Jan Teriete

Grundlagen der PHP7-Programmierung

Ein Webmasters Press Lernbuch

Version 7.2.0 vom 23.04.2019

Autorisiertes Curriculum für das Webmasters Europe Ausbildungs- und Zertifizierungsprogramm

— www.webmasters-europe.org —

Inhaltsverzeichnis

3 HTML und PHP

In dieser Lektion lernen Sie

- wie Sie PHP in Ihre HTML-Dateien einbetten können.
- was Sie dabei beachten müssen.
- die sechs Fehler kennen, die jeder PHP-Anfänger macht.
- die `echo`-Anweisung kennen.

3.1 Das Problem

Wenn Sie die letzte Lektion durchgelesen haben, wissen Sie nun, wie eine dynamische Webseite prinzipiell funktioniert. Jetzt wird es jedoch Zeit, dass Sie selbst eine PHP-Seite bauen, daher lassen Sie uns auch gleich loslegen. Sie haben inzwischen doch den Webserver und den Code-Editor installiert, wie es in Lektion 1 verlangt war, oder?

3.2 PHP in eine HTML-Seite einbetten

3.2.1 Eine statische HTML-Seite erstellen

Öffnen Sie Ihren bevorzugten Code-Editor und erstellen Sie nachfolgendes HTML-Dokument. Speichern Sie es im ***document root*** Ihres Webservers im Unterordner *phpschulung* unter dem Namen *erste_seite.php* ab. Wenn Sie beispielsweise eine Standardinstallation von XAMPP (siehe Lektion 23) durchgeführt haben, ist Ihr ***document root*** das Verzeichnis *c:\xampp\htdocs*. Bei MAMP unter Mac OS X ist der Standard */Applications/MAMP/htdocs* und auf Linux-Servern finden Sie dieses Verzeichnis oft unter */var/www*. Vergessen Sie nicht die korrekte Dateiendung *.php* zu benutzen, diese ist sehr wichtig.

Beispiel

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="utf-8" />
6      <title>Hallo Welt</title>
7  </head>
8
9  <body>
10     Hallo Welt
11 </body>
12
13 </html>
```

Codebeispiel 1 *erste_seite.php (Version 1)*

Rufen Sie die Datei nun in Ihrem Browser unter der URL `http://localhost/phpschulung/erste_seite.php` auf.

3.2.2 PHP-Code einfügen

So weit sollte das für Sie nichts Neues gewesen sein. Das Besondere an PHP ist allerdings, dass Sie es, ähnlich wie JavaScript, direkt in Ihre HTML-Dokumente einfügen können.

Um dem Webserver zu zeigen, wo PHP-Code in HTML-Dokumenten steht, wurde ein spezieller Tag entwickelt, der jeden PHP-Code umgeben muss. Kein PHP-Code darf außerhalb dieser Begrenzung stehen, oder der Webserver hält ihn für normales HTML und zeigt ihn einfach an.

Beispiel

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="utf-8" />
6      <title><?php echo "Hallo Welt"; ?></title>
7  </head>
8
9  <body>
10     <?php echo "Hallo Welt"; ?>
11 </body>
12
13 </html>
```

Codebeispiel 2 *erste_seite.php (Version 2)*

Was hat sich hier geändert? Wenn Sie sich die geänderte Seite in Ihren Browser ansehen, werden Sie keinen Unterschied feststellen. Die Seite hat immer noch den Titel `Hallo Welt` und gibt im `body`-Tag denselben Text noch einmal aus.

Der Unterschied ist, dass der Text `Hallo Welt` nun mittels PHP in die Seite geschrieben wurde. Wann immer Sie PHP-Code einfügen wollen, müssen Sie zuerst `<?php` schreiben, um zu zeigen, dass nun PHP-Code folgt. Wenn Sie mit dem PHP-Block fertig sind, zeigen Sie das durch den schließenden Tag `?>`. Um die Bedeutung des eigentlichen PHP-Codes müssen Sie sich im Moment noch nicht kümmern.[4]

Jeder PHP-Code muss zwischen `<?php` und dem schließenden Tag `?>` stehen. Ansonsten wird er nicht ausgewertet, sondern unverändert angezeigt. Dies sind die sogenannten ***PHP-Tags***, analog zu den bekannten HTML-Tags.

