Einführung in PHP

Horst Gierhardt
horst@gierhardt.de

15.01.2014

Inhaltsverzeichnis

1 Was ist PHP?

PHP steht für *PHP Hypertext Preprocessor* und ist eine Skriptsprache zur Programmierung von dynamischen Webseiten. PHP ist frei verfügbar bzw. als *Open Source* erhältlich. PHP ist eine komplette Programmiersprache, deren Ausgabe sich allerdings auf das beschränkt, was mit HTML darstellbar ist.

PHP läuft nicht auf dem lokalen Computer (*Client*), sondern auf einem entfernten *Server* im Internet oder Intranet.

1.1 Erstes Beispiel

Der folgende Text sei in einer Datei `index.php` gespeichert:

```
<HTML>
  <HEAD>
    <TITLE>PHP-Beispiel</TITLE>
  </HEAD>

  <BODY>
  Dieser Text ist ganz normal in HTML geschrieben.<br>
  Jetzt kommt der PHP-Teil:<br>

    <script language="php">
    $zahl1=7;
    $zahl2=19;
    $summe=$zahl1+$zahl2;
    echo "Das Ergebnis ist $summe.";
    echo "<br>";
    </script>

  Hier geht es wieder mit fettgedrucktem <b>HTML</b> weiter.
  </BODY>
</HTML>
```

Wird diese Datei vom Browser direkt von einer Festplatte (bei uns z.B. von Laufwerk Q) geladen, so wird der mit `script` eingeschlossene Teil der Seite einfach ignoriert. Browser können mit PHP nichts anfangen, sondern verstehen nur HTML.

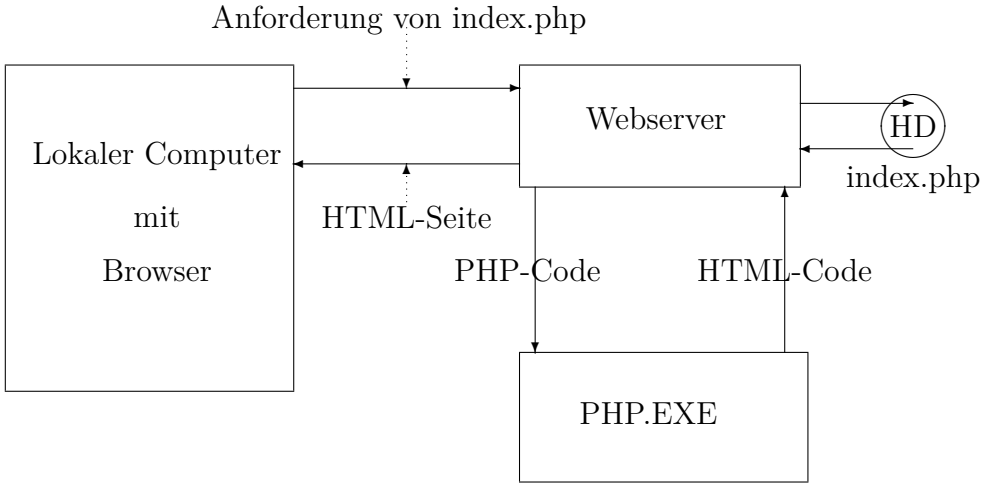
Speichert man die Datei `index.php` z.B. in `\\HP-Pro\Daten\htdocs\ThinkCentre05` ab, so erhält man bei uns durch die Browsereingabe `http://HP-Pro/ht/ThinkCentre05` ein ganz anderes Ergebnis. Nun wird das Ergebnis der Rechnung korrekt auf der Seite angezeigt. Betrachtet man den Quelltext der Seite im Browser, so ist von PHP nichts mehr zu sehen.

1.2 Wie kommt das Rechenergebnis auf die Browserseite?

Auf dem Intranet-Server (HP-Pro) ist ein Programm installiert, das man *Webserver* nennt. Dieses Programm hat die Aufgabe, Anforderungen der Webbrowser auf den Clients zu verarbeiten und diesen HTML-Seiten zurück zu senden. Der Webserver der Schule bedient allerdings nur Clients im lokalen Netzwerk, dem Intranet. Gibt man im Browser z.B. die

Adresse http://www.gymbala.de ein, so geht diese Anforderung an einen Webserver auf einem Computer in z.B. Karlsruhe.

