4 PHP-Basics

In diesem Kapitel erhalten Sie wichtige PHP-Grundkenntnisse. Sie erfahren, wie Sie PHP in HTML-Dokumente einbinden und wie Sie mit Variablen Ihre Skripte flexibel halten. Außerdem geht es um unterschiedliche Datentypen und speziell um Arrays zum Speichern mehrerer Elemente. Zum Schluss sehen Sie, wie Sie mit PHP Dateien einbinden können – praktisch, um auf mehreren Seiten vorkommende Inhalte zentral zu speichern.

4.1 PHP in HTML-Dokument einbinden

PHP-Code können Sie direkt in HTML-Dokumente einbinden. Damit der PHP-Parser die PHP-Befehle als solche erkennt, müssen diese innerhalb von `<?php` und `?>` notiert werden. Im folgenden Beispiel wird mit `echo` ein Text ausgegeben:

```
01  <!DOCTYPE html>
02  <html>
03  <head>
04    <meta charset="UTF-8" />
05    <title>PHP in HTML einbinden</title>
06  </head>
07  <body>
08  <?php
09     echo "Ein erstes PHP-Dokument";
10  ?>
11  </body>
12  </html>
```

Listing 4–1 *PHP-Code in ein HTML-Dokument einbinden (php_einbinden.php)*

Damit das Beispiel funktioniert, ist zweierlei notwendig: Zum einen muss die Datei im richtigen Verzeichnis abgespeichert sein, und zum anderen muss die Endung *.php* lauten. Falls es hierbei Probleme gibt, schauen Sie noch einmal in Kapitel 2 nach.

.php ist die übliche und gängigste Endung für PHP-Dateien. Was als Endung bestimmt wird, lässt sich in der Konfiguration des Webservers festlegen: Sie könnten auch eine beliebige andere Zeichenkombination als Endung für PHP festlegen. Bei manchen Providern gibt es beispielsweise die Option, eine Endung wie *.php4* zu verwenden, wenn man möchte, dass die Skripte mit der veralteten Version 4 von PHP verarbeitet werden sollen, oder umgekehrt, dass nur Skripte mit der Endung *.php5* auch mit PHP verarbeitet werden.

Wollen Sie hingegen, dass *.html*-Dateien vom PHP-Parser verarbeitet werden sollen, so können Sie das ebenfalls bestimmen. Ergänzen Sie dafür in dem Ordner mit den *.html*-Dateien eine *.htaccess*-Datei mit der folgenden Zeile:

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

Weitere Informationen zu *.htaccess*-Dateien finden Sie in Anhang A.

Wenn Sie die Datei im Unterverzeichnis *php-beispiele* abgespeichert haben, rufen Sie sie über *http://localhost/php-beispiele/php_einbinden.php* in Ihrem Browser auf.

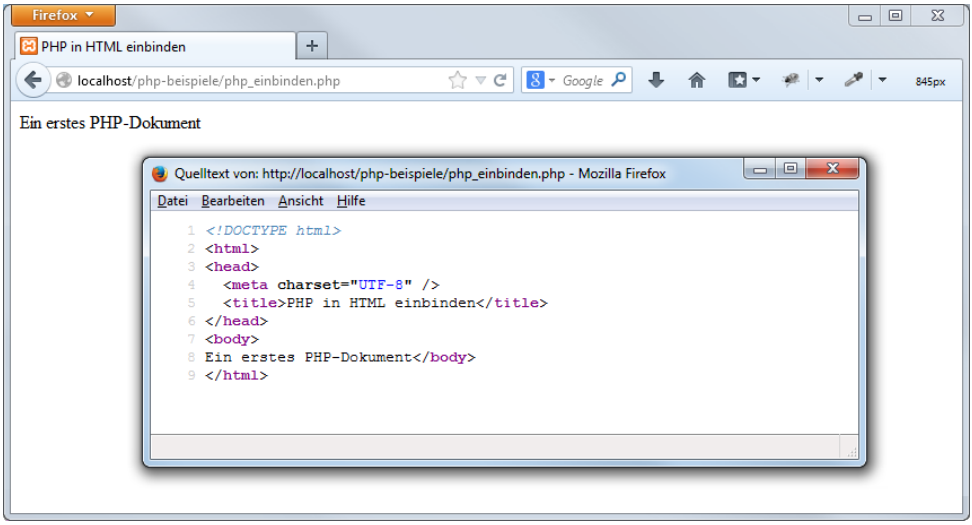


Abb. 4–1 *Dokument mit per PHP erzeugtem Text*

Wechseln Sie dann einmal in den Quellcode, im Firefox etwa über *Extras/Web-Entwickler/Seitenquelltext anzeigen*: Hier sehen Sie keinen PHP-Code, sondern nur HTML-Code. Wenn das so ist, hat alles geklappt.

Wir haben es hier mit zwei verschiedenen Quellcode-Dateien zu tun: Die Datei, die Sie in Ihrem Editor erstellt haben, enthält den PHP-Code, der mit HTML-Code gemischt sein soll. Das hingegen, was Sie als »Seitenquelltext« in Ihrem Browser sehen, ist das, was der PHP-Interpreter auf dem Server erzeugt hat – ein reiner HTML-Code ohne PHP-Befehle.

Jetzt genauer zum PHP-Code: Im Beispiel wird der PHP-Befehl `echo` eingesetzt, der zur Ausgabe dient. Handelt es sich um einen Text wie im Beispiel, müssen Sie diesen in Anführungszeichen schreiben: `echo "Unser erstes PHP-Dokument";`.

Für das, was hier allgemeinsprachlich mit *Text* bezeichnet wurde, gibt es die Fachbezeichnung *Zeichenkette* oder englisch *String*.

Außerdem sehen Sie am Ende einen Strichpunkt. Dieser dient in PHP dazu, Anweisungen abzuschließen.

4.1.1 Verschiedene Varianten der Einbindung

Im Beispiel wurde `<?php` und `?>` zum Einbinden des PHP-Codes benutzt. Das ist die gebräuchlichste und die empfohlene Variante, weil sie unabhängig von der Konfiguration immer funktioniert.

Es gibt daneben noch weitere Möglichkeiten:

- Mit dem `script`-Element:

```
<script language="php">
  echo "Eine andere Möglichkeit, PHP einzubinden";
</script>
```

- Eine Variante ohne das Wort `php` – das sogenannte Short-open-Tag:

```
<? echo "noch mal hallo"; ?>
```

Diese sehr kurze Option funktioniert nur, wenn die Konfigurationseinstellung `short_open_tag` auf `On` steht. Ob das bei Ihrer Installation der Fall ist, können Sie in der Ausgabe von `phpinfo()` nachsehen.

short_open_tag	On	On

Abb. 4–2 *Wenn die entsprechende Zeile in der Ausgabe von phpinfo() so aussieht, würden die Short-Open-Tags ebenfalls funktionieren – ansonsten müssen Sie die Konfiguration von PHP anpassen, wenn Sie die verkürzte Schreibweise nutzen wollen.*

Allerdings kommt es bei der Schreibweise <? zu Problemen, wenn Sie XHTML-Dokumente mit der XML-Deklaration am Anfang schreiben, wie `<?xml version="1.0" ?>`. Denn der Beginn der XML-Deklaration, also <?, würde als Anfang des PHP-Codes interpretiert werden. Als Abhilfe können Sie die XML-Deklaration über `echo` ausgeben lassen.

- Eine weitere Möglichkeit der Einbindung sind die sogenannten ASP-Tags:

```
<% echo "auch das ist möglich"; %>
```

Ob ASP-Tags möglich sind, ist ebenfalls von der Konfiguration abhängig. Die entsprechende Einstellung heißt `asp_tags`.

Fazit zur Einbindung: Wenn Sie sichergehen möchten, dass Ihre PHP-Skripten konfigurationsunabhängig laufen, sollten Sie nur die klassische Form `<?php` und `?>` benutzen.

4.1.2 PHP-Befehle überall

Die PHP-Befehle können Sie an beliebigen Stellen in Ihrem HTML einfügen – immer da, wo Sie sie brauchen. Also dort, wo Sie beispielsweise – wie später gezeigt – einen Wert aus der Datenbank ausgeben oder das Ergebnis einer Berechnung anzeigen lassen wollen:

Im folgenden Beispiel wird PHP-Code an mehreren Stellen eingefügt:

```
01  <?php date_default_timezone_set("Europe/Berlin");?>
02  <!DOCTYPE html>
03  <html>
04   <head>
05    <meta charset="UTF-8" />
06    <title><?php echo date("j.n.Y"); ?></title>
07  <style>
08    body { background-color: <?php echo "yellow"; ?>; }
09  </style>
10  </head>
11  <body>
12  <?php
13    echo "Schönen Tag auch!";
14  ?>
15  </body>
16  </html>
```

Listing 4–2 *PHP-Code kann an sich überall stehen (php_code_ueberall.php).*

In Zeile 1 – sogar vor der HTML-Dokumenttypangabe – steht ein erster Aufruf von PHP. Im Beispiel wird damit die Zeitzone gesetzt. In Zeile 6 folgt der nächste Aufruf von PHP: Hier wird im Seitentitel das Datum ausgegeben.

> Mehr zur Funktion, um die Zeitzone zu setzen, sowie zu `date()` zur Datumsausgabe in Kapitel 6.

In Zeile 8 wird noch einmal PHP aufgerufen: dieses Mal innerhalb der CSS-Angaben, und zwar bei der Zuweisung einer Hintergrundfarbe für das `body`-Element. Der letzte Aufruf von PHP erfolgt dann in Zeile 13, wo eine Begrüßung ausgegeben wird. Das alles ist problemlos möglich. Wichtig ist nur, dass das Ergebnis wieder korrektes HTML ist.

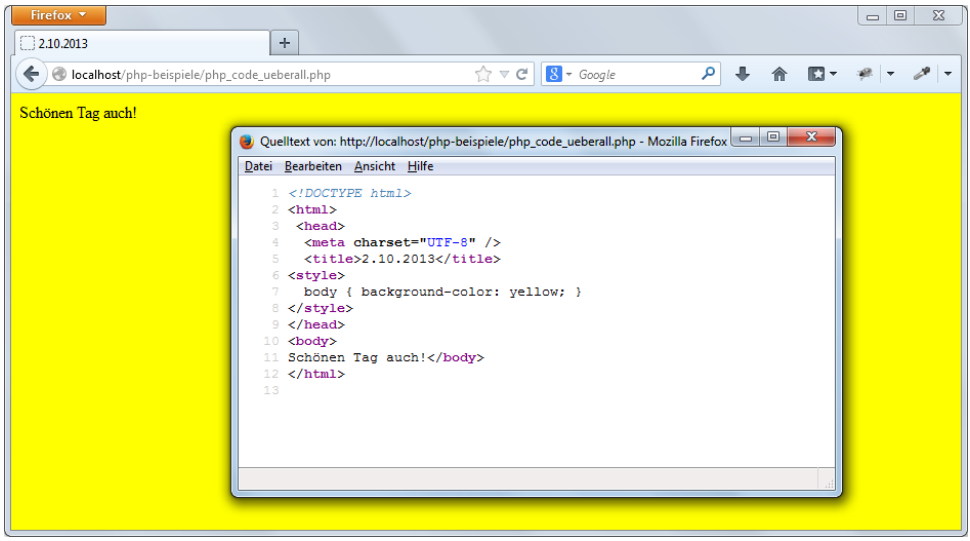


Abb. 4–3 *Das Ergebnis ist wieder korrektes HTML.*

Umgekehrt können Sie natürlich auch HTML-Code direkt innerhalb des Textes schreiben, der per `echo` ausgegeben wird:

```
<?php
  echo "<p>Schönen Tag auch!<br />Bis später</p>";
?>
```

In diesem Fall könnten Sie `<p>` und `</p>` auch außerhalb des PHP-Codes notieren – das macht keinen Unterschied:

```
<p>
<?php
  echo "Schönen Tag auch!<br />Bis später";
?>
</p>
```

Am Quellcode des HTML-Dokuments, das ausgeliefert wird, können Sie prinzipiell nicht feststellen, welche Teile über PHP-Befehle erzeugt werden und welche direkt im HTML-Code standen.

Im Beispiel wurde immer `echo` eingesetzt. Stattdessen können Sie übrigens auch `print` benutzen:

```
<?php
  print "Schönen Tag auch!<br />Bis später";
?>
```

Ob Sie `echo` oder `print` wählen, ist im Wesentlichen Geschmackssache. Es gibt allerdings kleinere Unterschiede, die im Normalfall nicht relevant sind: So gibt `print` einen Rückgabewert zurück, `echo` hingegen nicht. Und außerdem können Sie `echo` auch mehrere durch Komma getrennte Parameter übergeben, also beispielsweise `echo "eins", "zwei"`. Zu den Begriffen *Rückgabewert* und *Parameter* kommen wir später noch ausführlich.

Leerzeichen und neue Zeilen sind für PHP nicht relevant. Sie sind jedoch ganz essenziell für die Lesbarkeit des Skripts. Wie man diese geschickt einsetzt, erfahren Sie etwas später – in Kapitel 5 –, wenn Sie weitere PHP-Sprachelemente kennengelernt haben.

Innerhalb von Anführungszeichen sind die Leerzeichen hingegen schon relevant, sie werden eins zu eins so in den ausgegebenen Quellcode übernommen. Hier sind die meisten aber nicht sichtbar, da Browser Leerzeilen im HTML-Code ignorieren und mehrere Leerzeichen zu einem zusammenfassen.

4.2 Kommentare

Mit Kommentaren können Sie Erklärungen zu Ihrem Skript in den Quellcode schreiben, die vom PHP-Interpreter ignoriert werden. Vielleicht ist Ihnen heute bei einem Skript noch klar, warum Sie was an welche Stelle geschrieben haben, aber sehen Sie sich mal ein von Ihnen selbst geschriebenes Skript nach ein paar Monaten noch einmal an: Sie werden sich an wenig erinnern und froh sein, wenn Sie Hinweise finden, was die einzelnen Schritte bedeuten und warum sie durchgeführt wurden. Außerdem sind Kommentare ganz essenziell, wenn mehrere Personen an einem Skript arbeiten.

Kommentare können einzeilig sein:

```
//dies ist ein Kommentar
#dies ist auch ein Kommentar
```

Einzeilige Kommentare können auch als Anschluss an einen PHP-Befehl stehen:

```
echo "Hallo"; //gibt Hallo aus
```

Mehrzeilige Kommentare stehen zwischen /* und */:

```
/* dies ist ein
mehrzeiliger
Kommentar */
```

Kommentare können auch verwendet werden, um gerade nicht benötigte Codezeilen auszukommentieren. Im nächsten Beispiel wird die zweite Ausgabe auskommentiert:

```
<?php
  echo "<p>Schönen Tag auch!<br />Bis später</p>";
  /* echo "Der derzeitige Gesamtbetrag ist 42,50<br />"; */
  echo "Weitere interessante Produkte finden Sie unter ... ";
?>
```

Das kann man bei der Fehlersuche einsetzen, um festzustellen, ob die Fehlermeldung durch eine bestimmte Zeile bzw. einen bestimmten Codebereich hervorgerufen wurde.

