11 Ein Diskussionsforum mit PHP und MySQL

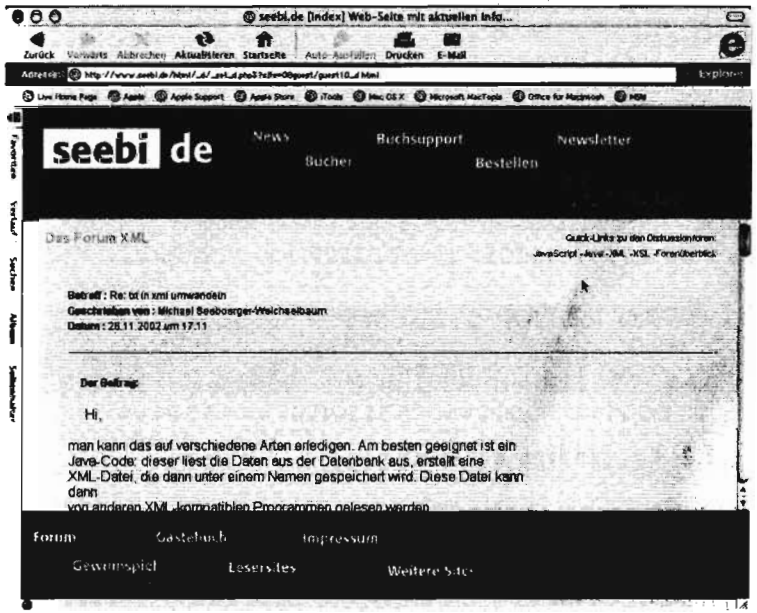
In diesem Kapitel vertiefen wir die Arbeit mit der Datenbank MySQL noch weiter. Wir beschäftigen uns damit, wie ein Diskussionsforum erstellt wird. Dazu ist eine Datenbank unabdingbar. In der Datenbank werden die einzelnen Beiträge gespeichert und ausgelesen. Dies ist im Prinzip sehr ähnlich wie viele der Anwendungen der letzten beiden Kapitel.

11.1 Aufbau eines Diskussionsforums

Ein konkretes Diskussionsforum sieht zunächst einfach aus. Der Code, der dahinter steht, ist jedoch etwas komplexer. Wir sehen uns ein Diskussionsforum in mehreren Schritten an. Wir werden in diesem Kapitel das Diskussionsforum einfach beginnen und sukzessive ausbauen.

11.1.1 Ein Beispiel für ein Diskussionsforum

Die Abbildung links zeigt ein typisches Diskussionsforum, das sich auf meiner Site *http://www.seebi.de* befindet. Dieses ist übrigens auch in PHP und MySQL programmiert.
Im Forumsüberblick wird ein Beitrag zunächst durch dessen Titelzeile aufgelistet, dann folgt der Absender des Beitrags und das Datum sowie die Uhrzeit. Jeder neue Eintrag wird ganz oben aufgelistet. Da es auf Beiträge einzelne (oder auch mehrere) Antworten geben kann, müssen die Antworten hierarchisch unterhalb des Original-Beitrages aufgelistet werden. Dieses ist nicht trivial, denn eine Antwort kann z. B. sehr spät



geschrieben worden sein. In der Zwischenzeit wurden komplett neue Beiträge mit anderen Themen geschrieben. Die Antwort darf dabei nicht als einzelner Beitrag ganz oben erscheinen, sondern muss als Antwort zu einem Beitrag eingerückt erscheinen.
Wird ein Beitrag (oder eine Antwort darauf) angeklickt, kann der komplette Beitrag gelesen werden (siehe Abbildung S. 307). Von hier aus kann über einen weiteren Link eine Antwort auf diesen Beitrag geschrieben werden.
Natürlich muss auch die Möglichkeit bestehen, einen neuen Beitrag zu schreiben (und nicht nur eine Antwort auf einen vorhandenen Beitrag). Dies geschieht über klassische Formularfelder (siehe nachfolgende Abbildung).

11.1.2 Notwendige Dateien

Bevor wir uns mit der Datenbank befassen, sollten wir uns überlegen, wie wir das Diskussionsforum in PHP realisieren. Wir benötigen insgesamt sechs Dateien:

Eine Übersichtsdatei *(index.php)*, die sämtliche Beiträge aus dem Diskussionsforum einzeilig auflistet (Beitrag, Absender, Datum und Uhrzeit). Jeder Beitrag kann angeklickt werden und ruft die Datei *read.php* auf.
Eine PHP-Datei *(read.php)*, die einen konkreten Beitrag aus der Datenbank ausliest.
Zwei PHP-Dateien *(replay.php* und *replay_entry.php)*, um eine Antwort auf einen Beitrag zu schreiben. In der Datei *replay.php* sind die Formularfelder für die Einträge enthalten, während die *replay_entry.php* die Einträge ausliest und die Daten in die Datenbank schreibt.
Eine HTML- und PHP-Datei (*new.html* und *new.php*), um eine neuen Beitrag zu erstellen. In der HTML-Datei sind die Formularfelder für die Einträge enthalten, während die PHP-Datei die Einträge ausliest und die Daten in die Datenbank schreibt.