Der Webserver holt die angeforderte Datei `index.php` von der Harddisk und erkennt an der Dateiendung `php`, dass PHP-Code enthalten sein muss. Diese Datei schickt er an das Programm `PHP.EXE`, das auf dem Server installiert ist. Dieses Programm verarbeitet und interpretiert bei Fehlerfreiheit den PHP-Code und schickt das Ergebnis als HTML-Code an den Webserver zurück. Fehlermeldungen gehen ebenso als HTML an den Webserver. Dieser schickt den HTML-Code an den anfordernden Client zurück, d.h. an den Browser. Demzufolge ist im Quelltext, der im Browser angezeigt wird, kein PHP mehr enthalten.



1.3 Was kann man mit PHP machen?

Im professionellen Bereich wird PHP zur Erzeugung dynamischer HTML-Seiten eingesetzt, d.h. die HTML-Seite wird erst bei Anforderung eines Clients auf dem Server erzeugt. Das wichtigste Einsatzgebiet ist wohl die Erzeugung von HTML-Seiten mit Ergebnissen einer Datenbankabfrage. Eine häufig auf Webservern benutzte Datenbank ist *MYSQL*, so dass man das Gespann PHP und MYSQL im Internet sehr häufig antrifft.

2 Grundlegende Sprachelemente von PHP

2.1 Grundlagen

2.1.1 Einbettung in HTML

Der PHP-Code wird in den HTML-Code durch

```
1 <script language="php">
2    echo "Das ist PHP.";
3 </script>
```

oder kurz mit

```
<?php
    echo "Das ist PHP.";
?>
```

eingebettet. Die ältere Variante `<? ... ?>` sollte nicht mehr benutzt werden. Die Ausgabe des echo-Befehls kann wieder HTML-Tags enthalten.

Achtung: Bei den HTML-Tags wird nicht zwischen Groß- und Kleinschreibung unterschieden. `<b>Fettdruck</b>` und `<B>Fettdruck</B>` führen zum gleichen Ergebnis. PHP ist aber *case sensitiv*, was eine häufige Fehlerquelle darstellt.

2.1.2 Kommentare

Kommentare werden wie in den Sprachen C++ oder Java behandelt:

```
<?php
    // Das ist ein Kommentar, der nur bis zum Zeilenende geht.
    echo "Das ist PHP."; // Alles, was hinter dem Doppelslash steht,
                         // wird ignoriert.
    /* Dieser Kommentar
       geht ueber
       mehrere Zeilen. */
?>
```

2.1.3 Inkludieren

Immer wieder verwendbare PHP-Codeteile kann man in eine externe Datei schreiben und mit z.B.

```
<?php
    include('bsp.php');
?>
```

einbinden.

2.2 Variablen und Konstanten

- Variablen werden durch ein vorangestelltes Dollarzeichen kenntlich gemacht. Beispiel: `$x=3;` [1] Ein Variablenname muss mit einem Buchstaben (Bitte keine Umlaute o.a. benutzen!) oder einem Unterstrich beginnen. Danach darf er Buchstaben, Ziffern oder den Unterstrich enthalten. Andere Sonderzeichen sind nicht erlaubt.

- Es sind keine Deklarationen von Variablen wie in vielen anderen Sprachen nötig. Durch das Verwenden einer Variablen wird sie erzeugt.

[1] Das Dollarzeichen deutet wahrscheinlich darauf hin, dass man mit PHP viel Geld im Internet verdienen kann.

- Der Variablentyp (Integer, String, double, ...) wird von PHP weitgehend automatisch erkannt, was aber manchmal zu recht merkwürdigen Ergebnissen führen kann:

```
<?php
   $x = 3;                    // ist Integer
   $y = $x + 7;               // ist auch Integer
   $z = "2 kleine Ferkel";    // ist ein String
   $a = $x + $z;              // ist Integer und hat den Wert 5.
   $g = 1.414;                // ist Double
   $h = $g + $x;              // wird Double
?>
```

- Strings werden durch doppelte Anführungszeichen eingegeben. Mehrere Strings können durch einen Punkt aneinandergefügt werden.

```
<?php
   $x = "Zwei";
   $y = " kleine Ferkel";
   $z = $x.$y
   $a = "Ene mene miste".                  // So kann man laengere Texte
        " es rappelt in der Kiste.";       // ueber mehrere Zeilen schreiben.
?>
```

