilchen_farbe = "blau";
print "Rosen sind $rosen_farbe, Veilchen sind $veilchen_farbe";
?>
```

Natürlich können Sie unter Verwendung einer der vorgestellten Stile für einzeilige Kommentare auch mehrzeilige Kommentare erstellen:

```
<?
// Datei: Beispiel.php
// Verfasser: WJ Gilmore
// Datum: 31. Juli 1975

print "Ein Beispiel mit Kommentaren";
?>
```

Mehrzeilige Kommentare

PHP stellt einen Mechanismus zur Erstellung ausführlicher Kommentare bereit, die mehr als eine Zeile beanspruchen. Dieser Kommentartyp ist im Kommentarstil von C gehalten und wird durch ein einleitendes **/*** und ein schließendes ***/** gekennzeichnet.

```
<?
/*
   Skript: Beispiel_mehrzeiliger_Kommentar.php
   Zweck: Beispiel für einen mehrzeiligen Kommentar
   Verfasser: WJ Gilmore
   Datum: 14. Juni 2000
*/

print "Oben in diesem Skript finden Sie einen mehrzeiligen Kommentar!";
?>
```

Wie Sie sehen, sind mehrzeilige Kommentare sehr nützlich, wenn Sie eine relativ umfangreiche Inhaltsangabe eines Skripts oder eines Skriptabschnitts anbringen müssen.

1.9 Wie geht es weiter?

In diesem Kapitel wurden Sie im Schnellverfahren mit den Schlüsselaspekten von PHP vertraut gemacht:

- Geschichte und Features von PHP
- Installation und Konfiguration
- PHP-Escapezeichen
- Kommentieren von PHP-Code

Diese Themen dienen als Einführung für die folgenden Kapitel, in denen Sie mehr über die Anwendungsentwicklung mit PHP erfahren werden. Am Ende des nächsten Kapitels werden Sie genug über PHP wissen, um mit der Erstellung Ihrer eigenen Programme zu beginnen. Sie werden dieses Wissen anschließend zur Entwicklung eines Veranstaltungskalenders einsetzen, der problemlos in eine bestehende Webseite eingefügt werden kann. Dieses Projekt dient als Vorbereitung auf die weitere Entwicklung der PHP-Rezepte-Webanwendung.

13 Cookies und Sitzungsverfolgung

Die Fähigkeit zur Benutzerverfolgung und Anpassung der Benutzerinformationen basierend auf persönlichen Einstellungen ist zu einem der aktuellsten und meistdiskutierten Features geworden, die im Web angeboten werden. Die Vorteile, die sich aus der Bereitstellung maßgeschneiderter Dienste für den Benutzer ergeben, sind unbestritten. Die Fähigkeit jedoch, einem Benutzer zu »folgen«, der von Seite zu Seite oder gar von Site zu Site navigiert, lässt Fragen zum Datenschutz laut werden.

Abgesehen von diesen Datenschutzfragen jedoch kann die Verfolgung von Benutzerinformationen über Cookies oder andere Technologien sowohl für den Benutzer als auch für den Siteanbieter von großem Nutzen sein. Es ist für den Benutzer vorteilhaft, dass diese Anbieterdienste die Möglichkeit zur Inhaltsanpassung bereitstellen, da so Informationen aussortiert werden können, die uninteressant oder nutzlos sind. Diese Fähigkeit ist für den Siteadministrator ebenfalls von Vorteil, da die Benutzervorlieben und -gewohnheiten ungeahnte Möglichkeiten für die Benutzerinteraktion bieten, z. B. ein zielgerichtetes Marketing und eine umfassende Analyse zur Akzeptanz des Siteinhalts. Im kommerziell dominierten Web sind diese Funktionen mittlerweile zum Standard geworden.

Die Idee, den Benutzer zu verfolgen, während sich dieser durch Ihre Site bewegt, kann als *Sitzungsverfolgung* definiert werden. Angesichts der umfangreichen Informationen, die über die Sitzungsverfolgung in Ihre Sitearchitektur eingespeist werden können, kann gesagt werden, dass die Vorteile, die sich aus der Sitzungsverfolgung und der Bereitstellung angepasster Inhalte ergeben, die Nachteile mehr als aufwiegen. Daher ist kein PHP-Handbuch vollständig, das nicht auch ein Kapitel den PHP-Features zur Sitzungsverfolgung widmet. Im vorliegenden Kapitel werde ich verschiedene Konzepte vorstellen, die eng mit der Sitzungsverfolgung verknüpft sind. Hierzu zählen Sitzungscookies und deren Verwendung, eindeutige Nummern zur Sitzungsidentifikation sowie ein Überblick über die vordefinierte PHP-Konfiguration für die Sitzungsverfolgung und vordefinierte Funktionen für die Sitzungsverfolgung.

13.1 Was ist ein Cookie?

Ein *Cookie* ist nichts weiter als eine kleines Informationspaket, das durch einen Webserver gesendet und auf einem Clientbrowser gespeichert wird. Dies kann für den Entwickler vorteilhaft sein, da so nützliche Benutzerdaten gespeichert und abgerufen werden können, was zu einem dauerhaften Status zwischen Client und Server führt. Cookies werden in vielen Internetsites sowohl zur Verbesserung des Surferlebnisses der Benutzer als auch zur Steigerung der Siteeffizienz eingesetzt,

da mit Cookies die Benutzernavigation, die Benutzeroperationen und -vorlieben nachvollzogen werden können. Die Fähigkeit zur Speicherung dieser Informationen ist ein Schlüsselfeature für Sites, mit der Dienste wie Onlinestores, Sitepersonalisierung und zielgerichtete Werbekampagnen realisiert werden können.

Aufgrund des benutzerzentrierten Zwecks von Cookies werden die Schlüsselinformationen im Allgemeinen in einer eindeutigen Benutzer-ID gespeichert (User Identification Number, UIN). Diese ID wird anschließend in einer Datenbank gespeichert und als Schlüssel für den Abruf von Datenbankinformationen eingesetzt, die mit dieser Benutzer-ID verknüpft sind. Natürlich ist es nicht zwingend erforderlich, dass Cookies zum Speichern einer Benutzer-ID eingesetzt werden; Sie könnten im Cookie auch ganz andere Informationen speichern – vorausgesetzt, diese Informationen sind nicht größer als 4 KB (4.096 Bytes).

13.1.1 Cookiekomponenten

Interessanterweise werden noch andere Informationen im Cookie gespeichert, die dem Entwickler eine Nutzungsanpassung im Hinblick auf Domäne, Zeitrahmen, Pfad und Sicherheit ermöglichen. Nachfolgend eine Beschreibung der verschiedenen Cookiekomponenten:

- **name** – Der Cookiename ist ein obligatorischer Parameter, da Cookies über diesen Namen referenziert werden. Den Cookienamen können Sie sich prinzipiell wie eine Variable vorstellen.
- **value** – Der Cookiewert ist einfach eine Informationseinheit, die dem Cookienamen zugeordnet wird. Hierbei kann es sich um eine Benutzer-ID, die Hintergrundfarbe, das Datum oder Ähnliches handeln.
- **expiration date** – Dieses Datum gibt die Lebensdauer des Cookies an. Sobald dieses Datum mit dem aktuellen Datum übereinstimmt, läuft der Cookie ab und wird unbrauchbar. Obwohl es sich laut Cookiespezifikationen bei diesem Ablaufdatum um einen optionalen Parameter handelt, besteht bei PHP (weiterhin) das Problem, dass nur bei Festlegung eines Ablaufdatums tatsächlich ein Cookie gesetzt wird. Gemäß den Cookiespezifikationen läuft der Cookie am Ende der Sitzung ab, wenn kein Ablaufdatum festgelegt wurde (wenn also der Benutzer die Site verlässt).
- **domain** – Die Domäne, die den Cookie erstellt hat und diesen lesen kann. Wenn eine Domäne über mehrere Server verfügt und sämtliche dieser Server auf das Cookie zugreifen sollen, kann die Domäne **.phprezepte.com** gesetzt werden. Auf diese Weise können alle potenziellen Domänen der dritten Ebene der PHP-Rezepte-Site, z. B. **wap.phprezeptecom** oder **nachrichten.phprezepte.com** auf den Cookie zugreifen. Aus Sicherheitsgründen kann ein Cookie nicht für eine Domäne als die gesetzt werden, die versucht, den Cookie zu set-

zen. Dieser Parameter ist optional. Sofern nicht eingeschlossen, wird standardmäßig der Domänenname verwendet, von dem der Cookie stammt.

- **path** – Die Pfadeinstellung gibt den gültigen URL-Pfad für den Cookie an. Jeder Versuch, ein Cookie von außerhalb dieses Pfades abzurufen, schlägt fehl. Die Pfadeinstellung ist optional. Wird dieser Parameter nicht gesetzt, wird standardmäßig der Pfad des Dokuments festgelegt, von dem aus der Cookie erstellt wurde.
- **security** – Dieser Parameter gibt an, ob der Cookie in einer unsicheren Umgebung abgerufen werden kann. Da der Cookie primär in unsicheren Umgebungen eingesetzt wird, lautet die Standardeinstellung für diesen Parameter **FALSE**.

Obwohl alle Cookies bei Ihrem Setzen den gleichen Syntaxregeln entsprechen müssen, ist das Speicherformat für Cookies browserabhängig. Der Netscape Communicator speichert Cookies beispielsweise in etwa diesem Format:

```
.phprezepte.com    FALSE    /    FALSE    971728956    bgcolor    blue
```

Im Internet Explorer wird derselbe Cookie so gespeichert:

```
bgcolor
blue
localhost/php4/php.exe/book/13/
0
2154887040
29374385
522625408
29374377
*
```

Zur richtigen Anzeige eines vom Internet Explorer gespeicherten Cookies öffnen Sie diesen einfach mit einem Texteditor. Beachten Sie, dass einige Texteditoren das Zeichen für eine neue Zeile nicht ordnungsgemäß verarbeiten und im Cookiedokument als Quadrate anzeigen.

Anmerkung Der Internet Explorer speichert Cookieinformationen in einem Ordner namens **Cookies**, im Netscape Communicator dagegen werden Cookies in einer einzigen Datei mit dem Namen **cookies** gespeichert. Führen Sie einen entsprechenden Suchlauf auf Ihrem Laufwerk aus, um diese Dateien zu finden.

13.1.2 Cookies und PHP

O.k., genug Hintergrundinformationen. Sie sind bestimmt schon gespannt darauf, zu erfahren, wie Sie mithilfe von PHP Ihre eigenen Cookies speichern und abrufen können. Sie werden sich freuen zu hören, dass dies erstaunlich einfach über einen einzigen Aufruf der vordefinierten Funktion **setcookie()** bewerkstelligt wird.

Die **setcookie()**-Funktion speichert ein Cookie auf dem Benutzercomputer. Die Syntax lautet:

```
int setcookie (string name [, string wert [, int datum [, string pfad [, string
domaene [, int sicher]]]]])
```

Wenn Sie sich die Zeit genommen haben, die obige Einleitung zu lesen, sind Ihnen die Parameter in der **setcookie()**-Syntax bereits vertraut. Wenn Sie diesen Abschnitt übersprungen haben und die Funktionsweise dauerhafter Cookies nicht kennen, sollten Sie zunächst doch die obige Einleitung lesen, dort erhalten Sie alle Informationen zu den **setcookie()**-Parametern.

Bevor wir fortfahren, rate ich Ihnen, den folgenden Satz nicht einmal, nicht zweimal, sondern dreimal zu lesen. Ein Cookie muss gesetzt werden, *bevor* seitenrelevante Informationen an den Browser gesendet werden. Schreiben Sie diesen Satz 500 Mal ab, lassen Sie sich diese Regel auf den Arm tätowieren, bringen Sie sie Ihrem Papagei bei – ganz egal, Hauptsache, Sie denken daran. Mit anderen Worten: Sie können ein Cookie nicht einfach irgendwann setzen. Cookies müssen vor dem Senden browserrelevanter Informationen gesetzt werden, *andernfalls funktionieren sie nicht*.

Eine weitere wichtige Einschränkung, die Sie beachten sollten, ist die, dass Sie ein Cookie nicht setzen und davon ausgehen können, es in derselben Seite verwenden zu können. Entweder muss der Benutzer die Seite aktualisieren (rechnen Sie nicht damit), oder Sie warten die nächste Seitenanforderung ab, bevor die Cookievariable verwendet werden kann.

Dieses Beispiel verdeutlicht, wie Sie mithilfe von **setcookie()** ein Cookie mit Benutzer-ID setzen können:

```
$benutzer_id = " 4139b31b7bab052";
$cookie_setzen = setcookie ("uid", $wert, time()+3600, "/", ".phpre-
zepte.com", 0);
```

Nach der Analyse dieses Codeabschnitts erhalten Sie folgende Ergebnisse:

- Nach dem erneuten Laden oder dem Navigieren zu einer Folgeseite wird die Variable **$benutzer_id** verfügbar und erzeugt die Benutzer-ID **4139b31b7bab052**.
- Der Cookie läuft exakt eine Stunde (3.600 Sekunden) nach dem Senden ab (und wird unbrauchbar).
- Der Cookie kann in allen Serververzeichnissen abgerufen werden.
- Auf diesen Cookie kann nur über die Domäne **phprezepte.com** zugegriffen werden.
- Der Cookie ist über ein unsicheres Protokoll zugänglich.

Das nächste Beispiel, Codebeispiel 13.1, veranschaulicht den Einsatz von Cookies zur Speicherung von Einstellungen zur Seitenformatierung, in diesem Fall die Hintergrundfarbe. Beachten Sie, dass der Cookie nur gesetzt wird, wenn die Formularaktion ausgeführt wurde.

Listing 13.1: Speichern der Lieblingshintergrundfarbe eines Benutzers

```
<?
// Wenn die Variable $hintergrund_farbe vorhanden ist…
if (isset($hintergrund_farbe)) :
    setcookie("hintergrundfarbe", $hintergrund_farbe, time()+3600);
?>

<html>
<body bgcolor="<?=$hintergrund_farbe;?>">

<?
// Andernfalls ($hintergrund_farbe nicht gesetzt) Formular anzeigen
else :
?>
<body bgcolor="with">
<form action="<? print $PHP_SELF; ?>" method="post">
    Welche ist Ihre Lieblingshintergrundfarbe?
    <select name="hintergrundfarbe">
        <option value="red">Rot
        <option value="blue">Blau
        <option value="green">Grün
        <option value="black">Schwarz
    </select>
        <input type="submit" value="Hintergrundfarbe festlegen">
</form>

<?
endif;
?>
</body>
</html>
```

Beim Laden dieser Seite in den Browser prüft das Skript, ob der Cookie namens **hintergrundfarbe** gesetzt wurde. Ist dies der Fall, wird die Hintergrundfarbe der Seite entsprechend dem in der Variable **$hintergrund_farbe** gesetzten Wert eingestellt. Andernfalls wird ein HTML-Formular angezeigt, in dem der Benutzer zur

Angabe seiner Lieblingshintergrundfarbe aufgefordert wird. Nach der Festlegung der Lieblingsfarbe wird beim erneuten Laden der Seite oder beim Anzeigen anderer Seiten der Cookiewert **$hintergrund_farbe** erkannt.

Interessanterweise können Sie Cookienamen auch unter Verwendung der Arraynotierung angeben. Sie könnten Cookienamen als **uid[1]**, **uid[2]**, **uid[3]** usw. angeben und auf diese Werte später wie bei einem normalen Array zugreifen. Ein entsprechendes Beispiel sehen Sie in Codebeispiel 13.2.

Listing 13.2: Zuweisen von Cookienamen nach Arrayindexwert

```
<?
setcookie("phprezepte[UID]", "4139b31b7bab052", time()+3600);
setcookie("phprezepte[Farbe]", "black", time()+3600);
setcookie("phprezepte[Einstellungen]", "english", time()+3600);

if (isset ($phprezepte)) {
while (list ($name, $wert) = each ($phprezepte)) {
echo "$name = $wert<br>\n";
}
}
?>
```

Die Skriptausführung führt neben dem Setzen von drei Cookies auf dem Benutzercomputer zu folgender Ausgabe:

```
UID = 4139b31b7bab052
Farbe = black
Einstellung = english
```

Anmerkung Obwohl die Verwendung arraybasierter Cookies zum Speichern aller möglichen Informationen geeignet ist, sollten Sie bedenken, dass einige Browser (z. B. der Netscape Communicator) die Anzahl der Cookies auf 20 pro Domäne beschränken.

Die vielleicht gängigste Verwendung von Cookies ist die Speicherung einer Benutzer-ID, die später zum Abrufen benutzerspezifischer Informationen eingesetzt wird. Dieser Vorgang wird im nächsten Codebeispiel verdeutlicht, in dem eine Benutzer-ID (UIN) in einer MySQL-Datenbank gespeichert wird. Die gespeicherten Informationen werden später abgerufen und zum Festlegen verschiedener Formatierungseinstellungen für die Seite verwendet.

Für das nächste Codebeispiel wird angenommen, dass in einer Datenbank namens **Benutzer** eine Tabelle mit dem Titel **Benutzerinfo** vorhanden ist. Die Tabelle

Benutzerinfo enthält drei Arten von Informationen, eine Benutzer-ID, den Vornamen und die E-Mail-Adresse. Diese Tabelle wurde unter Verwendung der folgenden Syntax erstellt:

```
mysql>create table Benutzerinfo (
->benutzer_id char(18),
->vorname char(15),
->email char(35));
```

Codebeispiel 13.3 setzt etwa auf der Hälfte von dem an, was ein vollständiges »Registrierungsskript« leistet, beginnend dort, wo Benutzerinformationen (Benutzer-ID, Vorname und E-Mail-Adresse bereits in die Datenbank eingefügt wurden. Damit der Benutzer sich später nicht anmelden muss, wird die Benutzer-ID (zu Beispielzwecken in Aufstellung 13.3 auf den Wert **15** gesetzt) mittels Cookie auf dem Benutzercomputer gespeichert.

Listing 13.3: Abrufen von Benutzerinformationen von einer Datenbank

```
<?
if (! isset($benutzer_id)) :
    $id = "15";
    setcookie ("benutzer_ID", $id, time()+3600);
    print "Auf Ihrem Computer wurde ein Cookie mit Ihrer Benutzer-ID
gespeichert. Bitte aktualisieren Sie die Seite, um Ihre Benutzerinformationen
abzurufen.";
else :
    @mysql_connect("localhost", "web", "4tf9zzzf") or die("Verbindung zum
MySQL-Server konnte nicht hergestellt werden!");
    @mysql_select_db("benutzer") or die("Benutzerdatenbank konnte nicht aus-
gewählt werden!");
    // Abfrage deklarieren
    $abfrage = "SELECT * FROM benutzer_info WHERE benutzer_id =
'$benutzer_id";
    // Abfrage ausführen
    $ergebnis = mysql_query($abfrage);