3.2.3 PHP erzeugt HTML

Wenn Sie sich den Quelltext der HTML-Seite anzeigen lassen (z. B. im *Firefox*-Browser mit Strg+U), werden Sie feststellen, dass dort nichts mehr von dem PHP-Code zu sehen ist. Bei genauerer Betrachtung werden Sie sogar erkennen, dass die erzeugte Seite mit der ersten, statischen Version aus Abschnitt 3.2.1 identisch ist. Vermutlich sind nicht alle Zeilenumbrüche da, wo Sie es erwarten. Zum Beispiel könnte der schließende body-Tag hinter das "Hallo Welt" gerutscht sein. Doch dies soll uns nicht weiter stören.

Exakt das ist damit gemeint, wenn man von PHP als einer serverseitigen Programmiersprache spricht. Der Webserver hat die PHP-Datei geöffnet, den PHP-Code durch den Interpreter auswerten lassen und, was als Ergebnis übrig blieb, an den Browser gesendet. Der Browser sieht nichts mehr von den ursprünglichen PHP-Tags und das ist auch gut so, denn Webbrowser haben keine Ahnung, was PHP überhaupt ist!

JavaScript wird vom Webbrowser ausgewertet. Also ist JavaScript-Code für den Browser sichtbar. PHP wird schon auf dem Webserver ausgewertet, lange bevor der Browser die Seite erhält. Also sollte ein Webbrowser niemals PHP-Code sehen.

4. Er sorgt dafür, dass der Text in den Anführungszeichen in der Webseite angezeigt wird.

3.3 Die sieben berühmten Fehler

Wenn Sie den Beispielen aus dem vorherigen Abschnitt problemlos folgen konnten, gratulieren Sie sich. Sie haben die Beispiele fehlerfrei abgetippt. Wenn Sie hingegen seltsame Meldungen oder schlimmer, gar nichts in Ihrem Browser gesehen haben, sind Sie wahrscheinlich in einen der sechs Fehler gelaufen, die jeder PHP-Anfänger macht. Sollte es Sie erwischt haben, grämen Sie sich nicht. Das bedeutet nur, dass Sie es schon hinter sich haben! Jeder PHP-Anfänger macht diese Fehler!

3.3.1 PHP-Code im HTML-Bereich schreiben

Einer der beliebtesten Fehler ist es, PHP-Code außerhalb des durch `<?php ?>` markierten Bereiches zu schreiben. Das bedeutet, dass dieser Code für HTML gehalten und einfach an den Browser geschickt wird.

Beispiel

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="utf-8" />
6      <title><?php echo "Hallo Welt"; ?></title>
7  </head>
8
9  <body>
10     echo "Hallo Welt";
11 </body>
12
13 </html>
```

Codebeispiel 3 *php_im_html.php*

Die erste `echo`-Anweisung in Zeile 6 wird korrekt ausgewertet. Im Browser steht an dieser Stelle nur noch `Hallo Welt`. In Zeile 10 jedoch haben wir vergessen, den PHP-Code in den passenden Tags zu verbergen. Das Ergebnis ist, dass der Browser an dieser Stelle das komplette `echo "Hallo Welt";` ausgibt.

3.3.2 HTML-Code im PHP-Bereich schreiben

Das Gegenteil des letzten Fehlers ist genauso verhängnisvoll. Wenn Sie HTML-Tags in den PHP-Bereich schreiben, versucht der PHP-Interpreter, aus den für ihn seltsamen Zeichen schlau zu werden und gibt schließlich mit einer Fehlermeldung auf.

Beispiel

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <?php <meta charset="utf-8" /> ?>
6      <title><?php echo "Hallo Welt"; ?></title>
7  </head>
8
9  <body>
10     <?php echo "Hallo Welt"; ?>
11 </body>
12
13 </html>
```

Codebeispiel 4 *html_im_php.php*

Jetzt liegt der Fehler in Zeile 5. Dort wird zwar ein PHP-Tag geöffnet, aber sein Inhalt besteht aus purem HTML. Bestraft wird dieses Verhalten von PHP mit einer hässlichen Fehlermeldung. Wenn Sie die Seite aufrufen, sehen Sie folgenden Text:

```
Parse error: syntax error, unexpected '<' in (...)\html_im_php.php on line 5
```

Genauso wenig, wie der Browser Ahnung von PHP hat, weiß der PHP-Interpreter etwas mit den HTML-Formatierungen anzufangen. Dementsprechend sagt er uns auch, dass er mit dem unerwarteten (unexpected) Zeichen `'<'` nichts anfangen kann.