Mehrzeilige Kommentare dürfen nicht verschachtelt werden. Das Folgende würde **nicht** funktionieren:

```
/* Das ist ein Kommentar
/* und hier fängt ein neuer Kommentar an */
Und erst hier wird der Kommentar beendet */
```

Das Ende des zweiten, im ersten verschachtelten Kommentars würde auch den ersten Kommentar beenden.

Prinzipiell verwendet man /* und */ für längere Kommentare zu Beginn eines Skripts oder eines Skriptbereichs, für die kleinen Schritte dazwischen hingegen //. In den Skripten in diesem Buch sehen Sie hingegen wesentlich häufiger die /* */-Kommentare. Das liegt daran, dass die Zeilen hier kürzer sind als sonst.

4.3 Variablen definieren und ausgeben

Sie haben bisher gesehen, wie Sie über PHP Texte ausgeben lassen können. Viele zusätzliche Möglichkeiten ergeben sich durch ein ganz wichtiges weiteres PHP-Sprachelement: die **Variablen**. Variablen sind Platzhalter für unterschiedliche Daten – z. B. Text oder Zahlen – und nichts anderes als ein symbolischer Bezeichner für einen Speicherbereich, in dem ein Wert abgelegt wird. Variablen sind beispielsweise notwendig, um Eingaben der Benutzer weiterzuverarbeiten: Sie wissen ja noch nicht, was die Benutzer eingeben, möchten aber trotzdem darauf zugreifen, um die Inhalte beispielsweise auszugeben.

Variablennamen beginnen in PHP immer mit einem Dollarzeichen: `$meineVariable`. Die Namen von Variablen vergeben Sie selbst. Dabei müssen Sie folgende Regeln beachten:

- Groß- und Kleinschreibung wird unterschieden. So sind `$meineVariable` und `$MeineVariable` unterschiedliche Variablen.
- Nach dem Dollarzeichen darf nicht direkt eine Zahl folgen: `$7kaese` wäre also kein korrekter Variablenname.
- Leerzeichen, Punkte, Ausrufezeichen oder Bindestriche sind in Variablennamen nicht erlaubt. Statt des Leerzeichens nehmen Sie am besten einen Unterstrich, z. B. `$brutto_preis`.

Um einer Variable einen Wert zuzuweisen, verwenden Sie den Zuweisungsoperator =:

```
$name = "Lola";
$alter = 2;
```

Das Gleichheitszeichen kennen Sie sicher auch aus der Mathematik. In der Mathematik bedeutet es »ist gleich«, hier in PHP hingegen »erhält den Wert«.

Ihren Variablen können Sie natürlich nicht nur einen festen Wert, sondern auch das Ergebnis einer Berechnung zuweisen:

```
$erg = 17 + 4;
```

4.3.1 Notice bei nicht initialisierten Variablen

Wenn Sie eine Variable einsetzen, der Sie keinen Wert zugewiesen haben, erhalten Sie eine entsprechende Notice – allerdings nur, wenn die Einstellung `error_reporting` wie in Kapitel 2 beschrieben angegeben ist. Ein Beispiel:

```
$zahl = 5;
$erg = $Zahl + 10;
```

Listing 4–3 *Nicht initialisierte Variable (nichtinitialisiert.php)*

Hier wird `$zahl` der Wert `5` zugewiesen, dann aber in der Berechnung `$Zahl` (mit Großbuchstaben) verwendet. Da die Groß-/Kleinschreibung von Variablen relevant ist, sind `$zahl` und `$Zahl` für PHP verschiedene Variablen, und `$Zahl` ist nicht initialisiert, das heißt, Sie haben ihr keinen expliziten Wert zugewiesen.

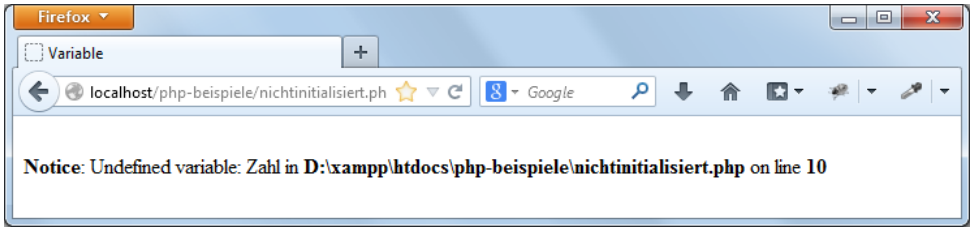


Abb. 4–4 *Fehlermeldung bei nicht initialisierter Variable*

Sie erhalten dann die in Abbildung 4–4 gezeigte Meldung – das Skript würde ansonsten aber trotzdem funktionieren, und der Variablen `$Zahl` würde der Defaultwert 0 zugewiesen. Die Fehlermeldung ist aber hier sehr hilfreich, da sie Ihnen einen Hinweis auf Ihren Tippfehler gibt.

4.3.2 Den Inhalt von Variablen ausgeben

Den Inhalt von Variablen können Sie per `echo` ausgeben:

```
echo $name;
```

Häufig möchte man Textinhalt mit dem Inhalt von Variablen kombinieren, also nicht nur »Lola« ausgeben lassen, sondern einen ganzen Satz. Das geht denkbar einfach: Sie können direkt Text und Variablen bei der Ausgabe kombinieren:

```
echo "$name ist $alter Jahre alt.";
```

Das gibt aus: »Lola ist 2 Jahre alt.«

Dieser Vorgang, dass innerhalb einer Zeichenkette Variablennamen erkannt und durch ihren Wert ersetzt werden, heißt *Variableninterpolation* und wird nur durchgeführt, wenn Sie den Text in doppelten Anführungszeichen schreiben. Verwenden Sie stattdessen einfache Anführungszeichen, sehen Sie *$name* anstelle von *Lola* in der Ausgabe und *$alter* anstelle von *2*:

```
echo '$name ist $alter Jahre alt.';
```

Häufig müssen Sie nur schnell in den PHP-Modus wechseln, um einen Wert ausgeben zu lassen:

```
<?php echo $wert; ?>
```

Genau für diesen Fall gibt es eine verkürzte Schreibweise. Sie schreiben direkt nach `<?` ein =-Zeichen und dann das, was Sie ausgeben lassen möchten:

```
<?=$wert?>
```

Kurz und praktisch – allerdings funktioniert diese Schreibweise bis einschließlich PHP 5.3.x nur, wenn die Konfigurationseinstellung `short_open_tag` auf `on` steht. Ab PHP 5.4 ist es unabhängig von dieser Einstellung und geht immer. Das heißt, wenn Sie diese schnelle Variante der Ausgabe einsetzen, sind Sie von der richtigen Konfiguration abhängig oder davon, dass PHP 5.4 oder größer benutzt wird.

Übung 1

Erstellen Sie ein Skript, in dem Sie mehrere Variablen für Ihren Vornamen, Ihren Nachnamen und Ihren Wohnort definieren. Lassen Sie dann »X Y wohnt in Z« ausgeben.

4.3.3 Sonderzeichen in Anführungszeichen

Möchten Sie z. B. innerhalb von doppelten Anführungszeichen wirklich ein Dollarzeichen ausgeben lassen, müssen Sie es *maskieren*: So stellen Sie sicher, dass PHP das Dollarzeichen als normales Dollarzeichen und nicht als Einleitung für eine Variable nimmt:

```
echo "Das Buch kostet 14 \$";
```

Genauso müssen Sie auch einen Backslash vor ein doppeltes Anführungszeichen schreiben, wenn Sie es innerhalb von doppelten Anführungszeichen einsetzen wollen. Das werden Sie häufig bei Attributwerten in HTML brauchen, die selbst in Anführungszeichen geschrieben werden:

```
echo "<img src=\"wiesen.jpg\" width=\"137\" height=\"103\" alt=\"Landschaft\" />";
```

ergibt dann als HTML-Code:

```
<img src="wiesen.jpg" width="137" height="103" alt="Landschaft" />
```

Wenn man sich die Datei im Browser ansieht, wird – sofern das Bild im Ordner vorhanden ist – die Landschaft angezeigt.

Anstatt die doppelten Anführungszeichen über \" zu maskieren, können Sie auch einfache Anführungszeichen für die Attributwerte in HTML verwenden:

```
echo "<img src='wiesen.jpg' width='137' height='103' alt='Landschaft' />";
```

Dies ließe sich auch umgekehrt schreiben, indem Sie außen die einfachen und innerhalb dieser die doppelten Anführungszeichen einsetzen.

```
echo '<img src="wiesen.jpg" width="137" height="103" alt="Landschaft" />';
```

Listing 4–4 fasst diese unterschiedlichen Verwendungen noch einmal zusammen.

Bei diesem Beispiel wurde das umfassende HTML-Grundgerüst nicht mehr mit abgedruckt. Das wird im Folgenden immer so gehandhabt, wenn der PHP-Teil ganz normal innerhalb von <body> und </body> steht.

```
01    $name = "Lola";
02    $alter = 2;
03    $erg = 17 + 4;
04    echo "<h3>Mit doppelten Anführungszeichen: </h3>";
05    echo "$name ist $alter Jahre alt.";
06    echo "<h3>Mit einfachen Anführungszeichen: </h3>";
07    echo '$name ist $alter Jahre alt.<br />';
08    echo "<h3>Und noch ein paar Bilder: </h3>";
09    echo "<img src=\"wiesen.jpg\" width=\"137\" height=\"103\"
                                                    alt=\"Landschaft\" />";
10    echo "<img src='wiesen.jpg' width='137' height='103' alt='Landschaft' />";
11    echo '<img src="wiesen.jpg" width="137" height="103" alt="Landschaft" />';
```

Listing 4–4 *Variablen ausgeben mit einfachen und doppelten Anführungszeichen (variablen_ausgeben.php)*

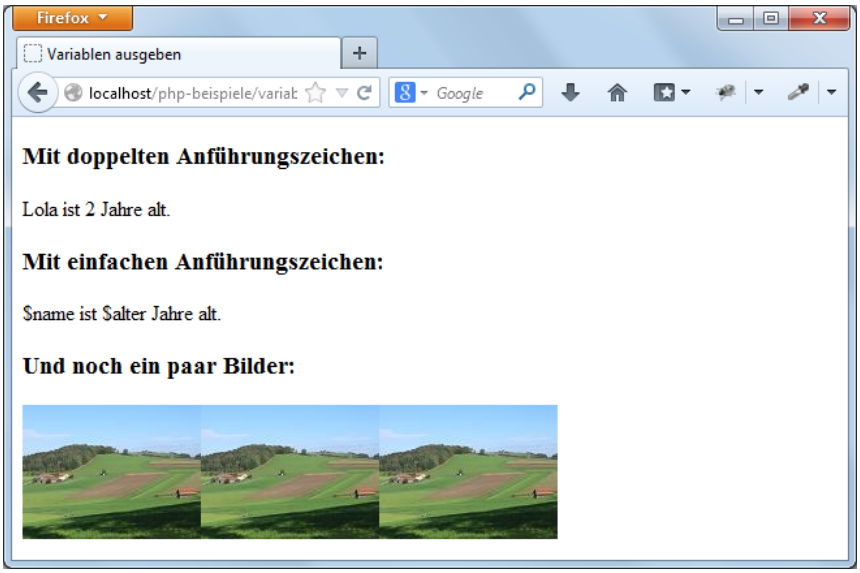


Abb. 4–5 *Unterschiedliche Verwendung von einfachen und doppelten Anführungszeichen*

Sie haben gesehen, wie Sie den Backslash innerhalb von doppelten Anführungszeichen einsetzen können, um Sonderzeichen wie das $-Zeichen oder doppelte Anführungszeichen selbst auszugeben. Daneben gibt es weitere Kombinationen von Backslash und Zeichen, die innerhalb von doppelten Anführungszeichen eine besondere Bedeutung haben.

\n und \t für einen übersichtlichen HTML-Quellcode

Ihren HTML-Quellcode strukturieren Sie in der Regel durch Zeilenumbrüche und Einrückungen. Um dies auch für den HTML-Code zu machen, den der PHP-Interpreter aus den PHP-Befehlen erzeugt, verwenden Sie `\n` und `\t`. `\n` erzeugt einen Zeilenumbruch, `\t` einen Tabulator:

```
echo "Unser erstes \nphp-Dokument. \n";
echo "\tUnser erstes \tphp-Dokument. \n";
```

Listing 4–5 *Tabulator und Newline im Einsatz (escapesequenzen.php)*

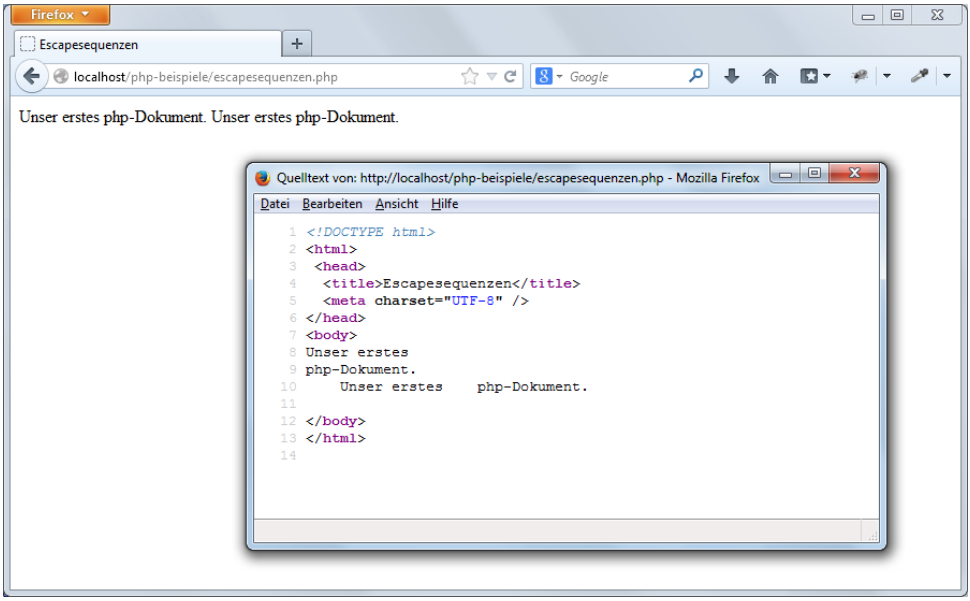


Abb. 4–6 *Sichtbar sind \t und \n nur im HTML-Quellcode, nicht in der Ausgabe des Browsers.*

In Abbildung 4–6 sehen Sie deutlich, dass `\t` und `\n` aus dem PHP-Code keine Auswirkung im Browser haben, sondern nur im HTML-Quellcode. Sinnvoll ist ihr Einsatz beispielsweise, wenn man mit PHP eine Tabelle ausgeben lässt. Hier kann man den Quellcode für eine bessere Lesbarkeit per `\t` und `\n` einrücken – das ist hilfreich, um mögliche Verschachtelungsfehler beim Einsatz von `<tr>` oder `<td>` zu finden.

> Nützlich ist \n ebenfalls für die Erzeugung von Zeilenumbrüchen in Textdateien (siehe Kap. 12) oder bei Textmails (Kap. 7).

