PHP
Einsteigerkurs

Autor: Michael Becker

Stand: 24. Mai 2009

Das Original auf dem neuesten Stand kann kostenlos von **http://www.bemi-online.de/php** heruntergeladen werden.

Inhaltsverzeichnis

1. Vorwort

Über PHP ist schon viel geschrieben worden. Doch meines Erachtens kommen bestimmte Fragen für Einsteiger viel zu kurz: Was ist PHP eigentlich? Was macht man damit? Brauche ich das überhaupt? Dieser Einsteigerkurs soll kein klassischer Schritt-für-Schritt-Ratgeber zum Lernen von PHP sein. Auch sollen nicht alle Funktionen im Detail besprochen werden. Dafür gibt es genügend Bücher am Markt. Dieser Einsteigerkurs soll einen Überblick darüber geben, was PHP kann und wie man PHP anwendet. Und es soll auch eine Entscheidungshilfe darstellen, ob man PHP für seine Einsatzzwecke verwenden kann und sollte.

Der Kurs wurde nach bestem Wissen zusammengestellt. Sollte dennoch etwas nicht richtig oder unklar sein, würde ich mich über eine kleine Nachricht freuen. Ebenfalls stehe ich für Fragen im Rahmen dieses Kurses gerne zur Verfügung.

Michael Becker

m.becker@go4more.de, ICQ: 147-997-803

2. PHP - Was ist das?

Als Anfänger ist man erst einmal damit beschäftigt, so genannte statische Seiten in HTML zu erstellen. Man platziert Texte, Verknüpfungen (Links), Bilder, Flash-Elemente und vielleicht noch einiges mehr auf der Seite. Das reicht in vielen Fällen am Anfang erst einmal aus.

Aber eine Programmiersprache wie PHP geht da noch weiter. Der Inhalt der Seiten kann dann auch noch variieren. Es werden in Abhängigkeit von den Benutzereingaben unterschiedliche Resultate erzielt. Wir alle kennen solche Seiten meist im großen Stil. Ich möchte die Vorgehensweise einmal an einem Forum beschreiben. Der Benutzer ruft das Forum auf. Und schon ist zum ersten Mal das Ergebnis der Programmierung zu sehen. Auf dem Bildschirm erscheinen nämlich die aktuellen Themen im Forum und welcher Benutzer was und wann zuletzt geschrieben hat. Eine Unmöglichkeit mit reinem statischem HTML. Denn diese Daten werden ganz aktuell bei dem Aufruf der Seite geschrieben. Nun geht es weiter: Der Benutzer gibt seinen Namen und sein Passwort ein und drückt den Einloggen-Schalter. Die Startseite des Forums wird neu aufgebaut. Doch dieses Mal werden wir freundlich mit einem "Hallo" und dem Benutzernamen dahinter begrüßt. Auch sind jetzt alle Themen in dem Forum markiert, die seit dem letzten Besuch neu hinzugekommen sind. Auch solche Informationen muss sich die Programmiersprache im Hintergrund besorgen und ganz frisch auf der Seite eintragen. Diese Beispiele könnte man noch viel weiterführen: Bei einem neuen Beitrag erscheint automatisch ein Benutzerbild, dass wir vorher einmal hochgeladen haben. Auch unsere persönliche Signatur steht unter den Beiträgen.

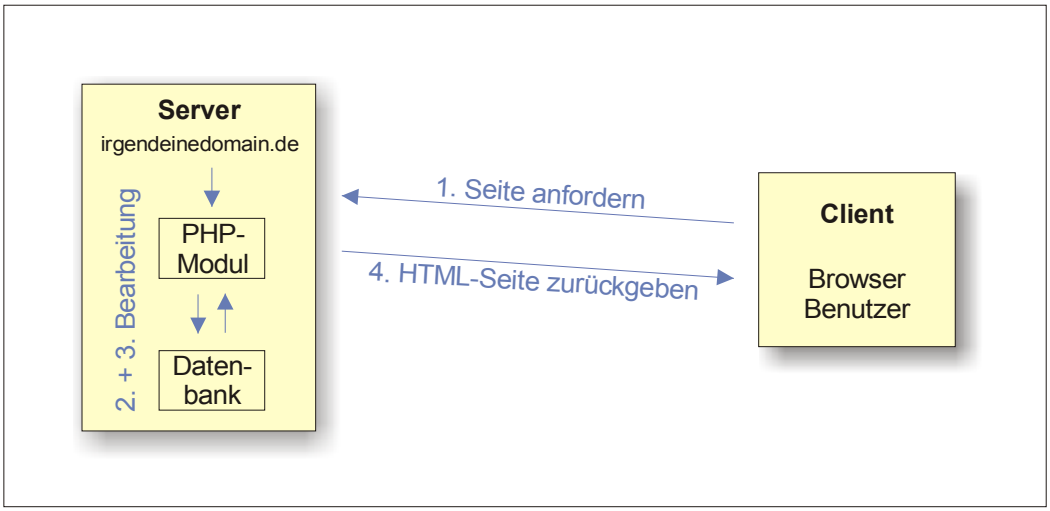
Natürlich gibt es auch noch viele weitere kleinere und größere Beispiele außer dem Forum: Gästebuch, Chat, Sportwetten, Online-Banking, SMS-Versand, Web-E-Mail, Wörterbuch, Nachrichten, Shopping, Besucherzähler, Spiele, Suchmaschinen etc.

Diese Beispiele haben alle eines gemeinsam: Sie werden erst zur Zeit des Aufrufs erzeugt. Man sagt, sie werden "dynamisch generiert". (Das ist nicht zu verwechseln mit "dynamischen Seiten", auf denen viele bunte und bewegte Bilder zu sehen sind und Texte über den Bildschirm fliegen.) Es ist auch egal, ob eine komplette Seite variablen Inhalt enthält oder nur ein kleiner Bereich wie der Besucherzähler oder der Witz des Tages. Ebenso muss das Ergebnis der Programmierung nicht einmal sichtbar sein. Eine solche Programmierung könnte zum Beispiel unsichtbar Statistiken über den verwendeten Browser und die Bildschirmauflösung protokollieren.

Es sollte noch dazu gesagt werden, dass PHP nicht die einzige Programmiersprache ist, genauso wie die Datenbank, die weiter unten noch angesprochen wird. Nur sind diese kostenlos verfügbar und auch weit verbreitet, so dass es später nicht schwer fallen wird, im Internet weiterführende Informationen, Erklärungen, Anleitungen, Hilfen etc. zu finden.

3. Wie funktioniert das alles denn nun?

Um die Vorgehensweise ein wenig anschaulicher zu gestalten, betrachten wir uns einmal das Beispiel eines Gästebuchs, und zwar wollen wir uns nur die bisherigen Beiträge anzeigen lassen. Wir rufen also das Gästebuch in unserem Browser auf oder - wie es üblicherweise ist - werden über einen Link dorthin gesteuert: http://www.irgendeineseite.de/gaestebuch.php.



Schritt 1: Der Browser versucht nun, die Seite gaestebuch.php von der fremden Domain zu laden. Man spricht davon, dass der Browser hier der Client ist, sozusagen der "Kunde", der etwas vom Server, dem "Anbieter", haben möchte.

Schritt 2: Der Server erkennt, dass eine Seite mit der Endung .php angefordert wurde, und weiß, dass diese Seite besonders behandelt werden muss. Sie muss nämlich erst einmal von einem PHP-Modul bearbeitet werden, bevor sie an uns wieder zurückgegeben werden kann.

Schritt 3: Die Befehle in der PHP-Datei werden nun ausgeführt. Dort steht, dass aus einer Datenbank mit allen Beiträgen die neuesten Beiträge gelesen werden müssen, und zwar in absteigender Reihenfolge. Diese Beiträge, zusammen mit dem Namen, der E-Mail-Adresse und der Homepage des Schreibers, sollen in einer Tabelle formatiert werden. Die Ausgabe entspricht wiederum dem HTML-Format! Denn das ist das einzige Format zur Seitengestaltung, das die Browser verstehen. Mit PHP-Befehlen können Browser nichts anfangen.

Schritt 4: Jetzt endlich wird die Seite wieder an den Browser zurückgegeben. Sie heißt immer noch gaestebuch.php, aber wenn wir uns den Quelltext im Browser ansehen, werden wir nichts anderes als HTML sehen. Das ist deswegen auch notwendig, weil in der PHP-Programmierung teilweise Passwörter für den Zugang zur Datenbank gespeichert sind.

Doch gehen wir noch einen Schritt weiter. Üblicherweise sieht man immer nur einen Teil des Gästebuchs, zum Beispiel 20 Beiträge pro Seite. Dann befindet sich irgendwo auf der Seite ein Link mit der Bezeichnung "weiter, "nächste Seite" oder auch einfach nur eine Seitenzahl, die man anklicken kann. Nun sieht der Link so aus: gaestebuch.php?seite=2. Jetzt folgen wieder die vier Schritte von oben. Nur dass dieses Mal das PHP-Modul in Schritt 3 darauf achtet, dass weitere Informationen übergeben wurden. Es wird also jetzt eine Tabelle mit den Beiträgen 21 bis 40

angezeigt. Dies entspricht der zweiten Gästebuchseite, wenn 20 Beiträge auf einer Seite stehen. Das bedeutet, die PHP-Programmierung lässt sich von außen steuern! Das Ergebnis ist abhängig von den Wünschen des Benutzers. Natürlich nur so weit, wie die übergebenen Parameter (unsere Wünsche) auch verarbeitet werden. Wenn wir zum Beispiel gaestebuch.php?seitevonhinten=1 aufrufen, würde gar nichts passieren, da gaestebuch.php den Parameter seitevonhinten gar nicht kennt und auch nicht verarbeitet.

Jetzt wollen wir noch einmal selbst einen Beitrag in das Gästebuch schreiben. Dazu erscheint auf dem Bildschirm ein Formular mit Eingabefeldern für den Namen, die E-Mail-Adresse, die Homepage und natürlich dem Text des Beitrags. Darunter befindet sich ein Schalter "Eintragen". Was soll ich noch weiter sagen? Es ist natürlich wieder die gleiche Vorgehensweise wie oben beschrieben. Das Formular ruft die Datei neuereintrag.php auf und übergibt gleichzeitig im Hintergrund, nicht sichtbar, die Inhalte der einzelnen Eingabefelder. Im dritten Schritt werden diese Daten zusammen mit dem Datum und der Uhrzeit in der Datenbank gespeichert. Doch was wird jetzt zurückgegeben? Üblicherweise steht in der zurückgelieferten HTML-Datei ein Text wie "Vielen Dank für deinen Eintrag." Oder aber die Programmierung wechselt nach der Speicherung der Daten automatisch wieder zu der bekannten Datei gaestebuch.php. Dann würden wir die aktuellen Beiträge sehen und natürlich ganz oben unseren eigenen, gerade erstellten Beitrag.

4. Anforderungen

Wenn man bis hierher gelesen hat, hat man sicherlich auch schon Überlegungen angestellt, ob und wie man PHP auf der eigenen Homepage benutzen kann. Doch halt, es gibt noch einige Hindernisse auf dem Weg zur eigenen Seite. Aber diese Hindernisse sind nicht unüberwindlich.

Der Server muss auch **PHP** verstehen. Das heißt, der Anbieter für den Web-Speicherplatz muss die entsprechenden Module zur Verfügung stellen. Ob er das tut, kann man auf seiner Homepage nachlesen. Ein kleiner Haken ist meist dabei, denn üblicherweise lassen sich die Anbieter diesen zusätzlichen Service bezahlen. In aller Regel wird gleichzeitig mit PHP auch die Datenbank **MySQL** angeboten. PHP und MySQL sind sehr weit verbreitet und dürfen zudem noch frei eingesetzt werden. Andere Programmiersprachen und Datenbanken gibt es normalerweise erst in Angeboten für den professionellen Einsatz.