3.3.3 Die PHP-Datei hat die Endung .html

Selbst wenn Sie das ursprüngliche Beispiel fehlerfrei abgetippt haben, kann es dennoch sein, dass Sie den kompletten Code mit allen PHP-Tags im Browser sehen. Das kann mehrere Gründe haben. Der häufigste ist jedoch, dass Sie die Datei mit der Endung `.html` versehen haben.

Wenn eine Datei, die PHP-Code enthält, nicht auf `.php` endet, weiß der Webserver für gewöhnlich nicht, dass sich PHP darin befindet, und schickt die Datei unverändert an den Browser.

Welche Dateiendungen der Webserver als PHP-Dateien ansieht, kann natürlich konfiguriert werden. Häufig funktionieren neben *.php* auch versionsspezifische Endungen wie beispielsweise *.php55*. Solange Sie keinen triftigen Grund haben, etwas anderes zu tun, sollten Sie immer die Endung *.php* verwenden, da diese immer funktioniert und Sie auf die genaue Konfiguration bei Ihrem Webhoster oft keinen Einfluss haben.

3.3.4 Die PHP-Datei liegt nicht im Zugriff des Webservers

Ein Webserver darf nur auf Dateien in einem vorher festgelegten Verzeichnis, genannt ***document root***, zugreifen. Wenn Sie Ihre PHP-Dateien außerhalb dieses Verzeichnisses speichern, wird der Webserver sie nicht finden und einen Fehler melden.

Dies ist ein, gerade von PHP-Anfängern, häufig gemachter Fehler, da die meisten Windows-Anwender es beispielsweise gewohnt sind, Dateien in *Eigene Dateien* oder auf dem *Desktop* abzulegen, wo der Webserver sie nicht finden kann.

> Wenn Sie häufiger in diese Falle tappen, legen Sie sich eine Verknüpfung zum ***document root*** auf Ihrem *Desktop* an. Dadurch sieht es so aus, als würden Sie die Dateien direkt in einem Ordner auf dem *Desktop* ablegen, und es funktioniert trotzdem.

3.3.5 Die PHP-Datei wurde direkt im Browser geöffnet

Kommen wir nun zum wahrscheinlich häufigsten Anfängerfehler. Anstatt die PHP-Datei über den Webserver aufzurufen, haben Sie die Datei direkt vom Dateisystem aus geöffnet. Das bedeutet, Ihr Webbrowser erhält direkt den PHP-Code und hat natürlich keine Ahnung, was er damit anfangen soll. Moderne Browser blenden meist alles zwischen `<?php ... ?>` einfach aus, aber gerade ältere Browser haben den Inhalt direkt angezeigt.

Sie merken am besten an der URL-Zeile im *Firefox*-Browser, dass Sie die Datei direkt geöffnet haben. Wenn Sie dort zu Anfang **file://** lesen, sind Sie in die Falle getappt. Gerade am Anfang Ihrer PHP-Karriere sollte der erste Blick der URL gelten, wenn etwas nicht so funktioniert, wie Sie es erwarten. In anderen Browsern wie beispielsweise Chrome wird eine PHP-Datei in einem solchen Fall übrigens gar nicht geöffnet, sondern zum Download angeboten.

3.3.6 Die echo-Anweisung fehlt

In den vorherigen Abschnitten haben Sie in die PHP-Tags das Wort `echo`, gefolgt von Text in Anführungszeichen, geschrieben. Das Ergebnis war, dass das Wort `echo` und die Anführungszeichen verschwunden sind und nur der Text selbst im Browser angekommen ist.