    $zeile = mysql_fetch_array($ergebnis);
    print "Hi ",$zeile["fname"].",<br>";
    print "Ihre E-Mail-Adresse lautet ".$zeile["email"];

    mysql_close();
endif;
?>
```

Aufstellung 13.3 zeigt, wie nützlich Cookies zur Benutzeridentifikation sind. Das obige Szenario kann auf viele andere Situationen übertragen werden, beispielsweise im Hinblick auf die Anmeldung und eine effektive Verfolgung der Benutzereinstellungen.

Das im nächsten Abschnitt (»Eindeutige Identifikationsnummern«) vorgestellte Codebeispiel zeigt den vollständigen Prozess der Benutzerregistrierung und die anschließende Speichern der eindeutigen Benutzer-ID.

Anmerkung Die in Aufstellung 13.3 verwendeten MySQL-Funktionen wurden in Kapitel 11, »Datenbanken«, vorgestellt.

13.2 Eindeutige Identifikationsnummern

Sie warten sicher schon ungeduldig darauf, eine einfache Methode zum Erstellen von eindeutigen Benutzer-IDs kennen zu lernen. Legen Sie Ihre alten Mathebücher weg, hierzu ist kein abgefahrener Algorithmus aus dem vorletzten Jahrhundert nötig. PHP bietet mit der vordefinierten Funktion **uniqid()** eine einfache Möglichkeit zum Erstellen einer eindeutigen Benutzer-ID.

Die **uniqid()**-Funktion generiert eine aus 13 Zeichen bestehende eindeutige Identifikationsnummer, die auf der aktuellen Uhrzeit basiert. Die Syntax lautet folgendermaßen:

```
int uniqid (string prefix [, boolean lcg])
```

Der Eingabeparameter **prefix** kann verwendet werden, um als Präfix der Benutzer-ID einen bestimmten Zeichenfolgenwert festzulegen. Da es sich bei **prefix** um einen erforderlichen Parameter handelt, muss zumindest ein leerer Wert zugewiesen werden. Falls auf **TRUE** gesetzt, weist der optionale Eingabeparameter **lcg** die **uniqid()**-Funktion zum Erzeugen einer Benutzer-ID an, die 23 Zeichen umfasst. Zur schnellen Erzeugung einer ID rufen Sie die Funktion **uniqid()** einfach mit einem leeren Wert als einzigem Eingabeparameter auf.

```
$eindeutige_id = uniqid("");
// Ein 13 Zeichen umfassender Wert wie beispielsweisevalue ' 39b3209ce8ef2'
wird erzeugt
```

Eine andere Möglichkeit zum Erstellen einer eindeutigen ID besteht darin, dem abgeleiteten Wert über den Eingabeparameter **prefix** eine Zeichenfolge voranzustellen:

```
$eindeutige_id = uniqid("php", TRUE);
// Ein 13 Zeichen umfassender Wert wie beispielsweise 'php39b3209ce8ef2' wird
erzeugt
```

Aufgrund der Tatsache, dass **uniqid()** die eindeutige ID basierend auf der aktuellen Systemzeit erstellt, besteht das (wenn auch geringe) Risiko, dass ein unbefugter Benutzer diese ID errät. Sie können sicherstellen, dass ein absolut willkürlicher ID-Wert generiert wird, indem Sie anhand einer weiteren vordefinierten PHP-Funktion namens **rand()** zunächst ein willkürliches Präfix erstellen. Ein Beispiel:

```
srand ((double) microtime() * 1000000);
$eindeutige_id = uniqid(rand());
```

Die Funktion **srand()** dient zur Initialisierung der Erzeugung einer willkürlichen Zahl. Wenn Sie sicherstellen möchten, dass **rand()** kontinuierlich für die Erzeugung willkürlicher Zahlen sorgt, müssen Sie zunächst **srand()** ausführen. Das Platzieren von **srand()** als Eingabeparameter in **uniqid()** führt dazu, dass **rand()** ausgeführt wird und einen Präfixwert an **uniqid()** zurückgibt. Anschließend wird **uniqid()** ausgeführt und generiert eine eindeutige ID, die sehr schwer zu erraten ist.

Ausgerüstet mit diesem Wissen über die Erstellung eindeutiger IDs können Sie sich nun an die Erstellung eines praxisorientierten Schemas zur Benutzerregistrierung wagen. Bei der ersten Anforderung des Skripts aus Codebeispiel 13.4 wird dem Benutzer ein kleines Formular angezeigt, in das er/sie Name und E-Mail-Adresse eingeben muss. Diese Informationen werden zusammen mit einer (zuvor generierten) eindeutigen ID in die Tabelle **benutzer_info** geschrieben, wie beschrieben in Codebeispiel 13.3. Außerdem wird ein Cookie mit dieser eindeutigen ID auf dem Benutzercomputer gespeichert. Bei nachfolgenden Besuchen dieser Seite wird die Datenbank über die Skriptanforderung basierend auf der eindeutigen Benutzer-ID im Cookie abgefragt, die Benutzerinformationen werden anschließend auf dem Bildschirm angezeigt.

Listing 13.4: Eine vollständige Benutzerregistrierung

```
<?
// Formular erstellen
$formular = "
<form action=\"Codebeispie 13.4.php\" method=\"post\">
<input type=\"hidden\" name=\"formular_gezeigt\" value=\"y\">
Wie lautet Ihr Vorname?:<br>
<input type=\"text\" name=\"vorname\" size=\"20\" maxlength=\"20\"
value=\"\"><br>
Wie lautet Ihre E-Mail-Adresse?:<br>
<input type=\"text\" name=\"email\" size=\"20\" maxlength=\"35\"
value=\"\"><br>
<input type=\"submit\" value=\"Registrieren!\">
</form>
";
```

```
// Wenn das Formular noch nicht angezeigt wurde und der Benutzer über keinen
Cookie verfügt
if ((! isset ($formular_gezeigt)) && (! isset ($benutzer_id))) :

    print $formular;

// Wenn das Formular angezeigt wurde, die Benutzerinformationen jedoch noch
nicht verarbeitet wurden
elseif (isset ($formular_gezeigt) && (! isset ($benutzer_id))) :

    srand ((double) microtime() * 1000000);
    $eindeutige_id = uniqid(rand());
    // MySQL-Serververbindung aufbauen und Benutzerdatenbank auswählen
    @mysql_pconnect("localhost", "web", "4tf9zzzf") or die("Verbindung zu
MySQL-Server nicht möglich!");
    @mysql_select_db("benutzer") or die("Benutzerdatenbank konnte nicht aus-
gewählt werden!");

    // Abfrage deklarieren und ausführen
    $abfrage = "INSERT INTO benutzer_info VALUES('$eindeutige_id', '$fname',
'$email')";
    $ergebnis = mysql_query($abfrage) or die("Benutzerinformationen konnten
nicht eingefügt werden!");

    // Ablaufdatum für Cookie "benutzer_id" auf einen Monat setzen
    setcookie ("benutzer_id", $eindeutige_id, time()+2592000);

    print "Herzlichen Glückwunsch, $vorname! Sie sind jetzt registriert! Ihre
Benutzerinformationen werden bei jedem Besuch dieser Seite angezeigt.";
// Falls Cookie vorhanden ist, Benutzer-ID zum Extrahieren der Informationen
aus der Benutzerdatenbank verwenden
elseif (isset($benutzer_id)) :
    // MySQL-Serververbindung aufbauen und Benutzerdatenbank auswählen
    @mysql_pconnect("localhost", "web", "4tf9zzzf") or die("Verbindung zu
MySQL-Server nicht möglich!");
    @mysql_select_db("benutzer") or die("Benutzerdatenbank konnte nicht aus-
gewählt werden!");

    // Abfrage deklarieren und ausführen
    $abfrage = "SELECT * FROM benutzer_info WHERE benutzer_id =
'$benutzer_id";
```

```
    $ergebnis = mysql_query($abfrage) or die("Benutzerinformationen konnten
nicht abgerufen werden!");

    $zeile = mysql_fetch_array($ergebnis);
    print "Hi ",$zeile["vorname"].",<br>";
    print "Ihre E-Mail-Adresse lautet ".$zeile["email"];

endif;