- *Arrays* müssen längenmäßig nicht vorab begrenzt werden. Man schreibt einfach in sie hinein.

```
<?php
   $tier[0] = "Ferkel";
   $tier[1] = "Hund";
   $tier[2] = "Saurier";
?>
```

Man kann ein Array auch direkt mit allen Werten füllen:

```
<?php
   $jahreszeit = array("Fruehjahr", "Sommer", "Herbst", "Winter");
   echo $jahreszeit[1];  // liefert Sommer; Nummerierung beginnt bei 0.
?>
```

- *Konstanten* kann man auch deklarieren. Man lässt im Unterschied zu den Variablen das Dollarzeichen weg.

```
<?php
   define("AKTUELLESJAHR", 2004);
   echo AKTUELLESJAHR;                // Ausgabe ist 2004
?>
```

- Der Datentyp *Boolean* ist auch möglich:

```
<?php
   $MorgenIstSchulfrei = TRUE; // Zuweisung der Konstanten TRUE.
   $EsRegnet = FALSE;          // Zuweisung der Konstanten FALSE.
   if ($MorgenIstSchulfrei and $EsRegnet) // Zu if unten mehr
        { echo "Ich bleibe zu Hause."; }
?>
```

2.3 Ausgabe

Die Ausgabe geschieht mit der echo-Anweisung. Man kann Variablen direkt oder zusammen mit Text in doppelten Anführungszeichen ausgeben. Die doppelten Anführungszeichen führen dazu, dass der Inhalt der Variablen expandiert wird. Bei einfachen Anführungszeichen passiert das nicht. Einfache Anführungszeichen benutzt man nur bei einfachen Texten.

```
1 <?php
2     $x = 2;
3     echo "$x kleine Ferkel"; // Ausgabe ist: 2 kleine Ferkel
4     echo '$x kleine Ferkel'; // Ausgabe ist: $x kleine Ferkel
5 ?>
```

2.4 Operatoren

2.4.1 Mathematische Operatoren

Operator	Bedeutung	Beispiel
`+`	Addition	`$x = 23 + 3;  // ergibt 26`
`-`	Subtraktion	`$x = 29 - 3;  // ergibt 26`
`*`	Multiplikation	`$x = 2 * 13;  // ergibt 26`
`/`	Division	`$x = 260 / 10; // ergibt 26`
`%`	Ganzzahliger Rest	`$x = 56 % 30;  // ergibt 26`

Es gilt „Punkt- vor Strichrechnung“ wie sonst auch in der Mathematik. Ansonsten sind eben Klammern geeignet zu setzen.

Informationen zu mathematischen Funktionen folgen später.

2.4.2 Vergleichsoperatoren

Für den Anfänger verwirrend ist, dass es in PHP (und z.B. auch in C++ oder Java) im Gegensatz zum Gebrauch in der Mathematik einen Unterschied zwischen dem Zuweisungsoperator `=` und dem Vergleichsoperator `==` gibt. In der Mathematik gibt es ja keine Zuweisungen und eine Zeile wie `x = x + 1` ist einfach nur eine Gleichung mit der Lösungmenge $L = \{\}$.

Wir benutzen Vergleichsoperatoren in Kontrollstrukturen (siehe unten).

Operator	Bedeutung	Beispiel
`==`	Gleich	`if ($x == 23)`
`!=`	Ungleich	`if ($x != 23)`
`>`	Größer	`if ($x >  23)`
`<`	Kleiner	`if ($x <  23)`
`>=`	Größer oder gleich	`if ($x >= 23)`
`<=`	Kleiner oder gleich	`if ($x <= 23)`

2.4.3 Logische Operatoren

Wir benutzen logische Operatoren in Kontrollstrukturen (siehe unten).

Operator	Bedeutung	Beispiel
`and`	Logisches UND	`if (($x == 23) and ($y == 17))`
`&&`	Logisches UND	`if (($x == 23) && ($y == 17))`
`or`	Logisches ODER	`if (($x == 23) or ($y == 17))`
`||`	Logisches ODER	`if (($x == 23) \|\| ($y == 17))`
`xor`	Logisches ENTWEDER ODER	`if (($x == 23) xor ($y == 17))`
`!`	Logisches NICHT	`if (!$EsRegnet)`

2.5 Grundlegende Kontrollstrukturen

2.5.1 if-Struktur

- Einfache if-Struktur

```
1  <?php
2      if ($EsRegnet)
3      {
4        echo "Die Strasse wird nass. <br>";
5        echo "Ich bleibe zu Hause.";
6      }
7  ?>
```

Die geschweiften Klammern werden wie in Java gesetzt. Die Bedingung steht in runden Klammern hinter `if` und kann mit den oben angeführten Vergleichs- und logischen Operatoren beliebig komplex gestaltet werden. Achtung: Die Zeile mit `if(...)` enthält kein Semikolon.