Wenn Sie in Ihrem Text selbst ein \ benutzen, so müssen Sie dieses ebenfalls durch einen weiteren Backslash maskieren:

```
$windowspfad = "C:\\xampp";
```

Alle möglichen Escapesequenzen

Die Kombination von Backslash plus Zeichen wird Escapesequenz genannt. Alle möglichen Escapesequenzen führt Tabelle 4–1 vor: Innerhalb von einfachen Anführungszeichen gibt es nur zwei Escapesequenzen: \' für ein einfaches Anführungszeichen innerhalb von Anführungszeichen und \\ für den Backslash innerhalb von einfachen Anführungszeichen selbst.

Kombination	Bedeutung
"\\"	\
"\n"	Neue Zeile
"\t"	Tabulator
"\$"	Dollarzeichen
"\""	"
"\r"	Wagenrücklauf
"\v"	Vertikaler Tabulator
"\f"	Seitenvorschub
"\100"	Das Zeichen, das der angegebenen Oktalzahl in der Codetabelle des Zeichensatzes entspricht – hier @
"\X40"	Das Zeichen, das der angegebenen Hexadezimalzahl in der Codetabelle des Zeichensatzes entspricht – hier @
'\\'	\
'\''	'

Tab. 4–1 *Escapesequenzen in einfachen und doppelten Anführungszeichen*

4.3.4 Variablennamen über {} kennzeichnen

Noch eine Besonderheit gibt es bei der Variableninterpolation: Sie haben ja gesehen, dass Sie den Wert von Variablen direkt in doppelten Anführungszeichen ausgeben lassen können. Was aber, wenn man direkt an den Wert etwas dranhängen möchte, beispielsweise ein Genitiv-s?

```
$vorname= "Amina";
```

Nehmen wir an, Sie möchten »Aminas Jacke« ausgeben lassen. Wenn Sie das so versuchen:

```
echo "$vornames Jacke";
```

versteht der PHP-Interpreter `$vornames` als Variablenname. Da Sie keine Variable mit diesem Namen definiert haben, wird nichts ausgegeben. Wenn – wie in Kapitel 2 beschrieben – die Anzeige der Fehlermeldungen so eingestellt ist, dass auch Hinweise (Notices) angezeigt werden, erhalten Sie eine Meldung, dass Sie eine nicht definierte Variable verwenden.

Aber es besteht natürlich eine Möglichkeit, etwas direkt an die Variable anzuhängen. Sie müssen PHP dabei nur mitteilen, wie weit der Variablenname geht und wo der zusätzliche Text ist. Dazu brauchen Sie geschweifte Klammern:

```
echo "{$vorname}s Jacke";
```

4.3.5 Komfortable Ausgabe über HereDoc und NowDoc

HereDoc und NowDoc sind weitere Möglichkeiten zur Ausgabe von Text. Wenn Sie mehr HTML-Tags und Variablen mischen wollen, ist das manchmal mühsam: Sie müssen immer darauf achten, die Anführungszeichen zu maskieren oder die jeweils anderen zu verwenden etc. Eine Vereinfachung kann die HereDoc-Syntax bringen.

Um etwas über HereDoc ausgeben zu lassen, schreiben Sie hinter `echo` drei spitze Klammern `<<<` und einen Bezeichner; im Beispiel ist es `DOC`. Danach geben Sie Ihren HTML-Code ganz »normal« an – Sie können beispielsweise Anführungszeichen unmaskiert verwenden. Sie beenden die HereDoc-Syntax mit dem Bezeichner, mit dem Sie das Ganze begonnen haben, und einem Strichpunkt.

```
echo <<<DOC

DOC;
```

Wichtig ist, dass der abschließende Bezeichner bei der HereDoc-Syntax ganz am Anfang der Zeile steht. Es darf kein Leerzeichen und auch kein anderes Zeichen davor stehen.

Das folgende Listing zeigt die HereDoc-Syntax zur Ausgabe einer Tabelle:

```
01  $vorname = "Amina";
02  $alter   = 3;
03  echo <<<DOC
04  <table border="1" cellpadding="5" cellspacing="0">
05    <tr>
06      <td>Name</td>
07      <td>Alter</td>
08    </tr>
09    <tr>
10      <td>$vorname</td>
11      <td>$alter</td>
12    </tr>
13  </table>
14  DOC;
```

Listing 4–6 *Ausgabe über die HereDoc-Syntax (heredoc.php)*

Sie müssen den Text nicht direkt ausgeben lassen, sondern können ihn auch in einer Variable speichern und später bei Bedarf ausgeben.

```
01  $vorname = "Amina";
02  $alter = 3;
03  $ausgabe = <<<DOC
04  <table border="1" cellpadding="5" cellspacing="0">
05    <tr>
06      <td>Name</td>
07      <td>Alter</td>
08     </tr>
```

```
09    <tr>
10      <td>$vorname</td>
11      <td>$alter</td>
12    </tr>
13  </table>
14  DOC;
15  echo $ausgabe;
```

Listing 4–7 *Dieses Mal wird der Text erst einmal in einer Variable gespeichert (heredoc_2.php).*

Der Text innerhalb der HereDoc-Syntax wird vom PHP-Interpreter so behandelt, als stünde er in doppelten Anführungszeichen – und Variablen werden interpoliert. Im Beispiel erscheint nach der Verarbeitung anstelle von `$vorname` der zugewiesene Wert Amina. Genau darin unterscheidet sich eine andere mögliche Konstruktion namens NowDoc von HereDoc. Bei NowDoc wird der Inhalt so behandelt, als stünde er in einfachen Anführungszeichen, und der Wert der Variablen wird nicht ausgegeben. NowDoc steht seit PHP 5.3 zur Verfügung.

NowDoc definieren Sie wie HereDoc – mit dem Unterschied, dass Sie den Bezeichner in einfachen Anführungszeichen schreiben (siehe Zeile 3).

```
01  $vorname = "Amina";
02  $alter = 3;
03  echo <<<'DOC'
04  <table border="1" cellpadding="5" cellspacing="0">
05    <tr>
06      <td>Name</td>
07      <td>Alter</td>
08     </tr>
09    <tr>
10      <td>$vorname</td>
11      <td>$alter</td>
12    </tr>
13  </table>
14  DOC;
```

Listing 4–8 *NowDoc (nowdoc.php)*

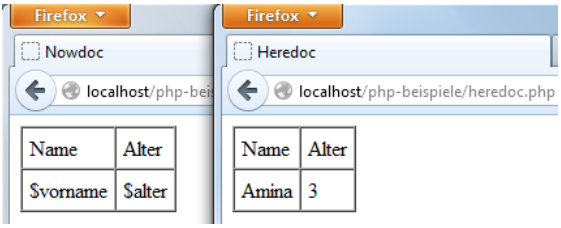


Abb. 4–7 *Links NowDoc ohne Variableninterpolation, rechts HereDoc-Syntax mit*

4.3.6 Qual der Wahl: einfache oder doppelte Anführungszeichen?

In den vorigen Abschnitten ging es um die Definition von Variablen und um die Ausgabe von Texten und Variablen. Dabei macht es ja einen Unterschied, ob Sie die doppelten oder die einfachen Anführungszeichen wählen. Was soll man jetzt im konkreten Fall jeweils nehmen – einfache oder doppelte Anführungszeichen? Der Unterschied ist ja bekanntlich, dass bei doppelten Anführungszeichen der Wert von Variablen ausgegeben wird, bei einfachen nicht.

Wenn man diesen Unterschied ernst nimmt und konsequent berücksichtigt, sollte man natürlich nur dann doppelte Anführungszeichen verwenden, wenn sie benötigt werden.

- Typischer Fall für doppelte Anführungszeichen: `echo "Hallo $name";`
- Eher ein Fall für einfache Anführungszeichen: `echo "Guten Morgen";`

Andererseits ist es – nach meiner Erfahrung aus Kursen – für PHP-Einsteiger relativ umständlich und mitunter verwirrend, wenn sie bei allen Ausgaben immer zuerst überlegen müssen, welche Anführungszeichen denn nun angebracht sind.

Deswegen werden hier im Buch konsequent doppelte Anführungszeichen eingesetzt, und wenn innerhalb dieser weitere benötigt werden – z. B. bei Attributwerten bei HTML-Tags –, einfache benutzt. Diese Regel lässt sich durchgehend anwenden und funktioniert immer.

4.3.7 Voll flexibel: variable Variablen

In PHP können Sie Variablennamen selbst in Variablen speichern und darüber auf die Variablen zugreifen. Dafür benutzen Sie zwei Dollarzeichen:

```
$varname = "beispiel";
$$varname = "php";
echo $beispiel;
```

Listing 4–9 *Variable Variable (variable_variablen.php)*

Im Beispiel wird eine Variable namens `$varname` definiert, mit dem String `"beispiel"` als Inhalt. Dann erhält `$$varname` den Inhalt `php`. Die Ausgabe von `echo $beispiel` ist `"php"`.

4.4 Konstanten definieren

Der Inhalt von Variablen ist, wie der Name sagt, variabel, er kann sich im Laufe des Skripts ändern. Wenn Sie hingegen mit feststehenden Werten in Ihrem Skript arbeiten, sollten Sie Konstanten einsetzen. Konstanten definieren Sie nicht über Zuweisung wie Variablen, sondern über die Funktion `define()`. In runden Klammern geben Sie zuerst den Namen der Konstanten an und nach einem Komma den Wert.

`define()` ist eine von PHP vorgegebene Funktion. PHP stellt Ihnen viele solcher vordefinierten Funktionen zur Verfügung, die Sie direkt einsetzen können. Hinter dem Funktionsnamen stehen runde Klammern, in denen Sie PHP die Parameter für die Funktion übergeben. Mit Parametern bestimmen Sie, mit was die Funktion operieren soll. Mehrere Parameter werden dabei durch Komma voneinander getrennt. Wie viele Parameter Sie angeben können und wie viele Sie angeben müssen, ist von Funktion zu Funktion unterschiedlich. In diesem und dem nächsten Kapitel werden Sie immer wieder weitere Funktionen kennenlernen. Vordefinierte Funktionen in PHP sind dann auch das alleinige Thema von Kapitel 6.

Durch folgende Zeile wird eine Konstante namens `MAXWERT` definiert und auf den Wert 10 gesetzt:

```
define("MAXWERT", 10);
```

Um im Skript auf die Konstante zuzugreifen, schreiben Sie sie direkt ohne Dollarzeichen. Das ist auch der formale Unterschied zu den Variablen.

```
echo MAXWERT; /* gibt 10 aus */
```

Wenn Sie versuchen, einer Konstanten einen neuen Wert zuzuweisen, erhalten Sie eine Fehlermeldung.

Seit PHP 5.6 können Sie übrigens bei der Zuweisung einer Konstanten direkt einen Operator einsetzen:

```
const ONE = 1;
const TWO = ONE * 2;
```

In Versionen vor PHP 5.6 erhalten Sie hingegen bei diesem Code eine Fehlermeldung.

Normalerweise spielt die Groß- und Kleinschreibung von Konstanten eine Rolle. Wenn diese hingegen nicht relevant sein soll, übergeben Sie einen dritten Parameter `true`:

```
define("MAXWERT", 10, true);
echo maxwert; /* gibt 10 aus */
```

Im Unterschied zu Variablen können Sie Konstanten nicht direkt in einem String in Anführungszeichen ausgeben lassen, da der PHP-Interpreter sie nicht von Text unterscheiden kann:

```
echo "Der maximale Wert ist MAXWERT"; /* Gibt aus: Der maximale Wert ist MAXWERT */
```

Über `define()` definieren Sie selbst Konstanten. Daneben stellt Ihnen PHP viele vordefinierte Konstanten zur Verfügung – z. B. mathematische Konstanten wie die Zahl Pi:

```
echo M_PI; /* 3.14159265359 */
```

Über eine weitere vordefinierte Konstante können Sie sich beispielsweise Informationen über die eingesetzte PHP-Version anzeigen lassen:

Mehr mathematische Konstanten finden Sie im PHP-Manual unter http://www.php.net/manual/de/math.constants.php.

```
echo "Verwendete PHP-Version" . PHP_VERSION . "<br />\n";
```

Sogenannte *magische Konstanten* liefern Ihnen Informationen über das aktuelle Skript. Sie werden mit zwei Unterstrichen am Anfang und am Ende geschrieben. `__LINE__` liefert Ihnen die aktuelle Zeile des Skripts, `__FILE__` den Namen der Datei und (ab PHP 5.3) `__DIR__` den Namen des Ordners, in dem sich das Skript befindet:

```
01  echo "PI: ";
02  echo M_PI;
03  echo "<br />\n";;
04  echo "Verwendete PHP-Version: ";
05  echo PHP_VERSION;
06  echo "<br />\n";
07  echo "Aktuelle Zeile des Skripts: ";
08  echo __LINE__;
09  echo "<br />\n";
10  echo "Name der Datei: ";
11  echo __FILE__;
12  echo "<br />\n";
13  echo "Name des Ordners: ";
14  echo __DIR__;
15  echo "<br />\n";
```

Listing 4–10 *Vordefinierte Konstanten (vordefinierte_konstanten.php)*

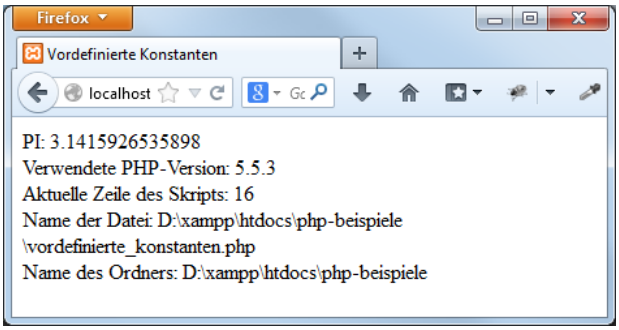


Abb. 4–8 *Ausgabe von vordefinierten Konstanten*

Wenn Sie eine Konstante einsetzen, die nicht definiert ist, so erhalten Sie bei entsprechendem Fehlermeldungsniveau einen Hinweis (Notice). PHP interpretiert diese Konstante aber ansonsten als String und gibt sie einfach direkt aus. Lassen Sie beispielsweise `__DIR__` in einer Version vor PHP 5.3 ausgeben, erhalten Sie die Notice und »__DIR__« wird ausgegeben.

1. http://www.php.net/manual/en/reserved.constants.php

Alle vordefinierten Konstanten in PHP finden Sie im Manual.[1]

4.5 Operatoren

Operatoren brauchen Sie für Berechnungen und zur Verkettung von Zeichenketten.