Dieses Wort `echo` ist für PHP eine Anweisung und veranlasst den PHP-Interpreter, den folgenden Text auszugeben. Im Gegensatz zu HTML, wo Sie jegliche Ausgabe einfach in die Datei schreiben, müssen Sie PHP ausdrücklich sagen, wenn Sie etwas auf dem Bildschirm ausgeben wollen. Für diesen Zweck gibt es in PHP eine ganze Reihe von Anweisungen, für den Anfang soll uns `echo` allerdings genügen, da Sie mit dieser Anweisung fast jede Situation meistern können.

Beispiel

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="utf-8" />
6      <title><?php echo "Hallo Welt"; ?></title>
7  </head>
8
9  <body>
10     <?php "Hallo Welt"; ?>
11 </body>
12
13 </html>
```

Codebeispiel 5 *echo_fehlt.php*

Fehlt die `echo`-Anweisung, so hat der PHP-Interpreter keine Ahnung, was er mit dem Text in Anführungszeichen anfangen soll, und dieser verfällt nach Ausführung von Zeile 10 ungenutzt. In diesem Fall erhalten wir somit eine leere Seite, da das `body`-Tag keinen weiteren Inhalt hat.

3.3.7 Das Leerzeichen wurde vergessen

Fehlt zwischen dem öffnenden Tag `<?php` und der nachfolgenden Anweisung das trennende Leerzeichen, so kann der PHP-Interpreter den PHP-Code nicht mehr erkennen.

Beispiel

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="utf-8" />
6      <title><?php echo "Hallo Welt"; ?></title>
7  </head>
8
9  <body>
10     <?phpecho "Hallo Welt"; ?>
11 </body>
12
13 </html>
```

Codebeispiel 6 *leerzeichen_fehlt.php*

Im Browser bekommen wir beim Aufruf dieser PHP-Datei eine leere Seite angezeigt. Es gibt auch keine Fehlermeldung des PHP-Interpreters, da dieser den PHP-Code ja gar nicht erkannt hat. Schaut man sich jedoch den Quelltext der HTML-Seite im Browser an, so kann man das Problem an dem im Quelltext enthaltenen PHP-Code erkennen. Eine PHP-IDE würde die fehlerhafte Zeile 10 übrigens sofort im Code markieren.

3.4 Mehrzeilige PHP-Anweisungen

3.4.1 Das Semikolon

Bisher haben Sie in jedem Beispiel am Ende einer PHP-Anweisung ein ***Semikolon***, zu Deutsch Strichpunkt, gesehen. Sollten Sie eines vergessen haben, werden Sie keinen Unterschied bemerkt haben. Der PHP-Code hat weiterhin genau dasselbe getan.

Wofür brauchen wir dieses Semikolon dann überhaupt? Es ist für die Trennung einzelner PHP-Anweisungen zuständig. Wenn Sie also mehr als eine Anweisung für PHP haben, müssen Sie dazwischen ein Semikolon setzen.

Beispiel

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="utf-8" />
6      <title><?php echo "Hallo Welt"; ?></title>
7  </head>
8
9  <body>
10     <?php echo "Hallo Welt "; echo "und willkommen beim PHP-Seminar!"; ?>
11     Das ist normales HTML.
12     <?php echo "Das ist korrekt." echo "Das nicht mehr!"; ?>
13 </body>
14
15 </html>
```

Codebeispiel 7 *mehrere_php_anweisungen.php*

Der Unterschied zwischen den Zeilen 10 und 12 besteht darin, dass in Zeile 12 vor dem zweiten `echo` kein Semikolon steht. PHP denkt also, hier stünde nur eine einzige lange Anweisung anstatt zwei. Da eine Anweisung mit zwei `echo`-Aufrufen für PHP kein gültiger Code ist, wird hier ein Fehler erzeugt:

```
Parse error: syntax error, unexpected 'echo' (T_ECHO), expecting ',' or ';' in
(...)\mehrere_php_anweisungen.php on line 12
```

PHP sagt uns hier, dass in Zeile 12 entweder ein Komma oder ein Semikolon erwartet wird. Tragen Sie also das in diesem Fall fehlende Semikolon nach, damit der Code funktioniert.

3.4.2 Mehrzeilige PHP-Anweisungen

Auch wenn es funktioniert, mehrere PHP-Anweisungen hintereinander zu schreiben, so ist es nicht wirklich übersichtlich. Vergleichen Sie im folgenden Listing die beiden PHP-Blöcke:

Beispiel

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="utf-8" />
6      <title>Hallo Welt</title>
7  </head>
8
9  <body>
10     <?php echo "Hallo Welt "; echo "und willkommen beim PHP-Seminar! "; echo
"Langsam wir es unübersichtlich. "; echo "Jetzt ist aber wirklich Schluss!"; ?>
```

```
11      <br />
12      <?php
13
14      echo "So sieht das schon viel besser aus. ";
15      echo "Wenn Sie jede Anweisung auf eine Zeile schreiben, ";
16      echo "behalten Sie leichter den Überblick. ";
17      echo "Trotzdem müssen Sie immer noch ";
18      echo "ein Semikolon nach jeder Anweisung schreiben, ";
19      echo "da PHP eine neue Zeile nicht als das Ende ";
20      echo "einer Anweisung sieht.";
21
22      ?>
23  </body>
24
25  </html>
```

Codebeispiel 8 *mehrzeiliges_php.php*

Der erste Abschnitt in Zeile 10 ist, obwohl PHP ihn versteht, nicht sehr übersichtlich. Sie müssen gezielt nach den Semikolons suchen, um den Beginn einer neuen Anweisung zu sehen. Der zweite Abschnitt in den Zeilen 12 bis 22 bringt jede Anweisung auf eine eigene Zeile, was es Ihnen leichter macht, den Code zu verstehen. Für PHP macht es allerdings keinen Unterschied.

Schreiben Sie jede Anweisung auf eine eigene Zeile, um die Lesbarkeit des Codes zu erhöhen. Wenn Sie mehrzeilige PHP-Anweisungen haben, schreiben Sie den öffnenden und den schließenden PHP-Tag auf jeweils eigene Zeilen.

3.5 Stilfragen

3.5.1 Saubere Trennung von HTML und PHP

In diesem Abschnitt werden Sie eines der wenigen Codebeispiele in diesem Lernbuch sehen, das eindeutig schlechten Stil darstellt. Ich bemühe mich natürlich, Ihnen nur Dinge beizubringen, die Sie später auch so benutzen sollten. Manchmal müssen Sie dafür aber auch Beispiele sehen, wie Sie es nicht machen sollten. Hier ist das erste:

Beispiel

```
1   <!DOCTYPE html>
2   <html>
3
4   <head>
5       <meta charset="utf-8" />
6       <title>Hallo Welt</title>
7   </head>
8
9   <body>
10      <?php
11
12      echo '<form action="test.php" method="post">';
13      echo '<input type="text" name="vorname" id="vorname" value="Arthur" />';
14      echo '<input type="text" name="nachname" id="nachname" value="Dent" />';
15      echo '<input type="submit" value="abschicken" />';
16      echo '</form>';
17
18      ?>
19  </body>
20
21  </html>
```

Codebeispiel 9 *stillos.php*

Vergleichen Sie das Beipiel eben mit dem folgenden, das exakt die gleiche Ausgabe erzeugt:

Beispiel

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <title>Hallo Welt</title>
</head>

<body>
    <form action="test.php" method="post">
        <input type="text" name="vorname" id="vorname" value="<?php echo 'Arthur';
?>" />
        <input type="text" name="nachname" id="nachname" value="<?php echo 'Dent';
?>" />
        <input type="submit" value="abschicken" />
    </form>
</body>

