Kapitel 3

jetzt lerne ich

Ein erstes Script

Im vorherigen Kapitel haben Sie PHP installiert und konfiguriert. Jetzt ist es an der Zeit, PHP zu testen. In diesem Kapitel werden Sie Ihr erstes Script erstellen und sich bereits ein wenig in die Syntax von PHP einarbeiten. Am Ende des Kapitels sollten Sie imstande sein, Dokumente zu erstellen, die sowohl HTML als auch PHP enthalten.

Im Einzelnen lernen Sie in diesem Kapitel:

- wie ein PHP-Script erstellt, hochgeladen und ausgeführt wird,
- wie HTML und PHP im gleichen Dokument untergebracht werden,
- wie Sie Ihren Code durch Kommentare verständlicher machen können.

3.1 Das erste Script

Gehen wir gleich in medias res und erstellen ein erstes PHP-Script. Zuerst öffnen Sie den von Ihnen bevorzugten Texteditor. Wie HTML-Dokumente bestehen auch PHP-Dateien aus reinem Text. Das heißt, Sie können PHP-Dateien mit jedem beliebigen Texteditor erstellen, zum Beispiel mit Notepad oder HomeSite unter Windows, Simple Text und BBEdit unter Mac OS oder vi und Emacs unter Unix. Die meisten der üblichen HTML-Editoren bieten zumindest rudimentäre Unterstützung für PHP.

Unter *http://phpeditors.linuxbackup.co.uk* finden Sie eine von Keith Edmunds immer auf den neuesten Stand gebrachte Liste von PHP-freundlichen Editoren.

Tippen Sie das Beispiel aus Listing 3.1 ab und speichern Sie es unter dem Dateinamen *listing3.1.php*.

Listing 3.1: Ein erstes PHP-Script

```
1: <?php
2: phpinfo();
3: ?>
```

Der Code in Listing 3.1 gibt Informationen über unsere PHP-Installation auf dem Browser aus. Die Funktion `phpinfo()` ist beim Debuggen von Scripts wegen ihrer vielfältigen systembezogenen Informationen sehr hilfreich.

Die Extension des PHP-Dokuments ist wichtig. Sie weist den Server an, die Datei als PHP-Code zu behandeln und die PHP-Engine aufzurufen. PHP-Dokumente tragen standardmäßig die Extension *.php*. Diese Einstellung kann jedoch über die Serverkonfiguration geändert werden. Die genaue Vorgehensweise wurde in Kapitel 2 beschrieben. Systemverwalter konfigurieren die Server gelegentlich für andere Extensionen; manche Server erwarten daher beispielsweise Extensionen wie *.phtml* oder *.php5*. Wie Sie im letzten Kapitel gelernt haben, verwendet Apache die Direktive `AddType`, um festzulegen, wie eine Datei behandelt werden soll. Sie finden `AddType` normalerweise in der Apache-Konfigurationsdatei *httpd.conf*.

```
AddType application/x-httpd-php .php
```

Wenn Sie nicht direkt auf dem Serverrechner arbeiten, der Ihr PHP-Script ausführt, werden Sie wahrscheinlich auf einen FTP-Client zurückgreifen müssen, wie WS-FTP für Windows oder Browser Lite für MacOS, um Ihr gespeichertes Dokument auf den Server hochzuladen.

Aus historischen Gründen verwenden die verschiedenen Betriebssysteme unterschiedliche Zeichenkombinationen, um die Zeilenenden in einem Text darzustellen. Speichern Sie Ihre PHP-Dokumente mit den Zeilenumbruchzeichen, die auf dem Betriebssystem gelten, unter dem der Server ausgeführt wird. Dokumente mit anderen Zeilenumbruchszeichen werden unter Umständen von der PHP-Engine als eine einzige, sehr lange Textzeile gelesen. Normalerweise stellt dies kein Problem dar, trotzdem können Fehler nicht ausgeschlossen werden. Die meisten guten Texteditoren erlauben Ihnen, das verwendete Betriebssystem anzugeben.

Nachdem das Dokument im richtigen Verzeichnis steht, sollten Sie darauf über Ihren Browser zugreifen können. Wenn alles geklappt hat, sollte die Aus-

gabe des Scripts erscheinen. Abbildung 3.1 zeigt die Ausgabe des Scripts *listing3.1.php*.

