Mini-Workshop

"Die HTML-PHP-Schnittstelle -- Ein- und Ausgabe der Daten"

Dirk Wiebel
28.07.03

1. HTML-Grundlagen

- SGML-basierte Formatiersprache (HyperText Markup Language)
- Basiert auf Tags und Attributen
- Enthält Head (Metainformationen) und Body (Informationen)

Einfachstes Beispiel:

```
<html>
  <head>
   <title> Hier steht der Titel </title>
  </head>
  <body>
    <p align="center"> Ich bin ein Abschnitt </p>
  </body>
</html>
```

Wichtigste Elemente:

	Paragraph/Abschnitt
	Zentrierter Abschnitt
 	Break/Zeilenumbruch
	Bold/Fett
	Italics/Kursiv
<u></u>	Underlined/Unterstrichen
	Non-breakable space/Festes Leerzeichen
ä	ä
Ä	Ä
ü	ü
&szlig:	ß
"	"

Uni Tübingen Verlinkung

...

+ Tabellenformatierung (s.u.)

2. Formulare in HTML

```
<form action=“zielseite.php“ method=“POST“>
  <input type=“text“ name=“textfeld1“ value=“Standardeintrag“>
  <input type=“submit“ value=“&Uuml;bertragen>
</form>
```

Mögliche Felder

submit
```
<input type=“text“ name=“textfeld1“ value=“Standardeintrag“>
```

reset
```
<input type=“text“ name=“textfeld1“ value=“Standardeintrag“>
```

text
```
<input type="text" size="25" maxlength="50" name="Name">
```

textarea:
```
<textarea cols="20" rows="5" name="Elementname">
Optionale Textvorbelegung
</textarea>
```

select:
```
<select size="Höhe" name="Name">
<option>Eintrag</option>
<option>anderer Eintrag</option>
</select>
```

checkbox
```
<input type="checkbox" name="Name" value="Wert"> Text
<input type="checkbox" checked name="Name" value="Wert"> Text
```

hidden
```
<input type="hidden" name="Name" value="Wert">
```

3. POST und GET (http-Methoden)

POST-Methode: Daten werden unsichtbar an das Ziel gesendet.
GET-Methode: Daten kommen an der URL angehaegt zum Ziel.

```
http://barlach.sfb.uni-tuebingen.de/test.php?Variable1=Wert1
```

Zum De-Buggen empfiehlt sich GET, ansonsten ist POST zu empfehlen, damit der User die Variablen nicht manipulieren kann.

4. Variablenübermittlung an PHP-Skripte

Das im Attribut action genannte PHP-Skript wird aufgerufen, die Namen der Eingabefelder werden dem PHP-Skript als Variablen mit den entsprechenden Namen und Werten übermittelt. Bei

```
http://barlach.sfb.uni-tuebingen.de/test.php?Variable1=Wert1
```

entspricht die Variable „Variable1“ dem Wert „Wert1“. Auch der Wert des Submit-Knopfes wird überliefert, dieser eignet sich hervorragend zur Überprüfung, ob eine Eingabe stattgefunden hat.

Tipp: Eine PHP-Seite kann sich selbst im Action-Attribut aufrufen:

```
<form action="<? echo $PHP_SELF; ?>" method="POST">
  <input type="text" name="textfeld1">
  <input type="submit" name="submit" value="Go!">
</form>

<?
if(isset($submit)){
   echo $textfeld1;
}
?>
```

5. Tabellen in HTML

Alle Größenangaben können absolut (in Pixeln) oder relativ (in Prozent) gemacht werden. Übliche Pixelanzahl bei gängigen Bildschirmen: 800x600, 1024x768 oder 1280x1024.

```
<table width="50%" border="2">
  <tr>
    <td>Hier kommt der Inhalt der ersten Spalte hin</td>
    <td>Hier kommt der Inhalt der zweiten Spalte hin</td>
  </tr>
  <tr>
    <td colspan="2"> Und hier die n&auml;chste Reihe...</td>
  </tr>
</table>
```

6. HTML-Formular, PHP-DBMS-Abfrage und HTML-Ausgabe in einem Skript

```
<html>

  <head>
    <title>Testskript</title>
  </head>


  <body>

    <p align=“center“>
      <font size=“+2“>Willkommen auf der Testseite!</font>
    </p>

    <table width=“50%“ align=“center“ border=“1“>
    <form action="<? echo $PHP_SELF; ?>" method="POST">
      <tr>
        <td>Abfrage: select id,title from movies where title like</td>
        <td><input type=“text“ name=“query“></td>
      </tr>
      <tr>
        <td>Abschicken?</td>
        <td><input type=“submit“ name=“submit“ value=“Los gehts!“></td>
      </tr>
    </form>
    </table>


 <?
  if(isset($submit)){                              //If-Kondition Start
                                                   //Verbindung aufbauen
    $verbindung =
      mysql_connect("localhost","workshopbenutzer","*****");
    $abfrage = "select id,title from movies where title like \"%$query%\"";
    $erg = mysql_db_query("db_workshop",$abfrage,$verbindung);
    $anzahl = mysql_num_rows($erg);

                                          //Tabelle anlegen
   echo “<table width=\“50%\“ border=\“1\“ align=\“center\“>“;

    $x = 0;                                        //Schleife starten
    while ($x <= $anzahl){
      echo “<tr>“;
      $row = mysql_fetch_array($erg);

         echo “<td>“;
           echo $row["id"];
         echo “</td>“;
         echo “<td>“;
           echo $row["title"];
         echo “</td>“;

       $x++;

      echo “</tr>“;
    }                                              //Schleife beenden

    echo “</table>“;                               //Tabelle beenden


  }                                                //if-Kondition Ende
 ?>

</body>
```

7. Sicherheitsaspekte in PHP

Problem: Jemand kennt/vermutet die Variablennamen (z.B. user, pass) und überschreibt die Werte der Variablen mit GET-Anweisungen.

Lösung: Variablen nicht „global“ akzeptieren. Man sollte immer angeben, von woher eine externe Variable kommt, bzw. wie sie übermittelt wird. Option in PHP (wird auf dem Server in der Konfigurationsdatei php.ini eingestellt):

register_globals=off

(s. auch http://www.php.net/register_globals)

In diesem Fall müssen alle Variablen in dem Skript definiert werden:

```
$username = $_COOKIE['username'];
oder
$username = $_GET['username'];
oder
$username = $_POST['username'];
```

Problem: Benutzerdaten/Passwort im Klartext in Include-Datei.

Lösungen:
Serverseitig den direkten Zugriff sperren per .htaccess o.Ä.
oder
Include-Dateien so benennen, dass sie mit .php enden. Dadurch werden sie geparst und nichts ausgegeben, was nicht per „echo“ ausdrücklich ausgegeben werden soll.
oder
Include-Dateien ausserhalb des Webserver-Trees ablegen.

8. Nachschlagehinweis

SelfHTML: Herausragende Dokumentation und Lernhilfen von Stefan Münz:
http://selfhtml.teamone.de/, auch als Printversion erhältlich.