11.1.3 Aufbau der Datenbank

Jetzt sollten wir uns noch intensive Gedanken um den Aufbau der Datenbank machen. Wir benötigen mehrere Felder, um die Informationen unterzubringen. Konkret brauchen wir acht Felder:

- Ein fortlaufende Nummer, die für jeden Neu-Eintrag (egal ob neuer Beitrag oder Antwort) hochgezählt wird. Anhand dieser Nummer kann jeder einzelne Beitrag aus der Datenbank ausgelesen werden. Das Feld bekommt den Namen *beitrags_id*. Dies ist der so genannte *Primärschlüssel*.
- Eine weitere Nummer, auf die sich der Beitrag bezieht. Handelt es sich um eine Antwort auf einen Beitrag, so wird hier die Nummer des Beitrags aufgelistet, auf die sich die Antwort bezieht. Handelt es sich um einen neuen Eintrag, ist die Nummer 0. Das Feld bekommt den Namen *bezugs_id*.
- Der Name des Internet-Surfers. Das Feld bekommt den Namen ***user***.
- Die E-Mail-Adresse des Internet-Surfers. Das Feld bekommt den Namen *email*.
- Das Datum des Beitrags. Das Feld bekommt den Namen ***datum***.
- Die Uhrzeit des Beitrags. Das Feld bekommt den Namen ***uhrzeit***.

- Die Betreff-Zeile. Das Feld bekommt den Namen *betreff*.
- Der konkrete Textbeitrag. Das Feld bekommt den Namen *beitragstext*.

Erstellen Sie ein neue Tabelle mit dem Namen *forum*:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><body>
<h3>Diskussions-Tabelle erstellen</h3>
<?php
   $db=mysql_connect("localhost","root","");
   mysql_select_db("manitu");
   $anfrage="CREATE TABLE forum (beitrags_id INT
                AUTO_INCREMENT, bezugs_id INT,
                user VARCHAR(50),email VARCHAR(50),
                datum VARCHAR(10),uhrzeit VARCHAR(10),
                betreff VARCHAR(255),
                beitragstext TEXT, PRIMARY KEY
                (beitrags_id))";
   mysql_query($anfrage);
   mysql_close($db);
?>
</body></html>
```

Bis auf die erste Spalte sind die weiteren Spalten *(bezugs_id, user, email, datum, uhrzeit, betreff, beitragstext)* vom Aufbau her bekannt. Neu ist nur die erste Spalte:

```
beitrags_id INT AUTO_INCREMENT
```

Die Spalte ist vom Typ *INT* und enthält den Eintrag *AUTO_INCREMENT*. Dadurch wird jeder neue Eintrag in dieser Spalte automatisiert hochgezählt. Wir brauchen uns nicht darum zu kümmern. Anhand dieser Zahl kann jeder Beitrag innerhalb des Forums eindeutig identifiziert werden. Gleichzeitig muss diese Spalte auch als so genannter *Primärschlüssel* definiert werden. Dieser identifiziert die Spalte als eine besondere Spalte, in der der hochzählende Wert nur einmal vorkommen kann. Dies ist nicht trivial, denn beispielsweise kann es bei der Wartung des Diskussionsforums (z. B. das manuelle Hinzufügen von

Beiträgen über phpMyAdmin) passieren, dass man auf den Wert nicht achtet und unvorsichtigerweise eine Zahl vergibt, die schon vorhanden ist. Das Kennzeichnen der ersten Spalte *beitrags_id* als Primärschlüssel schützt die Spalte vor derartigen fehlerhaften manuellen Eingriffen. Die Datenbank achtet darauf, dass der Wert in dieser Spalte immer hochgezählt und nur einmal vorhanden ist. Dadurch ist definitiv sicher gestellt, dass jeder Beitrag im Forum eine einmalige und eindeutige Zahl erhält, anhand deren der Beitrag genau identifiziert werden kann. In SQL wird eine Spalte als Primärschlüssel über *PRIMARY KEY* gekennzeichnet. Dies muss bei der Erzeugung der Tabelle durchgeführt werden. *PRIMARY KEY* wird an das Ende der SQL-Abfrage *CREATE TABLE* geschrieben. Die Spalte, die als Primärschlüssel definiert werden soll, wird in Klammern gesetzt:

```
PRIMARY KEY (beitrags_id)
```

11.2 Der Anfang des Diskussionsforums

Zum Start des Forum-Projekts brauchen wir zunächst nur vier Dateien, die wir jetzt anlegen werden:

- Die Datei *index.php*. Diese Datei liest sämtliche Einträge aus der *forum*-Tabelle aus und listet die Betreff-Zeilen untereinander auf.
- Die Datei *read.php*, die den konkreten Forumsbeitrag anzeigt.
- Die Dateien *new.html* und *new.php*, um einen neuen Beitrag hinzuzufügen.

Wir sehen uns die Basiscodes Schritt für Schritt an.

11.2.1 Der Forumsüberblick

Die Datei *index.php* bietet einen Forumsüberblick, indem die Beiträge untereinander aufgelistet werden (siehe nachfolgende Abbildung).
Die Datei *index.php* sieht so aus:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><body>
```


```
<h3>Das Diskussionsforum</h3>
<table border="1" width="500">
<th>Beitrag</th>
<th>von</th>
<th>Datum und Uhrzeit</th>
<?php
   $db=mysql_connect("localhost","root","");
   mysql_select_db("manitu");
   $anfrage="SELECT * FROM forum";
   $ergebnis=mysql_query($anfrage);
   $anz=mysql_num_rows($ergebnis);
   for($a=$anz-1;$a>-1;$a--) {
      mysql_data_seek($ergebnis,$a);
      $zeile=mysql_fetch_row($ergebnis);
      print("<tr align='center'>");
      print("<td>");
```



```
      print($zeile[6]);
      print("</td>");
      print("<td>");
      print($zeile[2]);
      print("</td>");
      print("<td>");
      print($zeile[4]);
      print(" um ");
      print($zeile[5]);
      print("</td>");
      print("</tr>");
   }
   print("</table>");
   mysql_close($db);
?>
</body></html>
```

Der Code ist relativ einfach, denn er basiert zunächst nur auf bekanntem Wissen. Das Forum ist eine HTML-Tabelle, in der die einzelnen Beiträge nur mit der Betreff-Zeile, dem Absender und dem Datum sowie der Uhrzeit aufgelistet werden. Im PHP-Code stellen wir den Datenbankconnect her und richten die SQL-Anfrage an die Datenbank. In *$ergebnis* liegt wieder das Ergebnis der Datenbankabfrage vor.
Der weitere Code ist anders als bisher. Denn wir verwenden jetzt eine *for*-Schleife zum Auslesen der einzelnen Zeilen aus dem Ergebnis. Da in der Datenbanktabelle die einzelnen neuen Einträge untereinander stehen, befindet sich am Ende der Datenbanktabelle *forum* der aktuelle Beitrag. Dieser muss natürlich in der HTML-Tabelle ganz am Anfang stehen. Dazu benutzen wir eine *for*-Schleife, die das Suchergebnis von hinten ausgibt. Das bedeutet, dass der aktuellste Datensatz in der HTML-Tabelle zuerst angezeigt wird:

```
$anz=mysql_num_rows($ergebnis);
for($a=$anz-1;$a>-1;$a--) {
```

In *$anz* liegt die Anzahl der Zeilen im Suchergebnis vor. Die *for*-Schleife beginnt jedoch mit der Initialisierung *$a=$anz-1*. Dies liegt daran, dass *$anz* die Anzahl der Zeilen enthält. Die Nummerierung der Zeilen

in der Datenbanktabelle beginnt jedoch bei 0, sodass der Initialisierungswert von *$a* bei *$anz-1* beginnen muss. Würde als Initialisierungswert stattdessen *$anz* genommen werden, gäbe es einen Fehler. Haben wir z. B. fünf Zeilen im Suchergebnis, sind dies in der Nummerierung der Datenbanktabelle die Zeilen 0 bis 4. Die Laufbedingung der *for*-Schleife ist *$a>-1*, d. h., die Schleife zählt so lange, bis der Wert 0 erreicht ist. Die Schleife wird über *$a--* heruntergezählt. Damit lesen wir das Suchergebnis aus der Datenbanktabelle von hinten nach vorne aus. Im nächsten Schritt nutzen wir dazu die sinnvolle Funktion *mysql_data_seek()*. Mit Hilfe dieser Funktion wählen wir konkret eine bestimmte Zeile aus der Suchanfrage aus. Man spricht auch davon, dass der aktuelle Zeiger an eine bestimmte Zeilenposition gesetzt wird. Anschließend liest *mysql_fetch_row()* die einzelnen Datensätze der Zeile aus und speichert diese in der Variablen *$zeile*:

```
mysql_data_seek($ergebnis,$a);
$zeile=mysql_fetch_row($ergebnis);
```

mysql_data_seek() erwartet zwei Übergabeparameter: zuerst die Variable, die das gesamte Suchergebnis enthält, und als zweiten Parameter die aktuelle Zeile, die ausgelesen werden soll. Dies ist natürlich immer *$a*, d. h. der aktuelle Wert.
Nacheinander geben wir jetzt die notwendigen Daten aus. Zuerst erscheint die Betreff-Zeile (*$zeile[6]*), dann der Absender (*$zeile[2]*) und schließlich das Datum sowie die Uhrzeit (*$zeile[4]* und *$zeile[5]*). Der konkrete Beitragstext muss hier noch nicht ausgegeben werden, denn hierbei handelt es sich um einen Forumsüberblick.