4.5.1 Arithmetische Operatoren

Natürlich gibt es in PHP auch die in Programmiersprachen üblichen arithmetischen Operatoren. Tabelle 4–2 listet die sechs gebräuchlichen Operatoren für Zahlen auf:

Operator	Operation	Beispiel
+	Addition	$i = 6 + 4; // 10
-	Subtraktion	$i = 6 - 4; // 2
*	Multiplikation	$i = 6 * 4; // 24
/	Division	$i = 6 / 4; // 1.5
%	Modulo	$i = 6 % 4; // 2
** (ab PHP 5.6)	Potenzieren	$i = 3 ** 2; //9

Tab. 4–2 *Arithmetische Operatoren*

Die meisten arithmetischen Operatoren kennen Sie sicher. Neu wird Ihnen aber eventuell der Modulo-Operator (`%`) sein, der den ganzzahligen Rest einer Division zurückgibt.

```
$i = 6 % 4;
```

Der Rest der Division von 6 durch 4 ist 2, so erhält `$i` den Wert 2. Mit dem Modulo-Operator lässt sich beispielsweise leicht ermitteln, ob eine Zahl gerade ist oder nicht. Denn wenn bei der Teilung durch 2 kein Rest übrig bleibt, ist die Zahl gerade.

```
$z = $i % 2; /* Wenn $z gleich 0, dann ist $i gerade */
```

Punkt vor Strich

Wenn Sie Berechnungen im PHP-Code durchführen, dann gilt, so wie man es erwarten würde, die Regel »Punkt vor Strich«. Das heißt, dass in einem Ausdruck wie

```
$i = 5 – 3 * 2;
```

zuerst die Multiplikation ausgeführt wird (`3 * 2`) und danach die Subtraktion. Deswegen erhält im obigen Beispiel `$i` den Wert `-1`. Wenn Sie hingegen wollen, dass zuerst eine andere Operation durchgeführt werden soll, müssen Sie Klammern einsetzen:

```
$k = (5 - 3) * 2;
```

Jetzt wird zuerst `5 – 3` berechnet und das Ergebnis mit `2` malgenommen, `$k` erhält also den Wert `4`.

Das sind die beiden wichtigsten Regeln zur Rangfolge der Operatoren. Weitere Regeln lesen Sie in Kapitel 5.

Kombinierte Operatoren

Häufig ändert man einen Wert und speichert den geänderten Wert wieder in der Variablen:

```
$i = 5;
$i = $i + 2;
```

`$i` hat jetzt den Wert 7.

An diesem Beispiel sehen Sie noch einmal deutlich, dass das »=«-Zeichen in PHP nicht »ist gleich« bedeutet, sondern »erhält den Wert«.

`$i = $i + 2;` lässt sich kürzer schreiben über einen sogenannten kombinierten Operator `+=`:

```
$i = 5;
$i += 2;
```

Diese kombinierten Operatoren gibt es ebenfalls für die anderen Operatoren.

```
$i *= 2; /* entspricht $i = $i * 2; */
$i -= 2; /* entspricht $i = $i - 2; */
$i /= 2; /* entspricht $i = $i / 2; */
$i %= 2; /* entspricht $i = $i % 2; */
```

Sehr häufig möchte man einen Wert um eins erhöhen:

```
$i += 1;
```

Speziell hierfür gibt es noch einen weiteren Operator – den Inkrementoperator:

```
$i++;
```

Entsprechend verringert der Dekrementoperator den Wert um 1:

```
$i--;
```

Übung 2

Definieren Sie zwei Variablen für Zahlen, und lassen Sie mit PHP eine Berechnung durchführen – wählen Sie dabei einen der arithmetischen Operatoren aus! Lassen Sie dann das Ergebnis ausgeben, beispielsweise als X + Y = Z.

4.5.2 Strings verknüpfen

Neben den Operatoren für Zahlen gibt es auch welche für Strings, also Texte. Am wichtigsten ist der Verknüpfungsoperator zur Verkettung von Strings – der Punkt:

```
$vorname = "Denis";
echo "Hallo " . " Welt. "; /* Hallo Welt */
echo "Hallo " . $vorname; /* Hallo Denis */
echo "<br />Guten Morgen, " . $vorname . ", – gut geschlafen?"; /* Guten Morgen,
Denis, gut geschlafen? */
```

Wie Sie sehen, können Sie über den Punkt auch Variablen anketten.

Der Verknüpfungsoperator für Strings lässt sich auch mit einer Zuweisung kombinieren, also `.=`.

Zuerst sehen Sie die ausführliche Variante:

```
$koffer = "Zahnbürste, ";
$koffer = $koffer . "Hose ";
$koffer = $koffer . "und T-Shirt";
echo "Ich nehme $koffer mit.";
```

Im Beispiel wird die Variable `$koffer` mit einem Anfangswert belegt, der dann nach und nach mit weiteren Strings ergänzt wird. Die Ausgabe ist: »Ich nehme Zahnbürste, Hose und T-Shirt mit.«

Das lässt sich über `.=` auch verkürzen:

```
$koffer = "Zahnbürste, ";
$koffer .= "Hose ";
$koffer .= "und T-Shirt";
echo "Ich nehme $koffer mit.";
```

Listing 4–11 *Verknüpfungsoperator für Strings (verknuepfungsoperator.php)*

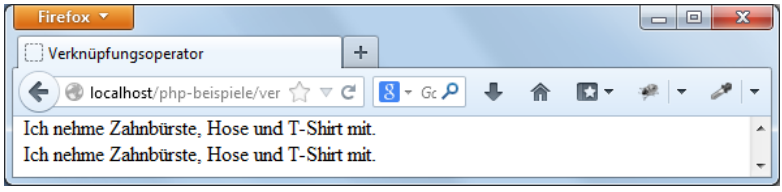


Abb. 4–9 *Die Ausgabe ist in beiden Fällen gleich.*

4.6 Datentypen

Es gibt verschiedene Typen von Daten, mit denen PHP arbeiten kann. Strings und Zahlen sind Ihnen bereits begegnet. Der richtige Zeitpunkt, diese und die weiteren möglichen Datentypen einmal genauer zu betrachten.

Die Datentypen werden in PHP – im Unterschied beispielsweise zu den streng typisierenden Sprachen wie Java – jedoch üblicherweise nicht vom Programmierer explizit gesetzt, sondern von PHP aus dem Kontext erkannt.

4.6.1 Strings

Den Typ String oder Zeichenkette haben Sie bereits kennengelernt. Ein String besteht aus ein oder mehreren Zeichen. Strings werden in einfachen oder doppelten Anführungszeichen notiert. Wahlweise können Sie auch die HereDoc- oder die NowDoc-Konstruktion benutzen:

```
$text  = "Das hier ist ein String";
$text2 = 'Das hier ist auch ein String';
$text3 = "7"; /* auch ein String */
$text4 = "10 Eier";
```

4.6.2 Integer und Float

Außerdem gibt es zwei unterschiedliche Typen für Zahlen: Integer für ganze Zahlen und Float für Fließkommazahlen.

Integer können positiv oder auch negativ sein und werden nicht in Anführungszeichen geschrieben.

```
$ganzezahl = 42;
$nocheine  = -13;
```

Integer werden sicher am häufigsten als Dezimalzahlen angegeben, das heißt mit 10 als Basis. Aber Sie können auch Zahlen definieren, die eine andere Basis als 10 haben, wie Oktalzahlen oder Hexadezimalzahlen. Bei Oktalzahlen, die als Basis 8 haben, wird eine 0 vorangestellt.

```
$oktal = 012; /* entspricht 10 */
```

Hexadezimalzahlen mit der Basis 16 kennen Sie von den Farbangaben in HTML/CSS: In PHP schreiben Sie am Anfang von Hexadezimalzahlen `0x`:

```
$hexadezimal = 0xFF; /* entspricht dezimal 255 */
```

Fließkommazahlen (Float) werden mit einem Punkt geschrieben:

```
$float = 1.5;
```

Ebenfalls möglich ist die wissenschaftliche Schreibweise für Fließkommazahlen:

```
$a = 1.2e3; /* entspricht 1.2 * 103, d.h. 1200 */
$b = 7e-2; /* entspricht 7 * 10-2 , d.h. 0.07 */
```

Neben Float finden Sie übrigens an manchen Stellen auch die Bezeichnung Double. Float und Double sind bei PHP identisch, und der Name Double taucht mitunter aus historischen Gründen auf.

4.6.3 Wahrheitswerte

Der boolesche Typ ist ein weiterer möglicher Datentyp, dabei handelt es sich um einen Wahrheitswert. Er kann nur `true` (wahr) oder `false` (falsch) annehmen. Sie haben ihn schon als drittes Argument von `define()` gesehen.

```
$regnen = true;
```

Die Groß- und Kleinschreibung ist dabei nicht relevant, Sie können auch `TRUE` und `FALSE` schreiben. Boolesche Werte brauchen Sie bei der Überprüfung von Bedingungen, d.h., wenn beispielsweise eine Meldung ausgegeben werden soll, sofern der Benutzer einen gültigen Benutzernamen eingegeben hat. Das Ergebnis einer Überprüfung ist dann `true` oder `false`. Mehr dazu in Kapitel 5.

4.6.4 Weitere Datentypen

Es gibt noch weitere sogenannte zusammengesetzte Typen: Arrays und Objekte. Zu Arrays folgt gleich mehr (Abschnitt 4.7) und Genaueres zu Objekten lesen Sie in Kapitel 5.

Außerdem gibt es noch Ressourcen, die eine Referenz auf eine externe Ressource beinhalten, wie beispielsweise auf eine geöffnete Datei oder auf eine Verbindung zu einer Datenbank. Wie Sie mit PHP auf Dateien zugreifen, ist Thema von Kapitel 12 und in Kapitel 11 geht es um Datenbankverbindungen.

`NULL` ist ein weiterer Datentyp und repräsentiert eine Variable ohne Wert. Das bedeutet: Diesem Typ gehört eine Variable an, der Sie entweder noch keinen Wert zugewiesen haben, die Sie explizit auf `NULL` gesetzt haben oder die Sie mit der PHP-Funktion `unset()` gelöscht haben.

4.6.5 Immer der richtige Typ

Eine Variable kann innerhalb eines Skripts beliebig den Wert wechseln:

```
$a = "Hallo"; // String
$a = 7; // Integer
$a = 3.5; // Float
```

In diesem Beispiel ist `$a` zuerst vom Typ String, dann ein Integer und schließlich eine Fließkommazahl. PHP führt Konvertierungen zwischen den einzelnen Variablentypen automatisch durch. Es ermittelt automatisch den Typ einer Variable aus dem Kontext.

Float und Integer

Was das Ergebnis einer Berechnung ist – eine Fließkommazahl oder ein Integer – ist eigentlich so, wie man es intuitiv erwarten würde. Um das genauer anzusehen, brauchen wir eine Methode, um zu ermitteln, welchem Datentyp eine bestimmte Variable

angehört. Hier bietet sich die Funktion `var_dump()` an. `var_dump()` übergeben Sie in runden Klammern die Variable, über die Sie mehr Informationen erhalten möchten. `var_dump()` gibt dann den Inhalt der Variablen und den Typ aus.

```
01  $a = 20;
02  $b = 3;
03  $c = 3.5;
04  $d = -3;
05  $e = -20;
06
07  $erg = $a / $b;
08  var_dump($erg);
09  echo "<br />\n";
10  $erg2 = $a + $b;
11  var_dump($erg2);
12  echo "<br />\n";
13  $erg3 = $a + $c;
14  var_dump($erg3);
15  echo "<br />\n";
16  $erg4 =  $a / $e;
17  var_dump($erg4);
```

Listing 4–12 *Jonglieren zwischen Integer und Float (integer_float.php)*

In den ersten fünf Zeilen werden Variablen mit Werten vorbelegt. Zeile 7 berechnet 20/3. Das Ergebnis samt Variablentyp wird über `var_dump()` in Zeile 8 ausgegeben: Ein Float mit dem Wert 6.666666667. In Zeile 10 werden zwei ganze Zahlen (20+3) addiert, und das Ergebnis ist ein Integer, wie man erwarten würde. Genauso einsichtig sind auch die beiden anderen Ergebnisse. Die Ausgabe des Skripts zeigt Abbildung 4–10.

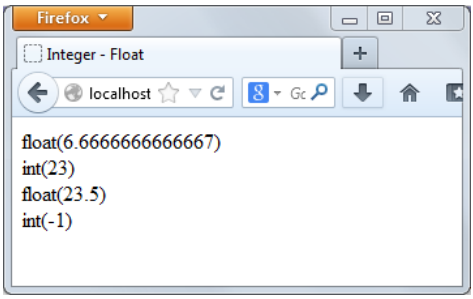


Abb. 4–10 *Das Ergebnis der Berechnungen mit Angabe des Dateityps*

Konvertierung von String in Zahlen

Werden Strings und Zahlen kombiniert, findet die Konvertierung automatisch statt. Die Konvertierung von Zahl zu String ist einfach, aus 7 wird eben »7«. Umgekehrt gibt es teilweise ungewöhnliche Ergebnisse.

Um das an einem Beispiel zu zeigen, benötigen wir wieder einen Kontext, der eine Konvertierung von einem String in eine Zahl auslöst, beispielsweise die Addition. Bei der Konvertierung in eine Zahl gilt folgendes Prinzip: Wenn ein String mit einer Zahl beginnt, wird diese genommen und der Rest verworfen. Wenn der String nicht mit einer Zahl beginnt, wird der String zu 0 konvertiert. Das gilt aber nur für das Ergebnis, der Inhalt der Variablen selbst bleibt unverändert. Ein Beispiel zeigt das:

```
01  $str1 = "10 Eier";
02  $str2 = "Schachtel mit 10 Eiern";
03  $str3 = "3.5 Äpfel";
04  $erg1 = $str1 + 2;
05  var_dump($erg1);
06  echo "<br />\n";
07  $erg2 = $str2 + 2;
08  var_dump($erg2);
09  echo "<br />\n";
10  $erg3 = $str3 + 2;
11  var_dump($erg3);
```

Listing 4–13 *Beispiel für die automatische Konvertierung von Strings (string_zu_zahl.php)*

In Zeile 4 wird »10 Eier« + 2 berechnet. Das Ergebnis ist 12. Das Ergebnis von »Schachtel mit 10 Eiern« + 2 ist hingegen 2. Denn »Schachtel mit 10 Eiern« beginnt nicht mit einer Zahl und wird als 0 ausgewertet. »3.5 Äpfel« + 2 (Zeile 10) ergibt dann entsprechend 5.5 und ist ein Float. Abbildung 4–11 zeigt die Ausgabe.

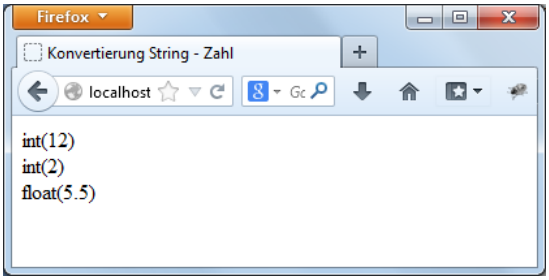


Abb. 4–11 *Das Ergebnis der Addition mit Strings und Zahlen*

Schön sind diese Umwandlungen nicht, und im Normalfall wird man vermeiden, so etwas zu tun. Wichtig ist aber für Sie: Falls Sie einen String mit einer Zahl addieren, wird PHP das klaglos durchführen; Sie erhalten keine Warnung oder Fehlermeldung.

4.6.6 TypeCasting

Anstatt Umwandlungen von PHP automatisch durchführen zu lassen, können Sie auch direkt eine Umwandlung anstoßen, beispielsweise über `(int)` in einen Integer, über `(float)` in eine Fließkommazahl oder über `(string)` in einen String. In folgen-

dem Beispiel wird ein String explizit in einen Integer verwandelt. Die Ausgabe von `var_dump()` ist entsprechend »int(22)«:

```
$string = "22";
$zahl = (int) $string;
var_dump($zahl);
```

Listing 4–14 *Umwandlungen direkt durchführen (typecasting.php)*

Außerdem stehen hierfür auch Funktionen zur Verfügung, nämlich `intval()`, `floatval()`, `stringval()` und ab PHP 5.5 auch `boolval()`.