Von Vorteil sind auch **Kenntnisse in HTML**. Ich hoffe, die Beispiele zuvor haben deutlich gemacht, dass PHP immer HTML-Seiten zurückgibt. Ob man nun seine Seiten komplett von Hand erstellt oder einen leistungsfähigen Editor wie Macromedia Dreamweaver oder Netobjects Fusion nimmt, der PHP-Code muss in HTML eingebettet werden.

Zum Schreiben von PHP-Programmen reicht (erst einmal) ein einfacher **Texteditor** aus, zum Beispiel aus dem Zubehör von Windows.

Eine weitere, und erst einmal letzte Anforderung, ist ein **FTP-Programm**. Wenn man seine HTML-Seiten bisher mit einem größerem Programm erstellt hat, hat dieses Programm vermutlich auch die Veröffentlichung im Internet vorgenommen. Mit einem separaten FTP-Programm kann man jede Art von Dateien auf den Web-Speicherplatz laden, also alle HTML- und PHP-Dateien sowie die dazugehörigen Bilder. Ein kostenloses Programm gibt es zum Beispiel unter http://www.filezilla.de. Man benötigt - wie bisher auch - die Zugangsdaten (Benutzer, Passwort, Adresse) von dem Speicherplatz-Anbieter.

Ganz hilfreich, aber keinesfalls ein Muss sind Englisch-Kenntnisse. Im Internet gibt es eine Vielzahl von Anleitungen, Beispielen, Tutorien und fertige Programme, allerdings zum Teil auch nur in englischer Sprache. Zudem sind alle PHP- und MySQL-Anweisungen in Englisch. Zur besseren Verständlichkeit benutzt dieser Kurs bei den eigenen Bezeichnungen nur deutsche Namen.

5. Erste Schritte

Gerade eben wurde es bereits erwähnt: Die minimale Anforderung zum Schreiben ist ein einfacher Texteditor, wie er zum Beispiel mit Windows mitgeliefert wird.

Damit der Server weiß, dass sich in der Textdatei ein PHP-Script befindet, muss - neben einer entsprechenden Dateiendung (dazu später mehr) - der PHP-Teil gekennzeichnet werden:

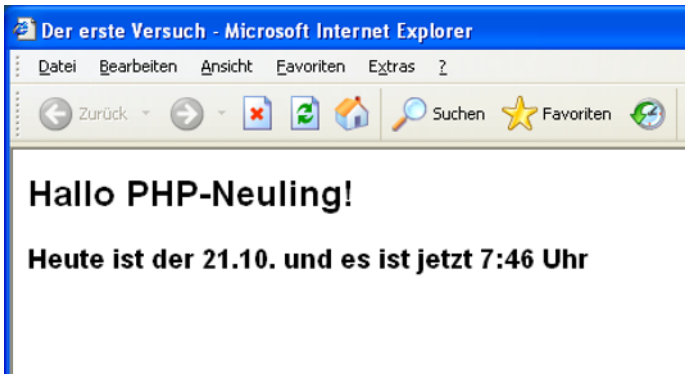
```
<?php
  Hier folgt das Programm.
?>
```

Wie bei HTML auch spielt es keine Rolle, wie viele Leerzeichen und Leerzeilen verwendet werden. Man selbst muss den Code hinterher überblicken können. Grundsätzlich hat es sich sinnvoll erwiesen, untergeordnete Teile einzurücken.

Wie gesagt, der PHP-Teil muss gekennzeichnet werden. Das heißt aber auch, dass noch ein HTML-Teil in der Datei vorkommen kann:

Datei: hallo.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
    <title>Der erste Versuch</title>
  </head>
  <body>
    <h2>Hallo PHP-Neuling!<h2>
    <?php
      $tag=date('j.n.');
      $uhrzeit=date('G:i');
      echo '<h3>Heute ist der '.$tag.' und es ist jetzt '.$uhrzeit.' Uhr</h3>';
    ?>
  </body>
</html>
```



Zugegeben das Ergebnis hätte man mit JavaScript auch noch nachbilden können. Aber wir sind ja noch am Anfang. Doch nun einmal ins Detail, denn das Beispiel enthält eine Vielzahl von Neuerung und erklärungsbedürftigen Stellen. Es sollte gezeigt werden, dass man HTML und PHP

in einer Datei mischen kann, und das man es sogar muss, wenn man dynamische Inhalte einbinden möchte. (Eine viel elegantere Lösung gibt es auf Seite 34 im Kapitel über Template-Systeme.)

Weiterhin ist zu sehen, dass am Ende eines jeden PHP-Befehls ein Semikolon stehen muss. Dies ist eine häufige Fehlerquelle.

Der echo-Befehl scheint hier eine Ausgabe zu bewirken. Er gibt den Inhalt an die HTML-Seite weiter; in diesem Fall ist es der ganze Teil im <h3>-Tag. An dieser Stelle sei noch einmal auf die Notwendigkeit von grundlegenden HTML-Kenntnissen hingewiesen! Das \n am Ende hat keine Auswirkung auf den sichtbaren Bereich, sondern erzeugt lediglich einen Zeilenumbruch im HTML-Code. Apropos HTML-Code: Wenn man sich jetzt im Browser den HTML-Code zeigen lässt, wird man - selbstverständlich - nichts mehr von PHP sehen. Dort sind Datum und Uhrzeit wie statische Elemente eingebettet.

$tag und $uhrzeit sind so genannte Variablen und können beliebige Inhalte unterschiedlicher Form, ob nun Text oder Zahlen oder gleich eine ganze Gruppe von Informationen, aufnehmen. Zu den Variablen gibt es noch später mehr. Variablen kann man auslesen, beschreiben, wiederbeschreiben, verketten und auch zum Rechnen verwenden. In dem Beispiel werden Datum und Uhrzeit in der HTML-Seite ausgegeben. Das muss ebenfalls im echo-Befehl geschehen. Man trennt die Variablen vom Textteil (auf beiden Seiten) mit einem Punkt ab.

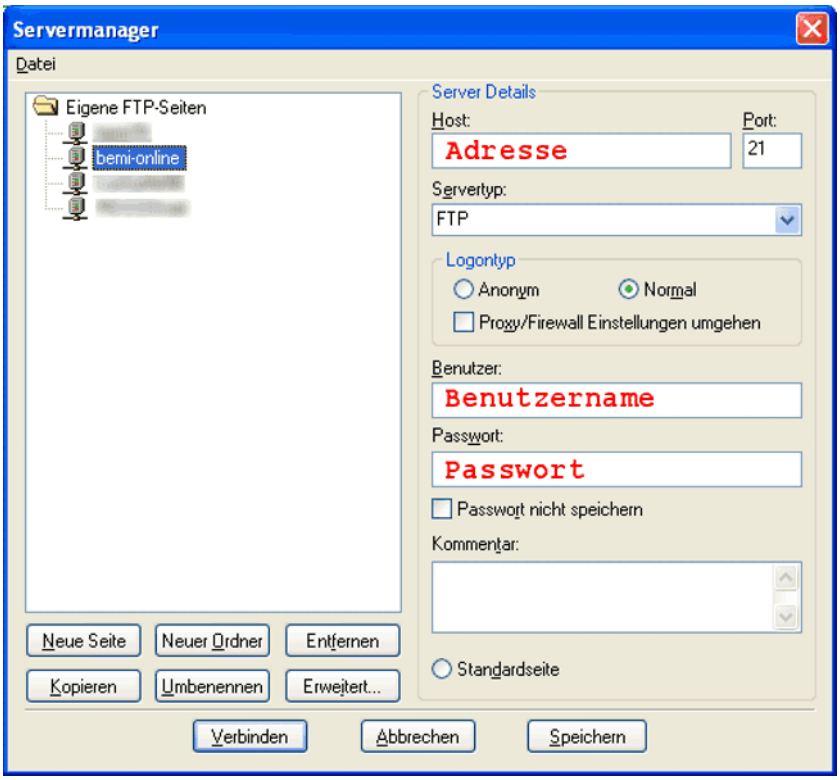
Bleibt noch date(...). Man spricht hier von einer Funktion. Eine Funktion ist ein Unterprogramm, das wahlweise auch einen Wert zurückgeben kann, wie in diesem Fall. Die date-Funktion ist Bestandteil von PHP und steht sofort zur Verfügung. Date erwartet einen Parameter, eine Textfolge, die in runden Klammern und Anführungszeichen angegeben werden muss. Bei dieser Funktion legt der Parameter das Aussehen des Datums fest. Hier noch eine kleine Auswahl neben den bereits im Beispiel angegeben Formatierungsoptionen: M für den abgekürzten Monat (z. B. "Feb"), s für Sekunden, W für die Wochennummer des Jahres, y für die zweistellige Jahreszahl (Groß- und Kleinschreibung beachten!).

6. Das erste Script testen

Die Programmierung befindet sich nun in der Textdatei hallo.php. Zum Testen muss sie auf einen Web-Speicherplatz geladen werden. Der Vorgang ist auch bekannt unter veröffentlichen, hochladen oder eingedeutscht uploaden. (Mit einem kleinen Aufwand kann man sein Werk auch auf der heimischen Festplatte testen. Siehe dazu das Kapitel zum lokalen Server auf Seite 16.)

Doch zuvor eine Anmerkung zum Dateinamen: Die Dateien dürfen jetzt natürlich nicht mehr auf .htm oder .html enden, denn sonst findet keine weitere Bearbeitung auf dem Server statt und die PHP-Datei würde in ihrem ursprünglichen Zustand an den Browser weitergegeben. Normalerweise verwendet man die Dateiendung .php, oder auch mit der Version angefügt, zum Beispiel .php4, sowie .phtml. Dies ist abhängig davon, wie der Server konfiguriert ist. Mit .php liegt man üblicherweise auf der richtigen Seite, ansonsten einfach ausprobieren, ob die Datei bearbeitet wird oder nicht. Des Weiteren gilt, wie bei allen anderen Dateien auch, dass Sonderzeichen wie Umlaute zu vermeiden sind. Mit den 26 Buchstaben, 10 Ziffern und dem _ Unterstrich kann man nicht viel verkehrt machen. Anders als Windows achten Betriebssysteme (auf dem Server) wie Unix oder Linux auf die Groß- und Kleinschreibung. Entweder beachtet man sehr gut die richtige Schreibweise oder schreibt gleich alles in Kleinbuchstaben.

Jetzt kommt das FTP-Programm aus den Anforderungen zum Einsatz, hier am Beispiel von FileZilla. Der erste Schalter in der Symbolleiste bzw. über die Seitenverwaltung im Menü Datei öffnet folgendes Fenster, in das die Daten des Web-Speicherplatz-Anbieters eingegeben werden müssen.



Anschließend wird eine Verbindung hergestellt und man kann nun auf dem Server Unterverzeichnisse anlegen und einfach von der eigenen Festplatte Daten auf den Server kopieren. Und bei Bedarf auch umgekehrt.

Ist die Datei hochgeladen, ruft man im Browser die eigene Seite auf, z. B. über

- http://www.neue-seite.com/unterverzeichnis/hallo.php

Wenn nun alles funktioniert hat, sollte die Seite wie im Beispiel zuvor erscheinen, natürlich mit aktuellem Datum und der augenblicklichen Uhrzeit.

Wird die Datei hingegen nicht gefunden, bitte noch einmal die Schreibweise überprüfen, insbesondere auch die Groß- und Kleinschreibung. Es könnte nun auch noch passieren, dass der Browser die Datei zum Download anbietet oder anstelle der netten Begrüßung den PHP-Code auf dem Bildschirm präsentiert. In diesem Fall ist die Datei auf dem Server nicht erst durch das PHP-Modul gelaufen. Das könnte zum Beispiel daran liegen, dass eine falsche Dateiendung gewählt wurde, oder schlimmstenfalls, dass der Anbieter überhaupt kein PHP unterstützt.

1. Variablen, Bedingungen, Schleifen und Funktionen

Im unseren ersten Beispiel haben wir gesehen, wie etwas in einer **Variablen** gespeichert wird und wie diese auch wieder ausgegeben werden kann. Wenn man etwas einer Variablen zuweisen will, muss der Name der Variablen auf der linken Seite vom Gleichheitszeichen stehen und mit einem Dollar-Zeichen beginnen. PHP erkennt automatisch den Typ einer Variablen.