- if-else-Struktur

```
1  <?php
2      if ($EsRegnet)
3      {
4        echo "Die Strasse wird nass. <br>";
5        echo "Ich bleibe zu Hause.";
6      }
7      else
8      {
9        echo "Die Strasse bleibt trocken. <br>";
10       echo "Ich kann Fahrrad fahren.";
11     }
12 ?>
```

- if-elseif-Struktur

```
1  <?php
2      if ($x > 0)
3      {
```

```
4       echo "Positive Zahl.";
5     }
6     elseif ($x < 0)
7     {
8       echo "Negative Zahl";
9     }
10    else
11    {
12      echo "Die Zahl ist Null.";
13    }
14  ?>
```

Es können beliebig viele elseif-Zweige benutzt werden.

2.5.2 while-Schleife

```
1  <?php
2      $x = 10;
3      while ($x > 0)
4      {
5        echo "$x kleine Negerlein, die ...";
6        $x = $x - 1;
7      }
8      echo "Das Lied ist zu Ende.";
9  ?>
```

Die Bedingung wird immer **vor** der Ausführung der Schleife überprüft. Wenn die Bedingung wahr ist, werden die Anweisungen in der Schleife ausgeführt. Wenn die Bedingung nicht wahr ist, wird mit der nächsten Anweisung hinter der Schleife fortgefahren. Es kann also passieren, dass die Anweisungen in der Schleife je nach Bedingung niemals ausgeführt werden.

2.5.3 do-while-Schleife

```
1  <?php
2      $x = 10;
3      do
4      {
5        echo "$x kleine Negerlein, die ...";
6        $x = $x - 1;
7      } while ($x >= 1);
8      echo "Das Lied ist zu Ende.";
9  ?>
```

Die Bedingung wird immer erst **nach** der Ausführung der Schleife überprüft. Wenn die Bedingung wahr ist, werden die Anweisungen in der Schleife ausgeführt. Wenn die Bedingung nicht wahr ist, wird mit der nächsten Anweisung hinter der Schleife fortgefahren. Die Anweisungen in der Schleife werden also unabhängig von der Bedingung auf alle Fälle mindestens einmal ausgeführt. Auch wenn oben z.B. `$x = -1;` vor der Schleife steht, wird die Schleife einmal durchlaufen. Prinzipiell ist somit ihr Einsatz „gefährlich.“.

Meistens kann man auf die do-Schleife verzichten und sie durch eine while-Schleife ersetzen. Sinnvoll ist sie nur, wenn auf alle Fälle erst einmal etwas getan werden muss, um sinnvoll weiter arbeiten zu können.

2.5.4 for-Schleife

Die for-Schleife ist eigentlich dazu gedacht, Anweisungen zu wiederholen, wobei die Anzahl der Wiederholungen vorab bekannt ist.

```
<?php
    for ($x = 10; $x > 0; $x = $x - 1)
    {
      echo "$x kleine Negerlein, die ...";
    }
    echo "Das Lied ist zu Ende.";
?>
```

Die Klammer hinter `for` enthält drei Teile:

1. Mit `$x = 10` wird die Schleifenvariable auf ihren Anfangswert gesetzt.
2. Die Bedingung `$x > 0` wird vor dem Durchlauf der Schleife überprüft. Nur wenn die Bedingung wahr ist, wird die Schleife ausgeführt. Ist sie nicht mehr wahr, so wird mit der nächsten Anweisung hinter der Schleife fortgefahren.
3. Die Anweisung `$x = $x - 1` wird am Ende jedes Durchlaufs vor der Überprüfung der Bedingung ausgeführt.

Die Syntax der for-Schleife ist wie bei Java. Ebenso sind die dort üblichen abkürzenden Schreibweisen wie `$x--` für `$x = $x -1` und `$x++` für `$x = $x + 1` üblich.