?>
```

Durch den sinnvollen Einsatz mehrerer **if**-Klauseln können Sie in einem Skript alle Schritte der Registrierung und anschließenden Benutzererkennung abdecken. In diesem Skript werden drei Situationen berücksichtigt:

- Dem Benutzer wurde das Formular noch nicht angezeigt, er/sie besitzt keinen gültigen Cookie. In diesem Fall wird dem Benutzer das Formular angezeigt.
- Dem Benutzer wurde das Formular angezeigt, aber er/sie besitzt noch keinen gültigen Cookie. In dieser Situation werden die Benutzerinformationen in die Datenbank eingegeben und der Cookie wird gesetzt (mit einem Ablaufdatum von einem Monat).
- Der Benutzer kehrt zum Skript zurück. Ist der Cookie noch gültig (er ist noch nicht abgelaufen), wird dieser eingelesen und die relevanten Informationen werden aus der Datenbank abgerufen.

Der allgemeine Prozess aus Codebeispiel 13.4 kann natürlich auf beliebige Datenbanken übertragen werden. Dies verdeutlicht, wenn auch auf einer grundlegenden Ebene, wie viele der größeren Sites benutzerspezifische Einstellungen auf Ihre Site anwenden können und damit das Seitenaussehen exakt auf die Wünsche eines Benutzers abstimmen.

Hier endet die Einführung in PHP und Cookies. Wenn Sie mehr über dieses Thema erfahren möchten, finden Sie weitere Informationen über die nachfolgend aufgeführten Onlineressourcen.

Relevante Links

Wenn Sie weitere Informationen zu Cookies und deren Verwendung suchen, sollten Sie sich folgende Ressourcensammlung ansehen:

- *http://www.cookiecentral.com*
- *http://home.netscape.com/newsref/std/cookie_spec.html*
- *http://builder.com/Programming/Cookies/ss01.html*
- *http://www.w3.org/Protocols/rfc2109/rfc2109*

Wie Sie bereits wissen, können Cookies äußerst nützlich sein, um sich an benutzerspezifische Informationen zu »erinnern«, wenn der Benutzer Ihre Site ein weiteres Mal besucht. Da der Benutzer seinen Browser jedoch so einstellen kann, dass keine Cookies akzeptiert werden, sollten Sie sich jedoch nicht nur auf diese Methode der Benutzerverfolgung stützen. Glücklicherweise stellt PHP eine weitere Methode zur dauerhaften Speicherung von Informationen bereit. Diese Methode wird als Sitzungsverfolgung bezeichnet und im nächsten Abschnitt behandelt.

13.3 Sitzungsverfolgung

Eine *Sitzung* kann am treffendsten als die Zeitspanne eines Sitebesuchs bezeichnet werden, beginnend mit dem Sitezugriff und endend beim Verlassen der Site. Im Verlauf dieser Sitzung können Sie verschiedene Variablen zuweisen, die den Benutzer bei der Navigation durch Ihre Site begleiten. Hierzu müssen Sie in PHP jedoch nicht bündelweise versteckte Formulare oder angehängte URL-Variablen codieren. Dieser ansonsten recht aufwendige Prozess wird durch die so genannte *Sitzungsverfolgung* erfreulich einfach.

Betrachten Sie folgendes Beispiel: Bei Einsatz der Sitzungsverfolgung wird einem Sitebesucher eine eindeutige Sitzungs-ID (SID) zugeordnet. Diese SID wird anschließend in einem Cookie namens **PHPSESSID** an den Benutzerbrowser gesendet. Zur gleichen Zeit wird eine Datei desselben Namens auf dem Server gespeichert. Während der Benutzer durch die Site navigiert, möchten Sie vielleicht bestimmte Variablen als Sitzungsvariablen aufzeichnen. Diese Variablen werden in der besagten Benutzerdatei gespeichert. Nachfolgende Aufrufe einer dieser Variablen vom Typ »Sitzung"« veranlassen den Server, die Benutzersitzungsdatei nach einer entsprechenden Variable zu durchsuchen. Und voilà! Die Sitzungsvariable wird angezeigt. Dies ist grob gesehen das, was bei der Sitzungsverfolgung geschieht. Sie können die Benutzerinformationen natürlich auch zur Speicherung an eine Datenbank oder eine andere Datei weiterleiten.

Klingt interessant? Darauf können Sie wetten. Ausgestattet mit diesen Informationen wird es Ihnen leicht fallen, die verschiedenen verfügbaren Konfigurationsaspekte zu verstehen, die ich nachfolgend erläutern werde. Es gibt drei besonders wichtige Konfigurationsflags. Das Erste, **--enable-trans-id**, muss in den Konfigurationsprozess einbezogen werden, wenn Sie die Vorteile dieses Features nutzen möchten (Erläuterung siehe unten). Die zwei weiteren Konfigurationsflags, **track_vars** und **register_globals**, können nach Bedarf in der Datei **php.ini** aktiviert und deaktiviert werden. Die Auswirkungen einer Aktivierung dieser drei Flags wird nachfolgend näher beschrieben.

13.3.1 --enable-trans-sid

Wenn Sie PHP mit diesem Flag kompilieren, wird an alle relativen URLs automatisch die Sitzungs-ID angehängt. Dieses Anhängen der Sitzungs-ID erfolgt in der Form **session-name=session-id**, wobei **session-name** (der Sitzungsname) in der Datei **php.ini** definiert wird (Erläuterung siehe weiter unten). Wenn Sie nicht diese Methode verwenden möchten, können Sie die konstante SID verwenden.

13.3.2 track_vars

Die Aktivierung von **track_vars** ermöglicht **$HTTP_*_VARS[]**-Arrays. Hierbei steht ***** für die EGPCS-Werte (**Environment**, **Get**, **Post**, **Cookie**, **Server**). Dieses Flag muss aktiviert werden, damit die SID von einer Seite zur anderen weitergegeben wird. Ab PHP-Version 4.03 ist diese Einstellung immer aktiviert.

13.3.3 register_globals

Das Aktivieren dieser Option führt dazu, dass alle EGCPS-Variablen global zugänglich sind. Wenn Sie nicht möchten, dass ein globales Array mit vielleicht unnötigen Daten gefüllt wird, können Sie diese Option deaktivieren. Ist diese Option deaktiviert und wurde gleichzeitig **track_vars** aktiviert, kann über die **$HTTP_*_VARS[]**-Arrays auf alle GPC-Variablen zugegriffen werden. Wenn beispielsweise **register_globals** deaktiviert ist, müssten Sie die vordefinierte Variable **$PHP_SELF** als **$HTTP_SERVER_VARS["PHP_SELF"]**.

Es gibt darüber hinaus einige Konfigurationsaspekte, die Sie berücksichtigen sollten. Diese Direktiven werden in Tabelle 13.1 beschrieben und können in der Datei **php.ini** in ihrer Standardeinstellung angezeigt werden. Die Reihenfolge entspricht der tatsächlichen Anzeigereihenfolge in der Datei.

Direktive	Beschreibung
Session.save_handler = files	Gibt an, wie die Sitzungsinformationen auf dem Server gespeichert werden. Es gibt drei Möglichkeiten: in einer Datei (**files**), in freigegebenem Speicher (**mm**) oder über benutzerdefinierte Funktionen (**User**). Die benutzerdefinierten Funktionen bieten eine einfache Möglichkeit zur Speicherung der Informationen in einem beliebigen Format, beispielsweise in einer Datenbank.
Session.save_path = /tmp	Weist das Verzeichnis zu, in dem PHP-Sitzungen gespeichert werden. Auf der Linux-Plattform reicht die Standardeinstellung (**'/tmp'**) sehr wahrscheinlich aus. Auf der Windows-Plattform sollte diese Einstellung in einen Windows-Pfad geändert werden, da ansonsten Fehler auftreten.

Tabelle 13.1 Direktiven zur Sitzungsverfolgung in der Datei **php.ini**

Direktive	Beschreibung
Session.use_cookies = 1	Falls aktiviert, werden Cookies zum Speichern der Sitzungs-ID auf dem Benutzercomputer eingesetzt.
Session.name = PHPSESSID	Wenn **session.use_cookies** aktiviert ist, wird **session.name** (Sitzungsname) als Cookiename eingesetzt. Der Name darf nur aus alphanumerischen Zeichen bestehen.
Session.auto_start = 0	Falls aktiviert, wird über **session.auto_start** automatisch eine Sitzung initialisiert, sobald ein Client eine Anforderung sendet.
Session.cookie_lifetime = 0	Wenn Sie **session.use_cookies** aktivieren, gibt **session.cookie_lifetime** die Lebensdauer der gesendeten Cookies an. Falls diese Einstellung auf den Wert 0 gesetzt wird, laufen die Cookies am Ende der Benutzersitzung ab.
Session.cookie_path = /	Bei Aktivierung von **session.use_cookies** legt **session.cookie_path** den Pfad des übergeordneten Verzeichnisses fest, in dem gesendete Cookies gültig sind.
Session.cookie_domain =	Bei Aktivierung von **session.use_cookies** legt **session.cookie_domain** die Domäne fest, in der die gesendeten Cookies Gültigkeit besitzen.
Session.serialize_handler = php	Diese Einstellung bestimmt den Namen des Handlers zur Datenserialisierung. Momentan sind für diese Einstellung zwei Werte verfügbar: **php** und **WDDX**.
Session.gc_probability = 1	Diese Einstellung legt die prozentuale Wahrscheinlichkeit fest, mit der die PHP-Bereinigungsroutine aktiviert wird.
Session.gc_maxlifetime = 1440	Gibt die Zeit (in Sekunden) an, nach der Sitzungsdaten als ungültig betrachtet und verworfen werden. Dieser Timer führt eine Rückwärtszählung durch und wird nach dem letzen Sitzungszugriff aktiviert.
Session.referer_check =	Wird diese Einstellung auf eine Zeichenfolge gesetzt, wird bei jeder Anforderung einer sitzungsaktivierten Seite zunächst geprüft, ob die angegebene Zeichenfolge in der globalen Variablen **$HTTP_REFERER** vorhanden ist. Wird die Zeichenfolge nicht gefunden, werden vorhandene Sitzungs-IDs ignoriert.
Session.entropy_file =	Zeigt auf eine externe Datei, die zusätzliche, willkürliche Informationen bereitstellt, die zur Erstellung der Sitzungs-ID verwendet werden. Es gibt zu diesem Zweck auf UNIX-Systemen üblicherweise zwei Geräte, **/dev/random** und **/dev/urandom**. Das **/dev/random**-Gerät sammelt willkürliche Daten aus dem Kernel, das Gerät **/dev/urandom** erzeugt unter Verwendung des MD5-Hashalogorithmus eine willkürliche Zeichenfolge. Kurz gesagt ist **/dev/random** schneller, **/dev/urandom** erzeugt jedoch eine »willkürlichere« Zeichenfolge.

Tabelle 13.1 Direktiven zur Sitzungsverfolgung in der Datei **php.ini** (Forts.)

Direktive	Beschreibung
Session.entropy_length = 0	Unter der Voraussetzung, dass **session.entropy_file** gesetzt ist, gibt **session.entropy_length** die Anzahl der Bytes an, die aus der über **session.entropy_file** angegebenen Datei gelesen werden sollen.
Session.cache_limiter = nocache	Legt die Cachesteuerungsmethode für Sitzungsseiten fest. Es gibt drei mögliche Werte für diese Einstellung, **nocache**, **public** und **private**.
Session.cache_expire = 180	Legt für zwischengespeicherte Sitzungsseiten die TTL (Time To Live, Lebensdauer) in Minuten fest.

Tabelle 13.1 Direktiven zur Sitzungsverfolgung in der Datei **php.ini** (Forts.)

Anmerkung Die Konfigurationsdirektive **session.save_handler** ist so nützlich, dass ich ihr einen ganzen Abschnitt widmen werde. Dieser befindet sich am Ende dieses Kapitels und trägt den Titel »Benutzerrückrufe als Speichermodule«.

Jetzt, da Sie wahrscheinlich die nötigen Konfigurationsanpassungen für Ihren Server vorgenommen haben, werde ich darauf eingehen, wie Sie die Sitzungsverfolgung auf Ihre Site anwenden können. Eigentlich handelt es sich hierbei um einen relativ simplen Prozess, der über den Einsatz verschiedener vordefinierter Funktionen realisiert wird. Das erste Konzept, das ich vorstellen möchte, ist die Initialisierung einer Sitzung über die Funktion **session_start()**. Sie können natürlich auch die zuvor genannte Einstellung **session.auto_start** in der Datei **php.ini** aktivieren. Für die verbleibenden Abschnitte dieses Kapitels werde ich jedoch voraussetzen, dass Sie diese Einstellung nicht vorgenommen haben, um die Konsistenz meiner Beispiele zu wahren. Die Syntax von **session_start()** ist einfach, es werden keine Eingabeparameter benötigt, als Rückgabe erhält der Entwickler einen booleschen Wert zum Erfolg der Funktionsausführung.

session_start()

Die Funktion **session_start()** hat zwei Verwendungszwecke. Sobald aufgerufen, prüft **session_start()**, ob der Benutzer bereits eine Sitzung gestartet hat. Ist dies nicht der Fall, wird eine Sitzung initialisiert. Die Syntax lautet:

```
boolean session_start()
```

Nach dem Sitzungsstart erfüllt **session_start()** drei Zwecke: Zuweisen einer SID für den Benutzer, Senden eines Cookies (falls **session.use_cookies** in der Datei **php.ini** aktiviert wurde) und Erstellen der Sitzungsdatei auf dem Server. Die zweite Funktion von **session_start()** besteht darin, die PHP-Engine darüber zu informieren, dass weitere Sitzungsvariablen in dem Skript verwendet werden können, in dem **session_start()** ausgeführt wird.

Eine Sitzung wird durch einen einfachen Aufruf von **session_start()** gestartet:

```
session_start();
```

Genau wie eine Sitzung erstellt werden kann, kann sie später auch wieder gelöscht werden. Dies wird über die Funktion **session_destroy()** erreicht.

> **Tipp** Die Funktion **session_start()** gibt unabhängig vom tatsächlichen Ergebnis immer **TRUE** aus. Daher sollte diese Funktion nicht in **if**-Klauseln oder zusammen mit **die()**-Anweisungen verwendet werden.

session_destroy()

Die Funktion **session_destroy()** löscht alle Daten, die in Verbindung zu der aktuellen Benutzersitzung stehen. Die Syntax lautet:

```
boolean session_destroy()
```

Beachten Sie, dass diese Funktion *nicht* zum Löschen von Cookies im Benutzerbrowser eingesetzt werden kann. Wenn Sie nicht an einer Verwendung des Cookies über das Sitzungsende hinaus interessiert sind, können Sie einfach in der Datei **php.ini** für die Einstellung **session.cookie_lifetime** den Wert 0 festlegen (Standardwert). Anwendungsbeispiel:

```
<?
session_start();
// ein bisschen Sitzungskram durchführen
session_destroy();
?>
```

Nun, da Sie wissen, wie Sitzungen erstellt und gelöscht werden, können Sie sich mit den verschiedenen Sitzungsvariablen beschäftigen. Die wichtigste hierbei ist wahrscheinlich die Sitzungs-ID. Diese kann über die Funktion **session_id()** leicht abgerufen werden.

session_id()

Die Funktion **session_id()** gibt die ursprünglich über **session_start()** erstellte SID zurück. Die Syntax lautet hierbei wie folgt:

```
string session_id([string sid])
```

Wenn Sie mithilfe eines optionalen Eingabeparameters **sid** eine Sitzungs-ID bereitstellen, wird die Sitzungs-ID des Benutzers geändert. Beachten Sie jedoch, dass hierdurch der Cookie *nicht* erneut gesendet wird. Sehen Sie sich das folgende Beispiel an:

```
<?
session_start ();
print "Ihre Sitzungs-ID lautet ".session_id();
session_destroy();
?>
```

Wenn Sie diesen Code ausführen, erhalten Sie in etwa die folgende Browseranzeige:

```
Ihre Sitzungs-ID lautet 967d992a949114ee9832f1c11cafc640
```

Wie erstellen Sie also eigene Sitzungsvariablen? Zu diesem Zweck steht die leicht einsetzbare Funktion **session_register()** zur Verfügung.

session_register()

Die Funktion **session_register()** registriert einen oder mehrere Variablennamen für die aktuelle Benutzersitzung. Die Syntax lautet:

```
boolean session_register (mixed varname1 [, mixed varname2 …])
```

Beachten Sie hierbei, dass Sie keine Variablen, sondern nur die Namen der Variablen registrieren. **Session_register()** ruft intern ebenfalls **session_start()** auf, sodass implizit eine neue Sitzung gestartet wird, falls eine solche noch nicht vorhanden ist.

Vor der weiteren Ausführung von **session_register()** möchte ich Ihnen eine weitere sitzungsorientierte Funktion vorstellen, mit der geprüft werden kann, ob eine bestimmte Variable registriert wurde oder nicht. Diese Funktion trägt den Namen **session_is_registered()**.

session_is_registered()

Es ist oft nützlich zu wissen, ob eine Variable bereits registriert wurde oder nicht. Diese Prüfung können Sie mithilfe der Funktion **session_is_registered()** vornehmen. Die Syntax lautet:

```
boolean session_is_registered (string varname)
```

Zur Veranschaulichung der Verwendung von **session_register()** und **session_is_registered()** werde ich das Beispiel heranziehen, dass eigentlich jeder verwendet: einen Zugriffszähler. Siehe hierzu Codebeispiel 13.5.

Listing 13.5: Ein benutzerspezifischer Zugriffszähler

```
<?
session_start();
if (! session_is_registered("zugriffe")) :
```

```
    session_register('zugriffe');
endif;
$zugriffe++;
print "Sie haben diese Seite $zugriffe Mal besucht";
?>
```

Sie können Sitzungsvariablen natürlich nicht nur erstellen, sondern auch löschen. Dies erreichen Sie mit der Funktion **session_unregister()**.

session_unregister()

Eine Sitzungsvariable kann durch einen Aufruf von **session_unregister()** gelöscht werden. Die Syntax lautet wie folgt:

```
boolean session_unregister (string varname)
```

Der Eingabeparameter **varname** ist der Name der Sitzungsvariablen, die Sie löschen möchten.

```
<?
session_start();
session_register('benutzername');
// Variable benutzername nach Bedarf verwenden
session_unregister('benutzername');
session_destroy();
?>
```

Wie im Falle von **session_register()** müssen Sie daran denken, dass Sie den Eingabeparameter **varname** nicht als tatsächliche Variable angeben (d.h. ohne Dollarzeichen [**$**]). Statt dessen verwenden Sie den *Namen* der Variablen.

session_encode()

Die Funktion **session_encode()** bietet eine besonders komfortable Methode zur Formatierung von Sitzungsvariablen zur Speicherung, beispielsweise in einer Datenbank. Die Syntax lautet folgendermaßen:

```
boolean session_encode()
```

Das Ausführen dieser Funktion führt dazu, dass alle Sitzungsdaten in einer einzigen Zeichenfolge formatiert werden. Diese Zeichenfolge kann anschließend zur Speicherung in eine Datenbank eingefügt werden.

In Codebeispiel 13.6 sehen Sie ein Beispiel zum Einsatz von **session_encode()**. Angenommen, für einen »registrierten« Benutzer wird die eindeutige Sitzungs-ID in einem Cookie gespeichert, der wiederum auf dem Benutzercomputer gespei-

chert wird. Wenn der Benutzer die in Codebeispiel 13.6 verwendete Seite anfordert, wird die Benutzer-ID aus dem Cookie abgerufen. Dieser Wert wird anschließend als Sitzungs-ID zugewiesen. Es werden bestimmte Sitzungsvariablen erstellt und mit Werten versehen, anschließend werden die Informationen mithilfe der Funktion **session_encode()** formatiert und in eine MySQL-Datenbank eingefügt.

Listing 13.6: Verwenden von session_encode() zur Speicherung von Daten in einer MySQL-Datenbank