```
$text='Hallo';
$zahl=100;
$dezimalzahl=5.3;  // Achtung: Punkt statt Komma
```

Nebenbei angemerkt: Eine Bemerkung steht entweder hinter zwei Schrägstrichen // bis zum Ende der Zeile oder eingebettet: /* Kommentar */. Dahinter kann Code stehen und die Bemerkung darf so auch über mehrere Zeilen gehen. Das ist insbesondere dann wichtig, wenn man zum Testen größere Teile unbearbeitet lassen möchte.

Genauso wie im Beispiel zuvor, als wir das Datum und die Uhrzeit mit den übrigen Textteilen der Ausgabe verbunden haben, lassen sich auch Variablen verknüpfen:

```
$teil1='Zurzeit gilt die ';
$teil2='Winterzeit';
$zusammen=$teil1.$teil2;
echo $zusammen;  // Ausgabe: Zurzeit gilt die Winterzeit
```

Mit den Variablen kann man selbstverständlich auch rechnen:

```
$a=6; $b=3;
$addition=$a+$b;
$subtraktion=$a-$b;
$multiplikation=$a*$b;
$division=$a/$b;
```

Interessant sind sicherlich auf die so genannten Arrays. Variablen diesen Typs enthalten nicht nur einen Wert, sondern gleich eine ganze Sammlung. Die einzelnen Datenelemente können an Hand eines Indexes abgefragt werden, wobei der erste Eintrag bei einer automatischen Nummerierung mit 0 beginnt, oder mittels eines zugeordneten Wertes anstelle vom (Zahlen-)Index. Der Zugriff auf die einzelnen Elemente des Arrays erfolgt dabei über eckige Klammern [] hinter dem Namen der Variablen. Hier ein paar Beispiele mit Zuordnung und anschließender Ausgabe:

```
$waehrung[]='EUR';
$waehrung[]='USD';
$waehrung[]='GBP';
$waehrung[]='JPY';
echo $waehrung[2];  // Ausgabe: GBP ... das dritte Element, Start bei 0!

$waehrung[2]='CHF';
echo $waehrung[2];  // Ausgabe: CHF ... einfach überschrieben

$waehrungsname['EUR']='Euro';
$waehrungsname['USD']='US-Dollar';
$waehrungsname['CHF']='Schweizer Franken';
$waehrungsname['JPY']='Japanischer Yen';
echo $waehrungsname['USD'];  // Ausgabe: US-Dollar
echo $waehrungsname[$waehrung[3]];  // Ausgabe: Japanischer Yen
```

```
/* alternative Zuweisung */
$waehrungsname=array('EUR'=>'Euro','USD'=>'US-Dollar','CHF'=>
  'Schweizer Franken','JPY'=>'Japanischer Yen');
echo $waehrungsname['EUR'];  // Ausgabe: Euro
```

Anführungszeichen und "Escapen"

Zuvor wurde bereits geschrieben, dass Texte immer in Anführungszeichen stehen müssen. PHP kann jedoch zwei Formen verarbeiten, die einfachen ' Anführungszeichen (Apostroph) und die doppelten " Anführungszeichen. (Der Apostroph ist auf der deutschen Tastatur über dem Doppelkreuz # zu finden und nicht mit den Akzent-Zeichen ´ und ` zu verwechseln.) Auf den ersten Blick scheinen beide gleich zu arbeiten:

```
echo 'Ein Text in Anführungszeichen';
echo "Ein Text in Anführungszeichen";
```

Der feine Unterschied besteht jedoch darin, was mit dem Inhalt geschieht. Dazu ein Beispiel mit einer Variablen:

```
$s='Text';
echo 'Ein $s in Anführungszeichen';  // Ausgabe: Ein $s in Anführungszeichen
echo "Ein $s in Anführungszeichen";  // Ausgabe: Ein Text in Anführungszeichen
```

Der Inhalt in einfachen Anführungszeichen wird von PHP nicht weiter verarbeitet, was zu einem kleinen Geschwindigkeitsvorteil führt. Grundsätzlich besteht jedoch weiterhin in beiden Versionen die Möglichkeit, Textteile zu verketten:

```
$s='Text';
echo 'Ein '.$s.' in Anführungszeichen';
echo "Ein ".$s." in Anführungszeichen";
// Ausgabe zweimal: Ein Text in Anführungszeichen
```

Allerdings ist es manchmal notwendig, dass bestimmte Textteile bearbeitet werden. Denn die Verarbeitung des Textes beschränkt sich nicht nur auf Variablen, sondern auch auf Sonderzeichen, die mit einem Backslash \ beginnen. Die Kombination \n steht für eine neue Zeile. Will man zum Beispiel eine Text-E-Mail mit der mail-Funktion verschicken, muss der Text mit dem Zeilenumbruch \n in doppelte Anführungszeichen geschrieben werden. Ansonsten steht sichtbar \n in der E-Mail - ohne einen Zeilenumbruch.

Jedoch sollte man vermeiden, Anführungszeichen, die als Begrenzung für einen Textteil dienen, zu mischen, denn sonst kann PHP nicht wissen, wo ein Text anfängt und aufhört. Allerdings besteht durch Austausch der äußeren Anführungszeichen die Möglichkeit, die Anführungszeichen im Text richtig auszugeben:

```
echo '<img src='bild.jpg' alt='Bild'>';  // Fehler!
echo "<img src="bild.jpg" alt="Bild">";  // Fehler!
echo '<img src="bild.jpg" alt="Bild">';  // jetzt funktioniert es
echo "<img src='bild.jpg' alt='Bild'>";  // und auch hier
```

Eine Besonderheit besteht darin, Sonderzeichen zu "**escapen**" (im Deutschen auch mit "maskieren" ausgedrückt). Das bedeutet nichts anderes, als dass man ein Backslash \ vor die Sonderzeichen stellt. Aus " wird \" und aus $ wird \$. Der Backslash selbst wird durch zwei \\ dargestellt.

Es gibt Situationen, in denen man Sonderzeichen übergeben muss, ohne dass man diese kennt. Zum Beispiel, wenn diese aus der Benutzereingabe in einem HTML-Formular stammen.

Ohne weiter auf das Thema im Rahmen dieses Einsteigerkurses einzugehen, soll auch nicht unerwähnt bleiben, dass man Daten von außen manipulieren kann, wenn man bei nicht sorgfältiger Programmierung Sonderzeichen nicht escapet (maskiert). Zum Schutz vor Missbrauch sollte man die nachfolgenden Zeilen beherzigen.

Das Beispiel soll zeigen, wie man Zeichen maskiert. (Es handelt sich um ein Beispiel, bei dem ein einzelner Beitrag aus einem Benutzerformular in eine Datenbank übertragen werden soll. Auf die Datenbank wird später noch ausführlicher eingegangen.)

```
$kommentar=$_POST['kommentar'];
mysql_query("INSERT INTO tabelle (kommentar) VALUES
  ('".mysql_real_escape_string($kommentar)."')");
```

Die erste Zeile liest das Kommentar-Feld aus dem Formular und speichert den Inhalt in der Variablen ab. Die zweite Zeile ist jedoch interessanter. Generell steht der Wert, der an die Datenbank übergeben soll, in einfachen Anführungszeichen, zum Beispiel VALUES('Mein Kommentar'). Der komplette SQL-Befehl besteht hier aus drei verketten Einzeltexten: den MySQL-Befehl, der mit mysql_real_escape_string() maskierte Text, und der Rest des MySQL-Befehls.

Die Funktion mysql_real_escape_string() sorgt nun dafür, dass alle Sonderzeichen, wie zuvor geschrieben, mit einem \ versehen werden und so korrekt in der Datenbank gespeichert werden. Wenn man sich später einmal den Eintrag in der Datenbank ansieht, wird man nur die Sonderzeichen sehen, nicht aber das einleitende Zeichen \.

Kommen wir noch einmal auf das Beispiel mit dem Bild zurück. Dann können wir die HTML-Ausgabe mit dem neuen Wissen auch so schreiben:

```
echo "<img src=\"bild.jpg\" alt=\"Bild\">\n\nBildtitel";
```

Jetzt weiß PHP, dass die Anführungszeichen um "bild.jpg" und um "Bild" Teil des Inhalts sind. Auch hier kann man sich anschließend den HTML-Code im Browser ansehen. Dort steht die korrekte HTML-Anweisung <img src="bild.jpg" alt="Bild">, gefolgt von einer Leerzeile und dem Wort "Bildtitel". Die beiden Zeilenumbrüche (\n) haben nur Auswirkung auf den Quelltext; Zeilenumbrüche in HTML werden bekanntlich mit
 erzeugt.

Aber Achtung: Das Beispiel funktioniert nur, wenn außen doppelte Anführungszeichen verwendet werden. Die Erklärung steht ganz am Anfang des Themas: Mit einfachen Anführungszeichen würden \" und \n nicht ausgewertet, sondern genau in dieser Form an den Browser übergeben.

Im obigen Beispiel mit der Datenbank wird ein Feld aus dem HTML-Formular geladen. Je nach Serverkonfiguration kann es allerdings sein, dass die Sonderzeichen bereits maskiert sind. Um das herauszufinden, benutzt man die Funktion get_magic_quotes_gpc() – bis PHP-Version 6. Da Funktionen und Abfragen erst später in diesem Kapitel erläutert werden und das Thema an dieser Stelle ein wenig zu weit gehen würde, gibt es in Kapitel 1 ein Beispiel dazu. Die Funktion wird in dem Quelltext zum Kontaktformular verwendet und auf Seite 22 beschrieben.

Aber wäre es nicht auch schön, mit Hilfe von Variablen nur bestimmte Texte auszugeben? Oder gar den Ablauf eines Programms zu steuern? Solche **Kontrollstrukturen** dürfen natürlich auch in

PHP nicht fehlen. Die einfachste ist die Wenn-dann-Abfrage, oder auch erweitert: Wenn-dann-sonst:

```
if ($stunde>0 and $stunde<=13) {
  echo 'Guten Morgen';
}
else {
  echo 'Guten Tag';
}
```

Die Bedingung selbst steht hierbei immer in runden Klammern und die Anweisungsblöcke (dann und sonst), die zu der Abfrage gehören, in geschweiften Klammern. (Folgen nur einzelne Befehle wie hier, könnte man die Klammern auch weglassen.) Ein Vergleich zwischen zwei Variablen bzw. Werten stellt man mit == für gleich, != für ungleich, > für größer, >= für größer gleich, < für kleiner und <= für kleiner gleich her. Daneben kann man mehrere Vergleiche verknüpfen und zwar mit and (beide Bedingungen müssen erfüllt sein), or (eine der beiden Bedingungen muss erfüllt sein) und xor (eine oder keine Bedingungen sind erfüllt).