4.7 Arrays

Die Typen von Variablen, die bisher besprochen wurden, speichern genau einen Wert. Manchmal möchte man aber gleichzeitig mit mehreren Werten arbeiten, beispielsweise mit einer Liste von möglichen Farben, einer Liste von Gästen, einer Liste von zur Verfügung stehenden Versionen oder Sprachen, einer Liste von Preisen oder Produkten etc. Genau dafür sind Arrays gedacht, die mitunter auch Felder genannt werden.

Wenn man in einer Variablen mehrere Werte speichert, stehen viele nützliche Möglichkeiten offen: Die Werte lassen sich sortieren und neu ausgeben, man kann auf einzelne gezielt zugreifen, sie vergleichen, zählen, weitere ergänzen und wieder ausgeben lassen.

4.7.1 Arrays erstellen

Um ein Array zu erstellen, verwenden Sie das Schlüsselwort `array()`. Hier einmal ein Beispiel für ein einfaches Array mit drei Elementen:

```
$antworten = array("nie", "manchmal", "oft");
```

In den Klammern hinter `array()` führen Sie bei der Definition eines Arrays die einzelnen Werte durch Komma getrennt auf. Wenn es Strings sind, schreiben Sie sie wie gewohnt in Anführungszeichen. Zahlen notieren Sie ohne:

```
$werte = array(42, 66, 3.5, 55, 7);
```

Innerhalb eines Arrays können auch verschiedene Typen kombiniert werden:

```
$antworten = array("nie", "manchmal", "oft", 42);
```

Seit PHP 5.4 gibt es eine alternative Schreibweise zur Definition von Arrays ohne das `array()`-Sprachkonstrukt: Sie geben die Werte für das Array in eckigen Klammern an:

```
$antworten = ["nie", "manchmal", "oft", 42];
```

Diese Schreibweise gibt es auch in anderen Sprachen, wie beispielsweise JavaScript. Sie finden ein entsprechendes Beispiel im Listing *arrays_alternativ.php*, das sich in Ihrem Listingarchiv zu diesem Kapitel befindet.

Die einzelnen Elemente werden von PHP automatisch durchnummeriert. Die Nummerierung beginnt dabei – das ist wichtig – bei 0. Das ist der sogenannte Index. Um ein einzelnes Element auszulesen, schreiben Sie den Namen des Arrays und in eckigen Klammern den Index:

```
echo $antworten[0]; /* nie */
echo "<br />\n";
echo $antworten[2]; /* oft */
```

Sie können Arrays auch problemlos im Nachhinein mit weiteren Elementen ergänzen.

Nehmen wir noch einmal das bestehende Array:

```
$antworten = array("nie", "manchmal", "oft", 42);
```

Dann können Sie durch folgende Zeile ein weiteres Element anhängen:

```
$antworten[] = "aus Prinzip nicht";
```

Und das ließe sich natürlich ausgeben:

```
echo $antworten[4];
```

Mit PHP 5.5 gibt es auch die Möglichkeit der Dereferenzierung von Arrays wie im folgenden Beispiel:

```
echo [1, 2, 3][0];
```

Damit wird 1 ausgegeben. Diese Syntax werden Sie wahrscheinlich nicht aktiv brauchen, aber es ist gut zu wissen, dass es sie gibt.

4.7.2 Informationen über Arrays ausgeben lassen

Wenn Sie versuchen, das Array als Ganzes per `echo` auszugeben, sieht das Ergebnis nicht wie gewünscht aus:

```
echo $antworten;
```

Das schreibt einfach »Array« auf den Bildschirm.

Um sich schnell einen Überblick über die Inhalte zu verschaffen, ist die PHP-Funktion `print_r()` praktisch.

```
print_r($antworten);
```

Listing 4–15 *Ausschnitt aus dem Listing arrays_print_r.php*

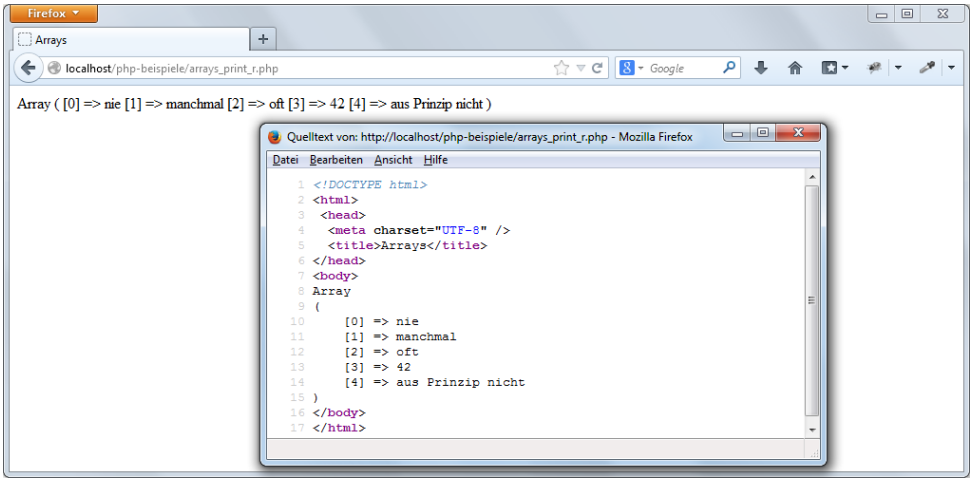


Abb. 4–12 *print_r() zeigt, was in Ihrem Array steckt.*

Abbildung 4–12 zeigt das Ergebnis von `print_r()`: Die Anzeige der Arrayinhalte mit den zugehörigen Indizes wird im Quellcode noch übersichtlicher angezeigt. Diese Einrückung wird natürlich im Browser nicht dargestellt, da Einrückungen im HTML-Quellcode vom Browser ignoriert werden.

Wollen Sie den Browser dazu bringen, die Inhalte wie im Quellcode anzuzeigen, inklusive aller Leerzeichen, können Sie das ansonsten selten verwendete HTML-Element `pre` benutzen und die Ausgabe von `print_r()` innerhalb der Start- und Endtags von `pre` schreiben:

```
echo "<pre>";
print_r($antworten);
echo "</pre>";
```

Listing 4–16 *Mit ergänztem HTML-Element pre (arrays_print_r_pre.php)*

Noch ausführlichere Informationen über Ihr Array erhalten Sie, wenn Sie anstelle von `print_r()` die Funktion `var_dump()` benutzen:

```
echo "<pre>";
var_dump($antworten);
echo "</pre>";
```

Listing 4–17 *Der Inhalt des Arrays wird dieses Mal über die Funktion var_dump() ausgegeben (arrays_var_dump.php).*

Sie sehen dann gleichzeitig, um welchen Datentyp es sich handelt, und bei Strings auch ihre Länge.

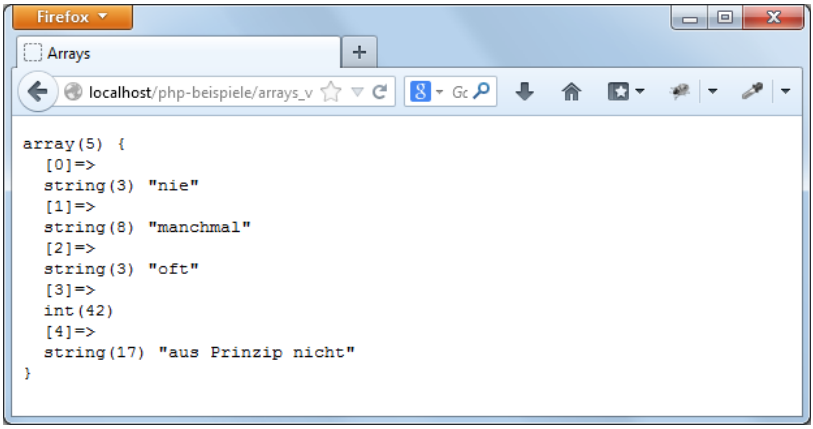


Abb. 4–13 *var_dump() liefert ausführlichere Informationen zu den Inhalten von Arrays.*

4.7.3 Arrays durchlaufen mit foreach

Die Ausgabe mit `var_dump()` oder `print_r()` ist nur geeignet, um sich bei der Programmierung einen schnellen Überblick über den Inhalt zu verschaffen – man könnte diese Ausgabe nicht einem normalen Benutzer zumuten. Dafür gibt es andere Wege: Speziell für die Ausgabe oder sonstige Bearbeitung aller Elemente eines Arrays existiert die Schleife `foreach`. Bei `foreach` werden Schritt für Schritt die einzelnen Elemente des Arrays durchlaufen und die von Ihnen festgelegten Anweisungen für jedes Element ausgeführt. Sie müssen `foreach` nicht sagen, wie oft es das durchführen soll, denn `foreach` wird durch die Anzahl der Arrayelemente selbst begrenzt.

In runden Klammern hinter `foreach` geben Sie zuerst das Array an, das Sie durchlaufen möchten. Danach folgt das Schlüsselwort `as` und danach der Name einer temporären Variablen, die den Wert der einzelnen Elemente zwischenspeichert. Der Name der Variable ist frei wählbar. In geschweiften Klammern steht der Code, der für jedes Element ausgeführt werden soll. Um jedes Element auszugeben, verwenden Sie den Namen, den Sie für die temporäre Variable eingesetzt haben.

Durch folgenden Code wird jedes Element des `$antworten`-Arrays ausgegeben – gefolgt jeweils von einem Zeilenumbruch:

```
foreach ($antworten as $aw) {
  echo "$aw <br />\n";
}
```

Wenn Sie außerhalb von `foreach` noch einmal auf die Variable `$aw` zugreifen, erhalten Sie den zuletzt dort gespeicherten Array-Wert:

```
foreach ($antworten as $aw) {
  echo "$aw <br />";
}
echo $aw; /* aus Prinzip nicht */
```

Listing 4–18 *Arrays können über foreach durchlaufen werden (arrays_foreach.php).*

Um die Anzahl der Elemente eines Arrays zu ermitteln, können Sie die Funktion `count()` einsetzen. Bei `count()` notieren Sie in runden Klammern das Array, dessen Elemente Sie zählen möchten. Als Rückgabewert erhalten Sie die Anzahl der Elemente:

```
$anzahl = count($antworten);
echo $anzahl; // 5
```

Übung 3

Erstellen Sie ein Array mit fünf Orten. Lassen Sie dann alle Orte in einer `foreach`-Schleife ausgeben, wobei nach jedem Ort immer ein Zeilenumbruch `<br />` eingefügt werden soll.

Übung 4

Modifizieren Sie die Ausgabe des Arrays aus der letzten Übung, sodass die Orte als ungeordnete Liste ausgegeben werden.

Sie erinnern sich: Eine ungeordnete Liste wird mit `<ul>` eingeleitet und mit `</ul>` beendet. Die einzelnen Punkte werden hingegen von `<li>` und `</li>` eingerahmt (siehe auch Kap. 3).

Kontrollieren Sie dann in der HTML-Quellcode-Ansicht, ob der erzeugte HTML-Code korrekt ist!

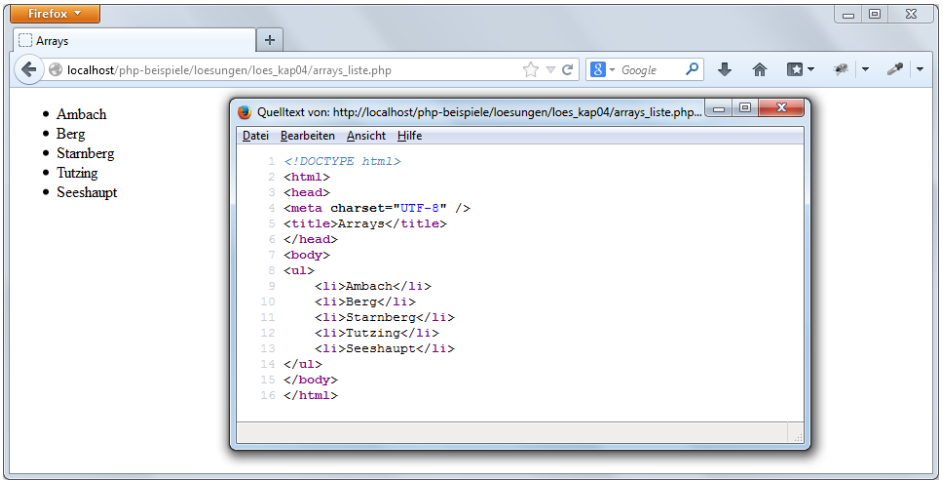


Abb. 4–14 *Eine mögliche Ausgabe mit dem erzeugten HTML-Code*

4.7.4 Zufällig ein Bild anzeigen lassen

Jetzt ein kleines Beispiel für die Verwendung von Arrays. Es soll zufällig eines von mehreren Bildern ausgegeben werden. Die Pfade zu den Bildern werden dafür in einem Array gespeichert.

Außerdem benötigen wir eine Funktion, die eine zufällige Zahl ermittelt. Genau dafür gibt es `rand()`. `rand()` erwartet in runden Klammern zwei Werte: Der eine bestimmt den minimalen Wert der Zufallszahl, der andere gibt den höchsten möglichen Wert an:

```
$zufallszahl = rand(0, 4);
```

Damit ist eine Zahl von 0 bis einschließlich 4 in `$zufallszahl` gespeichert.

Kommen wir zur zufälligen Ausgabe von Bildern:

```
01  <!DOCTYPE html>
02  <html>
03   <head>
04    <meta charset="UTF-8" />
05    <title>Zufallsbilder</title>
06  </head>
07  <body>
08  <?php
09  $bilder = array("blumen.jpg", "boot.jpg",
10                  "landschaft.jpg", "stadt_am_meer.jpg",
11                  "strand.jpg");
12  $max = count($bilder) - 1;
13  $zufallszahl = rand(0, $max);
14  echo "<img src='$bilder[$zufallszahl]' height='200' width='150' />";
15  ?>
16  </body>
17  </html>
```

Listing 4–19 *Welches Bild angezeigt wird, bestimmt der Zufall (zufallsbilder.php).*

In Zeile 9 wird ein Array namens `$bilder` angelegt. Es beinhaltet die Pfade zu den Bildern, die sich in demselben Ordner befinden wie das PHP-Skript selbst.

Zeile 12 ermittelt die Anzahl der Elemente des Arrays und zieht 1 davon ab. Damit haben wir in `$max` den höchsten Index des Arrays. Im Beispiel enthält das Array 5 Elemente. Der letzte Index ist aber 4 – da beim Index mit 0 zu zählen begonnen wird –, also eins weniger.

Zeile 13 ruft die Funktion `rand()` auf. Sie soll eine Zahl zwischen 0 und dem in `$max` gespeicherten höchsten Index generieren. Diese wird in der Variablen `$zufallszahl` gespeichert.