</html>
```

Codebeispiel 10 *stilvoll.php*

Im ersten Listing wird ein komplettes HTML-Formular mit mehreren `echo`-Anweisungen ausgegeben. Das ist prinzipiell möglich, da `echo` jede Art von Inhalten ausgeben kann, auch HTML-Formatierungen. Diese Herangehensweise hat allerdings mehrere Nachteile, von denen einige bereits jetzt offensichtlich sind:

- Da Sie wahrscheinlich (hoffentlich) mit einem Editor arbeiten, der Syntax-Highlighting beherrscht, wird auch der HTML-Code farblich hervorgehoben, z. B. werden Attribute und ihre Werte unterschiedlich dargestellt. Wenn Sie einen HTML-Tag in eine `echo`-Anweisung verpacken, verlieren Sie in einigen Editoren diesen Vorteil. Alles wird in einer Farbe dargestellt, was das Auffinden von Fehlern erschwert.
- Sie müssen alle Texte, die Sie mit `echo` ausgeben, in Anführungszeichen einschließen. Ich werde später noch im Detail darauf eingehen, aber Sie dürfen dieselben Anführungszeichen nicht mehr im Text verwenden, die Sie zum Einschließen verwendet haben. In unserem Beispiel wäre Folgendes also nicht erlaubt: `<?php echo 'Arthur's'; ?>`
- Jedes Zeichen, das Sie statt direkt im HTML mit `echo` ausgeben, sorgt dafür, dass die Seite langsamer lädt, da PHP mehr Daten verarbeiten muss. Der Unterschied mag bei kleinen Seiten nicht ins Gewicht fallen, aber auch dort gibt es keinen Grund, den Server unnötig zu belasten.
- Es ist schlicht und ergreifend hässlich!

Wenn Sie sich fertige PHP-Programme aus dem Internet herunterladen, werden Sie gerade bei älteren Projekten noch viel Code sehen, der wie die erste Version aussieht. Inzwischen ist man in der PHP-Community allerdings komplett von dieser Vorgehensweise abgerückt. Viele Projekte jüngeren Datums akzeptieren gar keinen Code mehr, der irgendwelche HTML-Formatierungen mit PHP-Anweisungen ausgibt. Sie werden später noch wesentlich mehr zu diesem Thema hören, also gewöhnen Sie es sich am besten gar nicht erst falsch an:

> Versuchen Sie, wo immer möglich, HTML-Code und PHP-Code getrennt zu halten. HTML-Tags mit PHP-Anweisungen auszugeben, ist extrem schlechter Stil und behindert die Übersichtlichkeit Ihres Codes.

3.5.2 Einrückungen

Ein zweiter Punkt beim Thema Stil sind Code-Einrückungen. Vergleichen Sie die beiden Beispiele:

Beispiel

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <title>Hallo Welt</title>
</head>

<body>
    <?php

    echo "So sieht das schon viel besser aus. ";
echo "Wenn Sie jede Anweisung auf eine Zeile schreiben, ";
      echo "behalten Sie leichter den Überblick. ";
  echo "Trotzdem müssen Sie immer noch ";
     echo "ein Semikolon nach jeder Anweisung schreiben, ";
   echo "da PHP eine neue Zeile nicht als das Ende ";
 echo "einer Anweisung sieht.";

    ?>
</body>

</html>
```

Codebeispiel 11 *schlechter_stil.php*

Beispiel

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <title>Hallo Welt</title>
</head>

<body>
    <?php

    echo "So sieht das schon viel besser aus. ";
    echo "Wenn Sie jede Anweisung auf eine Zeile schreiben, ";
    echo "behalten Sie leichter den Überblick. ";
    echo "Trotzdem müssen Sie immer noch ";
    echo "ein Semikolon nach jeder Anweisung schreiben, ";
    echo "da PHP eine neue Zeile nicht als das Ende ";
    echo "einer Anweisung sieht.";