```
if ($a!='de' and ($b<3 or $b>9))
```

Diese Bedingungen ist nur erfüllt, wenn in $a etwas anderes steht als "de" und wenn $b kleiner als 3 oder größer als 9 ist. Wichtig ist die richtige Verwendung der Klammern. In diesem Beispiel wird zuerst der Teil in den Klammern ($b<3 or $b>9) geprüft und dieses Ergebnis dann mittels and mit dem anderen Teil der Bedingung ($a!='de') verknüpft. Diese Art der Bedingungen gilt auch in gleicher Weise für die anderen Abfragen und Schleifen, die noch kommen. Möchte man eine einzelne Variable nach verschiedenen Inhalten prüfen, nimmt man statt der if-Struktur lieber diese:

```
switch ($endung) {
  case 'JPG': echo 'Bild'; break;
  case 'WAV': echo 'Musik'; break;
  case 'PPS': echo 'Präsentation'; break;
  case 'EXE': echo 'Programm'; break;
  default:    echo 'Sonstiges';
}
```

Es wäre echt mühsam, solche Abfragen über verschachtelte if-Abfragen zu realisieren: Wenn $endung gleich "JPG" gebe "Bild" aus, ansonsten: Wenn $endung gleich "WAV" gebe "Musik" aus, ansonsten: Wenn $endung gleich...

Mit **Schleifen** kann man schließlich eine Anweisungsblock mehrmals durchlaufen. Die erste fungiert einfach als Zähler. Dieses Beispiel gibt eine Zahlenreihe aus und zählt sie auch gleich zusammen:

```
$summe=0;
for ($zaehler=5; $zaehler<=9; $zaehler++) {
  echo $zaehler.'<br>';
  $summe=$summe+$zaehler;
}
echo $summe;
// Ausgabe in einzelnen Zeilen: 5, 6, 7, 8, 9, 35
```

Die for-Schleife erwartet drei Parameter, nämlich die Anweisung beim Beginn, die Bedingung zur Beendigung und die Anweisung, was in jedem Durchlauf passieren soll. $zaehler++ ist eine kurze Schreibweise für $zaehler=$zaehler+1. Also bei jedem Durchlauf die Variable um 1 erhöhen.

Grundsätzlich sollte man sich angewöhnen, die Variablen vor der ersten Verwendung zu belegen. In diesem Fall wurde der $summe 0 zugewiesen.

Eine weitere Variante der for-Struktur bezieht sich auf Arrays. Hier greifen wir noch einmal auf die Währungen aus dem Beispiel oben zu:

```
foreach ($waehrung as $einewaehrung) {
  echo $einewaehrung.'<br>';
}
// Ausgabe in einzelnen Zeilen: EUR, USD, GBP, JPY

/* Alternative */
for ($zaehler=0; $zaehler<=count($waehrung); $zaehler++) {
  echo $waehrung[$zaehler].'<br>';
}
```

Die letzte Schleife durchläuft einen Anweisungsblock so lange, bis eine Bedingung erfüllt ist. Die Bedingungen können dabei in gleicherweise verknüpft werden, wie wir es auch schon bei der if-Abfrage gesehen haben. Greifen wir noch einmal auf die Währungen zu:

```
$waehrungen=array('USD','AUD','GBP','CNY','DKK','EGP','EUR','HKD','JPY',
  'MXN','NZD','RUB','KRW','CHF','TRL');
$ziel='NZD';
$zaehler=0;
while ($waehrungen[$zaehler]!=$ziel and $zaehler<count($waehrungen)) {
  echo $waehrungen[$zaehler].'<br>';
  $zaehler++;
}
// Ausgabe in einzelnen Zeilen: USD, AUD, GBP, CNY, DKK, EGP, EUR, HKD, JPY, MXN
```

While ist das englische Wort für solange. Die Schleife besagt also im Klartext: Solange der Inhalt von $waehrungen an der Stelle $zaehler (wie erinnern uns: beginnt bei 0!) nicht dem Wert von $ziel entspricht, gebe die Kurzschreibweise der aktuellen Währung aus und setze den Zähler im 1 nach oben.

Dabei ist noch eine Sicherung eingebaut: Die Schleife beendet auch in dem Fall, dass $ziel sich gar nicht in der Liste befindet. Dann tritt der zweite Teil der Bedingung ein, dass der Zähler gleich der Anzahl ist. Auch darf man bei solchen Schleifen nicht vergessen, eine Zählvariable zu erhöhen. Ansonsten kommt man in eine so genannte Endlosschleife. Auf dem Server ist es jedoch so, dass PHP-Scripts nach einer bestimmten Zeit (meist unter einer Minute) abgebrochen werden.

Als letzte Gruppe in diesem Kapitel sollen die **Funktionen** vorgestellt werden. Funktionen stellen ein Unterprogramm dar und haben den wesentlichen Vorteil - neben einer übersichtlichen Strukturierung des Quelltextes -, dass der Programmteil in einer Funktion mehrfach aufgerufen werden kann, ohne ihn jedes Mal zu kopieren. Im Kapitel über Datenbanken befindet sich ein Beispiel, in dem eine Funktion in eine externe Datei ausgelagert wird. Und ein Beispiel haben wir weiter oben bei den ersten Schritten schon kennen gelernt. Dort hat die date-Funktion das aktuelle Datum mit einer bestimmten Formatierung ausgegeben.

Einer Funktion kann man bestimmte Werte übergeben, die so genannten Parameter. Die einzelnen Parameter werden mit Komma getrennt in runde Klammern geschrieben. Auch wenn ein Funktion keine Parameter erwartet, muss ein Paar runde Klammern ohne Inhalt geschrieben werden. Die meisten Funktionen geben einen Wert zurück. Diesen kann man wie eine Zahl oder eine Textfolge einer Variablen zuweisen.

Hier ein paar Beispiele:

```
echo strtolower('Der PHP-Einsteigerkurs hat über 30 Seiten!');
// Ausgabe: DER PHP-EINSTEIGERKURS HAT ÜBER 30 SEITEN!

echo strchr('Er fährt ein blaues Auto','ein');
// sucht nach in einem Strin einen Teilstring, Ausgabe: ein blaues Auto

$zahl=floor(2.7);
// floor rundet immer ab, Ausgabe: 2

$zahl=intval('7 Autos');
// intval liest eine Zahl aus einem String, Ausgabe: 7

$aktuell=time();
// das Ergebnis ist eine Zahl fortlaufende Sekundenzahl ab 1970

srand((double)microtime()*1000000);
echo rand(1,6);
// Zufalszahl eines Würfels (srand dient zur Initialisierung)

if (file_exists('test.dat'))
  echo 'Die Datei test.dat existiert';
else
  echo 'Die Datei test.dat existiert nicht';
```

Es gibt natürlich noch sehr viel mehr Funktionen. Eine gute Übersicht gibt es in SELFPHP wie auf Seite 19 beschrieben.

Aber viel interessanter wird ein PHP-Code, wenn man nicht nur die vorgefertigten Funktionen nimmt, sondern auch eigene erstellen kann. Eigene Funktionen können an beliebiger Stelle im PHP-Code stehen. Üblicherweise schreibt man sie aber wegen der Übersicht an den Anfang.

Das folgende Beispiel erwartet keine Parameter und gibt auch keinen Wert zurück:

```
function ausgabe() {
  echo 'Hallo<br>';
}
```

Im Folgenden wird es etwas umfangreicher. Weiter oben wurde eine Schleife vorgestellt, die eine Reihe von Zahlen addieren soll. Wir lagern diese Schleife jetzt aus und machen eine universelle Funktion davon. Das Ergebnis wird anschließend zurückgegeben.

```
function summe_reihe($von,$bis) {
  $summe=0;
  for ($i=$von; $i<=$bis; $i++)
    $summe=$summe+$i;
  return $summe;
}
$ergebnis=summe_reihe(2,5);  // Inhalt der Variablen: 14
echo summe_reihe(10,15);     // Ausgabe: 75
```

Die Parameter können von rechts nach links optional sein, das heißt, sie müssen nicht unbedingt übergeben und können mit einem Standardwert vorbelegt werden:

```
function begruessung($text,$name='') {
  echo $text;
  if ($name!='')
    echo ', '.$name;
```

```
  echo '!<br>';
}
begruessung('Guten Morgen','Nicki');  // Ausgabe: Guten Morgen, Nicki!
begruessung('Guten Tag');             // Ausgabe: Guten Tag!
begruessung();                        // Fehler! Erster Parameter erwartet
```

Zu erwähnen sei noch, dass Variablen in einer Funktion nicht automatisch auf die Variablen außerhalb der Funktionen zugreifen können. Auch umgekehrt verlieren Variablen, die in der Funktion erzeugt wurden, außerhalb ihre Gültigkeit. Es sei denn, man deklariert sie in der Funktion als "global". Das nachfolgende Beispiel soll dieses verdeutlichen:

```
function globale_variablen() {
  global $b;  // nur diese beiden Variablen verlieren nicht ihre Gültigkeit,
  global $c;  // $a hingegen schon
  $c=5;
  echo 'in der Funktion: a='.$a.'<br>';  // Ausgabe: (nichts, da unbekannt)
  echo 'in der Funktion: b='.$b.'<br>';  // Ausgabe: 7
  echo 'in der Funktion: c='.$c.'<br>';  // Ausgabe: 5
  $b=10;
}
$a=3;
$b=7;
globale_variablen();
echo 'außerhalb der Funktion: b='.$b.'<br>';  // Ausgabe: 10
echo 'außerhalb der Funktion: c='.$c.'<br>';  // Ausgabe: 5
```

2. Wie es weitergeht

Die Grundzüge sind in diesem Einsteigerkurs schon einmal erläutert. Doch wie geht es nun weiter? In den nächsten Kapiteln folgenden noch zwei etwas größere Anwendungsbeispiele, mit und ohne Benutzung einer Datenbank; "größer" natürlich nur in den Relation zu den bisher vorgestellten Beispielen. Denn richtige Scripts geben sich nicht mit einer halben bedruckten Seite zufrieden.

Als hervorragende Quelle zur Erklärung von PHP-Befehlen hat sich SELFPHP bewährt. Für den Anfang sei auf die "Funktionsübersicht" auf der Seite hingewiesen. Dort sind alle Befehle gruppiert und man findet zum Beispiel unter "String-Funktionen" jeweils mit Beschreibungen und Beispielen die Befehle, die man zum Bearbeiten von Textfolgen benötigt.

- http://www.selfphp.de

Zudem gibt es im Netz eine Vielzahl von Sammlungen kleinerer und größerer Scripts für PHP:

- http://www.php-resource.de
- http://www.phpwelt.de
- http://www.phparchiv.de
- http://www.php-free.de

1. Anwendungsbeispiel: Kontaktformular

Im Folgenden soll einmal ein vollständiger und auch etwas umfangreicherer Code für eine Seite dargestellt werden. Dies ist ein Kontaktformular, wie es auf vielen Seiten zu finden ist. Hier erst einmal der Code in einem Stück. Wir werden die einzelnen Bestandteile am Ende durchgehen. (Bitte nicht vergessen, beim Testen die eigene E-Mail-Adresse zu verwenden!)

Datei: kontaktformular.php

```
<?php

 /* Teil 1: Auswertung */
 $fehler='';
 $name=''; $email=''; $nachricht='';

 if ($_POST['gesendet']=='1') {
   // übergebene Formulardaten einlesen
   if (get_magic_quotes_gpc()) {
     $name=trim(stripslashes($_POST['name']));
     $email=trim(stripslashes($_POST['email']));
     $nachricht=trim(stripslashes($_POST['nachricht']));
   }
   else {
     $name=trim($_POST['name']);
     $email=trim($_POST['email']);
     $nachricht=trim($_POST['nachricht']);
   }