11.2.2 Einen Beitrag lesen

Um einen Forumsbeitrag lesen zu können, müssen wir sowohl die Datei *index.php* etwas verändern, wie auch eine neue Datei *read.php* erstellen. Jeder Forumsbeitrag, der in *index.php* aufgelistet ist, soll als Hyperlink realisiert werden. Klicken wir eine Betreff-Zeile eines Beitrags an, wird *read.php* aufgerufen. Diese Datei liest konkret den Forumsbeitrag aus der Datenbank aus und stellt ihn dar.
Das Entscheidende bei diesem Vorgang ist der Übergabeparameter an die Datei *read.php*. Diese Datei muss ja wissen, welcher Forumsbeitrag ausgelesen werden soll. Dazu dient uns das Feld *beitrags_id* aus der Datenbanktabelle. Anhand dieser einmaligen Zahl kann ja jeder Beitrag eindeutig identifiziert werden!

Sehen wir uns zunächst die Änderungen in der Datei *index.php* an. Hier müssen wir jede Betreff-Zeile als Hyperlink modifizieren. Die Änderungen sind nur in der *for*-Schleife nötig:

```
for($a=$anz-1;$a>-1;$a--) {
   mysql_data_seek($ergebnis,$a);
   $zeile=mysql_fetch_row($ergebnis);
   print("<tr align='center'>");
   print("<td>");
   print("<a href='read.php?forums_id=");
   print($zeile[0]);
   print("'>");
   print($zeile[6]);
   print("</a>");
   print("</td>");
   print("<td>");
   print($zeile[2]);
   print("</td>");
   print("<td>");
   print($zeile[4]);
   print(" um ");
   print($zeile[5]);
   print("</td>");
   print("</tr>");
}
```

In der ersten HTML-Tabellenzeile erfolgt ja die Ausgabe der Betreff-Zeile. Diese erscheint nun als Hyperlink. Dazu wird über PHP die Betreff-Zeile in ein *<a>*-Tag gelegt. Entscheidend ist nur der Aufruf der Datei *read.php*:

```
print("<a href='read.php?forums_id=");
print($zeile[0]);
print("'>");
```

An die Datei *read.php* übergeben wir noch eine Variable. Diese Übergabe wird über das Fragezeichen *(?)* erreicht. Wir hängen damit eine Va-

riable (und deren Wert) an die Anfrage im Rahmen des HTTP-Protokolls an. Dies ist ähnlich wie bei einem Formular, bei dem die Methode *GET* verwendet wird. Auch hier werden die einzelnen Daten über das Fragezeichen angehängt. Hier machen wir dies genauso, hängen jedoch die Variable und deren Wert manuell an. Die *POST*-Methode können wir hier übrigens nicht einsetzen. Dies liegt daran, dass wir keine andere Möglichkeit haben, die Variable an die Datei *read.php* weiterzugeben.

Nach dem Fragezeichen folgt der Name der Variablen *(forums_id)*. Über das Gleichheitszeichen weisen wir einen Wert zu. Der Wert, den wir hier übergeben müssen, ist der Wert, der in dem ersten Feld der Datenbanktabelle steht. Dieses ist ja die einmalige Zahl des Beitrags, d. h. dessen Primärschlüssel. Diese Zahl geben wir an die Datei *read.php*. In dem entsprechenden PHP-Code kann dann mit Hilfe dieser Zahl der Beitrag ausgelesen werden. Die Datei *read.php* sieht so aus:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><body>
<h3>Der Forumsbeitrag</h3>
<div style="width: 400px">
<?php
   $id=$HTTP_GET_VARS['forums_id'];
   $db=mysql_connect("localhost","root","");
   mysql_select_db("manitu");
   $anfrage="SELECT * FROM forum WHERE beitrags_id
             LIKE '";
   $anfrage.=$id;
   $anfrage.="'";
   $ergebnis=mysql_query($anfrage);
   $zeile=mysql_fetch_row($ergebnis);
   print("<b>Betreff: </b>");
   print($zeile[6]);
   print("<br><b>von: </b>");
   print($zeile[2]);
   print("<br><b>E-Mail: </b>");
   print($zeile[3]);
   print("<br><b>Geschrieben am: </b>");
   print($zeile[4]);
   print(" um ");
```