In Zeile 14 erfolgt die Ausgabe des Zufallsbildes über das hierfür benötigte `img`-Element, das beim Attribut `src` den Pfad zur Datei erwartet. Hier wird auf das Array `$bilder` zurückgegriffen und als Index die Variable `$zufallszahl` benutzt, die ja einen Wert zwischen 0 und dem letzten Index enthält. Damit wird immer ein anderes Bild aus dem Bilderarray ausgelesen.

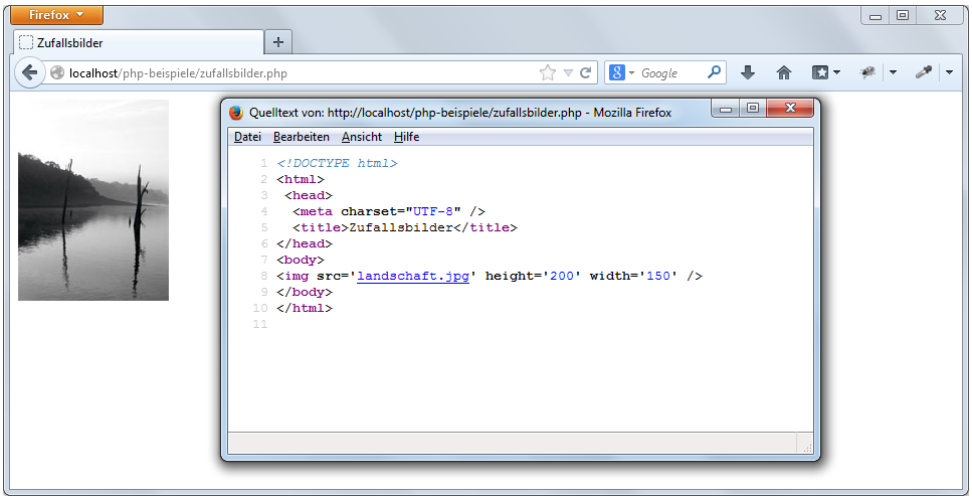


Abb. 4–15 *Zufallsbild – und anbei der erzeugte Quellcode*

Wenn Sie das Skript testen, klicken Sie mehrmals auf den Reload-Button: Welche Bilder angezeigt werden, wird zufällig bestimmt.

Übung 5

Ändern Sie das Beispiel *zufallsbilder.php* so ab, dass zufällig einer von mehreren Texten angezeigt wird. Dafür müssen Sie natürlich zuerst ein Array mit mehreren Strings definieren!

Das Beispiel sollte den Zusammenhang von Index und Anzahl der Elemente eines Arrays illustrieren. Sonst hätte man sich eine Zeile Code sparen können, indem man die von PHP zur Verfügung gestellte Funktion `array_rand()` benutzt, die aus dem Array, das man ihr in Klammern übergibt, zufällig einen Index wählt. Das Beispiel finden Sie unter dem Namen *zufallsbilder_array_rand.php* ebenfalls in den Listings, die Sie auf der Webseite zu diesem Buch unter www.dpunkt.de/php56 herunterladen können.

4.7.5 Assoziative Arrays

Bisher haben wir die einzelnen Elemente über Nummern angesprochen. Manchmal möchte man aber die Arrayelemente über Namen ansprechen. Solche Schlüssel-Wert-Paare können Sie einsetzen, wenn Sie beispielsweise deutsche Farbnamen den entsprechenden in HTML/CSS üblichen hexadezimalen Farbbezeichnungen zuordnen möchten. Oder um Vorwahlnummern Städten zuzuordnen, Produktklassen zu Mehrwertsteuersätzen usw. Auch das ist mit Arrays möglich. Diese Sorte von Arrays wird

im Gegensatz zu den gerade besprochenen indizierten Arrays als *assoziative Arrays* bezeichnet.

Zur Erstellung eines assoziativen Arrays verwenden Sie wieder `array()`, schreiben aber in runde Klammern immer die Schlüssel-Wert-Paare, die durch `=>` verknüpft werden:

```
$farben = array ("rot"  => "#FF0000",
                 "grün" => "#00FF00",
                 "blau" => "#0000FF");
```

Falls Sie die Funktion `array()` nicht einsetzen wollen, können Sie die Elemente eines assoziativen Arrays auch einzeln definieren:

```
$farben["rot"]  = "#FF0000";
$farben["grün"] = "#00FF00";
```

Auf diese Art lassen sich auch nachträglich weitere Elemente ergänzen:

```
$farben["schwarz"] = "#000000";
```

Einzelne Werte sprechen Sie an, indem Sie in eckigen Klammern den Schlüssel schreiben:

```
echo $farben["rot"];
```

Es gibt auch viele in PHP vordefinierte assoziative Arrays. So können Sie über `$_SERVER["PHP_SELF"]` auf den Pfad zum aktuellen Skript zugreifen oder über `$_GET["name"]` oder `$_POST["name"]` auf den Inhalt von Formulardaten. `$_SERVER` lernen Sie in der nächsten Übung kurz kennen, die anderen assoziativen Arrays sind Thema von Kapitel 7.

Einen schnellen Überblick über den Inhalt eines Arrays verschaffen Sie sich wiederum mit `print_r()` oder `var_dump()`:

```
print_r($farben);
```

Um die Inhalte ansprechender auszugeben, brauchen Sie `foreach`. In runden Klammern geben Sie zuerst den Namen des Arrays an, das durchlaufen werden soll. Dann folgen das Schlüsselwort `as` und zwei Variablen, die als temporäre Speicher für jeweils den Schlüssel und den dazugehörigen Wert dienen und durch `=>` getrennt werden.

```
foreach ($farben as $k => $v){
  echo "Schlüssel: $k, Wert: $v<br />\n";
}
```

Listing 4–20 *Assoziative Arrays können ebenfalls über foreach ausgegeben werden (ass_array.php).*

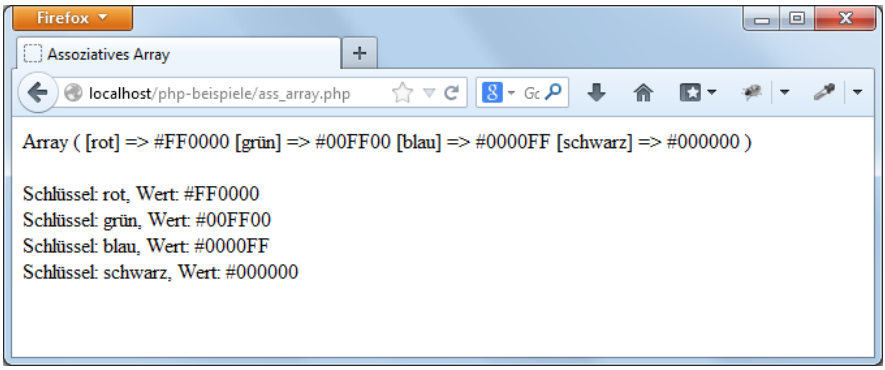


Abb. 4–16 *Das assoziative Array wird ausgegeben: oben über print_r(), unten über foreach.*

Übung 6

Erstellen Sie eine `foreach`-Schleife, die das vordefinierte Array `$_SERVER` ausgibt. Den Code können Sie ganz parallel zum Beispiel *ass_array.php* aufbauen – mit dem einzigen Unterschied, dass Sie `$_SERVER` nicht erst definieren müssen.

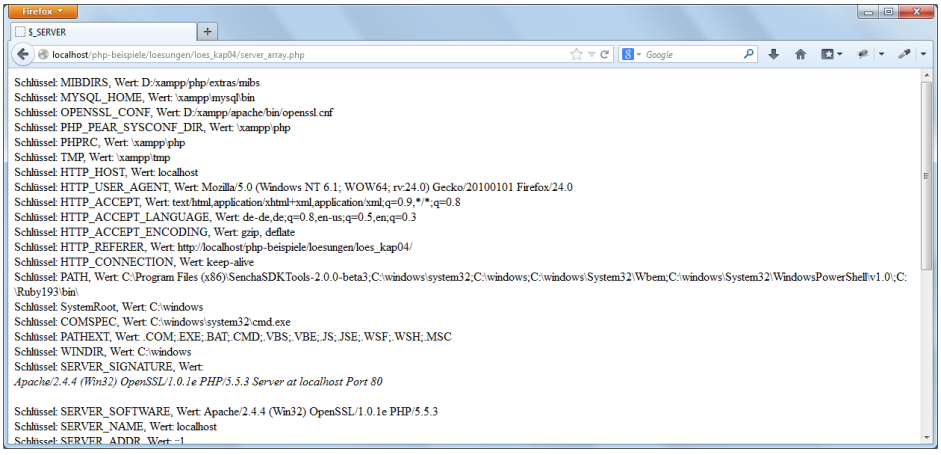
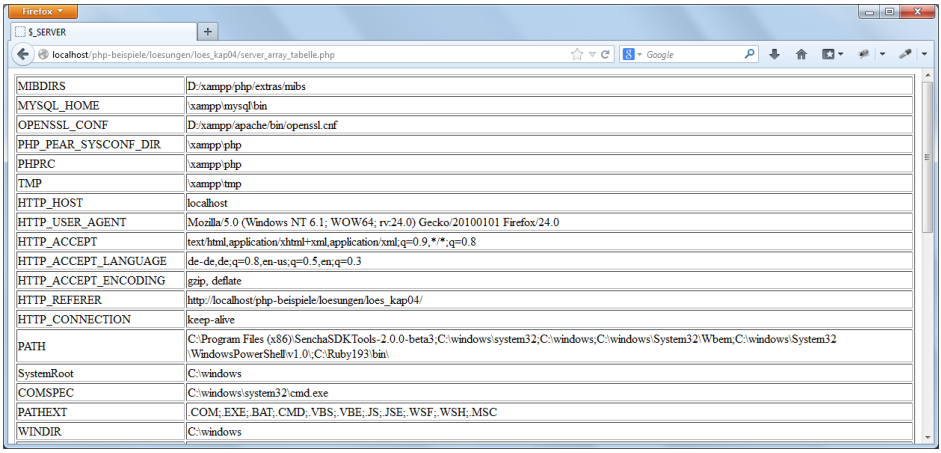


Abb. 4–17 *Ausgabe des $_SERVER-Arrays*

Übung 7

Modifizieren Sie das Beispiel aus der letzten Übung so, dass Sie das `$_SERVER`-Array innerhalb einer Tabelle ausgeben lassen. Innerhalb der ersten Spalte soll jeweils der Schlüssel ausgegeben werden, innerhalb der zweiten Spalte der Wert.



MIBDIRS	D:/xampp/php/extras/mibs
MYSQL_HOME	\xampp\mysql\bin
OPENSSL_CONF	D:/xampp/apache/bin/openssl.cnf
PHP_PEAR_SYSCONF_DIR	\xampp\php
PHPRC	\xampp\php
TMP	\xampp\tmp
HTTP_HOST	localhost
HTTP_USER_AGENT	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0
HTTP_ACCEPT	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE	de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
HTTP_ACCEPT_ENCODING	gzip, deflate
HTTP_REFERER	http://localhost/php-beispiele/loesungen/loes_kap04/
HTTP_CONNECTION	keep-alive
PATH	C:\Program Files (x86)\SenchaSDKTools-2.0.0-beta3;C:\windows\system32;C:\windows;C:\windows\System32\Wbem;C:\windows\System32\WindowsPowerShell\v1.0\;C:\Ruby193\bin\
SystemRoot	C:\windows
COMSPEC	C:\windows\system32\cmd.exe
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
WINDIR	C:\windows

Abb. 4–18 *Die Tabelle mit den Inhalten des $_SERVER-Arrays*

Bei Bedarf sehen Sie noch einmal in Kapitel 3 nach, wie Tabellen in HTML erstellt werden.

4.7.6 Schlüssel von Arrays richtig angeben

Kommen wir noch einmal zur Ausgabe eines einzelnen Elements bei assoziativen Arrays. Dafür schreiben Sie den Schlüssel in den eckigen Klammern in Anführungszeichen, sofern es sich um einen String handelt:

```
echo $farben["rot"];
```

Wenn Sie bei diesem Schlüssel, der ein String ist, die Anführungszeichen weglassen, erhalten Sie einen Hinweis (Notice). PHP beschwert sich, dass eine nicht definierte Konstante verwendet wird:

```
echo $farben[rot];
```

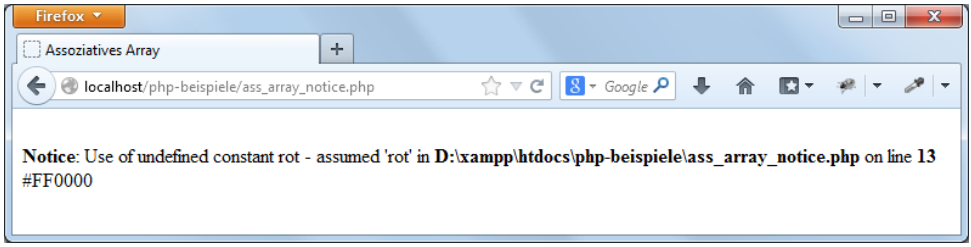


Abb. 4–19 *Ein Hinweis erscheint, wenn man bei einem Schlüssel, der ein String ist, keine Anführungszeichen setzt.*

Sie erinnern sich? Konstanten werden ohne Dollarzeichen geschrieben. Bei nicht definierten Konstanten nimmt PHP an, dass es sich um den entsprechenden String handelt, und gibt diesen aus.

Diese Schreibweise ohne Anführungszeichen begegnet Ihnen mitunter noch in älteren Skripten, Sie sollten sie aber vermeiden. Es könnte zu Problemen führen, wenn Sie einmal in einem Skript wirklich eine Konstante mit demselben Namen – hier in unserem Fall `rot` – definiert hätten.

Handelt es sich hingegen um eine Zahl beim Schlüssel, verwenden Sie natürlich keine Anführungszeichen:

```
echo $antworten[0];
```

4.7.7 Arrays und Variableninterpolation

Nun zu Besonderheiten bei der Interpolation von Arrayvariablen in Strings. Wenn der Schlüssel eine Zahl ist, können Sie den Wert des Arrayelements problemlos in doppelten Anführungszeichen ausgeben lassen:

```
echo "Sag niemals $antwort[0]";
```

Wenn Sie hingegen einen String als Schlüssel haben, funktioniert das so schon einmal nicht:

```
echo "die Farbe ist $farben["rot"]"; /* geht nicht */
```

Auch mit einfachen Anführungszeichen geht es nicht:

```
echo "die Farbe ist $farben['rot']"; /* geht nicht */
```

Mit einfachen Anführungszeichen klappt es hingegen, wenn Sie – wie bereits in Abschnitt 4.3.4 vorgestellt – die geschweiften Klammern zur Klammerung des Ausdrucks verwenden:

```
echo "die Farbe ist {$farben['rot']}; /* geht */
```

Zwei weitere Varianten gibt es noch: Sie können den Verknüpfungsoperator einsetzen, um das Problem elegant zu umgehen:

```
echo "die Farbe ist " . $farben["rot"]; /* geht auch */
```

Es funktioniert außerdem noch, wenn Sie den Schlüssel ohne Anführungszeichen schreiben:

```
echo "die Farbe ist $farben[rot]"; /* geht auch */
```

4.7.8 Verschachtelte Arrays am Beispiel

Arrays können Sie auch verschachteln. Eben hatten wir ja ein Beispiel, in dem zufällig eins von mehreren Bildern angezeigt wurde. Dabei wurde ein `img`-Element mit unterschiedlichen Pfadangaben ausgegeben. Was aber, wenn man noch mehr Informationen zum jeweiligen Bild ausgeben lassen möchte? Obligatorisch wäre ja eigentlich das `alt`-Attribut für einen alternativen Text, außerdem könnte man das `img`-Element noch mit einem `title`-Attribut bestücken. Der Inhalt des `title`-Attributs wird von Browsern in Form eines Tooltipps angezeigt. Damit müsste beispielsweise folgender Code erzeugt werden:

```
<img src='stadt_am_meer.jpg' height='200' width='150' alt='Häuser'
title='Griechische Häuser am Abend' />
```

Dieses Mal soll sich also nicht nur der Inhalt des `src`-Attributs ändern, sondern es sollen auch andere Texte für `alt` und `title` gezeigt werden.