   // Fehlerprüfung
   if ($name=='')
     $fehler='Bitte den Namen eingeben!<br>';
   if ($email=='')
     $fehler.='Bitte die E-Mail-Adresse eingeben!<br>';
   if ($nachricht=='')
     $fehler.='Bitte eine Nachricht eingeben!';

   if ($fehler=='') {
     // alles ok, E-Mail versenden -- eigene Adresse eintragen!
     $emailheader="From: keineantwort@neue-seite.com\nContent-type: text/plain";
     $emailempfaenger='eigeneadresse@neue-seite.com';
     $emailbetreff='Formulardaten';
     $emailnachricht="Name: ".$name."\n\nE-Mail: ".$email."\n\n";
     $emailnachricht.="Nachricht:\n".$nachricht;
     mail($emailempfaenger,$emailbetreff,$emailnachricht,$emailheader);
     header('location:kontakt_danke.htm'); die();
   }
   else {
     // Daten für die Ausgabe vorbereiten
     $name=htmlspacialchars($name);
     $email=htmlspecialchars($email);
     $nachricht=htmlspecialchars($nachricht);
   }
 }

 /* Teil 2: HTML-Formular */
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
    <title>Nehmen Sie Kontakt zu uns auf!</title>
  </head>
  <body>
    <h2>Wir sind für Sie da!</h2>
    <h3>Schreiben Sie uns eine E-Mail mit Ihrem Anliegen.
    Wir versprechen Ihnen, dass wir uns umgehend mit Ihnen in
    Verbindung setzen!</h3>
    <?php
      // falls Fehler auftreten:
      if ($fehler!='')
        echo "<p style=\"color:#C00000; font-weight:bold\">".$fehler."</p>\n";
    ?>
    <form action="kontaktformular.php" method="post">
      <input type="hidden" name="gesendet" value="1">
      <p>
        Ihr Name:<br>
        <input type="text" name="name" value="<?php echo $name; ?>" size="30">
      </p>
      <p>
        Ihre E-Mail-Adresse:<br>
        <input type="text" name="email" value="<?php echo $email; ?>" size="30">
      </p>
      <p>
        Ihre Nachricht:<br>
        <textarea name="nachricht" cols="70"
          rows="6"><?php echo $nachricht; ?></textarea>
      </p>
      <p><input type="submit" value="Abschicken"></p>
    </form>
  </body>
</html>
```

Datei: kontakt_danke.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
    <title>Ihr direkter Draht zu uns</title>
  </head>
  <body>
    <h1>Vielen Dank!</h1>
    <h3>Ihre Nachricht wurde an uns weitergeleitet!</h3>
    <h4>Sie werden innerhalb von 24 Stunden eine Antwort bekommen.</h4>
  </body>
</html>
```

Die zweite Datei hat nur die kleine Aufgabe, sich bei dem Benutzer freundlich zu bedanken, und besteht aus reinem HTML. Alle weiteren Beschreibungen beziehen sich nur auf die PHP-Datei.

Es mag etwas befremdlich wirken, dass die Auswertung (Teil 1) vor dem eigentlichen Formular (Teil 2) kommt. Doch der Schein trügt ein wenig. Tatsächlich wird die Auswertung beim ersten Aufrufen einfach übersprungen. Schauen wir uns doch erst einmal den zweiten Teil an. Dieser besteht aus ein paar Überschriften und Informationstexten und dem Formular-Teil.

Zuerst stoßen wir auf einen eingebetteten PHP-Teil, in dem abgefragt wird, ob ein Fehler vorlag. Dies ist für später wichtig, nachdem das Formular geprüft wurde. Beim ersten Aufruf kann noch nichts in dieser Variablen stehen, da sie gleich am Anfang (Teil 1) gelöscht und dann auch nicht weiter bearbeitet wird.

Nun kommen zwei Besonderheiten in dem <form>-Tag zum Vorschein. Zum einen wird hier unter action die eigene PHP-Datei wieder aufgerufen (man hätte die Prüfung auch in eine andere Datei verlagern können, aber das macht diese kleine Anwendung nicht gerade übersichtlicher), und zum anderen wird method="post" angegeben. Dies bewirkt, dass beim Absenden des Formulars die Daten im Hintergrund und für den Benutzer unsichtbar übergeben werden. Auf diese "Post"-Daten werden wir gleich noch zurückgreifen. Zur Prüfung, ob ein Formular abgesendet wurde, schicken wir über ein verstecktes Feld (hidden) einen beliebigen Wert ungleich null mit.

Die letzte Besonderheit im Formular ist die Vorbelegung der einzelnen Felder über den Parameter value im <input>- bzw. <textarea>-Tag. Auch hier kann beim ersten Aufruf nichts in den HTML-Quelltext geschrieben werden, denn die Variablen $name, $email und $nachricht sind am Anfang leer.

Nun stellen wir uns vor, das Formular wurde ausgefüllt und der "Absenden"-Schalter betätigt. Gemäß den Angaben im <form>-Tag wird wieder die gleiche Datei aufgerufen und die Daten werden mit der Post-Methode verschickt. Die Post-Daten können mit $_POST['Feldname'] ausgelesen werden. In diesem Fall wird gefragt, ob das Formular geschickt wurde ($_POST['gesendet']). Da diese Überprüfung positiv ist, erfolgt nun die Auswertung der Formulardaten (Teil 1).

Hinweis:Im Folgenden werden zwei Konfigurationseinstellungen beschrieben, die nur noch kurze Zeit gültig sind: Ab PHP-Version 6 sind die Optionen zu "register globals" und "magic quotes" abgeschafft, d. h. PHP verhält sich so, als wenn die beiden Optionen, wie nun beschrieben, deaktiviert sind.

Zunächst einmal wollen wir die drei Formularfelder in eigenständige Variablen laden. Hier werden wir leider mit zwei Einstellungen des Servers konfrontiert; deshalb an dieser Stelle ein kleiner Exkurs (oder die nächsten beiden Absätze überspringen und das Schema aus dem Beispiel übernehmen). Je nach Einstellung ist PHP selbst in der Lage, diese Variablen zuzuweisen. Dies nennt man "register globals". Wenn diese Option serverseitig aktiviert ist, könnte man sich den Weg über $_POST (oder auch $_GET bei der Get-Methode) sparen. Der Server würde automatisch eine gleichnamige Variable anlegen. Doch sei hier Vorsicht geboten. Denn ein Außenstehender könnte ein fiktives Formular erzeugen bzw. den URL manipulieren. Aus diesem Grund sollte man die Variablen auch vor der ersten Verwendung immer löschen oder mit einem sonstigen Startwert belegen. Um zu sehen, ob diese Konfiguration gesetzt ist, bitte den letzten Absatz im Kapitel über lokale Server lesen.

Eine zweite Konfiguration bezieht sich auf die "magic quotes". Anführungszeichen sind bei der Datenübertragung immer ein heikles Thema (mehr dazu im Thema über die Variablen). Auch wenn es im obigen Beispiel nicht wahrscheinlich ist, dass man zum Beispiel beim Namen ein Anführungszeichen verwendet, sollte das Programm doch auf diese Eventualität vorbereitet sein, um keine böse Überraschung zu erleben. "magic quotes" sagt aus, ob die übergebenen Formulardaten automatisch maskiert (escapet) werden, das heißt, ob aus einem Anführungszeichen \" wird. Auch hier bietet sich unter Umständen wieder eine Angriffsfläche für Hacker, insbesondere später bei der Verwendung einer Datenbank.

In unserem Beispiel brauchen wir jedoch diese Maskierung mit dem Backslash davor nicht. Abhängig von der Konfiguration des Servers werden wir diese Zeichen mittels der Funktion

stripslashes wieder entfernen. Nur wenn man genau die Serverkonfiguration kennt, kann man sich den Aufwand sparen. Man sollte jedoch auch bedenken, dass man seine Homepage einmal auf einen anderen Server übertragen möchte oder muss. Üblicherweise hat man bei den günstigeren Angeboten keinen Zugriff auf die PHP-Konfiguration und ist auf die Voreinstellung angewiesen.

Die trim-Funktion sorgt dafür, dass Leerzeichen am Anfang und am Ende entfernt werden. So können wir dann auch leere Eingaben filtern, wenn nur ein Leerzeichen in ein Formularfeld eingegeben wurde.

Einfach ausgedrückt bedeuten die ersten Zeilen der Auswertung: Lese die übergebenen Post-Daten ein und speichere sie in den drei Variablen. Die Leerzeichen am Anfang und Ende sind auf jeden Fall zu entfernen, die Backslashs bei Bedarf.

Die nachfolgenden Zeilen der Fehlerüberprüfung sind weniger spektakulär. Wenn die drei Variablen Name, E-Mail-Adresse und Nachricht leer sind, wird ein entsprechender Text zur Variablen $fehler hinzugefügt.

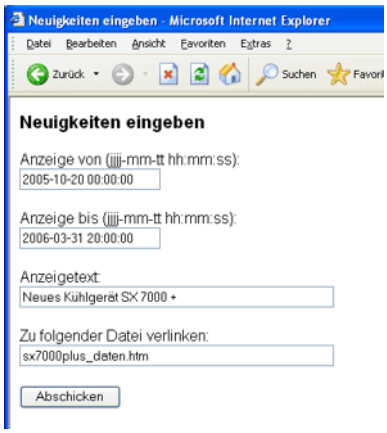
Diese Variable $fehler machen wir uns jetzt auch zu Nutze. Wurde kein Fehlertext hinzugefügt, wenn also die Variable noch leer ist, können die Formulardaten jetzt endlich per E-Mail verschickt werden. Voraussetzung ist, dass diese Funktion auf dem Server erlaubt und auch richtig eingestellt ist, was aber im Allgemeinen kein Problem darstellt. Zum Verschicken von E-Mails ist nur eine einzige Funktion notwendig. Die mail-Funktion erwartet als Parameter die Empfänger-Adresse (in diesem Fall die eigene), die Betreffzeile, die Nachricht selbst und den Kopf der E-Mail (Header). Im Nachrichtentext werden Zeilenumbrüche durch \n erreicht. Im Header wird zum einen die Absenderadresse (from) übergeben und zum anderen der Typ des Inhalts (content-type), in diesem Fall reiner Text, der für ein Kontaktformular ausreichend ist. Noch ein Wort zum Absender: Sicherlich kann man hier auch eine gefälschte Adresse angeben, allerdings lässt sich die wahre Herkunft nicht ohne Weiteres verschleiern.

Interessant ist auch die header-Funktion und in diesem Beispiel in Verbindung mit "location:neue Adresse". Hiermit wird eine Weiterleitung auf dem Server erzeugt. Die Seite wird hier zu kontakt_danke.htm weitergeleitet und nur diese bekommt der Benutzer hinterher zu sehen. Wichtig ist, dass vor der header-Funktion noch nichts an den Browser übergeben wurde, das heißt, kein HTML-Befehl und nicht einmal eine leere Zeile. Die weitere Bearbeitung endet hier mit der die-Funktion.

Bleibt zum Schluss nur noch die Möglichkeit, dass etwas in $fehler steht, wenn nicht alle Felder ausgefüllt wurden. In diesem Fall sollen die anderen Daten, die eingegeben wurden, auch wieder in die Felder des Formulars geschrieben werden. Doch auch dieses Mal machen uns wieder die Sonderzeichen einen Strich durch die Rechnung. Stellen wir uns einmal die zwar unwahrscheinlichen, aber dennoch nicht ausgeschlossenen Beispiele vor, dass jemand vielleicht seinen Spitznamen in besonderer Form eingibt: Franz "Franky" Müller oder Franz <Franky> Müller. In diesem Fall würde später im HTML-Quelltext stehen: <input type="text" name="name" value="Franz "Franky" Müller" size="30"> oder auch mit value="Franz <Franky> Müller". Diese besonderen Varianten kann der Browser nicht verarbeiten, weswegen wir die Daten mit der htmlspecialchars-Funktion vorher aufbereiten. Alle Sonderzeichen werden zu HTML-spezifischen Darstellungen umgewandelt. Unter anderem wir das Anführungszeichen zu ", das Kleinerzeichen zu < und das Größerzeichen zu >. Der Benutzer sieht davon auf der Seite gar nichts.