Wenn PHP nicht auf dem Server installiert ist oder die Extension der Datei nicht erkannt wird, können Sie die in Abbildung 3.1 gezeigte Ausgabe nicht sehen. In diesen Fällen wird vielleicht der in Listing 3.1 erstellte Quellcode angezeigt oder Sie werden aufgefordert, die Datei herunterzuladen! Welche Auswirkung eine fehlerhafte Konfiguration hat, hängt von Ihrer Plattform, Ihrem Server und Ihrem Browser ab. Abbildung 3.1 zeigt die Reaktion des Internet Explorers, wenn ein Apache-Server, der PHP als Modul unter Linux ausführt, auf eine unbekannte Extension trifft.

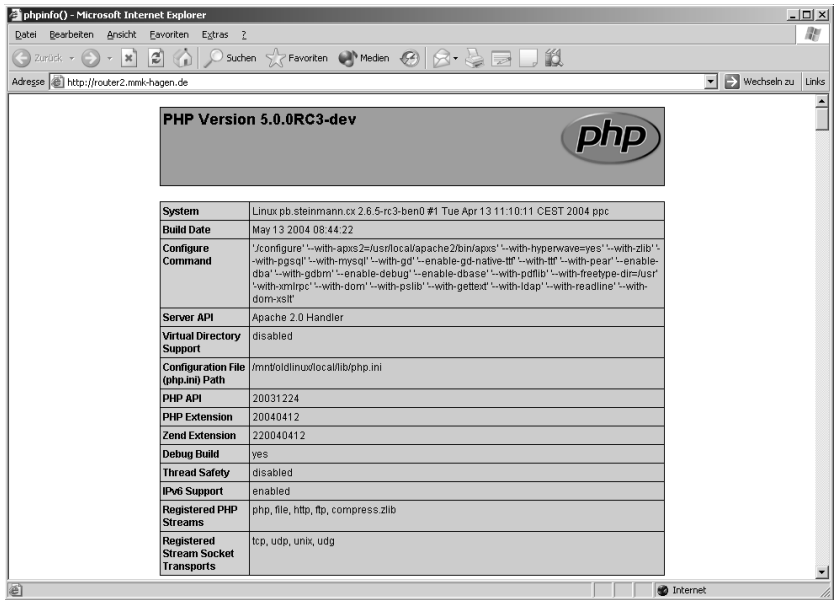

Abb. 3.1: Erfolgreich: Die Ausgabe von Listing 3.1.

Abb. 3.2: Fehlgeschlagen: Die Extension wird nicht erkannt.

Wenn Sie diese Ausgabe erhalten oder den Quellcode des Scripts im Browserfenster angezeigt bekommen, sollten Sie zuerst prüfen, mit welcher Extension Sie Ihr PHP-Script abgespeichert haben. Wenn Sie normalerweise mit HTML-Dateien arbeiten, stellen Sie sicher, dass Ihr Script nicht die Extension *.html* trägt. Ist die Datei-Extension korrekt, sollten Sie prüfen, ob PHP ordnungsgemäß installiert wurde und ob die Konfiguration Ihres Servers die Verwendung Ihrer Script-Extension vorsieht. In Kapitel 2 ist ausführlich beschrieben, wie Sie PHP installieren und konfigurieren. Um die Ausgabe in Abbildung 3.2 zu erzeugen, haben wir die Direktive `AddType` aus der Apache-Konfigurationsdatei entfernt.

Die Konfigurationsprobleme sind von Server zu Server und von Plattform zu Plattform unterschiedlich. Für den Apache treten beispielsweise Fehler auf, wenn PHP als CGI-Script ausgeführt wird und die `Action`-Direktive in der Apache-Datei *httpd.conf* fehlt.

Treten unerwartete Fehler auf, sollten Sie in den Installationsanweisungen für Ihren Server nachlesen. Vollständige Anweisungen zum Einrichten von PHP auf den meisten Servern finden Sie unter *http://www.php.net/installation*.

Nachdem Sie inzwischen Ihr Script hochgeladen und getestet haben, können wir den Code etwas genauer in Augenschein nehmen.

3.1.1 Anfang und Ende eines PHP-Anweisungsblocks

Wenn Sie PHP-Code schreiben, müssen Sie der PHP-Engine mitteilen, dass Ihre Befehle ausgeführt werden sollen. Andernfalls wird der von Ihnen aufgesetzte Code für HTML gehalten und im Browser ausgegeben. Sie verwenden hierzu besondere Tags, die Anfang und Ende des PHP-Codeblocks kennzeichnen. In Tabelle 3.1 finden Sie vier verschiedene Formen von PHP-Begrenzungstags.