3 Funktionen

3.1 Funktionen ohne Parameter

Funktionen entsprechen in ihrer einfachen Form den void-Methoden von Java. In gewisser Weise kann man also in PHP Funktionen „missbrauchen“, um irgend Etwas zu tun, ohne einen Funktionswert zu liefern.

```
<?php
    function refrain()
    {
      echo "Can't wait until tonight, baby...";
    } // Ende der Funktion

    echo "La La Li ...";
    refrain();
    echo "Li La Lu ...";
    refrain();
    echo "Thank you!";
?>
```

3.2 Funktionen mit Parametern

Die Funktionen können auch etwas tun in Abhängigkeit von einem Parameter:

```
<?php
    function negerlein($nummer)
    {
      echo "$nummer kleine Negerlein, die ...";
    }

    for ($x = 10; $x > 0; $x = $x - 1)
    {
      negerlein($x);
    }
    echo "Das Lied ist zu Ende.";
?>
```

Bei jedem Aufruf der Funktion wird der aktuelle Wert von `$x` in die Variable `$nummer` kopiert und dann die Funktion ausgeführt.

3.3 Funktionen mit Rückgabewerten

Funktionen sind eigentlich dazu da, Funktionswerte zu liefern. Ein Funktionswert wird an die aufrufende Stelle im Programm zurück geliefert und einer Variablen zugewiesen.

```
<?php
    function dasDoppelte($zahl)
    {
      return($zahl + $zahl);
    }

    for ($x = 1; $x <= 10; $x = $x + 1)
    {
      $y = dasDoppelte($x);
      echo $y;
    }
?>
```

3.4 Einfache mathematische Funktionen

- `$y = -17;`
  `$x = abs($y);` liefert für `$x` den Wert 17, d.h. den Betrag der Integerzahl `$y`.
- `$y = floor(3.1425);` liefert für `$y` den Wert 3, d.h. die nächstkleinere Integerzahl.
- `$y = round(3.1415);` liefert für `$y` den Wert 3.
- `$y = round(3.1415, 2);` liefert für `$y` den Wert 3,14.
- `$y = pi();` liefert für `$y` den Wert 3,14159265... wie mit dem Taschenrechner.
- `$y = sqrt(2);` liefert für `$y` den Wert 1,414213... wie mit dem Taschenrechner.

- `$y = pow(2, 3);` liefert für `$y` den Wert $8 = 2^3$ wie mit dem Taschenrechner.
- `srand(date('s'));` initialisiert den Zufallsgenerator mit der aktuellen Sekunde. Bei jedem Gebrauch von Zufallszahlen sollte der Zufallszahlengenerator vorher mit einer beliebigen und immer anderen Zahl initialisiert werden, damit er nicht immer die gleichen Zufallszahlen liefert.
- `$x = rand(1, 6);` liefert eine ganze Zufallszahl, die wie bei einem Würfel die Werte von 1 bis 6 annehmen kann.
- `mt_srand(date('s'));` initialisiert einen Zufallsgenerator mit der aktuellen Sekunde. Hier wird der sogenannte „Mersenne Twister (MT)“-Zufallszahlengenerator benutzt. Ansonsten gilt das Gleiche wie bei `srand()`.
- `$x = mt_rand(1, 6);` liefert eine ganze Zufallszahl mit dem MT-Zufallszahlengenerator. Ansonsten gilt das Gleiche wie bei `rand()`.

4 HTML-Formulare und PHP

4.1 Allgemeines

Zuerst ein einfaches Beispiel:

Inhalt der Datei *Textfeld.html*:

```
<HTML><BODY>
<FORM method="post" action="Textfeld.php">
 1. Summand: <INPUT type="text" name="zahl1" size="5" maxlength="10">
 2. Summand: <INPUT type="text" name="zahl2" size="5" maxlength="10">
 <br>
 <INPUT type="submit" value="Berechnen">
</FORM>
</BODY></HTML>
```

Formulare werden in HTML mit den Tags `<FORM>` und `</FORM>` definiert.

Das Attribut `method="post"` bedeutet so viel wie „Formulardaten wegschicken“.

Das Attribut `action="Textfeld.php"` gibt an, wohin die Formulardaten gesendet werden sollen. Bei uns ist es immer eine PHP-Datei zur Auswertung der Formulardaten. Die Daten können aber z.B. auch als E-Mail verschickt werden (siehe unten).