2. Datenbanken

Die Aufgabe: Wir wollen auf einer Internetseite Neuigkeiten präsentieren. Die Neuigkeiten werden in einer Datenbank gespeichert und können über weitere Seiten hinzugefügt und gelöscht werden. Zudem sollen alle Neuigkeiten zeitgesteuert eingeben werden können, das heißt, wir geben einen Anfang- und einen Endtermin für die Anzeige ein.



Doch beschäftigen wir uns zunächst einmal mit den Datenbanken an sich. Eine Datenbank ist eine Sammlung von Tabellen. So könnte die Homepage in diesem Beispiel neben den Neuigkeiten noch Tabellen für Benutzerdaten, Produkte und Beiträge eines Forums enthalten. Eine **Tabelle** hingegen enthält Datensätze gleicher Art. Das bedeutet, für unser Beispiel steht ein Datensatz für eine anzuzeigende Neuigkeit. Und dieser **Datensatz** ist immer gleich aufgebaut: laufende Nummer (Index), Beginn-Datum, Ende-Datum, der angezeigte Text und der Link zu einer Seite mit weiterführenden Informationen. In unserem Beispiel steht hinter der Neuigkeit "mehr", was mit dem Link verknüpft ist (siehe erste Bildschirmkopie).

Jedes einzelne **Datenfeld** eines Datensatzes muss von einem bestimmten **Datentyp** sein, zum Beispiel Text, Zahl, Datum/Uhrzeit oder auch ganze Objekte wie Grafiken. Man sollte sich vorher gut Gedanken über die Struktur einer Tabelle machen, um später nicht alle entsprechenden Stellen im PHP-Code ändern zu müssen.

Zuerst einmal wollen wir eine neue Tabelle anlegen. Üblicherweise stellt der Anbieter des Web-Speicherplatzes das Modul phpMyAdmin zur Verfügung. Mit phpMyAdmin kann man bequem die einzelnen Datenfelder aus Auswahllisten aussuchen. Wenn auf dem System kein phpMyAdmin oder ein ähnliches Modul installiert ist, muss die Tabelle über einen MySQL-Befehl eingegeben werden.

Wie weiter oben bereits beschrieben wurde, brauchen wir für unser Beispiel fünf Datenfelder. Die laufende Nummer (Feldname id) und auch die beiden Datumsangaben (Feldnamen von und bis) werden als Zahl (int) gespeichert. Für die Datumsangaben verwenden wir den so genannten Unix-Timestamp. Dabei werden die Sekunden ab dem 1. Januar 1970 um exakt 0 Uhr (Zeitzone GMT) gezählt. Sowohl MySQL als auch PHP stellen Funktionen zum einfachen Umwandeln bereit. Der Anzeigetext und der Link sind vom Typ tinytext (Es gibt Text-Datentypen unterschiedlicher Länge).

Die laufende Nummer bekommt die Eigenschaft "auto_increment", so dass dieses Feld automatisch hochgezählt wird. Es kann keine Nummer doppelt vergeben werden. Dies gilt auch für Nummern bereits gelöschter Datensätze.

Nachfolgende Bildschirmkopie zeigt, wie die fertige Tabelle in phpMyAdmin aussieht. Darunter steht der SQL-Befehl (es ist nur einer über mehrere Zeilen verteilt), der genau diese Tabelle erzeugt.

SQL-Befehl zum Erstellen der neuen Seite (Alternative zu manuellen Auswahl):

```
CREATE TABLE `neuigkeiten` (
  `id` int(11) unsigned NOT NULL auto_increment,
  `von` int(11) default NULL,
  `bis` int(11) default NULL,
  `anzeige` tinytext NOT NULL,
  `link` tinytext NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM;
```

Wenn jetzt alles richtig gemacht wurde, dann wird phpMyAdmin erst einmal nicht mehr benötigt. Wenden wir uns einmal den Dateien unseres Projekts zu.

Zum Schutz vor unbefugtem Zugriff werden wir alle (vier) Dateien bis auf die Startseite in ein Unterverzeichnis speichern. Dieses Unterverzeichnis heißt in unserem Beispiel "admin" und sollte am besten mit einem Passwortschutz versehen werden. Dazu bieten meistens die Anbieter des Web-Speicherplatzes eine einfache Möglichkeit an. Etwas aufwendiger ist es, wenn man wie hier: http://www.drweb.de/htaccess/zugriffsschutz_1.shtml die Datei .htaccess selbst erzeugt. Aber die Sicherheit sollte diese einmalige Arbeit wert sein.

Hier folgt die erste Datei. Weiter oben bei der Beschreibung von Funktionen wurde bereits gesagt, dass Funktionen auch in separate Dateien ausgelagert werden können. Das soll jetzt einmal geschehen, denn diese Funktion wird von alle anderen drei Dateien benötigt.

Datei: zugang.php

```
<?php
  function verbindedatenbank() {
    $host=      'localhost';
    $benutzer=  '...';          // eigene Daten einsetzen!
    $datenbank= '...';
    $passwort=  '...';
```

```
    $verbindung=@mysql_connect($host,$benutzer,$passwort);
    if ($verbindung) {
      mysql_select_db($datenbank);
      return $verbindung;
    }
    else {
      echo '<h3>Fehler bei der Verbindung zur Datenbank. ';
      echo 'Bitte noch einmal versuchen!</h3>';
      die();
    }
  }
?>
```

Auch wenn diese Datei später Teil einer anderen PHP-Datei wird, muss trotzdem noch einmal der Inhalt als PHP-Teil gekennzeichnet werden. Würde man diese Datei vom Browser aus aufrufen (angenommen, die Datei steht nicht in einem passwortgeschützen Unterverzeichnis), würde nichts passieren, da die Funktion hier nicht aufgerufen wird.

In diese Datei müssen die eigenen Benutzerdaten für MySQL eingegeben werden, die vom Anbieter bereitgestellt werden. In dieser Funktion wird versucht, eine Verbindung herzustellen und die richtige Datenbank auszuwählen. Gelingt dies nicht, wird eine Meldung für den Benutzer ausgegeben und die komplette Bearbeitung, also auch die der aufrufenden Datei, abgebrochen.

Datei: startseite.php

```
<?php
  include('admin/zugang.php');
  verbindedatenbank();
  $neuigkeiten='';
  $abfrage=mysql_query("SELECT * FROM neuigkeiten WHERE UNIX_TIMESTAMP()
    BETWEEN von AND bis ORDER BY id DESC");
  if (mysql_num_rows($abfrage)==0)
    $neuigkeiten='Es gibt leider keine Neuigkeiten!';
  else {
    while ($datensatz=mysql_fetch_assoc($abfrage)) {
      if ($neuigkeiten!='')
        $neuigkeiten.='<br>';
      $neuigkeiten.=htmlspecialchars($datensatz['anzeige']).' ';
      $neuigkeiten.='<a href="'.htmlspecialchars($datensatz['link']).'">[mehr]</a>';
    }
  }
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
    <title>Mega Cooling</title>
    <style type="text/css">
      p.neues {
        background-color:#E0E0FF;
        text-size:9pt; color:#000080;
        padding:10px; border:2px solid #C0C0FF;
       }
    </style>
  </head>
  <body>
    <h2>Willkommen auf der Seite von Mega Cooling</h2>
    <p class="neues">
      <?php echo $neuigkeiten; ?>
```

```
    </p>
  </body>
</html>
```

Die Startseite versucht erst einmal, sich mit der Datenbank zu verbinden. Dazu wird die Datei zugang.php im untergeordneten Verzeichnis eingebunden und die entsprechende Funktion aufgerufen.

Der Befehl mysql_query schickt eine Anfrage an die Datenbank, in diesem Fall eine Select-Anfrage, um Daten zu erhalten. (Groß- und Kleinschreibung spielt bei den vordefinierten Wörtern innerhalb einer Anfrage keine Rolle.) Das Sternchen hinter Select steht für alle Felder, in unserem Beispiel also alle von id bis link. Der From-Teil gibt an, woher die Daten kommen sollen, denn in der Praxis ist es wahrscheinlich, mehr als nur eine Tabelle zu haben. Der interessantere Teil ist der Where-Teil, in dem die Bedingung steht, nach der die Daten gefiltert werden. Wir erinnern uns, dass wir die Datumsangaben als fortlaufende Sekunden-Zahl (Unix-Timestamp) gespeichert haben. Die MySQL-Anweisung Unix-Timestamp ohne einen Parameter gibt das aktuelle Datum mit der aktuellen Zeit zurück. Order by sortiert nach einem bestimmten Feld und in diesem Fall absteigend (Desc).

Damit lautet die Abfrage in unserem Beispiel: Wähle alle Felder aus der Tabelle neuigkeiten aus, bei denen die aktuelle Zeit zwischen den Daten in den Feldern von und bis liegt, und sortiere die Ergebnisse anschließend so, dass die neuesten Eingaben oben stehen. Im Hintergrund wir nun eine neue Tabelle erzeugt, die nur diese ausgewählten Datensätze in der angegebenen Reihenfolge enthält. Der Schlüssel für die neue Tabelle wird in $abfrage gespeichert, damit wir darauf weiter zugreifen können. Diese nicht sichtbare Tabelle bleibt bis zum Ende des Scripts erhalten. Der Schlüssel ist wichtig, da selbstverständlich auch mehrere Abfragen gleichzeitig gestartet werden können.

Mysql_num_rows($abfrage) liefert die Anzahl der Datensätze zurück. Wenn bei unseren Neuigkeiten mindestens ein Datensatz gefunden wird, speichern wir die Ergebnisse in $neuigkeiten. Jeden einzelnen Datensatz bekommen wir über mysql_fetch_assoc($abfrage). Diese Funktion sorgt dafür, dass der ganze Datensatz in einem Array gespeichert wird. Als Index für das Array dient der Feldname. Zum Beispiel können wir später über $datensatz["anzeige"] auf den Anzeigetext des aktuellen Datensatzes auswählen. Nachdem ein Datensatz über mysql_fetch_assoc abfragt wurde, wird automatisch der interne Zähler erhöht, so dass eine erneute Abfrage auf den nächsten Datensatz zugreift.

Die Daten für die angezeigten Text und den Link werden vorab wieder mit htmlentities bearbeitet. Dazu wurde bereits am Ende des Kontaktformular-Beispiels etwas geschrieben.

Die Anzeige der vorhandenen Daten ist noch relativ einfach. Man könnte diese Daten auch direkt in der Datenbank-Tabelle speichern, bearbeiten und löschen. Doch für dieses Beispiel erstellen wir eigens Seiten dafür.