Tabelle 3.1: PHP Start- und End-Tags

Tag-Stil	Start-Tag	End-Tag
Standard-Tags	`<?php`	`?>`
Tags in Kurzschreibweise	`<?`	`?>`
ASP-Tags	`<%`	`%>`
Script-Tags	`<script language="php">`	`</script>`

Von den in Tabelle 3.1 aufgeführten Tags lassen sich nur die Standard- und die Script-Tags für jede Konfiguration verwenden. Die Tags in Kurzschreib-

weise und die ASP-Tags müssen in Ihrer *php.ini*, die Sie in Kapitel 2 kennen gelernt haben, explizit aktiviert werden.

Die Erkennung der verkürzten Schreibweise aktivieren Sie, indem Sie die Einstellung `short_open_tag` in *php.ini* wie folgt auf `"On"` setzen:

```
short_open_tag = On;
```

Die Kurzform der PHP-Tags ist standardmäßig aktiviert, sodass Sie *php.ini* nur bearbeiten müssen, wenn Sie diese deaktivieren wollen.

Um die Erkennung von ASP-Tags zu aktivieren, müssen Sie die `asp_tags`-Einstellung setzen:

```
asp_tags = On;
```

Nachdem Sie Ihre *php.ini* bearbeitet haben, steht es Ihnen theoretisch frei, welchen der vier Stile Sie in Ihren Scripts verwenden wollen. Dennoch empfehlen wir Ihnen, bei den Standard-Tags `<?php ?>` zu bleiben. Es ist die offiziell unterstützte Syntax, die sich gut mit XML kombinieren lässt und in jedem PHP-Kontext funktioniert. Außerdem gibt es keine Garantie, dass die verkürzte Schreibweise der Tags auf Dauer unterstützt wird.

Die Zeichenfolge `<?` teilt XML-Parsern mit, dass sie eine Verarbeitungsanweisung zu erwarten haben.

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

Diese Syntax überschneidet sich mit der Kurzform der PHP-Tags. Eingebettet in XML- oder XHTML-Dokumente können diese Tags Fehlinterpretationen durch den XML-Validator, die PHP-Engine oder beide verursachen. Deaktivieren Sie daher die verkürzte Schreibweise, wenn Sie PHP in einem XML-Kontext verwenden wollen.

Die Kurzform der PHP-Tags oder die ASP-Tags sind auch dann zu vermeiden, wenn Ihr Script portierbar sein soll. Andere Server sind unter Umständen so konfiguriert, dass sie die Tags nicht unterstützen.

Betrachten Sie ein paar der Möglichkeiten, wie der Code aus Listing 3.1 korrekt aufgesetzt werden kann. Sie können eine der oben genannten vier verschiedenen Start- und End-Tags von PHP verwenden:

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

```
<%
phpinfo();
%>
<script language="php">
phpinfo();
</script>
```

Einzelne PHP-Codezeilen können zusammen mit den Start- und End-Tags in einer Zeile stehen:

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

Nachdem Sie sich jetzt davon überzeugt haben, dass PHP funktioniert, können Sie fortfahren und etwas mehr Code schreiben.

3.1.2 Die Funktion print()

In Listing 3.1 haben wir die Funktion `phpinfo()` verwendet. In Listing 3.2 schreiben wir unser Script um und geben eine Mitteilung an uns selbst im Browser aus.

Listing 3.2: Eine Nachricht ausgeben

```
1: <?php
2: print "Hallo Web!";
3: ?>
```

`print()` ist ein Sprachkonstrukt, das Daten ausgibt. Es ist zwar keine Funktion, aber es verhält sich so: Es übernimmt eine Reihe von Zeichen, die auch als String bezeichnet wird. Strings müssen entweder in einfachen oder doppelten Anführungszeichen stehen. Der an `print()` übergebene String, wird ausgegeben – üblicherweise im Browser oder auf der Befehlszeile.

Der `print()`-Anweisung verwandt ist die Anweisung `echo()`, die dasselbe Verhalten zeigt (aber keinen Rückgabewert liefert). In den meisten Beispielen in diesem Buch könnten Sie `print()` durch `echo()` ersetzen, ohne dass es einen Unterschied machen würde.

In Funktionsaufrufen muss grundsätzlich ein Klammerpaar auf den Funktionsnamen folgen – unabhängig davon, ob die Funktion Argumente übernimmt oder nicht. `print()` ist allerdings eher ein Sprachkonstrukt als eine Funktion und bedarf daher keiner Klammern. Da die Klammern traditionell weggelassen werden, werden wir in unseren Beispielen ebenfalls keine Klammern verwenden.