Mit `<INPUT type="submit" value="Berechnen">` wird ein Button mit der Aufschrift „Berechnen“ erzeugt. Ein Klick darauf führt zum Abschicken der Formulardaten.

Inhalt der Datei *Textfeld.php*:

```
<HTML><BODY>
<?php
$x = $_POST["zahl1"];
$y = $_POST["zahl2"];
$summe = $x + $y;
echo "$x + $y = $summe";
?>
</BODY></HTML>
```

Die Datei *Textfeld.php* erhält die Formulardaten und kann auf die im Formular über `name` definierten Variablen zugreifen.

4.2 Einzeiliges Eingabefeld

Im oben angegebenen Beispiel werden zwei Textfelder erzeugt. Z.B. führt

`<INPUT type="text" name="zahl1" size="5" maxlength="10">`

dazu, dass ein Formularfeld vom Typ *text* erzeugt wird. Mit `size="5"` wird die Breite des Eingabefeldes auf 5 Zeichen gesetzt. Man kann aber bei Bedarf mehr Zeichen eingeben. Die maximale Zeichenzahl wird hier mit `maxlength="10"` auf 10 Zeichen gesetzt.

Geheime Eingabe: Benutzt man `type="password"`, so wird die Eingabe mit Sternchen vor den neugierigen Blicken mitlesender Menschen geschützt.

Geheime Übertragung: Das Beispiel

```
<INPUT type="hidden" name="wert" value="356">
```

zeigt die Übertragung eines Wertes, ohne dass irgendetwas auf dem Bildschirm dazu erscheint.

4.3 Mehrzeiliges Eingabefeld

Mehrzeilige Eingabefelder werden nicht mit `<INPUT>`, sondern mit den Tags `<TEXTAREA>` und `</TEXTAREA>` eingeschlossen. Z.B. wird mit

```
<TEXTAREA name="eingabe" rows="3" cols="4">Das steht schon drin.</TEXTAREA>
```

ein Texteingabefeld mit drei Zeilen und vier Spalten festgelegt, wobei der Text „Das steht schon drin.“ als Vorbelegung gedacht ist.

4.4 Radio-Button

Das Beispiel demonstriert die prinzipielle Arbeitsweise mit Radio-Buttons. Von mehreren Buttons kann nur einer angeklickt bzw. gewählt werden. Mit dem Attribut `checked` wird festgelegt, welcher Button standardmäßig vorausgewählt sein soll.

Inhalt der Datei *RadioButton.html*:

```
<HTML><BODY>
<FORM method="post" action="RadioButton.php">
 Markieren Sie Ihr Geschlecht:
 <INPUT type="radio" name="geschlecht" value="w"         >weiblich
 <INPUT type="radio" name="geschlecht" value="m" checked>m&auml;nnlich
 <BR>
 <INPUT type="submit" value="Abschicken">
</FORM>
</BODY></HTML>
```

Inhalt der Datei *RadioButton.php*:

```
<HTML><BODY>
<?php
$geschl = $_POST["geschlecht"];
if ($geschl == "w")
     { $anrede = "Sehr geehrte Frau "; }
else { $anrede = "Sehr geehrter Herr "; }
echo "$anrede Dingsbums.";
?>
</BODY></HTML>
```

4.5 Mehrfachauswahl mit checkbox

Das Beispiel demonstriert die prinzipielle Arbeitsweise mit sogenannten Checkboxen. Von mehreren Kästchen können alle oder keines angeklickt bzw. angekreuzt werden. Mit dem Attribut `checked` wird festgelegt, welcher Button standardmäßig vorausgewählt sein soll.

Inhalt der Datei *Checkbox.html*:

```
<HTML><BODY>
<FORM method="post" action="Checkbox.php">
 Markieren Sie Ihre bisher besuchten Schulformen:
 <INPUT type="checkbox" name="schule1" value="haupt">      Hauptschule
 <INPUT type="checkbox" name="schule2" value="real">       Realschule
 <INPUT type="checkbox" name="schule3" value="gym" checked>Gymnasium
 <BR>
 <INPUT type="submit" value="Abschicken">
</FORM>
</BODY></HTML>
```