Datei: neuigkeiten_eingabe.php

```
<?php
  include('zugang.php');
  verbindedatenbank();
  $von=''; $bis='';
  $anzeige=''; $link='';
  $fehler=''; $info='';
  if ($_POST['gesendet']=='1') {
    if (get_magic_quotes_gpc()) {
```

```
      $von=trim(stripslashes($_POST['von']));
      $bis=trim(stripslashes($_POST['bis']));
      $anzeige=trim(stripslashes($_POST['anzeige']));
      $link=trim(stripslashes($_POST['link']));
    }
    else {
      $von=trim($_POST['von']);
      $bis=trim($_POST['bis']);
      $anzeige=trim($_POST['anzeige']);
      $link=trim($_POST['link']);
    }
    if ($von=='' or $bis=='' or $anzeige=='' or $link=='')
      $fehler='Es müssen alle Felder ausgefüllt werden!';
    if ($fehler=='') {
      mysql_query("INSERT INTO neuigkeiten SET
        von=UNIX_TIMESTAMP('".mysql_real_escape_string($von)."'),
        bis=UNIX_TIMESTAMP('".mysql_real_escape_string($bis)."'),
        anzeige='".mysql_real_escape_string($anzeige)."',
        link='".mysql_real_escape_string($link)."'");
      $info='Daten wurden eingetragen!';
      $von=''; $bis=''; $anzeige=''; $link='';
    }
    else {
      $von=htmlspecialchars($von);
      $bis=htmlspecialchars($bis);
      $anzeige=htmlspecialchars($anzeige);
      $link=htmlspecialchars($link);
    }
  }
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
    <title>Neuigkeiten eingeben</title>
  </head>
  <body>
    <h3>Neuigkeiten eingeben</h3>
    <?php
      if ($fehler!='')
        echo '<p style="color:#C00000">'.$fehler.'</p>';
      if ($info!='')
        echo '<p style="color:#00C000">'.$info.'</p>';
    ?>
    <form action="neuigkeiten_eingabe.php" method="post">
      <input type="hidden" name="gesendet" value="1">
      <p>Anzeige von (jjjj-mm-tt hh:mm:ss):<br>
        <input type="text" name="von" value="<?php echo $von; ?>" size="20">
      </p>
      <p>Anzeige bis (jjjj-mm-tt hh:mm:ss):<br>
        <input type="text" name="bis" value="<?php echo $bis; ?>" size="20">
      </p>
      <p>Anzeigetext:<br>
        <input type="text" name="anzeige" value="<?php echo $anzeige; ?>" size="50">
      </p>
      <p>Zu folgender Datei verlinken:<br>
        <input type="text" name="link" value="<?php echo $link; ?>" size="50">
      </p>
      <p><input type="submit" value="Abschicken"></p>
    </form>
```

```
  </body>
</html>
```

Na, kommt der Aufbau bekannt vor? Im Großen und Ganzen unterscheidet sich diese Datei nicht viel von dem Kontaktformular im vorherigen Beispiel. Beim ersten Aufruf wird der gesamte Teil der Auswertung in PHP übersprungen. Im HTML-Teil folgt ein Formular zur Eingabe der vier Daten Anfang-Datum, Ende-Datum, Text und Link. Es müssen wieder alle Felder ausgefüllt werden.

Die Datumsangaben werden hier in einer bestimmten Form erwartet. Da es in diesem Kapitel vorrangig um Datenbanken geht, erfolgt keine oder Fehlerprüfung. Falscheingaben haben zur Folge, dass 0 in der Datenbank gespeichert wird und somit keine Ausgabe auf der Startseite erfolgt. Wer weitere Informationen zur Überprüfung und Umwandlung spezieller Eingabeformate sucht, sollte sich einmal im Internet oder der Literatur nach den komplexen, aber äußerst effizienten "regulären Ausdrücken" umsehen.

Anstelle der E-Mail im Kontaktformular-Beispiel wird hier wieder eine SQL-Abfrage gestartet. Nicht nur das Auslesen von Daten, sondern auch das Einfügung und Löschen nennt man in SQL eine Abfrage und deswegen wird auch der gleiche Befehl mysql_query verwendet. Diese Abfrage sollte wieder nur aus einer einzelnen Zeile bestehen, aber das Papier ist leider nicht breit genug.

Insert into bedeutet einfügen und eingefügt wird in die Tabelle neuigkeiten. Nach dem Set folgen die einzelnen Zuweisungen der Werte zu den Feldern, jeweils mit einem Gleichheitszeichen dazwischen und einem Komma zwischen den Einträgen. Die fortlaufende Nummer (id) braucht - oder besser gesagt: darf - in diesem Fall nicht weiter berücksichtigt werden, da diese automatisch mit dem nächst verfügbaren Wert belegt wird. Die einzelnen Werte stehen immer in einfachen Anführungszeichen. Vorsichtshalber maskieren wir die Eingaben mit der bekannten Funktion mysql_real_escape_string. Damit wird den Sonderzeichen wieder ein Backslash vorangestellt und eine bewusst oder unbewusst falsche Eingabe vermieden. Die Eingaben von Datum und Uhrzeit werden mit der MySQL-Funktion Unix-Timestamp in das Format der fortlaufenden Sekundenzählung gebracht (deswegen auch das festgeschriebene Format der Eingabe).

Die Variablen aus dem Formular werden anschließend gelöscht, damit sie nicht noch einmal angezeigt werden, und ein Hinweistext wird ausgegeben.

Datei: neuigkeiten_loeschen.php

```
<?php
  include('zugang.php');
  verbindedatenbank();
  $info='';
  $loescheindex=$_GET['loesche'];
  if ($loescheindex!='') {
    $loescheindex=intval($loescheindex);
    if ($loescheindex==0)
      $info='Ungültige Eingabe';
    else {
      mysql_query("DELETE FROM neuigkeiten WHERE id='".$loescheindex."'");
      $info="Neuigkeit Nr. ".$loescheindex." gelöscht!";
    }
  }
  function alleanzeigen() {
    $abfrage=mysql_query("SELECT * FROM neuigkeiten ORDER BY id ASC");
    if (mysql_num_rows($abfrage)==0)
      echo '<p>Keine Daten vorhanden!</p>';
    else
```

```
      while ($datensatz=mysql_fetch_assoc($abfrage)) {
        echo '<p>Lfd. Nummer: '.$datensatz['id'].'<br>';
        echo 'Anzeige von: '.date('d.m.Y H:i',$datensatz['von']).'<br>';
        echo 'Anzeige bis: '.date('d.m.Y H:i',$datensatz['bis']).'<br>';
        echo 'Text: <b>'.htmlspecialchars($datensatz['anzeige']).'</b><br>';
        echo 'Link: '.htmlspecialchars($datensatz['link']).'<br>';
        echo '<a href="?loesche='.$datensatz['id'].'">Löschen!</a></p>\n';
      }
  }
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
    <title>Neuigkeiten löschen</title>
  </head>
  <body>
    <h3>Neuigkeiten löschen</h3>
    <?php
      if ($info!='')
        echo '<p style="color:#C00000">'.$info.'</p>';
      alleanzeigen();
    ?>
  </body>
</html>
```

Das Grundgerüst ist wieder einmal dasselbe. Am Anfang steht die Auswertung gefolgt von einer Funktion, die für die Anzeige der bereits eingegeben Daten zuständig ist. Die Funktion alleanzeigen() wird im Hauptteil aufgerufen.

Das Auslesen der Daten ist bereits von der startseite.php bekannt. Jedoch wird als letzter Punkt für jeden Datensatz ein Link "Löschen!" angefügt. Dieser Link führt zu "?loesche=" und dahinter die fortlaufende Nummer (id) des Datensatzes. Da hier der eigentliche Dateiname fehlt, ergänzt der Browser die Datei und setzten den Teil ab dem Fragezeichen hinten an. Bei den Formularen haben wir immer die Post-Methode gesehen. Doch nun verwenden wir einmal die Get-Methode, bei der die Daten mittels dem URL weitergegeben werden. Generell sieht das Format folgendermaßen aus: datei.php?parameter1=wert1&parameter2=wert2&parameter3=wert3... In unserem Beispiel benötigen wir nur den einen Parameter (loesche) und einen Wert (Nummer des Datensatzes).

Im Teil der Auswertung wird der Parameter loesche mittels $_GET abgefragt. Wenn dieser Parameter vorhanden ist, dann wird er vorsorglich der intval-Funktion unterzogen. Intval sorgt dafür, dass eine Zahl aus einem String erzeugt wird. Zum Beispiel "1" bleibt 1, aus "2ab" wird 2 und "c3d" ist ungültig und wird somit 0. Wir können also davon ausgehen, dass das Ergebnis 0 eine falsche Eingabe bedeutet. Auch der automatische Zähler in der Datenbank vergibt die Nummer 1 für den ersten Datensatz.

Nun folgt wieder eine MySQL-Abfrage. Der Aufbau zum Löschen ist der zum Auslesen ähnlich. Aus der Tabelle neuigkeiten sollen die Datensätze gelöscht werden, bei denen id dem mit der Get-Methode übergebenen Wert entspricht. der Wert muss wieder in einfachen Anführungszeichen stehen. In diesem Fall ist die mysql_real_escape_string-Funktion wie bei der Neueingabe nicht erforderlich, da wir bereits sichergestellt haben, dass es sich nur um eine Zahl handelt.

In diesem Beispiel wurde nicht überprüft, ob ein Datensatz mit der geforderten Nummer wirklich existiert. Denn es könnte zum Beispiel jemand über einen direkten Aufruf im Browser mit

neuigkeiten_loeschen.php?loesche=148 versuchen, den 148. Eintrag zu löschen. Aber erstens befindet sich diese Datei - wenn man dem Rat oben befolgt hat - in einem geschützten Unterverzeichnis und zweitens kann man keine nicht existierenden Daten löschen. Eine Fehlermeldung von MySQL oder PHP gibt es nicht. Die Anzeige, dass ein Datensatz nicht existiert, ist lediglich ein weiterer Schritt zum Nutzerkomfort.

3. Lokaler Server

Bisher war der Server immer nur über das Internet zu erreichen. Man überträgt die PHP-Dateien und alle anderen Dateien online zu dem Anbieter. Doch wäre es nicht schön, diese vorab ohne solchen Aufwand zu testen? Das ist kein Problem, es muss nur ein lokaler Server eingerichtet werden. Wie das Wort lokal schon sagt, ist es also ein Server, der sich bei einem selbst auf der Festplatte befindet.

Hier ist leider ein wenig Arbeit - einmalig - notwendig. Aber um es gleich vorweg zu nehmen, ist ein lokaler Server kein Muss. Man kann auch erst einmal dieses Kapitel ruhigen Gewissens überspringen und später darauf zurückkommen.

Wenn wir unsere Dateien lokal testen wollen, müssen wir die gleiche Umgebung wie auf dem Server des Anbieters herstellen. Das heißt, wir brauchen einen Server, das PHP-Modul und die Datenbank.

Die einzelnen Module müssen aus dem Internet heruntergeladen und installiert werden. Wie zuvor schon mehrfach beschrieben wurde, können verschiedene Umgebungen eingerichtet werden, aber wir wollen uns einmal auf die gebräuchlichsten beschränken:

- Apache-Server: http://www.apache.org
- PHP-Modul: http://www.php.net
- MySQL-Datenbank: http://www.mysql.com

Doch halt! Wenden wir uns vorher einmal einer anderen Seite zu. Es gibt mittlerweile Systeme, die alle benötigten Module auf einmal installieren. Dies ist für Anfänger sicherlich der leichtere Weg. Zum Beispiel wird unter http://www.apachefriends.de das kostenlose Produkt **XAMPP** angeboten, das die zuvor genannten Module und noch andere Erweiterungen enthält, die später einmal notwendig sind. Wenn man sich zu dem Newsletter anmeldet, wird man ständig über neue Versionen auf dem Laufenden gehalten. XAMPP wird regelmäßig bei Erscheinen von neueren Modulen aktualisiert.