Wir haben unsere einzige Codezeile in Listing 3.2 mit einem Semikolon abgeschlossen. Das Semikolon informiert die PHP-Engine, dass die Anweisung zu Ende ist.

Eine *Anweisung* ist ein Befehl an die PHP-Engine. Allgemein könnte man auch sagen, dass eine Anweisung für PHP das ist, was ein Satz für die geschriebene oder gesprochene deutsche Sprache ist. Ein Satz endet in der Regel mit einem Punkt, eine Anweisung mit einem Semikolon. Ausnahmen hierzu bilden die Anweisungen, die andere Anweisungen enthalten, und Anweisungen, die einen Codeblock beenden. Meist jedoch bringt ein fehlendes Semikolon am Ende einer Anweisung die PHP-Engine aus dem Takt und führt dazu, dass ein Fehler für die folgende Zeile im Script gemeldet wird.

Da die Anweisung in Listing 3.3 die letzte im Codeblock ist, ist das Semikolon optional.

3.2 HTML und PHP kombinieren

Der Code in Listing 3.1 und Listing 3.2 ist reines PHP. Sie können diesen Code in ein HTML-Dokument einbetten, indem Sie einfach um die Start- und End-Tags von PHP herum HTML-Code hinzufügen. Ein Beispiel sehen Sie in Listing 3.3.

Listing 3.3: Ein PHP-Script mit HTML

```
 1: <!DOCTYPE html PUBLIC
 2:   "-//W3C//DTD XHTML 1.0 Strict//EN"
 3: "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 4: <html>
 5: <head>
 6: <title>Listing 3.2 Ein PHP-Script mit HTML</title>
 7: </head>
 8: <body>
 9: <div><b>
10: <?php
11: print "Hallo Welt";
12: ?>
13: </b></div>
14: </body>
15: </html>
```

Angesichts der vielen neuen Geräte, die heute auf das Web zugreifen und neue Browser auf noch mehr Plattformen ausführen, gewinnen die Standards immer mehr an Bedeutung. Soweit es möglich war, entspricht die Ausgabe der Codebeispiele in diesem Buch dem XHTML-Standard. XHTML (Extensible Hypertext Markup Language) ist eine XML-orientierte Version von HTML, die sich parsen und validieren lässt. Deshalb ist dieser Standard auch gut für schlanke Browser geeignet, die auf kleinen Speichergeräten laufen. Außerdem trägt XHTML zur Verbreitung einer wirklich browserübergreifenden Auszeichnung bei. Mehr zum Thema XHTML finden Sie unter *http://www.w3.org/TR/xhtml1/*.

Sicher ist Ihnen aufgefallen, dass Listing 3.3 mit einem sehr komplexen Element beginnt. Dieses Element wird als `DOCTYPE`-Deklaration bezeichnet und gibt an, welcher XHTML-Version das Dokument entspricht.

Wie Sie sehen können, ist die Einbettung von HTML in ein PHP-Dokument denkbar einfach. Sie müssen nur den HTML-Code hinzufügen. Die PHP-Engine ignoriert alles, was außerhalb der öffnenden und schließenden PHP-Tags steht. Wenn Sie das Listing 3.3 in einem Browser betrachten (siehe Abbildung 3.3), sehen Sie den String `Hallo Welt` in Fettdruck. Wenn Sie sich den Quelltext des Dokuments anzeigen lassen (siehe Abbildung 3.4), sieht das Listing wie ein normales HTML-Dokument aus.

Sie können beliebig viele PHP-Codeblöcke in ein Dokument einfügen und dazwischen nach Bedarf HTML verteilen. Die verschiedenen Codeblöcke in einem Dokument bilden zusammen ein einzelnes Script. Alle Variablen, die Sie im ersten Block definiert haben, sind normalerweise für alle nachfolgenden Blöcke verfügbar.