Bei der Auswertung mit PHP ist darauf zu achten, dass einige Werte undefiniert bleiben können. Inhalt der Datei *Checkbox.php*:

```
<HTML>
<HEAD>
<TITLE>Checkbox-Beispiel</TITLE>
</HEAD>
<BODY>
<?php
if (!empty($_POST['schule1']))
     $schule1 = $_POST['schule1'];
else $schule1 = "keine";
if (!empty($_POST['schule2']))
     $schule2 = $_POST['schule2'];
else $schule2 = "keine";
if (!empty($_POST['schule3']))
     $schule3 = $_POST['schule3'];
else $schule3 = "keine";

echo "Das␣Ergebnis␣der␣Eingabe␣<BR>";
if ($schule1=="haupt")
     echo "Sie␣besuchten␣die␣Hauptschule.<BR>";
if ($schule2=="real")
     echo "Sie␣besuchten␣die␣Realschule.<BR>";
if ($schule3=="gym")
     echo "Sie␣besuchten␣das␣Gymnasium.<BR>";
echo "<BR>";
?>
</BODY>
</HTML>
```

4.6 Auswahlliste

Mit `<SELECT>` und `</SELECT>` wird eine Liste von Auswahlmöglichkeiten erstellt. Mit `size="2"` wird hier festgelegt, dass zwei Einträge der Liste sichtbar sind.

Inhalt der Datei *Auswahlliste.html*:

```
<HTML><BODY>
<FORM method="post" action="Auswahlliste.php">
  Markieren Sie Ihren Lieblingsgitarristen:<BR>
  <SELECT name="bestergitarrist" size="4">
  <option>Wes Montgomery
  <option>Frank Zappa
  <option>Steve Vai
  <option>Carlos Santana
  <option>Eric Clapton
  </SELECT>
  <INPUT type="submit" value="Abschicken">
</FORM>
</BODY></HTML>
```

Inhalt der Datei *Auswahlliste.php*:

```
<HTML><BODY>
<?php
$best = $_POST["bestergitarrist"];
echo "Der beste Gitarrist ist $best.";
?>
</BODY></HTML>
```

4.7 Formulardaten als E-Mail wegschicken

Mit

```
<HTML><BODY>
<FORM method="post" action="mailto:horst@gierhardt.de">
 Vorname :<INPUT type="text" name="vorname"  size="10" maxlength="20">
 Nachname:<INPUT type="text" name="nachname" size="10" maxlength="20">
 <br>
 <INPUT type="submit" value="Abschicken">
</FORM>
</BODY></HTML>
```

wird eine E-Mail erzeugt und an die angegebene Adresse abgeschickt. In der E-Mail sind dann die Einträge für *vorname* und *nachname* enthalten.

4.8 Einfache Dateioperationen: Ein Besucherzähler

Das folgende Beispiel zeigt das einfache Lesen und Schreiben mit Dateien. Konkret wird die Variable `$zahl` bei jedem Aufruf der Seite aus einer Datei gelesen, um 1 erhöht und wieder in die Datei geschrieben.

```
<?php

  $fn = "zaehler.txt";     // Dateiname festlegen

  // Auf  Existenz  pruefen
  if (file_exists($fn) == FALSE)
       { $fp = fopen($fn, "w");  // Erzeugt die Datei,
         fclose($fp);            // wenn nicht vorhanden.
                                 // w+ oeffnet und loescht den Inhalt
       }

  // Lesen
  $fp = fopen($fn, "r");       // Oeffnet die Datei zum Lesen
  $zahl = fread($fp, 20);      // Liest max. bis zum 20. Zeichen
  if ($zahl == '') $zahl = 0; // Datei ist am Anfang leer.
  fclose($fp);                 // Schliesst Datei

  // Schreiben
  $fp = fopen($fn, "r+");  // Oeffnet die Datei zum Lesen und Schreiben
  fseek($fp, 0);           // Springt zum Anfang der Datei.
  $zahl = $zahl + 1;
  fwrite($fp, $zahl);      // Schreibt dort in die Datei
  fclose($fp);             // Schliesst Datei
?>

<HTML>
<BODY>
Sie haben diese Seite schon <?php echo $zahl ?>  mal besucht.
</BODY>
</HTML>
```

4.9 Verbindung mit MySQL-Datenbank herstellen

In der Datei *Hauptteil.html* wird ein einfaches Formular zur Eingabe von Befehlen für die MySQL-Datenbank erzeugt.

