PHP und MySQL

Projekt II

Witzedatenbank: Idee von PHP/ MySQL-Guru Kevin Yank, der mehrere gute Bücher darüber verfasst hat.--> Bei Bedarf bitte unbedingt eine neue Auflage verwenden, weil sich besonders PHP in den letzten Jahren – vor allem wegen Sicherheitsproblemen – stark verändert hat.

Schritt 3: Eintragen eigener Witze in die Datenbank

1. *Legen Sie eine neue Datei in Notepad++ an und speichern Sie diese unter dem Namen* witzeschreiben.php *in htdocs. Notepad++ versteht wegen der Endung, dass es sich um eine PHP-Datei handelt.*
2. *Fügen Sie den gewohnten HTML-Rahmen ein.*
3. *Wenn* witzeschreiben.php *das erste Mal aufgerufen wird, soll der User die Möglichkeit bekommen, sich einzuloggen, denn jemand, der in der Witzedatenbank Eintragungen vornimmt, sollte uns nicht ganz fremd sein. Damit sich der User anmelden kann, erzeugen wir ein Formular.*

```
<form method="post" action=" witzeschreiben.php " name="frmLogin">
```

Dieses Formular hat als action*-Attribut seinen eigenen Namen stehen: das bedeutet, dass es sich selbst wieder aufrufen wird, wenn der User den Abschicken-Knopf betätigt. In diesem Fall wird wegen der Methode* POST *automatisch das $_Post-Array erzeugt, das alle Usereingaben in die Formularfelder enthalten wird, mit denen wir dann arbeiten können.*

4. *Wir überprüfen nun, ob* witzeschreiben.php *erstmalig aufgerufen wird, indem wir eine PHP- if-Verzweigung verwenden:*

```
<?php
if (empty($_POST))
```

Nur dann, wenn die in Klammern stehende Bedingung wahr ist, wenn also witzeschreiben.php erstmals angezeigt wird, *soll das Login-Formular angezeigt werde:*
Wir können nun kurzfristig den PHP-Tag schließen und normalen HTML-Code eingeben, oder aber mit dem echo-Befehl von PHP den HTML-Code später ausgeben lassen. Dabei muss beachtet werden, dass die HTML-Anführungszeichen sich mit den Anführungszeichen des PHP-echo-Befehls beißen: innerhalb der Anführungsstriche des echo-Befehls verwendet man deshalb Hochkommas – es besteht auch die Möglichkeit, vor die Anführungsstriche einen Backslash zu setzen, was aber unübersichtlich ist und nur im Norfall gemacht werden sollte.

```
{echo "<h1>Login</h1>";
echo "<table border='2' width='50%' align='center'>";
echo "<tr><td>Host</td><td><input type='text' name='host' value='localhost'></td></tr>";
echo "<tr><td>Host</td><td><input type='text' name=' benutzername' value='root'></td></tr>";
echo "<tr><td>Host</td><td><input type='password' name='passwort'></td></tr>";
echo "<tr><td colspan='2'><input type='submit' value='Einloggen'></td></tr></table>";
exit(); }
```

Die geschweiften Klammern umschließen den Code, der nur ausgeführt wird, wenn die in runden Klammern stehende Bedingung der if-Verzweigung wahr ist. Insgesamt wird hier eine Tabelle mit drei Eingabefeldern erzeugt, in welche der User die Daten eingeben kann, die er für eine Verbindung mit der MySQL-Datenbank benötigt. Es ist von großer Bedeutung, dass jedes Eingabefeld einen Namen zugewiesen bekommt, denn über diesen Namen können die Eingaben später aus dem $_POST-Array ausgelesen werden. Das value*-Attribut legt einen vorgegebenen Standardwert fest – In der richtigen Welt würde man bei Login-Daten keine Vorgaben machen.*
Am Tabellen-Ende wird ein submit-Button erzeugt, mit welchem der User das Formular später abschicken kann. Der Exit-Befehl führt zum Abbrechen der Code-Ausführung – alles, was sich jetzt nun anschließt, soll erst ausgeführt werden, wenn der Benutzer sich angemeldet hat.