Dafür braucht man ein verschachteltes Array. Die Bildinformationen zu einem einzelnen Bild werden als assoziatives Array gespeichert:

```
array("pfad" => "stadt_am_meer.jpg",
     "alt"   => "Häuser",
     "titel" => "Griechische Häuser am Abend");
```

Entsprechend geht das auch für die anderen Bilder. Aus diesen Arrays wird dann ein verschachteltes Array gebaut, das ist ein Array, das selbst wieder Arrays als Elemente hat. Im Beispiel heißt das Array `$bilder` und enthält als Elemente die Arrays mit den einzelnen Bildinformationen:

```
01  $bilder = array(
02                array("pfad"  => "blumen.jpg",
03                      "alt"   => "rote Blumen",
04                      "titel" => "Strauß aus roten Blumen"),
05                array("pfad"  => "landschaft.jpg",
06                      "alt"   => "Landschaft",
07                      "titel" => "Landschaft im Nebel"),
08                array("pfad"  => "stadt_am_meer.jpg",
09                      "alt"   => "Häuser",
10                      "titel" => "Griechische Häuser am Abend"),
11                array("pfad"  => "strand.jpg",
12                      "alt"   => "Strand",
13                      "titel" => "Strand mit Bergen"),
14                array("pfad"  => "boot.jpg",
15                      "alt"   => "Boot",
16                      "titel" => "Boot auf einem Felsen")
17            );
```

Um auf einzelne Werte zuzugreifen, schreiben Sie zuerst den Namen des Arrays, also `$bilder`. Dahinter folgen zwei eckige Klammernpaare. In das erste schreiben Sie den Index des verschachtelten Arrays, auf das Sie zugreifen wollen, und in die zweiten eckigen Klammern schreiben Sie den Namen des Werts, den Sie auslesen möchten:

```
echo $bilder[0]["pfad"]; // blumen.jpg
```

Damit lässt sich das Skript zur zufälligen Ausgabe von Bildern mit mehr Informationen folgendermaßen erstellen:

```
    /*Definition des verschachtelten Arrays wie oben */
18  $max = count($bilder) - 1;
19  $zufallszahl = rand(0, $max);
20  echo "<img src='{$bilder[$zufallszahl]['pfad']}'
21        height='200' width='150'
22        alt='{$bilder[$zufallszahl]['alt']}'
23        title='{$bilder[$zufallszahl]['titel']}' />\n";
```

Listing 4–21 *Dieses Mal können bei den einzelnen per Zufall angezeigten Bildern die jeweils passenden alt- und title-Werte bestimmt werden (zufallsbilder_erweitert.php).*

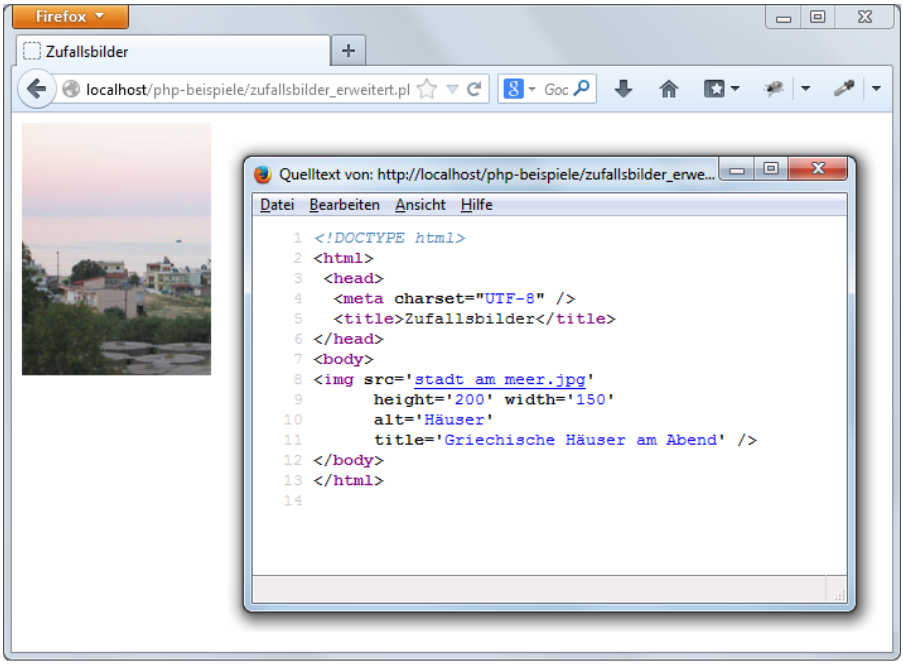


Abb. 4–20 *Zufallsbilder mit den richtigen Attributen, wie man in der HTML-Code-Ansicht sieht*

PHP stellt viele nützliche Funktionen zur Arbeit mit Arrays bereit. Mehr dazu in Kapitel 6.

4.8 Nützlich für alle Zwecke: Dateien einbinden

Zum Abschluss des Kapitels geht es um eine praktische Funktion zum Einbinden von Dateien. Oft haben Sie bei Webprojekten Bereiche, die auf allen Webseiten vorkommen – beispielsweise einen Kopfbereich oder eine Fußzeile. Wenn sich der Inhalt bei einem dieser Bereiche ändert, müssen Sie jede Datei einzeln bearbeiten, wo dieser Bereich vorkommt. Praktischer ist es, diese Bereiche in einzelne Dateien auszulagern und dann per PHP einzubinden.

Genau hierfür gibt es in PHP zwei Sprachkonstrukte, nämlich `include` und `require`. Sehen wir uns erst einmal die Funktionsweise von `include` an. An die Stelle, an der Sie die externe Datei einbinden wollen, notieren Sie `include` und dahinter den Pfad zu der Datei, die Sie einbinden möchten.

Im folgenden Beispiel wird `include` zweimal eingesetzt: Am Anfang des Dokuments wird damit ein Begrüßungstext ausgegeben, und am Ende des Dokuments wird über eine externe Datei ein Copyright-Vermerk ergänzt.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8" />
  <title>Dateien einbinden</title>
</head>
<body>
<?php
include "header.php";
?>
<h2>Lorem ipsum dolor </h2>
<p>sit amet ....</p>
<?php
include "copyright.php";
?>
</body>
</html>
```

Listing 4–22 *Zwei Dateien werden per include eingebunden (include_beispiel.php).*

Kommen wir zu den eingebundenen Dateien. Der Inhalt von *copyright.php* ist ganz kurz, die Datei besteht nur aus einer Zeile (kein HTML-Gerüst drumherum!):

```
<p>&copy; 2015 Example.com</p>
```

Listing 4–23 *Die Datei copyright.php ist einzeilig.*

In *copyright.php* steht nur HTML-Code: ein Absatz mit einem ©-Zeichen, der Jahreszahl und einer fiktiver Domain.

Nun zur zweiten eingebundenen Datei: *header.php*. Diese beinhaltet hingegen PHP-Code: Hier wird ein Willkommensgruß mit dem aktuellen Datum ausgegeben.

```
<?php
date_default_timezone_set("Europe/Berlin");
echo "<h1>Willkommen am ";
echo date("j.n.Y");
echo "</h1>\n";
?>
```

Listing 4–24 *Der Inhalt von header.php*

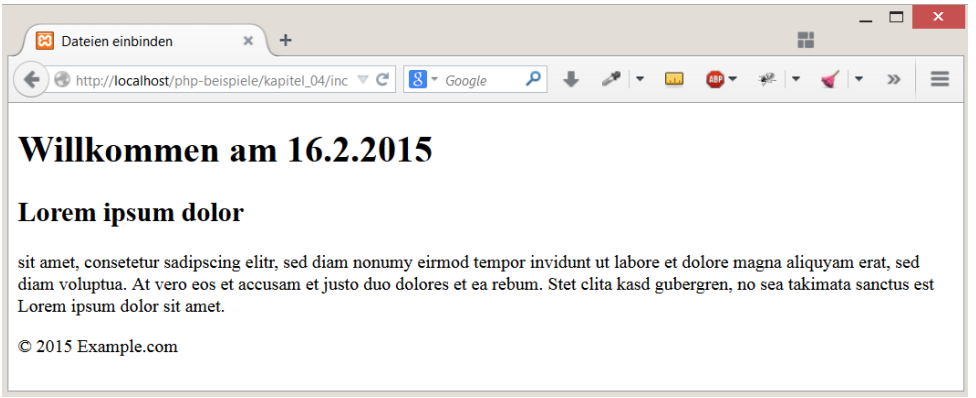


Abb. 4–21 *Die Ausgabe des Dokuments mit den zwei eingebundenen Dateien*

Wie Sie gesehen haben, können Sie mit `include` Dateien einbinden, die nur HTML-Code enthalten, aber Sie können auch in den eingebundenen Dateien PHP-Befehle schreiben. Wenn Sie PHP-Code einbinden wollen, müssen Sie in der eingebundenen Datei dann aber den Code auch mit `<?php` einleiten und mit `?>` beenden, wie Sie in der Datei *header.php* sehen. Das schließende `?>` könnten Sie allerdings auch weglassen.

Neben `include` gibt es `require`, das prinzipiell genauso funktioniert:

```
require "header.php";
```

Der Unterschied zwischen `require` und `include` zeigt sich nur, wenn die angegebene Datei *nicht* geladen werden kann. In beiden Fällen wird eine Warnung ausgegeben, aber bei `require` zusätzlich noch ein fataler Fehler, und die Abarbeitung des Skripts wird abgebrochen.

Wie bereits erwähnt, sollte die Ausgabe der Fehlermeldungen beim echten Einsatz der Skripte unterbunden werden. Und dann wird der Unterschied zwischen `include` und `require` sehr deutlich: Bei `include` wird der restliche Inhalt der Seite normal angezeigt, bei `require` hingegen nicht. Das heißt, `require` verwenden Sie zur Einbindung von essenziellem Code, ohne den der Rest der Verarbeitung nicht mehr sinnvoll ist. `include` benutzen Sie hingegen für Fälle wie im Beispiel. Hier wäre es sinnvoll, die Seite trotzdem ausgeben zu lassen, auch wenn z. B. die Copyright-Information fehlt.

Eine Einstellung, die für `include` und `require` relevant ist, ist der sogenannte `include-path`. Dieser sagt dem Skript, wo es nach eingebundenen Dateien nachsehen

soll. Standardmäßig sind hier ein Punkt und der Pfad zu PEAR angegeben. Worauf der `include-path` gesetzt ist, sehen Sie wieder in der Ausgabe von `phpinfo()`.

include_path	.;D:\xampp\php\PEAR	.;D:\xampp\php\PEAR

Abb. 4–22 *Einstellung für den include_path bei XAMPP unter Windows*

Bei XAMPP unter Windows steht hier beispielsweise: `.;LAUFWERK:\xampp\php\pear\`. Der Punkt am Anfang steht für das aktuelle Verzeichnis, danach kommt der Strichpunkt als Trennzeichen für die Angabe von mehreren Verzeichnissen und noch der Pfad `LAUFWERK:\xampp\php\pear`. Das bedeutet: Wird `include` oder `require` eingesetzt, wird zuerst nach der entsprechenden Datei ausgehend vom aktuellen Verzeichnis nachgesehen. Falls sie hier nicht gefunden wird, geht die Suche im Verzeichnis *LAUFWERK:\xampp\php\pear* weiter.

Unter Linux/Unix wird als Trennzeichen für mehrere Pfadangaben nicht der Strichpunkt, sondern der Doppelpunkt eingesetzt.

Wenn Sie mit mehreren verschachtelten Includes in Unterverzeichnissen arbeiten, sollten Sie absolute Pfade verwenden. Benutzen Sie dann:

```
include __DIR__ . "/pfad/zur/include/datei";
```

Die Konstante `__DIR__` steht erst ab PHP 5.3 zur Verfügung. Vorher mussten Sie Folgendes schreiben:

```
include dirname( __FILE__ ) . '/pfad/zur/include/datei;
```

Übung 8

- Definieren Sie ein Array mit den Namen der Übungsdateien dieses Kapitels.
- Lassen Sie das Array mit einer `foreach`-Schleife ausgeben. Modifizieren Sie dann die Ausgabe so, dass die Dateinamen nicht nur erscheinen, sondern zu anklickbaren Links werden.
- Zur Erinnerung: Einen Link erstellen Sie in HTML etwa über