Inhalt der Datei *Hauptteil.html*:

```
<HTML>
<HEAD>
  <TITLE>MySQL-Datenbank kontaktieren</TITLE>
</HEAD>
<BODY>
  <DIV align=center>
  <h1> MySQL-Datenbank kontaktieren</h1>

  <FORM action="AllesAnzeigen.php" method="post">

    Hier SQL-Abfrage eingeben:<BR>
    <TEXTAREA name="Eingabe" rows="5" cols="60"> SELECT * FROM lehrer;
    </TEXTAREA><BR>
    <INPUT type="Submit" name="Senden" value="Senden">

  </FORM>
  </DIV>
</BODY>
</HTML>
```

Der Inhalt der Textarea mit dem Namen `Eingabe` wird an die PHP-Datei *AllesAnzeigen.php* geschickt. Diese stellt die Verbindung zur MySQL-Datenbank her. Hier sind folgende Vorgaben gemacht worden:

Server-Adresse:	localhost
Benutzer-Name:	sqladminq2if2
Benutzer-Kennwort:	enemenemu
Datenbank-Name	schueler (enthält die Tabelle lehrer)

Nach erfolgreicher Verbindung und Übertragung der SQL-Anweisung in der Variablen `$abfrage` wird in der Variablen `$ergebnis` das Ergebnis der Datenbank gespeichert und an die Methode `mysql_Tabelle_schreiben`, die in der Datei *dbtabelle.php* steht, übergeben. Dort wird eine HTML-Tabelle zusammen gebaut.

Inhalt der Datei *AllesAnzeigen.php*:

```
<HTML>
<BODY>
<DIV align=center>
<h1>Ergebnis der Datenbankabfrage</h1>
<br>

<SCRIPT language ="php">

include("dbtabelle.php");

$kennung = mysql_connect("localhost","sqladminq2if2","enemenemu") or
           die ("Konnte nicht mit der Datenbank verbinden.");

```

```
$db = mysql_select_db("schueler") or
      die("Konnte die Datenbank nicht selektieren.");

$abfrage = $_POST['Eingabe'];

echo "Folgende SQL-Anweisung wurde an die Datenbank gesendet:<BR>";
echo "<B>$abfrage</B><BR>";

$ergebnis = mysql_query($abfrage)
            or die(mysql_error());
$tabAttribute = "width='80%' BORDER=1 ".
                "bordercolor=red bgcolor=gray";
if (is_bool($ergebnis))
    { if ($ergebnis)
            echo "Erfolgreicher Vorgang in Datenbank.";
      else echo "Da hat etwas in der Datenbank nicht geklappt.";
    }
else mysql_Tabelle_schreiben($ergebnis, $tabAttribute);

mysql_close();
</SCRIPT>

</BODY>
</HTML>
```

Inhalt der Datei *dbtabelle.php*:

```
<SCRIPT language = "php">

function mysql_Tabelle_schreiben($param_res, $param_form)
{
   $anzahl = mysql_num_fields($param_res)        // Anzahl der Spalten
             or die ("Die Tabelle ist leer.");
   echo "<TABLE $param_form>";                   // Tabellenkopf schreiben
   echo "<TR>";
   for ($i=0;$i<$anzahl;$i++)
   {    echo "<TH>";
        echo mysql_field_name($param_res,$i);     // Spaltennamen auslesen
        echo "</TH>";
   }
   echo "</TR><TR>";                              // Ende der Kopfzeile
   while ($zeile = mysql_fetch_row($param_res))   // array zeile
   {    $spaltenzahl = mysql_num_fields($param_res);
        for ($i=0; $i<$spaltenzahl; $i++)
        {    echo "<TD>$zeile[$i]</TD>";         // Spalteninhalte schreiben
        }
        echo "</TR><TR>";
   }
   echo "</TABLE>";
}
</SCRIPT>
```

4.9.1 Aufgaben:

1. Ergänze das Formular für die Eingabe der SQL-Anfrage um Textfelder zur Eingabe der Serveradresse, des Benutzernamens, des Benutzerkennwortes und des Datenbanknamens.

2. Teste das Textfeld für das Benutzerkennwort mit `type = "password"`.

3. Baue die Eingabefelder in eine Tabelle ein.

4. Nach der Ausgabe der Ergebnistabelle soll die Anzahl der gefundenen Datensätze angegeben werden. Dazu ist die Datei *dbtabelle.php* geeignet zu verändern.

Fortsetzung folgt