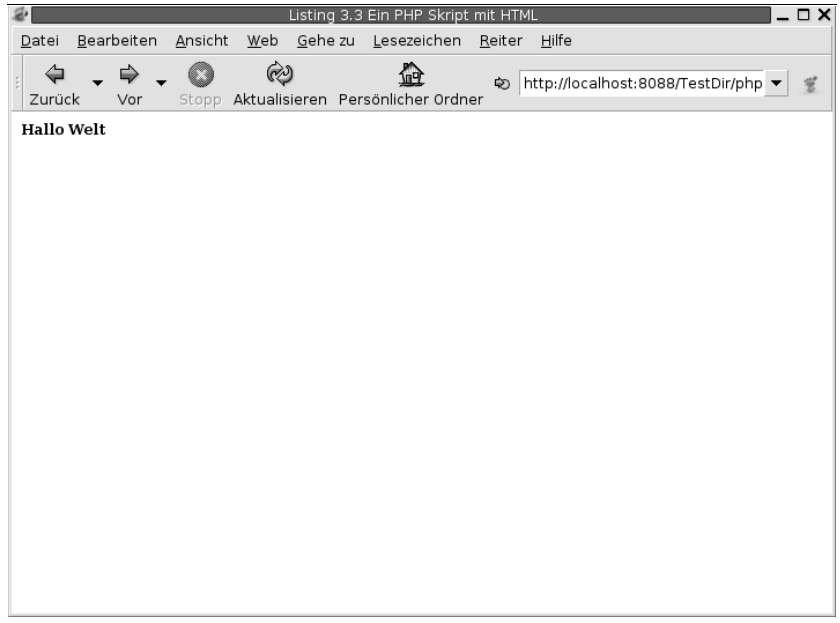

Abb. 3.3: Die Ausgabe von Listing 3.3, wie sie in einem Browser erscheint

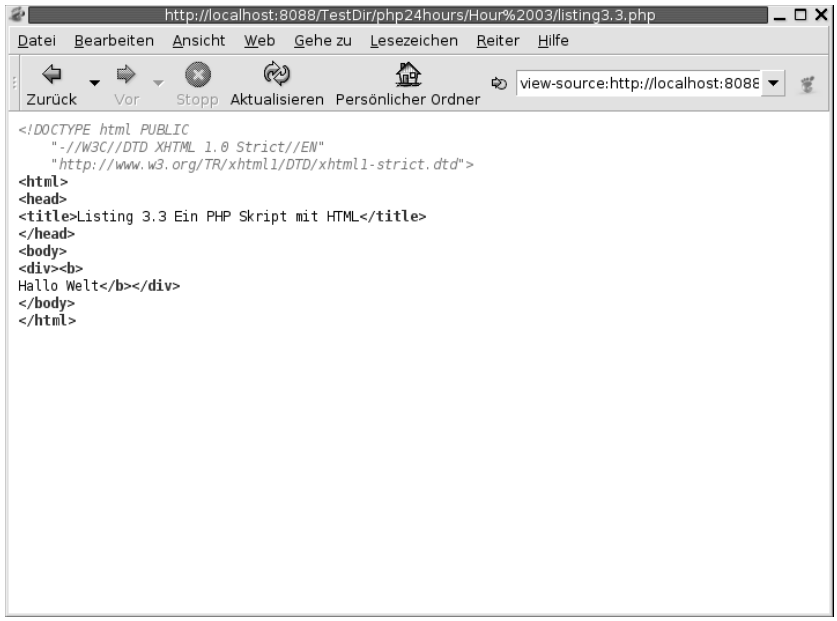

Abb. 3.4: Die Ausgabe von Listing 3.3 als HTML-Quelltext

3.3 Kommentare im PHP-Code

Code, der beim Schreiben noch ganz logisch erscheint, kann sich bei seiner Korrektur sechs Monate später als hoffnungsloses Durcheinander entpuppen. Mit Kommentaren, die Sie beim Aufsetzen des Codes einfügen, können Sie sich später viel Zeit sparen und anderen Programmierern, die sich mit Ihrem Code auseinander setzen müssen, die Arbeit erleichtern.

Ein *Kommentar* ist ein Text in einem Script, der von der PHP-Engine ignoriert wird. Kommentare werden eingesetzt, um den Code lesbarer zu machen oder ein Script mit Anmerkungen zu versehen.

Einzeilige Kommentare beginnen mit zwei Schrägstrichen (//) oder einem einfachen Hash-Zeichen (#). Der gesamte nachfolgende Text wird bis zum Ende der Zeile bzw. dem schließenden PHP-Tag ignoriert. Hier ein Beispiel:

```
// ein Kommentar
# noch ein Kommentar
```

Mehrzeilige Kommentare beginnen mit einem Schrägstrich gefolgt von einem Sternchen (/*) und enden mit einem Sternchen gefolgt von einem Schrägstrich (*/):

```
/*
dieser Kommentar
wird nicht von
der PHP-Engine
geparst
*/
```

PEAR (PHP Extension and Application Repository) ist eine ständig wachsende Sammlung von Bibliotheken und Scripts, die die Funktionalität von PHP erweitern. Zu diesen gehört auch ein Paket namens *phpDocumentor*, das Ihre Inline-Kommentare in eine Hyperlink-Dokumentation umwandelt und für die Pflege größerer Projekte von unschätzbarem Wert ist. Nähere Informationen zu PHPDocumentor finden Sie unter *http://phpdocu.sourceforge.net*.