  ```
  <a href='arrays.php'> arrays.php </a>.
  ```

- Erstellen Sie ein weiteres Dokument, das nur eine `h1`-Überschrift enthält, beispielsweise mit dem Text »PHP-Übungen«.
- Unter dieser Überschrift sollen die Links per `include` eingebunden werden.
- Sehen Sie sich auf jeden Fall den erzeugten HTML-Code an. Es ist wichtig, dass Sie in dieser Ausgabe nicht zwei ineinander verschachtelte HTML-Strukturen haben. Es darf also `<html><head> . ..</head><body> ..</body></html>` nur einmal vorkommen! (s. Abb. 4–23)

→

Übung 8

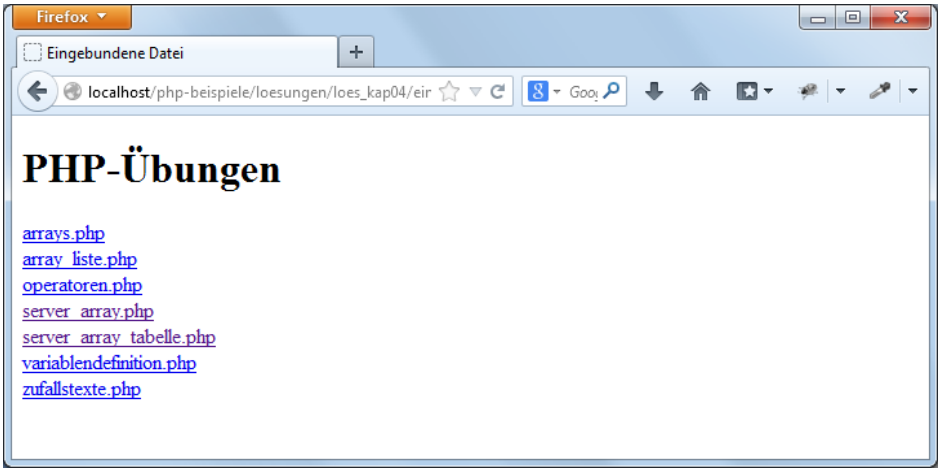


Abb. 4–23 *Ein Dokument mit Links zu den Übungen dieses Kapitels*

4.9 Zusammenfassung

Das Kapitel hat Ihnen wichtige Basics zu PHP vermittelt. Sie haben erfahren, dass Sie den PHP-Code innerhalb von `<?php` und `?>` in Ihr Dokument einbinden. Außerdem haben Sie gesehen, wie Sie mit Variablen arbeiten, die in PHP immer mit einem Dollarzeichen beginnen. Ein weiteres Thema war die Variableninterpolation, d.h., dass innerhalb von doppelten Anführungszeichen der Wert von Variablen ausgegeben wird. Schließlich haben Sie unterschiedliche Datentypen kennengelernt, wie Strings, Integer, Float und boolesche Werte. Ausführlicher haben wir uns mit Arrays beschäftigt – der Möglichkeit, mehrere Werte unter einem Namen anzusprechen. Arrays können Sie mit `foreach`-Schleifen durchlaufen, und über `count()` lässt sich die Anzahl der Elemente in einem Array ermitteln. Schließlich haben Sie noch `include` und `require` kennengelernt, die praktisch sind, um externe Dateien einzubinden.

Mit `foreach` haben Sie eine erste Schleife kennengelernt – um weitere Schleifen geht es im nächsten Kapitel, das Ihnen mehr wichtige PHP-Basics vermittelt.

Inhaltsübersicht

Florence Maurice gibt Trainings, Inhouseschulungen und individuelle Coachings zu Webthemen, setzt eigene Webprojekte um und schreibt regelmäßig Artikel in Fachzeitschriften. Sie ist Autorin mehrerer Fachbücher zu CSS, PHP und MySQL sowie mobilem Webdesign.

Florence Maurice

PHP 5.6 und MySQL 5.7

Ihr praktischer Einstieg in die Programmierung dynamischer Websites

4., aktualisierte und erweiterte Auflage

dpunkt.verlag

Florence Maurice
florence@maurice-web.de

Lektorat: René Schönfeldt
Copy-Editing: Friederike Daenecke, Zülpich
Herstellung: Birgit Bäuerlein
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

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

ISBN
Buch 978-3-86490-281-9
PDF 978-3-86491-675-5
ePub 978-3-86491-676-2

4., aktualisierte und erweiterte Auflage 2015

Wieblinger Weg 17
69123 Heidelberg

5 4 3 2 1 0

Vorwort

PHP ist eine äußerst beliebte Skriptsprache zur serverseitigen Programmierung. Mit PHP können Sie sogenannte dynamische Seiten erstellen. Das sind Seiten, die jedes Mal, wenn sie aufgerufen werden, neu, d.h. meist mit aktuellen Daten, erzeugt werden. Besonders beliebt ist PHP in der Kombination mit dem Datenbanksystem MySQL, da beide kostenlos zur Verfügung stehen. Mit PHP können Blogs, Content-Management-Systeme, Shop-Systeme, Foren, Bildergalerien usw. programmiert werden. Die Abkürzung PHP selbst steht für *PHP Hypertext Preprocessor*.

PHP hat viele Vorteile:

- Es ist speziell für dynamische Webseiten entwickelt worden – das bedeutet, alle Funktionen sind genau darauf zugeschnitten.
- Es ist relativ einfach zu erlernen ...
- ... und trotzdem ausgereift: PHP liegt derzeit in Version 5.6 vor.
- PHP kann sowohl prozedural als auch objektorientiert programmiert werden und ist damit auch für den Einsatz bei größeren Projekten geeignet.
- PHP ist eine äußerst mächtige Skriptsprache. Sie ermöglicht das Arbeiten mit Datenbanken sowie mit Dateien, aber auch vieles mehr, wie etwa die Erstellung von PDFs oder Bildbearbeitung. Letzteres ist praktisch, um beispielsweise Vorschaubilder automatisch zu erzeugen oder dynamische Diagramme, basierend auf aktuellen Umfragewerten, ausgeben zu lassen.
- Alles, was Sie zur Arbeit mit PHP brauchen, steht frei zur Verfügung. In diesem Buch werden Sie erfahren, wie Sie sich Ihre Entwicklungsumgebung mit wenigen Mausklicks einrichten.
- Webhosting-Angebote mit PHP-Unterstützung sind inzwischen relativ preiswert.
- PHP ist weit verbreitet. Das bedeutet: Im Internet finden Sie auch bei spezielleren Fragen Hilfe, und es gibt auch für ausgefallenere Anforderungen Lösungen.
- Viele bekannte Open-Source-Anwendungen wie das Blogsystem WordPress oder die Content-Management-Systeme Joomla!, Drupal und TYPO3 basieren auf PHP.

- PHP ist für große Anwendungen wie Content-Management-Systeme geeignet, aber auch für kleine: Wenn Sie die Daten aus dem Kontakt-Formular selbst verarbeiten, überprüfen und sich per Mail zusenden lassen wollen, ist PHP ebenfalls die richtige Wahl.
- Für große Projekte gibt es inzwischen mehrere Frameworks, die auf PHP aufsetzen.
- PHP macht Spaß!

Sie sehen, es sprechen viele Gründe dafür, PHP zu lernen.

Das sollten Sie schon können

Welche Vorkenntnisse brauchen Sie, wenn Sie PHP lernen möchten? Mit PHP erstellen Sie dynamisch HTML-Seiten, die dann an den Browser ausgeliefert werden (Genaueres dazu in Kap. 1). Deswegen sollten Sie über grundlegende HTML-Kenntnisse verfügen. Einen Crashkurs dazu gibt Ihnen Kapitel 3, aber falls Sie noch keine HTML-Seite erstellt haben, sollten Sie für die Einarbeitung in HTML zusätzliche Zeit einplanen und sich noch mit einem speziellen HTML-Buch eindecken.

Und grundsätzlich sollten Sie natürlich Lust haben, sich in die Welt der Programmierung hineinzudenken.

Vorweg: Das behandelt das Buch

Um dynamische Webseiten zu erstellen, bei denen die Inhalte aus einer Datenbank stammen, brauchen Sie PHP für die Programmierung und MySQL für die Datenbankoperationen. Wie das alles funktioniert, lernen Sie in diesem Buch.

Der Schwerpunkt des Buches liegt dabei auf der Programmierung mit PHP – Sie lernen alle wichtigen Techniken im Zusammenhang mit PHP kennen – von den Basics der Sprache über nützliche Funktionen bis zur Verarbeitung von Formularen und der Arbeit mit Sessions und Cookies. Das sind wichtige Techniken gerade auch im Zusammenhang mit MySQL: So können beispielsweise die in ein Formular eingetragenen Daten mit PHP entgegengenommen und in einer MySQL-Datenbank gespeichert werden. Auch weiterführende Techniken sind ein wichtiges Thema des Buchs: Sie finden einen Einstieg in die objektorientierte Programmierung und erfahren, wie Sie mit PHP Dateien bearbeiten, PDFs erstellen und Bilder erzeugen. Außerdem befassen wir uns mit einem Template-System (Smarty) und dem attraktiven PHP-Framework Laravel, und Sie lernen, wie PHP in WordPress bei der Arbeit mit Child-Themes verwendet wird und wie das Zusammenspiel mit dem beliebten jQuery funktioniert.

Das Buch, das Sie in den Händen halten, ist die vierte vollständig überarbeitete und aktualisierte Fassung meines ursprünglich bei Addison-Wesley erschienenen Buches. Behandelt werden jetzt alle neuen Features von PHP 5.6 wie der Operator zum Potenzieren oder der Splat-Operator für variadische Funktionen. Neu hinzugekommen ist ein Unterkapitel zur Installation von MAMP für Mac OS. Außerdem gibt es ein eigenes Kapitel zum Thema PHP-Frameworks anhand eines Einstiegs in das

PHP-Framework Laravel – denn in der Praxis werden Sie häufig bei größeren Projekten mit Frameworks zu tun haben.

PHP-Versionen im Blick

Regelmäßig erscheinen neue Versionen von PHP, so wie es etwa auch bei Microsoft Office immer wieder aktualisierte Versionen gibt. Die Änderungen bei neuen Office-Versionen sind oft weitreichend; es kann sein, dass sich einzelne Menüpunkte nun an einer ganz anderen Stelle befinden. So etwas kann Ihnen mit PHP nicht passieren, denn die grundlegenden Dinge verändern sich bei kleineren Versionssprüngen nicht. Es kommen aber natürlich einzelne neue Features hinzu, andere Features werden vielleicht als unerwünscht gekennzeichnet (ein Hinweis, dass man sie nicht mehr verwenden sollte, weil sie in einer späteren Version eventuell entfernt werden).

Im Buch erfahren Sie das Wichtigste der aktuellen Version; wenn ein Feature erst vor ein paar Versionen hinzugekommen ist, wird das eigens vermerkt. Das ist eine wichtige Information, denn die klassischen Hostingangebote aktualisieren oft nicht direkt auf die neueste Version. Es kann Ihnen durchaus passieren, dass bei Ihrem Hoster noch PHP 5.3x installiert ist, obwohl längst PHP 5.6x aktuell ist. Deswegen kann es sinnvoll sein, nicht direkt die neuesten Features einer neuen Version zu nutzen – aber es ist wichtig zu wissen, wohin der Trend geht, um beispielsweise schon vorab auf den Einsatz von als veraltet gekennzeichneten Features zu verzichten.

Ausführliche Übersicht über die Kapitel

In Kapitel 1 geht es erst einmal um die Grundlagen von PHP – Sie erfahren, was der Unterschied zwischen statischen HTML-Seiten und dynamisch per PHP erzeugten Seiten ist. Kapitel 2 zeigt Ihnen, wie Sie auf Ihrem Computer eine Entwicklungsumgebung installieren. Außerdem sehen Sie am Beispiel, wie Sie PHP konfigurieren. Kapitel 3 vermittelt Ihnen im Schnelldurchlauf die wichtigsten HTML/CSS-Basics.

> In diesem wie auch in den weiteren Kapiteln (mit Ausnahme der späteren) finden Sie immer *kleine Übungen*, um das Gelesene selbst auszuprobieren und zu testen. Die Lösungen dazu stehen im Anhang und bei den Listings zu diesem Buch.

In Kapitel 4 geht es um die Sprachelemente von PHP: Sie erfahren, wie Sie PHP in HTML-Dateien einbetten und welche Datentypen und Operatoren es gibt. Ebenfalls befassen wir uns damit, wie Sie immer wieder vorkommende Bestandteile von Webseiten zentral erstellen und mit PHP einfügen können – eine sehr nützliche Technik. Außerdem lernen Sie Arrays kennen, und zwar anhand eines Beispiels, bei dem zufällig eines von mehreren Bildern angezeigt wird. Kapitel 5 führt weitere wichtige Sprachelemente ein – Sie erfahren, wie man Programme mit Bedingungen und Schleifen flexibel gestaltet und Funktionen erstellen kann. In Kapitel 6 sehen Sie wichtige fertige Funktionen, die Ihnen PHP zur Verfügung stellt: Mit diesen lassen sich Texte auf

jede erdenkliche Art bearbeiten oder Arrays manipulieren. Einige Funktionen sind auch speziell für die Arbeit mit Datum und Uhrzeit gedacht.

Möchten Sie mit Ihren Benutzern kommunizieren, bieten sich dafür Formulare an. Kapitel 7 vermittelt Ihnen die wichtigsten Techniken zu Formularen, und Sie erfahren auch, wie – und warum – Sie diese absichern müssen. Außerdem sehen Sie, wie Sie mit PHP Mails versenden, und erfahren am Beispiel, wie sich ein Bild-Upload per Formular realisieren lässt.

Cookies und Sessions sind eine weitere zentrale Webtechnologie: Mit Cookies und Sessions können Sie Zustände speichern, was Sie beispielsweise brauchen, um Warenkörbe zu realisieren. Den Details zu Cookies und Sessions widmet sich Kapitel 8; außerdem erhalten Sie einen Einblick in die Erzeugung von Passwort-Hashs mit den neuen Funktionen aus PHP 5.5.

Durch die objektorientierte Programmierung lassen sich Programme besser warten und einzelne Komponenten leichter wiederverwenden. Kapitel 9 widmet sich detailliert der Objektorientierung und zeigt auch fortgeschrittene Möglichkeiten auf, wie Namespaces aus PHP 5.3 und Traits aus PHP 5.4.

Wenn Sie mit umfangreichen Datenmengen arbeiten, diese verändern und auslesen möchten, so empfiehlt sich der Einsatz einer Datenbank. Kapitel 10 liefert Ihnen die wichtigsten MySQL-Grundlagen. Sie werden mit phpMyAdmin vertraut gemacht und lernen zudem, die wichtigsten MySQL-Befehle selbst zu schreiben. Das brauchen Sie dann in Kapitel 11, wenn es darum geht, per PHP auf MySQL-Datenbanken zuzugreifen.

Nicht immer sind die Daten, die man bearbeiten möchte, in einer Datenbank gespeichert, manchmal liegen sie auch in Textdateien vor. Kapitel 12 zeigt Ihnen, wie sich Sie Inhalte aus Textdateien auslesen und per PHP in Textdateien schreiben können. Im Weiteren sehen Sie, wie Sie einfach über die Schnittstelle simpleXML auf XML-Dateien zugreifen können, um beispielsweise Newsfeeds von anderen Seiten in Ihre Seite zu integrieren. Zudem befassen wir uns mit den in PHP 5.3 neu eingeführten Phar-Archiven und der Erzeugung von PDF-Dateien.

PHP kann mehr, als Texte bearbeiten – Sie können mit PHP auch dynamisch Grafiken erzeugen oder vorhandene Bilder bearbeiten. Wie das geht, sehen Sie in Kapitel 13 anhand von zwei Beispielen: Sie erfahren, wie Sie automatisch kleine Vorschaubilder von größeren Bildern erstellen lassen und wie Sie Diagramme dynamisch realisieren.

Bisher wurden immer der HTML- und der PHP-Code gemischt. Um diese zu trennen, gibt es sogenannte Template-Systeme. Ein Beispiel für ein Template-System – Smarty – lernen Sie in Kapitel 14 kennen.

Die letzten drei Kapitel gehen etwas über PHP pur hinaus: In Kapitel 15 lernen Sie das äußerst mächtige und beliebte PHP-Framework Laravel kennen. Kapitel 16 demonstriert, wie man in WordPress PHP bei der Erstellung eines Child Themes einsetzt. Schließlich zeigt Kapitel 17, wie die beliebte JavaScript-Bibliothek jQuery funktioniert und wie jQuery und PHP zusammenarbeiten.

Den Abschluss bildet der Anhang mit Lösungen zu den Übungen und mit Informationen zu Möglichkeiten, PHP zu konfigurieren, sowie mit einem Einblick ins Debugging von PHP-Skripten.

Den gesamten Code der Listings können Sie auf der Website zu diesem Buch unter www.dpunkt.de/php56 herunterladen.

Damit wissen Sie alles Wichtige zum Buch und können mit PHP loslegen. Ich wünsche Ihnen viel Spaß dabei!