5. *Nun müssen wir ein Formular gestalten, über welches der User einen neuen Witz eintragen, also einen neuen Datensatz in der Tabelle* witze *anlegen kann. Ein Datensatz (Tupel) der* witze*-Tabelle besteht aus vier unterschiedlichen Informationen: Da ist zunächst die ID – die muss der User nicht eingeben, weil wir bei der Anlage der Tabelle ID auf Autoincrement gestellt haben, so dass MySQL automatisch bei jedem neuen Datensatz die nächsthöhere ganze Zahl einfügt. Dann folgen das* datum*, der* autor *und der eigentliche* witz*. Wir müssen für diese drei Felder geeignete Eingabemöglichkeiten für den User gestalten.*
6. *Wieder legen wir eine Tabelle an. Für das Datumsfeldschlagen wir das aktuelle Datum als Standardwert vor:*

```
echo "<table border='2' width='50%' align='center'>";
echo "<tr><td>Datum</td><td><input type='text' name='datum' value='";
echo date ("d.m.Y", time()) ."'></td></tr>";
```

Die date*-Funktion von PHP erwartet zwei Parameter: Formatierungsinformationen als Text und eine Zeitangabe, die entsprechend formatiert wird.* d *gibt den Tag zweistellig aus, der Punkt macht einfach einen Punkt,* m *gibt die zweistellige Monatszahl aus, der Punkt macht wieder einen Punkt und* Y *ergibt die vierstellige Jahreszahl: so entstehen immer genau 10 Ziffern, die in das Datumsfeld unserer Tabelle passen.*
Wir könnten das Datumsfeld verstecken (Attribut type='hidden' *statt* 'text' *setzen), und so sicher sein, dass immer das aktuelle Systemdatum des Webservers eingetragen wird – dann könnte der User aber kein von heute abweichendes Datum eintragen. Der Punkt vor dem Anführungszeichen nach* time)) *ist übrigens der Kombinationsoperator von PHP und er verbindet das Datum mit den folgenden Tags. Alternativ hätte man die Tags auch in eine neue* echo*-Zeile schreiben können.*

7. *Weil in unserer Datenbank die Daten normalisiert vorliegen und also die Autoren und die Witze in unterschiedlichen miteinander verbundenen Tabellen abgelegt sind, müssen wir sicherstellen, dass ein neuer Witz nur von einem bereits existierenden Autor hinzugefügt werden kann. Deshalb bieten wir unserem User eine Liste oder ein Dropdown-Feld mit allen Autoren der Datenbank an und nehmen ihm die Möglichkeit, einen eigenen Autor selbst einzutragen.*
   Ein Dropdown-Feld (Kombinationsfeld) hat in HTML die Syntax <select> <option> </option> <option> </option> <option> </option> </select>Die einzelnen Elemente (bei uns die Autoren) werden von einem option*-Tag umschlossen. Diese Syntax eignet sich für eine PHP-While-Schleife: in jedem Durchgang soll ein Eintrag in das Dropdown-Feld eingetragen werden. Dazu müssen wir uns zunächst in die Witzedatenbank einloggen und dort die Autoren-Tabelle auslesen.*
8. *Exkurs: Dropdown-Feld mit den Witze-Autoren.*
   Die Daten zum Einloggen liegen im $_Post-Array für uns bereit. Wir verwenden Sie, um uns bei MySQL anzumelden. Da wir uns später nochmal mit der Datenbank verbinden müssen, wenn wir den neuen Witz eintragen wollen, müssen wir die Daten aus dem aktuellen $_Post-Array so ablegen, dass sie uns auch später wieder zur Verfügung stehen, ohne dass der User sie nochmals umständlich eingeben muss.
   Deshalb tragen wir sie in unsichtbare Felder ein. Parallel legen wir die Daten auch in Variablen ab, so dass man im Folgenden einfacher auf die Daten zugreifen kann:

```
$host=$_POST["host"];
echo "<input type='hidden' name='host' value=$host>"; //unsichtbares Formularfeld
$benutzer=$_POST["benutzername"];
echo "<input type='hidden' name='benutzername' value=$benutzer>"; //unsichtbares Formularfeld
$passwort=$_POST["passwort"];
echo "<input type='hidden' name='passwort' value=$passwort>"; //unsichtbares Formularfeld
```

   Mit diesen Daten kann sich der User bei MySQL anmelden:

```
mysql_connect($host,$benutzer,$passwort) or die ("Verbinung mit MySQL misslungen!");
mysql_select_db ("witze") or die ("Verbindung mit der Witze-Datenbank misslungen!");
```

   Wenn die Verbindung nicht gelingt, endet das Skript mit der angegebenen Fehlermeldung.
   Nun übergeben wir den SQL-Code, mit dem wir alle Autoren der Witze-Datenbank ermitteln werden, an eine Textvariable und diese lassen wir dann von MySQL ausführen:

```
$SQLBefehl1="SELECT * FROM autoren ORDER BY autornn";
$autoren=mysql_query ($SQLBefehl1);
```