Und was macht man nun mit einem lokalen Server? Die eigenen Dateien müssen in einem besonderen Verzeichnis gespeichert werden (das man mit einigen Kenntnissen auch ändern kann). In Fall von XAMPP von Apachefriends ist es das Unterverzeichnis htdocs. Am besten legt man für jedes Projekt dort ein eigenes Verzeichnis an. Jetzt können wir die eigene mit PHP erstellte Seite auch von unserer Festplatte aufrufen, und zwar mit - zum Beispiel -

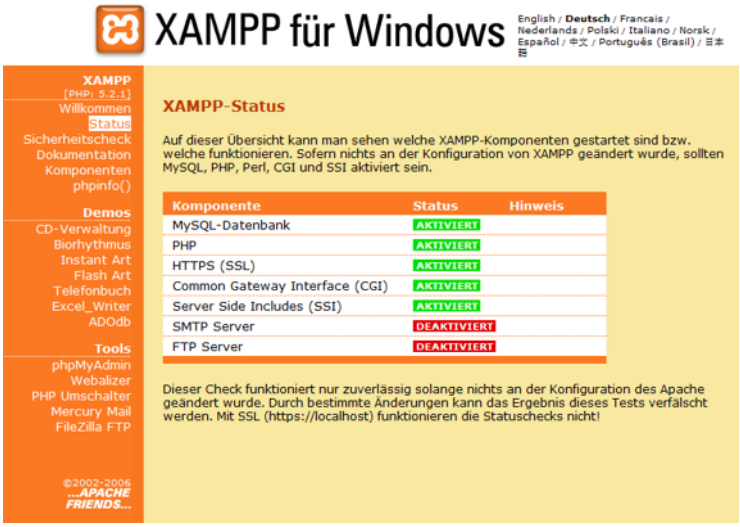
- http://localhost/unterverzeichnis/index.php

Wenn man weiter ins Detail geht und besondere Funktionen ausführen möchte, müssen eventuell einige Optionen angepasst werden. Dieses würde aber für einen Einsteigerkurs zu weit führen. Die einzelnen Konfigurationen, ob nun Apache, PHP oder ein anderes Modul, findet man in den entsprechenden Unterverzeichnissen des kompletten Programms.

Interessant ist aber noch eine PHP-Funktion, die uns alle Einstellungen auf einen Blick präsentiert. Dazu schreibt man am besten ein kleines PHP-Programm, das man auf den Server des Anbieters und auf den lokalen Server kopiert und so eine Vergleichsmöglichkeit hat:

Datei: phpinfo.php

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



1. Template-Systeme

Template ist das englische Wort für Vorlage oder Schablone. Bisher haben wir in diesem Kurs gesehen, dass PHP und HTML in einer Datei gemischt vorgekommen sind. Entweder haben wir PHP- und HTML-Teile abwechselnd geschrieben oder auch von PHP HTML-Code mittels dem echo-Befehl erzeugen lassen. Das muss aber nicht unbedingt so sein. Denn es gibt auch im Internet (frei verfügbare) Template-Systeme.

Der Vorteil ist die klare Trennung von der Programmierung und der Seitengestaltung. Dies bringt in erster Linie Übersichtlichkeit. Aber ein weiterer positiver Effekt ist darin zu sehen, wenn mehrere Personen an einem Projekt arbeiten. Zum einen wertet der Programmierer die Eingaben (über einen Link, über ein Formular) aus und der Designer erstellt seine Seiten noch ohne konkrete Inhalte. Erst später werden diese leeren Seiten (deswegen auch Vorlagen) mit Hilfe von Variablen mit Leben gefüllt. Das komplette Handling wird dabei dem Template-System überlassen.

Gute Template-Systeme gehen aber noch weiter: Sie bieten Funktionen innerhalb der Vorlagen an, so dass zum Beispiel auf einfache Weise Arrays durchlaufen werden können und eine Tabelle erstellt werden kann (ansonsten müsste die Tabelle von PHP kommen). Auch können bereits verarbeitete Seiten zwischengespeichert werden (Cache), damit Seiten, die nur in längeren Abständen aktualisiert werden müssen, schneller geladen werden können.

Ein solches Template-System findet man zum Beispiel auf der Seite http://smarty.php.net. Ein Beispiel mit Smarty soll - stellvertretend für alle Systeme - die Arbeitsweise verdeutlichen. In einem Online-Shop sollen Waren einer bestimmten Kategorie angezeigt werden:

Datei: warenanzeigen.php

```
<?php
  define('SMARTY_DIR','smarty/libs/');
  include(SMARTY_DIR.'Smarty_class.php');
  $smarty=new Smarty;
  $smarty->template_dir='./smarty/templates';
  $smarty->compile_dir='./smarty/templates_c';

  // die folgenden Daten würden natürlich aus einer Datenbank kommen,
  // der Einfachheit und Übersichtlichkeit halber die Version mit Arrays;
  // konzentrieren wir uns auf die Vorlagen

  $produkte[]=array('bild'=>'mixer.gif','beschreibung'=>
                    'Mixer Turbo 2005','preis'=>'49,95 EUR');
  $produkte[]=array('bild'=>'kaffee.gif','beschreibung'=>
                    'Kaffeemaschine Super Plus','preis'=>'94,90 EUR');
  $produkte[]=array('bild'=>'brotback.gif','beschreibung'=>
                    'Brotbackautomat B3','preis'=>'66,00 EUR');

  $smarty->assign('kategorie','Haushaltswaren');
  $smarty->assign('produkte',$produkte);
  $smarty->display('warenuebersicht.tpl');
?>
```

Datei: warenuebersicht.tpl

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
  <head>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
    <title>Online-Shop</title>
  </head>
  <body>
    <h1>Online-Shop</h1>
    <h2>Kategorie: {$kategorie}</h2>
    <table border="1">
      <tr>
        <td>Produktbild</td>
        <td>Beschreibung</td>
        <td>Preis</td>
      </tr>

      {foreach name=produktliste item=produkt from=$produkte}
      <tr>
        <td><img src="{$produkt.bild}" alt=""></td>
        <td>{$produkt.beschreibung}</td>
        <td>{$produkt.preis}</td>
      </tr>
      {/foreach}

    </table>
  </body>
</html>
```

In der PHP-Datei werden vorab ein paar Einstellungen vorgenommen, damit die entsprechenden Unterverzeichnisse bekannt sind. template_dir steht für das Verzeichnis, in dem die Vorlagen gespeichert sind (könnte natürlich auch in einem Verzeichnis mit den PHP-Dateien stehen) und in compile_dir sind bereits vom System vorbereitete PHP-Dateien zu finden. Anschließend werden die Variablen zugewiesen (engl. assign), die später in die Vorlage übernommen werden sollen. In unserem Beispiel handelt es sich um einen einzelnen Text, nämlich die Kategorie, und um ein mehrfaches (mehrdimensionales) Array; drei Datensätze jeweils mit einem Bild (Datei), einer Beschreibung und einem Preis. Die Beschaffung der Daten aus einer Datenbank ist ein anderes Thema. Zum Schluss teilen wir Smarty noch mit, welche Vorlage verwendet werden soll. Die Dateiendung der Template-Datei spielt keine Rolle.

In der Vorlage findet sich die Kategorie in Form von {$kategorie} wieder. Für die Schleife, also das Durchlaufen des Produkt-Arrays, verwendet Smarty einen eigenen Befehl, nämlich foreach. Der foreach-Befehl bezieht sich wieder auf die Variable $produkte, die im PHP-Code vergeben wurde, und kann bei jedem Durchlauf auf die einzelnen Datenelemente zugreifen. Was jetzt noch recht übersichtlich aussieht, wird später im Quelltext der kompletten HTML-Seite eine größere Tabelle. Für jeden Datensatz wird ein <tr>...</tr>-Block angelegt.

Nun ist es für den Designer der HTML-Seite auch ein Leichtes, ohne Eingriff in die Programmierung das Aussehen der Seite zu ändern:

Auszug aus einer möglicherweise überarbeiteten Datei: warenuebersicht.tpl

```
<h2>Kategorie: {$kategorie}</h2>

{foreach name=produktliste item=produkt from=$produkte}
<img src="{$produkt.bild}" alt=""><br>
{$produkt.beschreibung} ..... <b>jetzt nur {$produkt.preis}</b><br>
<hr>
{/foreach}
```

Wie gesagt, Smarty ist nur eines von verfügbaren Template-Systemen und das vorgestellte Beispiel bezieht sich speziell hierauf. Andere Systeme arbeiten auf ähnliche Weise.

Wer gerne das Rad neu erfinden möchte: Es muss kein fertiges Template-System sein. Hier folgt das einfachste Template-System. Wer allerdings ernsthaft damit arbeiten möchte, wird schnell den Komfort vermissen und doch viel Arbeit aufwenden müssen.

Die Vorlage: hauptseite.tpl

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
    <title>Die Erfindung des Templates</title>
  </head>
  <body>
    <h2>$ausgabe</h2>
  </body>
</html>
```

... wird verarbeitet mit: index.php

```
<?php
  $dateiname='hauptseite.tpl';
  $datei=fopen($dateiname,'r');
  $inhalt=fread($datei,filesize($dateiname));

  $ausgabe='Mit Vorlage erstellt!';  // dieser Text soll später erscheinen

  eval("\$inhalt=\"$inhalt\";");
  echo $inhalt;
?>
```

Wie erwartet, erscheint im Browser der Text "Mit Vorlage erstellt!" anstelle der Variablen $ausgabe. Die ersten drei Zeilen in der index.php laden nur die komplette Datei (Vorlage) in $inhalt. Interessant ist der eval-Befehl, eine Kurzform des englischen Worts evaluate, also auswerten. Die Variable $inhalt wird ausgewertet, das heißt, auch die Variablen in ihr werden eingefügt. Anschließend wird der gesamte Ausdruck wieder unter gleichem Namen gespeichert und ausgegeben. (Eval ist wieder ein solcher Fall, wo man mit dem falschen Setzen von Anführungszeichen und Backslashes eine Menge Verwirrung erreichen kann.)

Aber das Ziel eines Template-Systems ist erreicht: HTML und PHP sind getrennt!

Und noch einen Tipp zum Schluss: Mit Template-Systemen ist es auch ganz einfach möglich, mehrsprachige Seiten zu erstellen:

Auszug aus der HTML-Datei:

```
<a href="hauptseite.php?sprache=de">Deutsche Seite</a> -
<a href="hauptseite.php?sprache=en">English page</a> -
<a href="hauptseite.php?sprache=es">Página española</a> -
<a href="hauptseite.php?sprache=fr">Page française</a>
```

Auszug aus der PHP-Datei:

```
switch ($sprache) {
  case "en": $vorlage="en_hauptseite.tpl"; break;
  case "es": $vorlage="es_hauptseite.tpl"; break;
  case "fr": $vorlage="fr_hauptseite.tpl"; break;
```

```
  default:   $vorlage="de_hauptseite.tpl";
}
```

Auszug aus de_hauptseite.tpl:

```
<h1>Guten Tag, {$benutzername}!</h1>
<p>Neue Nachrichten: {$neuenachrichten}</p>
```

Auszug aus en_hauptseite.tpl:

```
<h1>Good afternoon, {$benutzername}!</h1>
<p>New messages: {$neuenachrichten}</p>
```

Auszug aus es_hauptseite.tpl:

```
<h1>¡Buenos días, {$benutzername}!</h1>
<p>Nuevos mensajes: {$neuenachrichten}</p>
```

Auszug aus fr_hauptseite.tpl:

```
<h1>Bonjour, {$benutzername} !</h1>
<p>Nouveaux messages: {$neuenachrichten}</p>
```

Kleiner Nachteil dieser Methode ist, dass man bei späteren Design-Änderungen alle Template-Dateien überprüfen und korrigieren muss. Aber wie gesagt, es gibt noch mehr Template-Systeme und auch Möglichkeiten.

Stichwortverzeichnis