3.4 Zusammenfassung

Damit sollten Sie jetzt über alle Hilfsmittel verfügen, um ein einfaches PHP-Script auf einem ordnungsgemäß konfigurierten Server auszuführen.

In diesem Kapitel haben Sie Ihr erstes PHP-Script erstellt. Sie haben gelernt, wie Sie mit einem Texteditor ein PHP-Dokument erstellen und benennen. Ihnen wurden vier verschiedene Tag-Arten vorgestellt, mit denen Sie einen PHP-Codeblock einleiten und beenden können, und Sie haben erfahren, wie Sie mit `print()` Daten an den Browser senden. Anschließend haben Sie HTML und PHP zusammen in einem Script untergebracht. Zu guter Letzt haben Sie gelernt, was Kommentare sind und wie Sie diese in ein PHP-Dokument einfügen.

Im nächsten Kapitel werden Sie diese Kenntnisse benötigen, um einige der wesentlichen Bausteine von PHP, einschließlich Variablen, Datentypen und Operatoren, zu testen.

3.5 Fragen und Antworten

F: Welche Start- und End-Tags sollte man am besten verwenden?

A: Das hängt vor allem davon ab, wo Sie Ihre Präferenzen setzen. Mit Hinblick auf die Portabilität sind die Standard-Tags (`<?php` und `?>`) zweifelsohne die sicherste Wahl. Die Kurzschreibweise, die standardmäßig aktiviert ist, hat den Vorzug, dass sie Tipparbeit spart. Der besseren Portabilität wegen sollten Sie jedoch auf ihren Einsatz verzichten.

F: Welche Editoren sollte ich möglichst nicht zum Erstellen von PHP-Code verwenden?

A: Vermeiden Sie Textverarbeitungssysteme, die den Text für den Druck formatieren (wie Word oder OpenOffice). Auch wenn Sie Dateien, die mit diesen Textverarbeitungssystemen erstellt wurden, im Nur-Text-Format abspeichern, können sich verborgene Zeichen in Ihren Code einschleichen.

F: Wann sollte ich meinen Code kommentieren?

A: Auch dies bleibt ganz Ihnen überlassen. Kurze Scripts sind meist auch nach einer längeren Pause immer noch selbsterklärend. Längere oder komplizierte Scripts sollten Sie kommentieren, um sich langfristig Zeit und Ärger zu sparen.

3.6 Übungen

1. Kann der Benutzer den Quellcode eines PHP-Scripts lesen, das Sie erfolgreich installiert haben?
2. Wie sehen die Standard-Begrenzungstags von PHP aus?
3. Wie sehen die ASP-Begrenzungstags von PHP aus?
4. Wie sehen die Script-Begrenzungstags von PHP aus?
5. Welche Syntax würden Sie verwenden, um eine Zeichenkette im Browser auszugeben?
6. Machen Sie sich mit dem Erstellen, Hochladen und Ausführen von PHP-Scripts vertraut. Sie sollten vor allem Ihr eigenes »Hallo Welt«-Script erstellen. Fügen Sie HTML-Code und weitere PHP-Codeblöcke hinzu. Spielen Sie ein wenig mit den verschiedenen Begrenzungstags in PHP herum. Welche sind in Ihrer Konfiguration aktiviert? Rufen Sie Ihre *php.ini*-Datei auf und bestätigen Sie Ihre Vermutungen. Vergessen Sie nicht, Ihren Code mit Kommentaren zu versehen.

3.6.1 Lösungen

1. Nein, der Benutzer sieht nur die Ausgabe Ihres Scripts.
2. Die Standard-Begrenzungstags von PHP:

```
<?php
// Ihr Code
?>
```

3. Die ASP-Begrenzungstags von PHP:

```
<%
// Ihr Code
%>
```

4. Die Script-Begrenzungstags von PHP:

```
<script language="php">
// Ihr Code
</script>
```

5. Sie würden normalerweise mit `print()` an den Browser schreiben, auch wenn Sie mit `echo()` das gleiche Ergebnis erzielen könnten.