```
<?
// Sitzung initiieren und einige Sitzungsvariablen erstellen
session_register('hintergrundfarbe');
session_register('schriftfarbe');

// Annahme, dass Variable $benutzer_id (mit eindeutiger Benutzer-ID)
// in einem Cookie auf dem Benutzerrechner gespeichert ist

// session_id() verwenden, um Sitzungs-ID auf die in Cookie
// und Datenbank gespeicherte Benutzer-ID zu setzen
$id = session_id($benutzer_id);

// diese Variablen könnten per HTML-Formular über den Benutzer eingestellt
werden
$hintergrundfarbe = "white";
$schriftfarbe = "blue";

// alle Sitzungsdaten in einer einzigen Zeichenfolge formatieren
$benutzer_daten = session_encode();

// MySQL-Serververbindung aufbauen und Benutzerdatenbank auswählen
@mysql_pconnect("localhost", "web", "4tf9zzzf") or die("Verbindung zu MySQL-
Server nicht möglich!");
@mysql_select_db("benutzer") or die("Benutzerdatenbank konnte nicht ausge-
wählt werden!");

// Seiteneinstellungen des Benutzers aktualisieren
$abfrage = "UPDATE benutzer_info set seiten_daten='$benutzer_data' WHERE
benutzer_id= '$id'";
$ergebnis = mysql_query($abfrage) or die("Benutzerinformationen konnten nicht
aktualisiert werden!");
?>
```

Wie Sie sehen können, wird durch das schnelle Konvertieren aller Sitzungsvariablen in eine einzige Zeichenfolge die Ablaufverfolgung verschiedener Spaltennamen zum Speichern und Abrufen von Daten überflüssig. Gleichzeitig sparen Sie einige Codezeilen, die normalerweise für das Speichern und Abrufen der Daten erforderlich wären.

session_decode()

Über **session_encode()** formatierte Daten können mithilfe von **session_decode()** wieder decodiert werden. Die Syntax lautet folgendermaßen:

```
string session_decode(string sitzungs_daten)
```

Der Eingabeparameter **sitzungs_daten** ist die formatierte Zeichenfolge (bestehend aus Sitzungsvariablen), wahrscheinlich das Ergebnis einer Datei- oder Datenbankabfrage. Die Zeichenfolge wird decodiert und alle Sitzungsvariablen in der Zeichenfolge werden in ihr ursprüngliches Variablenformat zurückkonvertiert.

Codebeispiel 13.7 zeigt, wie zuvor codierte Sitzungsvariablen mithilfe von **session_decode()** wieder decodiert werden. Angenommen, Sie verfügen über eine MySQL-Tabelle namens **benutzer_info**, die aus zwei Spalten besteht, **benutzer_id** und **seiten_daten**. Die eindeutige Benutzer-ID, die in einem Cookie auf dem Benutzercomputer gespeichert wird, wird zum Abrufen codierter Sitzungsdaten verwendet, die in der Spalte **seiten_daten** gespeichert sind. Die Spalte **seiten_daten** speichert eine codierte Zeichenfolge aus Variablen, von denen eine die bevorzugte Hintergrundfarbe des Benutzers angibt, gespeichert in der Variablen **$hintergrundfarbe**.

Listing 13.7: Decodieren der in einer MySQL-Datenbank gespeicherten Sitzungsdaten

```
<?
// Annahme, dass Variable $benutzer_id (mit eindeutiger Benutzer-ID)
// in einem Cookie auf dem Benutzerrechner gespeichert ist

$id = session_id($benutzer_id);

// MySQL-Serververbindung aufbauen und Benutzerdatenbank auswählen
@mysql_pconnect("localhost", "web", "4tf9zzzf") or die("Verbindung zu MySQL-
Server nicht möglich!");
@mysql_select_db("benutzer") or die("Benutzerdatenbank konnte nicht ausge-
wählt werden!");

// Daten aus der MySQL-Tabelle auswählen
$abfrage = "SELECT seiten_daten FROM benutzer_info WHERE benutzer_id= '$id'";
```

```
$ergebnis = mysql_query($abfrage);
$benutzer_daten = mysql_result($ergebnis, 0, "fname");

// Daten decodieren
session_decode($benutzer_daten);

// eine der neu erstellten Sitzungsvariablen ausgeben
print "HINTERGRUNDFARBE: $hintergrundfarbe";

?>
```

Wie Sie aus den zwei vorherigen Codebeispielen ersehen können, können **session_encode()** und **session_decode()** beim Speichern und Abrufen von Sitzungsdaten äußerst effizient eingesetzt werden.

13.3.4 Benutzerrückrufe als Speichermodule

Obwohl sich Daten sehr gut in Dateien speichern lassen, möchten Sie Ihre Daten vielleicht unter Verwendung eines anderen Mediums speichern, z. B. in einer Datenbank. Oder vielleicht möchten Sie Skripts in anderen Sites wieder verwenden, jedoch unter Einsatz anderer Datenbanken. Ein weiteres Dilemma ist die gemeinsame Nutzung von Sitzungsdaten über verschiedene Server hinweg. Dies gestaltet sich ziemlich schwierig, wenn Sie die PHP-Standardroutinen zur Speicherung von Sitzungsdaten in einer Datei verwenden. Sie werden glücklich sein zu hören, dass sämtliche dieser Erweiterungen zur PHP-Sitzungsverfolgung leicht zu realisieren sind, da PHP dem Entwickler über die vordefinierte Funktion **session_set_save_handler()** eine Methode zur Festlegung eigener Speicherroutinen an die Hand gibt.

Die Funktion **session_set_save_handler()** definiert Speicher- und Abruffunktionen auf Benutzerebene. Die Syntax lautet folgendermaßen:

```
void session_set_save_handler (string open, string close, string read, string
write, string destroy, string gc)
```

Die sechs Eingabeparameter entsprechen den sechs Funktionen, die allgemein als die PHP-Funktionen zur Sitzungsverarbeitung bezeichnet werden. Die Funktion **session_set_save_handler()** ermöglicht Ihnen eine Neudefinition dieser Funktionen, ohne dass dies Auswirkungen auf Skripts hat, die die vordefinierten PHP-Sitzungsfunktionen aufrufen. Obwohl Sie die Namen der Funktionen beliebig abändern können, muss jede der Funktionen als Eingabe mit einem bestimmten Parametersatz versehen werden. Bevor wir zu einem Beispiel kommen, sollten Sie sich kurz Tabelle 13.2 anschauen. Dort werden die sechs Funktionen und ihre Eingabeparameter erläutert.

Anmerkung Um **session_set_save_handler()** verwenden zu können, müssen Sie die Einstellung **session.save_handler** in der Datei **php.ini** auf den Wert **user** setzen.

Parameter	Beschreibung
sess_close()	Wird aufgerufen, wenn ein Skript mit den Sitzungsfunktionen beendet wird. Dieser Parameter ist *nicht* mit **sess_destroy()** identisch, der zum Löschen der Sitzungsvariablen eingesetzt wird. Für **sess_close()** gibt es keine Eingabeparameter.
sess_destroy($session_id)	Löscht alle Sitzungsdaten. Der Eingabeparameter **$session_id** gibt an, welche Sitzung gelöscht werden soll.
sess_gc($maxlifetime)	Löscht Sitzungen, die abgelaufen sind. Die Lebensdauer wird durch den Eingabeparameter **$maxlifetime** angegeben (in Sekunden). Dieser Parameter wird aus der Datei **php.ini** gelesen und entspricht **session.gc_lifetime**.
sess_open($sess_path, $sess_name)	Wird aufgerufen, wenn eine neue Sitzung initialisiert wird, entweder über **session_start()** oder **session_register()**. Die zwei Eingabeparameter **$sess_path** und **$sess_name** werden aus der Datei **php.ini** gelesen und entsprechen jeweils den Parametern **session.save_path** und **session.name**.
sess_read($key)	Wird zum Abruf der Werte einer Sitzungsvariablen verwendet, angegeben durch den Eingabeparameter **$key**.
sess_write($key, $value)	Wird zum Schreiben der Sitzungsdaten eingesetzt. Sämtliche über **sess_write()** gespeicherte Daten können später über **sess_read()** abgerufen werden. Der Eingabeparameter **$key** entspricht einem Sitzungsvariablennamen, **$value** entspricht dem **$key** zugewiesenen Wert.

Tabelle 13.2 Die sechs Eingabeparameter für die Funktion **session_set_savehandler()**

Jetzt, da Sie mehr über die zu definierenden Funktionen wissen, werde ich ein Beispiel zur MySQL-basierten Implementierung der Funktionen zur Sitzungsverfolgung anführen. Siehe hierzu Codebeispiel 13.8.

Listing 13.8: MySQL-Implementierung der Funktionen zur Sitzungsverfolgung

```
<?
// MySQL-Implementierung der Funktionen zur Sitzungsverfolgung

// MySQL-Serverhost, Benutzername und Kennwortwerte
$HOST = "localhost";
$USER = "web";
$PSWD = "4tf9zzzf";
```

```
// Datenbank und Tabellennamen
$SESS_DBNAME = "benutzer";
$SESS_TBLNAME = "benutzer_sitzungs_daten";