    ?>
</body>

</html>
```

Codebeispiel 12 *guter_stil.php*

Sie sollten auf den ersten Blick sehen, was im oberen Beispiel falsch gelaufen ist. In PHP gelten im Prinzip dieselben Regeln wie in HTML: Sie sollten alles einheitlich formatieren und einrücken. Wenn Sie mehrere `echo`-Anweisungen nacheinander schreiben, sollten alle gleich eingerückt sein.

Wenn Sie sich jetzt fragen, warum ich das überhaupt erwähne, seien Sie versichert: Solange Ihnen das Einrücken nicht in Fleisch und Blut übergegangen ist, werden Sie sich immer wieder dabei ertappen, dass Ihr Code nicht einheitlich formatiert ist.

Sobald Sie mehr PHP-Anweisungen beherrschen, werde ich noch mehr Beispiele zu diesem Thema bringen. Merken Sie sich aber schon mal:

Formatieren Sie Ihren Code immer auf die gleiche Weise. Der (in)offizielle PHP-Standard ist es, jede Ebene um zusätzliche **vier Leerzeichen** einzurücken. Viele Editoren unterstützen Sie dabei, indem Sie einen einzelnen Tab in vier Leerzeichen umwandeln. Die einzige Ausnahme bildet bei einer mehrzeiligen PHP-Anweisung die erste Ebene, welche wir nicht (weiter) einrücken. Diese ist somit bündig zum öffnenden und schließenden PHP-Tag.

Noch besserer Stil ist es übrigens, wenn man statischen Text gar nicht erst mit `echo` ausgibt, aber zu den variablen Inhalten kommen wir erst in der nächsten Lektion.

3.6 Zusammenfassung

In dieser Lektion haben Sie die ersten Gehversuche in PHP unternommen. Sie haben gelernt, dass Sie mit den Tags `<?php` und `?>` PHP-Code direkt in HTML einbetten können. Sie haben erfahren, dass Sie mehrere Anweisungen mit dem Semikolon trennen müssen, und welchen Zweck die `echo`-Anweisung erfüllt.

Zusätzlich haben Sie aber auch schon jetzt die ersten Hinweise erhalten, wie Sie sauberen Code schreiben, der Ihnen später keinen Ärger macht. Je früher Sie sich an diese Tipps halten, desto besser, denn einmal falsch gelernte Angewohnheiten sind nur schwer wieder loszuwerden. Auch wenn Ihnen das im Moment vielleicht übertrieben erscheint, werden Sie später sehr dankbar sein, sich nicht umgewöhnen zu müssen.

3.7 Testen Sie Ihr Wissen!

1. Wie können Sie PHP-Code in HTML einbetten?
2. Wofür ist die `echo`-Anweisung gut?
3. Was müssen Sie bei der Kombination von HTML und PHP beachten?
4. Wie können Sie mehrere Anweisungen in einem PHP-Tag unterbringen?
5. Warum sollten Sie HTML-Tags nicht mit `echo` ausgeben?
6. Welchen Vorteil hat es, Code einheitlich einzurücken?

3.8 Übungen

Übung 1:

Legen Sie einen Unterordner *lektion03* im Ordner *phpschulung* an. Speichern Sie dort ein HTML-Dokument namens *spass.php*, das im `<body>`-Tag den Text `PHP macht Spaß!` stehen hat.

Hinweis: Legen Sie fortan für alle Übungen einer Lektion zunächst einen entsprechenden Unterordner an.

Übung 2:

Verändern Sie das Dokument. Der Text `PHP macht Spaß!` soll nun mit der PHP-Anweisung `echo` ausgegeben werden.

Übung 3:

Öffnen Sie nun die Datei *spass.html* direkt im *Firefox*-Browser über **Datei → öffnen** und beobachten Sie, was passiert.

Hinweis: Seit Version 4 ist im *Mozilla Firefox* die Menüleiste standardmäßig deaktiviert. Sie kann jedoch durch Drücken der (linken) Alt-Taste eingeblendet werden. Alternativ lässt sich die Datei auch einfach auf den Anzeigebereich des Browsers ziehen, wodurch sie im Browser geöffnet wird.

Übung 4:

Öffnen Sie die Datei nun über den Webserver, also über `http://localhost/phpschulung/lektion03/spass.php`, und beobachten Sie, was passiert.

Übung 5:

Benennen Sie die Datei in *spass.html* um, öffnen Sie sie über `http://localhost/phpschulung/lektion03/spass.html` und beobachten Sie den Unterschied in der Darstellung.

3.9 Optionale Übungen

Übung 6:

Schreiben Sie ein HTML-Dokument mit dem Namen *formular.php*, das ein Formular mit den folgenden Feldern enthält:

- `<input type="text" name="benutzername" value="" />`,
- `<input type="password" name="passwort1" value="" />`,
- `<input type="password" name="passwort2" value="" />` und
- `<input type="submit" value="registrieren" />`.

Übung 7:

In dem Feld `benutzername` soll nun standardmäßig "Mr. X" stehen, was über das `value`-Attribut möglich ist. Der Text soll dort mittels einer PHP-`echo`-Anweisung eingetragen werden.

Übung 8:

In den beiden `password`-Feldern sollen jeweils fünf Sterne (`*****`) stehen. Auch diese sollen dort mit `echo` eingetragen werden. Wie werden die Sterne im Browser dargestellt?