   Wir selektieren alles (*) aus der Tabelle autoren *und sortieren das Ergebnis aufsteigend nach dem Autornachnamen (steht im Feld* autornn *der Tabelle* autoren*). Mit* mysql_query() *schicken wir die Abfrage an MySQL und weisen deren Rückgabewert (eine namentlich sortierte Liste aller Witze-Autoren) der Variablen* $autoren *zu. Nun erstellen wir eine* while*-Schleife, die nacheinander jeden Autoren aufruft, indem sie immer wieder die gleiche Funktion mysql_fetch_row() mit dem frisch erzeugten Autoren-Array* $autoren *aufruft: diese Funktion liest nicht nur immer einen vollständigen Datensatz mit allen zugehörigen Spalten aus, sondern setzt den Zeiger danach immer auf den nächsten Datensatz. Das hat zur Folge, dass bei jedem Aufruf dieser Funktion ein neuer Autor ausgelesen wird. Da wir mehrere Informationen zu jedem einzelnen Autor auslesen wollen, müssen wir das Ergebnis des Aufrufs also in einer Variablen zwischen speichern, die wir dann beliebig häufig aufrufen können. Da ein Aufruf von* mysql_fetch_row *den Wert* false *zurückgibt, wenn der Zeiger auf keinen Datensatz mehr springen kann, weil das Array vollständig durchgegangen worden ist, eignet sich dieser Ausdruck auch als Bedingung einer* while*-Schleife:*

```
echo "<tr><td>Autor</td><td><select name='autor'>";
while ($row=mysql_fetch_row($autoren))
  { echo "<option value=\"$row[0]\">$row[2] $row[1]</option>"; }
```

   *Die Schleife läuft nun alle Datensätze des $*Autoren*-Arrays durch, das seinerseits alle Autoren der Tabelle* autoren *beinhaltet. Bei jedem Durchgang erstellt sie einen neuen Eintrag des Dropdown-Feldes und belegt dieses mit dem Wert* $row[0]*, also mit dem ersten Wert aus dem $*row*-Array, das den je aktuellen Autordatensatz enthält – der erste Wert mit dem Index 0 ist die ID des Autors, also genau der Wert, den wir als Fremdschlüssel in die Tabelle* witze *eintragen wollen. Das Formular übergibt später diesen Wert* value *des vom User ausgewählten Ele-*

ments im Dropdown-Feld. Zwischen den <option>*-Tags schreibt die Schleife erst den Vor- und dann den Zunamen des aktuellen Autors, so dass der User später die ID übergibt, aber den vollständigen Realnamen sehen kann:*

```
echo "</select></td></tr>";
echo "</tr><td>Witz:</td><td><textarea rows='8' name='txtwitz'></textarea></td></tr>";
echo "<td colspan='2'><input type='submit' value='Witz eintragen'></td></table>";
exit();
```

Am Ende erzeugen wir einen Submit-Button, der dazu führt, dass sich die Seite witzeschreiben.php *nun zum dritten Mal aufruft: nun erst soll sie die Eingaben endlich in die Datenbank eintragen.*
Damit der eben ausgeführte Code nicht nochmal ausgeführt werden muss, stellen wir ihm noch eine Verzweigung vor, die dafür sorgt, dass der User nurt dann einen neuen Witz eingeben kann, wenn das Feld txtwitz*, das beim zweiten Laden gefüllt wird, einen Inhalt besitzt:*

```
if (! isset($_POST["txtwitz"]))
```

Das führende Ausrufezeichen in der Bedingungs-Klammer negiert die Bedingung. Der folgende Code in geschweiften Klammern wird also nur ausgeführt, wenn das Feld txtwitz *noch nicht gefüllt worden ist. Hat der User bereits einen Witz eingegeben und dann auf* submit *geklickt, dann besitzt* $_POST["txtwitz"] *einen Wert: nämlich den eigentlichen Witztext.*
Obige Verzweigung wird unmittelbar hinter die erste Verzweigung platziert und der gesamte obige Code wird zwischen ein geschweiftes Klammerpaar gesetzt. Mit exit() am Ende wird der Code an dieser Stelle abgebrochen und so wird verhindert, dass man in den folgenden Block läuft, der erst nach der Eingabe eines Witzes ausgeführt werden soll.

9. *Beim nun erfolgenden dritten Aufruf der Seite werden die Bedingungen beider Verzweigungen nicht mehr erfüllt und also wird nur ausgeführt, was nun folgt.*
   Nun steht das eigentliche Eintragen der Daten in die Datenbank an. Das erledigen wir mit dem INSERT-Befehl von SQL, von dem es mehrere Varianten gibt (vgl. http://dev.mysql.com/doc/refman/5.1/de/insert.html)
   Dazu greifen wir auf das $_POST*-Array zurück, das alle Formulardaten des zweiten Seitenaufrufs enthält.*

```
$host=$_POST["host"];
$benutzer=$_POST["benutzername"];
$passwort=$_POST["passwort"];
$datum=$_POST["datum"];
$autor=$_POST["autor"];
$witz=$_POST["txtwitz"];
```