// sess.gc_lifetime-Wert aus der php.ini-Datei abrufen
$lebensdauer_sitzung = get_cfg_var("sess.gc_lifetime");

// Funktion: mysql_sess_open()
// über mysql_sess_open() Verbindung zum MySQL-Server herstellen
// und Datenbank auswählen

function mysql_sess_open($speicher_pfad, $sitzungs_name) {
    GLOBAL $HOST, $USER, $PSWD, $SESS_DBNAME;

    @mysql_pconnect($HOST, $USER, $PSWD) or die("Verbindung zu MySQL-Server
nicht möglich!");
    @mysql_select_db($SESS_DBNAME) or die("Datenbank kann nicht ausgewählt
werden!");
}

// Funktion: mysql_sess_close()
// mysql_sess_close() wird in der MySQL-Implementierung nicht benötigt.
// *ABER* muss trotzdem definiert werden

function mysql_sess_close() {
    return true;
}

// Funktion: mysql_sess_read()
// mysql_sess_read() liest die Informationen aus der MySQL-Datenbank

function mysql_sess_read($schluessel) {
    GLOBAL $SESS_TBLNAME;

    $abfrage = "SELECT value FROM $SESS_TBLNAME WHERE sess_key = '$schluessel'
AND
                sess_expiration >". time();
    $ergebnis = mysql_query($abfrage);

    // wenn Sitzungswert gefunden, zurückgeben
    if (list($wert) = mysql_fetch_row($ergebnis)) :
```

```
        return $wert;
    endif;

    return false;
}

// Funktion: mysql_sess_write()
// mysql_sess_write() schreibt die Informationen in die MySQL-Datenbank

function mysql_sess_write($schluessel, $wer) {
    GLOBAL $lebensdauer_sitzung, $SESS_TBLNAME;

    // Ablaufdatum festlegen
    $ablauf = time() + $lebensdauer_sitzung;

    $abfrage = "INSERT INTO $SESS_TBLNAME VALUES('$schluessel, '$ablauf,
'$wert)";
    $ergebnis = mysql_query($abfrage);

    // wenn die Einfügeanforderung aufgrund der primären Schlüsseleinstellung
für die Spalte sess_key fehlschlägt,
    // statt dessen Aktualisierung ausführen

    if (! $ergebnis) :
$abfrage = "UPDATE $SESS_TBLNAME SET sess_expiration = '$ablauf,
                          sess_value='$wert' WHERE sess_key = '$schluessel";
           $ergebnis = mysql_query($ergebnis);
    endif;
}

// Funktion: mysql_sess_destroy()
// mysql_sess_destroy() löscht alle Tabellenzeilen mit dem Sitzungsschluessel
=$sitzungs_id

function mysql_sess_destroy($sitzungs_id) {
    GLOBAL $SESS_TBLNAME;

    $abfrage = "DELETE FROM $SESS_TBLNAME WHERE sess_key = '$sitzungs_id'";
    $ergebnis = mysql_result($abfrage);
```

```
    return $ergebnis;
}

// Funktion: mysql_sess_gc()
// mysql_sess_gc() löscht alle Tabellenzeilen,
// für die gilt: Ablauf < aktuelle Uhrzeit – session.gc_lifetime

function mysql_sess_gc($max_Lebensdauer) {
    GLOBAL $SESS_TBLNAME;

    $abfrage = "DELETE FROM $SESS_TBLNAME WHERE sess_expiration < " . time();
    $ergebnis = mysql_query($abfrage);

    return mysql_affected_rows();
}

session_set_save_handler("mysql_sess_open", "mysql_sess_close",
"mysql_sess_read", "mysql_sess_write", "mysql_sess_destroy",
"mysql_sess_gc");
?>
```

Nachdem Sie diese sechs Funktionen definiert haben, können Sie jede dieser Funktionen über den abstrakten Namen ausführen (**sess_close()**, **sess_destroy()**, **sess_gc()**, **sess_open()**, **sess_read()** oder **sess_write()**). Das Schöne daran ist, dass Sie anschließend so viele Implementierungen wie nötig schreiben und **session_set_save_handler()** nach Bedarf neu definieren können.

13.4 Projekt: Erstellen eines Besucherprotokolls

Es ist häufig nützlich, Informationen zu den Besuchern einer Site aufzuzeichnen. Wie Sie bereits wissen, ist dies eine häufig eingesetzte Methode bei Webmarketingagenturen, Webportalen und verschiedenen anderen Sites, die daran interessiert sind, mehr über ihre jeweiligen Besucher zu erfahren. Obwohl diese Systeme sehr schnell äußerst kompliziert werden können, bieten sie doch verschiedene Vorteile, die auch bei der Erstellung eines relativ simplen Protokollierungssystems genutzt werden können. Ich werde Ihnen zeigen, wie Sie ein solches einfaches System unter Verwendung von PHP, MySQL und Cookies implementieren.

Achtung Dieses Projekt beinhaltet das in Kapitel 8 behandelte Projekt zur Browsererkennung. Wenn Sie Kapitel 8 bzw. dieses Projekt übersprungen haben, sollten Sie zumindest den Projektcode überfliegen, bevor Sie fortfahren.

Wie schon gesagt, unser System wird relativ einfach gestaltet sein und nur die Besucherzugriffe der Siteindexseite überwachen. Wenn der Besucher auf die Seite zugreift, prüft PHP, ob auf dem Besuchercomputer ein gültiger Cookie vorhanden ist. Ist ein solcher Cookie vorhanden, kann angenommen werden, dass der Besucher innerhalb eines vorgegebenen Zeitrahmens (vom Siteadministrator in einer Initialisierungsdatei festgesetzt) das letzte Mal auf die Site zugegriffen hat. Das Skript nimmt diesen Zugriff daher nicht in die Zählung auf. Ist kein Cookie oder ein abgelaufener Cookie vorhanden, hat der Benutzer die Site entweder noch nie besucht oder zwischen zwei Zugriffen ist der festgelegte Zeitrahmen überschritten worden. In diesem Fall werden Informationen in der MySQL-Tabelle aufgezeichnet. Darüber hinaus wird ein neuer Cookie an den Benutzercomputer gesendet.

Wie kann in PHP ein solches Skript erstellt werden? Zunächst müssen Sie eine MySQL-Tabelle zur Informationsspeicherung erstellen.

```
mysql>create table Besucher (
       ->browser char(85) NOT NULL,
       ->ip char(30) NOT NULL,
       ->host char(85) NOT NULL,
       ->zugriffszeit datetime NOT NULL
       ->);
```

Wozu dient jede dieser Spalten? Die Browserspalte enthält Informationen, die in direktem Zusammenhang mit dem Benutzerbrowser stehen. Diese Informationen werden durch die PHP-Variable **$HTTP_USER_AGENT** bereitgestellt. Die **ip**-Spalte enthält die IP-Adresse des Benutzers. Die Spalte **host** enthält ISP-Informationen zum Ursprung der IP-Adresse. Die Spalte **zugriffszeit** schließlich gibt Datum und Uhrzeit des Benutzerzugriffs auf die Site an.

Anmerkung Jean-Pierre Dézélus, Webmaster von phpinfo.net *(http://www.phpinfo.net)*, hat eine sehr leistungsstarke Anwendung zur Besucherprotokollierung entwickelt. Eine Implementierung dieser Anwendung finden Sie auf der vorgenannten Site, der Code steht sogar zum Download bereit. Sie sollten allerdings besser Ihr Französischwörterbuch hervorkramen, bevor Sie diese Site besuchen.

Als Nächstes wird die Initialisierungsdatei für die Anwendung erstellt, **init.inc**, siehe Codebeispiel 13.9. In dieser Datei werden sowohl die globalen Variablen als auch Hauptfunktionen gespeichert. Beachten Sie, dass die Funktionalität des in Kapitel 8 erarbeiteten Skripts **sniffer.php** in der Funktion **viewStats()** eingesetzt wird. Dieses Skript wird, falls erforderlich, zusammen mit der **init.inc**-Datei eingeschlossen. Nehmen Sie sich einen Moment Zeit, um das Skript und die Kommentare zu lesen.

Listing 13.9: Erstellen der Anwendungsinitialisierungsdatei (init.inc)

```
<?
// Dateiname: init.inc
// Zweck: Initialisierungsdatei für das Besucherprotokollierungs-Projekt

// Variablen für die Datenbankverbindung
$host = "localhost";
$benutzer = "web";
$kennwort = "4tf9zzzf";

// Datenbankname
$datenbank = "meinSniffer";

// Tabellenname abrufen
$besucher_tabelle = "Besucher";

// Verbindung zum MySQL-Server herstellen
@mysql_pconnect($host, $benutzer, $kennwort) or die("Verbindung zu MySQL-Ser-
ver nicht möglich!");

// Datenbank auswählen
@mysql_select_db($datenbank) or die("Auswahl der Datenbank $datenbank nicht
möglich!");

// Anzahl der letzten Besucher in einer Tabelle anzeigen
$maxAnzBesucher = "5";

// Cookiename. Dieser kann auf einen beliebigen Wert gesetzt werden. Die aktu-
elle Einstellung ist o.k.
$cookieName = "besucherProtokoll";

// Cookiewert. Dieser kann auf einen beliebigen Wert gesetzt werden. Die aktu-
elle Einstellung ist o.k.
$cookieWert="1";

/*
Zeitrahmen, nach dem erneuter Besuch durch denselben Benutzer als neuer
Zugriff erkannt wird
Wenn $zeitLimit auf 0 gesetzt wird, wird jeder Besucherzugriff auf der Seite
aufgezeichnet,
```

```
unabhängig von der Frequenz. Alle weiteren Ganzzahleneinstellungen werden als
Anzahl in SEKUNDEN gewertet,
die zwischen zwei Besuchen verstreichen müssen, bevor der Zugriff erneut pro-
tokolliert wird.
*/
$zeitLimit = 3600;

// Wie soll die Tabelle angezeigt werden?
$titel_farbe = "#cbda74";
$tabellen_farbe = "#000080";
$zeilen_farbe = "#c0c0c0";
$schrift_farbe = "#000000";
$schrift_art = "Arial, Times New Roman, Verdana";
$schrift_groesse = "-1";

// Funktion: benutzerProtokoll
// Zweck: Aufzeichnen von Benutzerinformationen in der MySQL-Tabelle
$besucher_tabelle
function benutzerProtokoll() {

    GLOBAL $besucher_tabelle, $HTTP_USER_AGENT, $REMOTE_ADDR, $REMOTE_HOST;

    // Falls Benutzer auf dem Internetserver arbeitet, $REMOTE_HOST auf
'localhost' setzen
    // Sie möchten vielleicht interne Besucher nicht protokollieren, da es
sich wahrscheinlich
    // um weitere Mitglieder des Entwicklungsteams handelt.
    if ($REMOTE_HOST == "") :
        $REMOTE_HOST = "localhost";
    endif;

    // Gültiges MySQL-Format für Datum und Uhrzeit formatieren
    $zeitstempel = date("d.m.y, H:i:s");

    // Benutzerdaten in MySQL-Tabelle einfügen
    $abfrage = "INSERT INTO $lesezeichen_tabelle
                VALUES('$HTTP_USER_AGENT', '$REMOTE_ADDR', '$REMOTE_HOST',
'$zeitstempel)";

    $ergebnis = @mysql_query($abfrage);
```

```
} // benutzerProtokoll

// Funktion: zeigeStat
// Zweck: Extrahieren und Formatieren von Informationen in der MySQL-Tabelle
$besucher_tabelle
function zeigeStat() {

    // Einige globale Variablen einschließen
    GLOBAL $besucher_tabelle, $maxAnzBesucher, $tabellen_farbe,
$titel_farbe;
    GLOBAL $zeilen_farbe, $schrift_farbe, $schrift_art, $schrift_groesse;

    // Aktuellen Wert für $maxAnzBesucher aus der MySQL-Tabelle auswählen
    $abfrage = "SELECT browser, ip, host, zugriffszeit FROM $besucher_tabelle
                ORDER BY zugriffszeit desc LIMIT 0, $maxAnzBesucher";

    $ergebnis = @mysql_query($abfrage);

    // Abgerufene Daten formatieren und ausgeben
    print "<table cellpadding=\"2\" cellspacing=\"1\" width = \"800\" border
= \"0\" bgcolor=\"$tabellen_farbe\">";
    print "<tr bgcolor= \"$titel_farbe\"><th>Browser</th><th>IP</
th><th>Host</th><th>Zeit des Zugriffs</th></tr>";

    while($zeile = mysql_fetch_array($ergebnis)) :

        // Diese Funktionen sind in 'sniffer.inc'
        list ($browser_typ, $browser_version) = browser_info
($zeile["browser"]);
        $betriebs_system = opsys_info ($zeile["browser"]);

        print "<tr bgcolor=\"$zeilen_farbe\">";
        print "<td><font color=\"$schrift_farbe\" face=\"$schrift_art\"
size=\"$schrift_groesse\">";
        print "$browser_typ $browser_version - $betriebs_system</font></td>";
        print "<td><font color=\"$schrift_farbe\" face=\"$schrift_art\"
size=\"$schrift_groesse\">".$zeile["ip"]."</font></td>";
        print "<td><font color=\"$schrift_farbe\" face=\"$schrift_art\"
size=\"$schrift_groesse\">".$zeile["host"]."</font></td>";
        print "<td><font color=\"$schrift_farbe\" face=\"$schrift_art\"
size=\"$schrift_groesse\">";
```

```
        print $zeile["Zeit des Zugriffs"]."</font></td>";
        print "</tr>";

    endwhile;
    print "</table>";
} // zeigeStat
?>
```

Als Nächstes fügen Sie das in Codebeispiel 13.10 gezeigte Skript ein. Über dieses Skript prüfen Sie, ob ein gültiger Cookie vorhanden ist und rufen (falls erforderlich) die Funktion **benutzerProtokoll()** auf. Ich werde diesen Code zusammen mit einer sehr einfachen Indexdatei namens **index.php** einschließen.

Listing 13.10: Prüfen auf einen gültigen Cookie (index.php)

```
<?
include("init.inc");
// Wenn kein gültiger Cookie gefunden wird
if (! isset($$cookieName)) :
    // Neuen Cookie setzen
    setcookie($cookieName, $cookieWert, time()+$zeitLimit);
    // Besucherinformationen aufzeichnen
    benutzerProtokoll();
endif;
?>

<html>
<head>
<title>Willkommen auf meiner Site!</title>
</head>
<body bgcolor="#c0c0c0" text="#000000" link="#808040" vlink="#808040"
alink="#808040">
Willkommen auf meiner Site. <a href = "besucher.php">Hier können Sie sehen,
wer ebenfalls vor kurzem diese Site besucht hat</a>.
</body>
</html>
```

Wie werden die in der MySQL-Datenbank gespeicherten Daten im Browser angezeigt? Sie erreichen dies, indem Sie die Funktion **zeigeStat()** einfach in einer separaten Datei (**besucher.php**) platzieren:

```
<html>
<?
// Funktionalität zur Browsererkennung einschließen
```

```
include("sniffer.inc");
// Initialisierungsdatei einschließen
include("init.inc");
?>
<head>
<title>Aktuell <?=$maxAnzBesucher;?> Besucher</title>
</head>
<body bgcolor="#ffffff" text="#000000" link="#808040" vlink="#808040"
alink="#808040">
<?
zeigeStat();
?>
</body>
</html>
```

Alternativ könnten Sie den gesamten HTML-Code in die **zeigeStat()**-Funktion packen, die Dateien **sniffer.inc** und **init.inc** einschließen und **zeigeStat()** in einer separaten Datei aufrufen. Die Vorgehensweise richtet sich danach, wie stark Sie die Formatierung der Seite konsolidieren möchten. Die in Abbildung 13.1 gezeigte Ausgabe wurde mithilfe der aktuellen Tabellenformatierungseinstellungen in **init.inc** und unter Verwendung der Funktion **zeigeStat()** erzeugt.

Browser	IP	Host	Zeit des Zugriffs
Netscape 4.75 - Linux	134.43.112.41	host12.cob.ohio-state.edu	23.01.01 12:16:00
Opera 4.02 - Windows	212.23.456.123	nat-7.cat.com	23.01.01 11:55:00
IE 5.0 - Windows	62.4.156.931	paris11-nas4-46-208.dial.proxad.net	23.01.01 11:24:00
IE 5.0 - Windows	213.43.48.52	ppp-127.dialup.osu.edu	23.01.01 11:21:00
IE 5.0 - Windows	245.12.234.512	ca.ol23.att.net	23.01.01 11:20:00

Abbildung 13.1 Über **zeigeStat()** erzeugte Beispielausgabe

Dieses Skript könnte auf vielfältige Weise weiter verbessert werden. Eine häufig eingesetzte Methode zur Besucherprotokollierung besteht in der Zuweisung einer Identifikationsnummer zu jeder Seite, die protokolliert werden soll. Anschließend können Sie die Besucher verfolgen, während diese von Seite zu Seite navigieren. Dies im obigen Projektbeispiel etwa durch eine einfache Erweiterung der MySQL-Tabelle um eine Spalte zur Speicherung des Seitenindexwertes geschehen. Anschließend bearbeiten Sie die Funktion **benutzerProtokoll()** so, dass diese mit einem Eingabeparameter versehen werden kann, aus dem diese ID zur Protokollierung übergeben werden könnte. Darüber hinaus könnten Sie jeden Cookie so abändern, dass dieser die Seiten-ID speichert. Auf diese Weise können Sie bei der Besucheranforderung einer protokollierten Seite prüfen, ob ein entsprechender Cookie vorhanden ist.

13.5 Wie geht es weiter?

In diesem Kapitel wurde eines der spannendsten Features von PHP behandelt – die Sitzungsverfolgung. Im Einzelnen wurden folgende Themen behandelt:

- Grundlegendes zu Cookies
- Cookies und PHP
- Eindeutige Identifikationsnummern
- Szenarien zur Benutzerregistrierung
- Einführung in Sitzungen
- Die Sitzungsparameter der Datei **php.ini**
- Die vordefinierten Sitzungsfunktionen von PHP
- Die Funktion **session_set_save_handler()**

Sitzungen bieten insbesondere den Entwicklern ungeahnte Möglichkeiten, die an der Entwicklung wirklich benutzerorientierter Websites interessiert sind. Ich rate Ihnen dringend, sich eingehender mit der PHP-Funktionalität zur Sitzungsverarbeitung auseinander zu setzen, da die bereitgestellten Funktionen häufig sehr nützlich sein können.

Mit diesem Kapitel endet Teil II dieses Buches. In Teil III, »PHP für Fortgeschrittene«, erhalten Sie einen Überblick über die Integration von PHP und XML. Bleiben Sie dran, jetzt wird es erst richtig interessant!