10. *Mit diesen Angaben können wir uns nochmals mit MySQL verbinden und die Witze-Datenbank öffnen:*

```
mysql_connect($host,$benutzer,$passwort) or die ("Verbinung mit MySQL misslungen!");
mysql_select_db ("witze") or die ("Verbindung mit der Witze-Datenbank misslungen!");
```

11. *Anschließend formulieren wir den SQL-Befehl, mit dem wir den neuen Witz in die Datenbank eintragen:*

```
$SQLBefehl2="INSERT INTO witze (ID,datum,autor,witz) VALUES ('NULL',\"$datum\",\"$autor\",\"$witz\")";
```

Dieser Befehl liest sich zunächst schwierig, weil um alle Textfelder maskierte Anführungszeichen gesetzt werden müssen. Man hätte auch mit Hochkommata und dem Kombinationsoperator arbeiten können:

```
$SQLBefehl2="INSERT INTO witze (ID,datum,autor,witz) VALUES ('NULL','" .$datum ."','" .$autor ."','" .$witz ."')";
```

NULL ist ein Schlüsselbegriff, der nicht 0, sondern keinen Wert, ein leeres Feld bezeichnet

12. *Das Abschicken dieses SQL-Befehls kann mit einer Fehlerroutine verbunden werden:*

```
if (mysql_query($SQLBefehl2))
   {  echo("<p>Neuer Witz eingetragen</p>");
     echo "<a href='witzelesen.php'>Zur&uuml;ck zum Anfang</a>";    }
   else
   {   echo("<h2>Fehler</h2>");
      echo("<p>Die Fehlerbeschreibung lautet:</p>");
      echo (mysql_error());   }
```

Wenn der Witzeintrag nicht klappt, dann gibt mysql_query false *zurück und löst die Fehlerbehandlung aus.*
Die PHP-Funktion mysql_error() *gibt eine Beschreibung des letzten Fehlers aus.*

Die wichtigsten PHP/ MySQL/ SQL-Befehle:

PHP-Befehle für MySQL

mysql_connect(host, benutzer, passwort); → verbindet einen User mit MySQL
bsp.: mysql_connect("localhost","root"); → verbindet den User *root* (der alles darf) der bei XAMPP kein Passwort besitzt mit dem auf *localhostlaufenden MySQL*

mysql_select_db(*Datenbank*); → verbindet nach dem Login mit einer MySQL-Datenbank
bsp.: mysql_select_db("witze"); → stellt eine Verbindung *zur Datenbank witze* her

mysql_query(*SQL-Befehl*); → Führt eine Abfrage in der ausgewählten Datenbank aus
bsp.: mysql_query(„SELECT * FROM tabwitze); → wählt alle Datensätze in der Tabelle tabwitze aus.

mysql_num_rows(*Tabellenhandle*); → Sagt, wie viele Datensätze eine Tabelle beinhaltet
bsp.: mysql_num_rows($Witze); → Sagt, wie viele Witze in dem Array *$Witze* abgelegt sind

mysql_error(); → Gibt eine Beschreibung des letzten Fehlers aus *(Klammern ohne Inhalt)*

mysql_fetch_row(*Tabellenhandle*); → Liest einen Datensatz aus einer Tabelle aus und setzt den Zeiger auf den nächten Datensatz
bsp.: mysql_fetch_row($Witze); → liest beim ersten Aufruf den ersten Datensatz/ Witz aus *$Witze* aus – beim fünften Aufruf entsprechend den fünften Datensatz/ Witz

---

PHP-Befehle

date ("d.m.Y", time()); → gibt einen formatierten Zeit-Ausdruck aus

substr(Ausdruck,Start,Länge) → schneidet etwas aus einen *Ausdruck* heraus.
Echo substr("Hallo, du Schlafmütze",7,2); → würde von dem Text ab der 8.Stelle 2 Stellen ausgeben: *du*

SQL-Befehle

*SELECT * FROM tabelle*→ Auswahl aller Datensätze einer Tabelle

*SELECT * FROM tabelle ORDER BY Datum DESC*→ Auswahl aller Datensätze einer Tabelle, absteigend nach dem Feld *Datum* sortiert

*SELECT * FROM tabelle ORDER BY ID* → Auswahl aller Datensätze einer Tabelle, aufsteigend nach dem Feld *ID* sortiert

*SELECT * FROM tabelle WHERE ID='5'* →Auswahl aller Datensätze einer Tabelle, die im Feld *ID* den Wert 5 besitzen

INSERT INTO tabAutoren (Nr, NachName, VorName, GebJahr)
VALUES (1, 'Böll', 'Heinrich', 1917); → *Trägt Heinrich Böll in die Tabelle tabAutoren ein*