```
   print($zeile[5]);
   print("<br><br><hr><br><br>");
   print($zeile[7]);
   mysql_close($db);
?>
</div></body></html>
```

Im PHP-Code muss zuerst die übergebene Variable *forums_id* ausgelesen werden. Nach dem Aufbau der Datenbankverbindung erstellen wir die Datenbankanfrage:

```
$anfrage="SELECT * FROM forum WHERE beitrags_idLIKE '";
$anfrage.=$id;
$anfrage.="'";
```

Als *WHERE*-Parameter verwenden wir *beitrags_id*, denn in dieser Spalten wollen wir suchen. Für den *LIKE*-Wert setzen wir natürlich die ermittelte Variable *$id* ein. Anschließend senden wir die Anfrage an die Datenbank, erhalten das Ergebnis und lassen es uns ausgeben. Die Abbildung auf der nächsten Seite zeigt eine Anzeige eines Forumsbeitrags im Browser.

11.2.3 Einen neuen Beitrag schreiben

Das Hinzufügen eines neuen Forumbeitrags geschieht klassisch über Formularfelder. Dazu benötigen wir zwei Dateien: *new.html* enthält die entsprechenden HTML-Formularfelder, und *new.php* wertet diese aus. Anschließend schreibt ein PHP-Code in *new.php* die Daten in die Datenbank. Unberücksichtigt sind jetzt noch die Antworten auf einen Forumsbeitrag – mit diesem Problem beschäftigen wir uns später. Die Datei *new.html* sieht so aus:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<title>PHP-Forumsbeitrag</title></head>
<body>
<div style="font-family:arial">
<h2>Ihr Forumsbeitrag:</h2>
<form method="post" action="new.php">
```


```
<table border="0">
<tr>
<td>Ihr Name</td>
<td><input type="text" name="user"></td></tr>
<tr>
<td>Ihre E-Mail-Adresse</td>
<td><input type="text" name="mail"></td></tr>
<tr>
<td>Betreff-Zeile</td>
<td><input type="text" name="betreff"></td></tr>
<tr><td>Ihr Eintrag</td>
<td><textarea name="forumsbeitrag" cols="40" rows="5">
</textarea></td></tr>
<tr>
```



```
<td><input type="submit" value="Abschicken">
<input type="reset" value="Löschen"></td></tr>
</table>
</form></div>
</body></html>
```

Das Formular enthält mehrere Formularfelder. Die Variablen lauten *user* (für den Namen des Internet-Surfers), *mail* (für dessen E-Mail-Adresse), *betreff* (die Betreff-Zeile) und *forumsbeitrag* (der konkrete Beitragstext). Datum und Uhrzeit werden wir über das PHP-Skript ermitteln. Auch die einmalige Zahl (*beitrags_id*) brauchen wir nicht selbst zu setzen. Die Datei *new.php*, die die Auswertung und das Schreiben in die Datenbank vornimmt, sieht so aus:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><body>
<?php
   $user=$HTTP_POST_VARS['user'];
   $mail=$HTTP_POST_VARS['mail'];
   $betreff=$HTTP_POST_VARS['betreff'];
   $forumsbeitrag=$HTTP_POST_VARS['forumsbeitrag'];
   $punkt=".";
   $datum=date("d");
   $datum.=$punkt;
   $datum.=date("m");
   $datum.=$punkt;
   $datum.=date("Y");
   $zeit=date("G");
   $zeit.=$punkt;
   $zeit.=date("i");
   $zeit.=$punkt;
   $zeit.=date("s");
   //Sonderzeichen beachten:
   $user=htmlspecialchars($user);
   $user=htmlentities($user);
   $betreff=htmlspecialchars($betreff);
   $betreff=htmlentities($betreff);
   $forumsbeitrag=htmlspecialchars($forumsbeitrag);
   $forumsbeitrag=htmlentities($forumsbeitrag);
```

```
    $forumsbeitrag=nl2br($forumsbeitrag);
    $db=mysql_connect("localhost","root","")
       or die("<b>Kein Connect zum
                Datenbankserver!</b>");
    mysql_select_db("manitu")
       or die("<b>Datenbank konnte nicht angesprochen
                werden</b>");
    $anfrage="INSERT INTO forum VALUES ('";
    $anfrage.="0', '0', '";
    $anfrage.=$user;
    $anfrage.="', '";
    $anfrage.=$mail;
    $anfrage.="', '";
    $anfrage.=$datum;
    $anfrage.="', '";
    $anfrage.=$zeit;
    $anfrage.="', '";
    $anfrage.=$betreff;
    $anfrage.="', '";
    $anfrage.=$forumsbeitrag;
    $anfrage.="')";
    mysql_query($anfrage)
       or die("<b>Fehler bei der
                Datenbankanfrage</b>");
    mysql_close($db);
    print("<p>Vielen Dank f&uuml;r Ihren
           Beitrag!</p>");
    print("<a href='index.php'>Zur&uuml;ck zum
           Forums&uuml;berblick</a>");
?>
</body></html>
```

Der Code dürfte Ihnen teilweise bekannt vorkommen. Er ist ähnlich dem PHP-Code aus Kapitel 10.3, in dem die Lesersites per Datenbank eingetragen wurden. Zuerst ermitteln wir Datum und Uhrzeit. Anschließend holen wir uns die vier übergebenen Variablen:

```
$user=$HTTP_POST_VARS['user'];
$mail=$HTTP_POST_VARS['mail'];
```



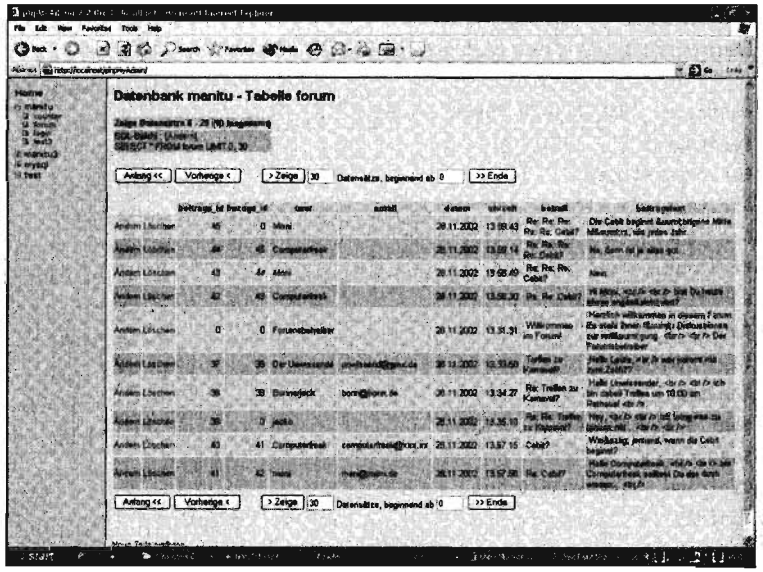
```
$betreff=$HTTP_POST_VARS['betreff'];
$forumsbeitrag=$HTTP_POST_VARS['forumsbeitrag'];
```

Anders als im letzten Abschnitt nutzen wir jetzt statt *$HTTP_GET_VARS* nun *$HTTP_POST_VARS*, da ja die Variablen vom Formularfeld über die *POST*-Methode gesendet werden. In den nächsten Zeilen müssen wir die Variablen *$user*, *$betreff* und *$forumsbeitrag* nachbearbeiten, indem wir die Sonderzeichen codieren und im Beitragstext noch Zeilenumbrüche über *
* hinzufügen.
Dann erstellen wir den SQL-Befehl *INSERT INTO*:

```
$anfrage="INSERT INTO forum VALUES ('";
$anfrage.="0', '0', '";
$anfrage.=$user;
$anfrage.="', '";
$anfrage.=$mail;
$anfrage.="', '";
$anfrage.=$datum;
$anfrage.="', '";
$anfrage.=$zeit;
$anfrage.="', '";
$anfrage.=$betreff;
$anfrage.="', '";
$anfrage.=$forumsbeitrag;
$anfrage.="')";
```

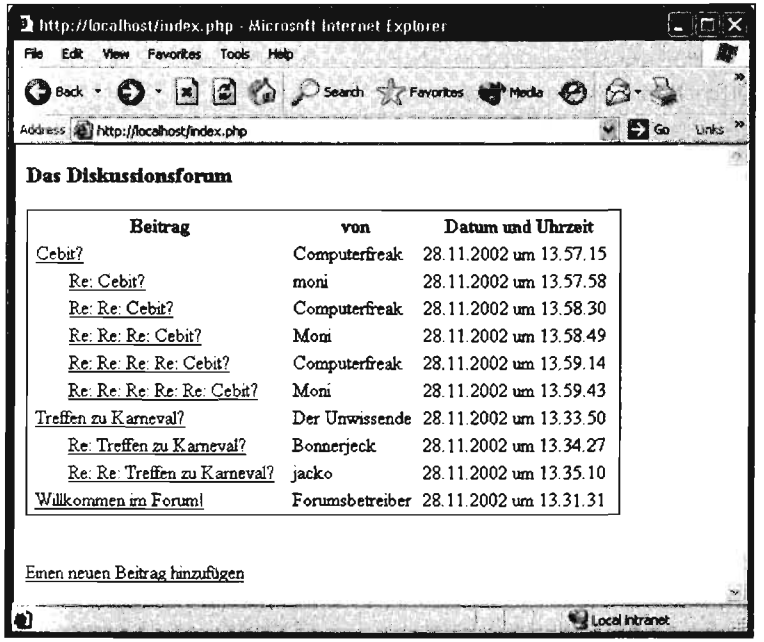
Die vier übergebenen Variablen, das Datum und die Uhrzeit werden nacheinander in die Variable *$anfrage* gepackt. Achten Sie aber besonders auf die ersten beiden Werte, die ja in der Datenbanktabelle den Primärschlüssel und einen Bezugswert (für eine Antwort) angeben. Beide sind hier interessanterweise auf 0 gesetzt. Es könnten bei dieser Fassung auch andere Werte stehen. Die erste Spalte in der Datenbanktabelle ist immer der Primärschlüssel, der von der Datenbank automatisch hochgezählt wird. Wir übergeben hier eine 0, doch dieser Wert ist egal. Er wird auf jeden Fall von der Datenbank überschrieben. Der zweite Wert ist ebenfalls 0. Diesen werden wir erst später ändern, daher kann dieser zunächst 0 sein. Der Rest des Code ist nicht neu, Sie kennen ihn aus dem letzten Kapitel.
Schreiben Sie jetzt über die Datei *new.html* mehrere neue Test-Beiträge

in das Diskussionsforum. Sie werden sehen, dass die Datei *index.php* alle neuen Beiträge auflistet. Sie können jeden Beitrag anklicken und lesen. Sehen Sie sich auch die Beiträge in der Datenbank über phpMyAdmin einmal an (siehe nachfolgende Abbildung). Sie erkennen die Beiträge wieder und sehen sofort, dass der Primärschlüssel immer hochgezählt wird – egal was Sie bei *INSERT INTO* als ersten Wert übergeben haben.



11.3 Antworten auf Beiträge berücksichtigen

Wir haben in unserem Diskussionsforum Antworten auf Beiträge noch nicht berücksichtigt. Diese Antworten müssen auch entsprechend grafisch dargestellt werden, d. h., bei dem Forumsüberblick muss erkennbar sein, welche Antwort zu welchem Beitrag gehört (siehe nachfolgende Abbildung).



Dies erreichen wir in zwei Schritten. Zunächst wird jede Antwort als normaler Forumsbeitrag angesehen. Dabei müssen wir jedoch den Ursprungsbeitrag, auf den die Antwort geschrieben wird, aktualisieren. Wir müssen in der zweiten Spalte der Datenbanktabelle jetzt die Zahl des Primärschlüssels der Antwort angeben, d. h. die Zahl aus der Spalte *beitrags_id* der Antwort. Bei dem Forumsüberblick müssen wir im zweiten Schritt den Inhalt dieser Spalte auslesen und entsprechend unterhalb des Ursprungsbeitrags anzeigen. Dieses Vorgehen führen wir in zwei Schritten aus:

1. Die Antwort erstellen, in die Datenbank schreiben und Ursprungsbeitrag aktualisieren.
2. Den Forumsüberblick neu programmieren und die Antworten berücksichtigen.

11.3.1 Eine Antwort schreiben

Um eine Antwort in die Datenbank zu schreiben, muss Folgendes modifiziert bzw. neu erstellt werden:

- Die Datei *read.php*, die einen Forumsbeitrag anzeigt.
- Die Dateien *replay.php* und *replay_entry.php*, die die Antwort konkret in die Datenbank schreibt.

Eine Antwort kann nur auf einen existierenden Beitrag geschrieben werden. Daher müssen wir die Datei *read.php*, die ja den Inhalt eines Forumsbeitrags anzeigt, etwas verändern. Nachdem der Forumsbeitrag aufgelistet wurde, müssen wir am Ende einen Link auf die Datei *replay.php* legen. Der Link zeigt auf die Datei *replay.php* und übergibt gleichzeitig die aktuelle Zahl, die sich in der Datenbanktabelle in der Spalte *beitrags_id* befindet:

```
print("<a href='replay.php?forums_id=");
print($id);
print("'>Schreiben Sie hier eine Antwort auf diesen
          Beitrag!</a>");
```

In der Variablen *$id* befindet sich die aktuelle Zahl aus der Spalte *beitrags_id*, die wir an *replay.php* übergeben und dort als Variable *forums_id* bezeichnen. Warum müssen wir diese überhaupt übergeben? Dies liegt daran, dass wir später den Ursprungsbeitrag, auf den wir die Antwort schreiben (über die Dateien *replay.php* und *replay_entry.php*), in der Datenbank aktualisieren müssen. In der zweiten Datenbankspalte *(bezugs_id)* müssen wir die Zahl des Primärschlüssels der Antwort einfügen.
Interessant sind jetzt die beiden Dateien *replay.php* und *replay_entry. php*. Sehen wir uns zuerst *replay.php* an. Diese enthält die Formularfelder, um die konkrete Antwort auf den Beitrag zu schreiben, während *replay_entry.php* den Eintrag in der Datenbank vornimmt. Wir müssen die übergebene Zahl an *replay_entry.php* weiterreichen. Daher muss *replay.php* eine PHP-Datei sein und keine HTML-Datei:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
```



```
<title>PHP-Forumsbeitrag - Antwort</title></head>
<body>
<div style="font-family:arial">
<h2>Ihre Antwort auf den Forumsbeitrag:</h2>
<form method="post" action="replay_entry.php">
<table border="0">
<tr>
<td>Ihr Name</td>
<td><input type="text" name="user"></td></tr>
<tr>
<td>Ihre E-Mail-Adresse</td>
<td><input type="text" name="mail"></td></tr>
<tr>
<td>Betreff-Zeile</td>
<td>
<?php
   $id=$HTTP_GET_VARS['forums_id'];
   $db=mysql_connect("localhost","root","");
   mysql_select_db("manitu");
   $anfrage="SELECT * FROM forum WHERE beitrags_id LIKE '";
   $anfrage.=$id;
   $anfrage.="'";
   $ergebnis=mysql_query($anfrage);
   $zeile=mysql_fetch_row($ergebnis);
   $betreff="<input type='text' name='betreff' value='";
   $betreff.="Re: ";
   $betreff.=$zeile[6];
   $betreff.="'>";
   print($betreff);
   print("<input type='hidden' name='forums_id' value='");
   print($id);
   print("'>");
?>
</td></tr>
<tr><td>Ihr Eintrag</td>
<td><textarea name="forumsbeitrag" cols="40" rows="5">
</textarea></td></tr>
```

```
<tr>
<td><input type="submit" value="Abschicken">
<input type="reset" value="Löschen"></td></tr>
</table>
</form></div>
</body></html>
```

Zu Beginn der Datei befinden sich die bekannten Formularfelder. Die Datei ist im Prinzip sehr ähnlich zu *new.html* für einen neuen Forumsbeitrag. Anders als bei *new.html* haben wir zwei Änderungen im Code:

- Die Betreff-Zeile wird automatisch aus der Datenbank ermittelt und mit einem *Re:* (für Replay) versehen.
- Wir müssen den Primarschlüssel des Forumsbeitrags an die Datei *replay_entry.php* zur dortigen Auswertung weiterreichen.

Im PHP-Code ermitteln wir zuerst die übergebene Variable:

```
$id=$HTTP_GET_VARS['forums_id'];
```

Diese werden wir weiter unten in der Datei an *replay_entry.php* weiterreichen.
Sehr sinnvoll ist es, die Betreff-Zeile aus der Datenbank zu ermitteln und diese im Formularfeld anzeigen zu lassen. Dazu müssen wir einen Datenbankzugriff durchführen:

```
$db=mysql_connect("localhost","root","");
mysql_select_db("manitu");
$anfrage="SELECT * FROM forum WHERE beitrags_id LIKE '";
$anfrage.=$id;
$anfrage.="'";
$ergebnis=mysql_query($anfrage);
$zeile=mysql_fetch_row($ergebnis);
```

Wir müssen dazu den *SELECT*-Befehl einsetzen und als Suchkriterium den Primärschlüssel (*$id*) in der Spalte *beitrags_id* angeben. Aus dem Ergebnis in *$zeile* benötigen wir nur die Betreff-Zeile, die sich in *$zeile[6]*

befindet. Jetzt erstellen wir von PHP aus das Formularfeld, das die Betreff-Zeile anzeigt, und fügen die ausgelesene Betreff-Zeile direkt ein:

```
$betreff="<input type='text' name='betreff' value='";
$betreff.="Re: ";
$betreff.=$zeile[6];
$betreff.="'>";
print($betreff);
```

Um die Betreff-Zeile in das Formularfeld zu bekommen, verwenden wir das Attribut *value* des *<input>*-Tags, das eine Vorgabe für das Formularfeld tätigt. Die Betreff-Zeile wird noch mit *Re:* versehen, Sie können auch statt dessen das verbreitete *AW:* (für Antwort) verwenden.
Abschließend erstellen wir noch ein weiteres Formularfeld. Dieses ist versteckt und enthält den Primärschlüssel, den wir ja an *replay_entry.php* übergeben müssen:

```
print("<input type='hidden' name='forums_id' value='");
print($id);
print("'>");
```

Über das Attribut *type='hidden'* wird das Formularfeld versteckt, d. h., es ist für den Internet-Surfer nicht sichtbar. Über das Attribut *value* schreiben wir den Wert der Variablen *$id* in das unsichtbare Formularfeld hinein.
Warum hängen wir die Variable *$id* nicht einfach über das Fragezeichen (*?*) an die URL an? Der Grund liegt darin, dass wir zur Versendung des Formulars die *POST*-Methode einsetzen. Das Fragezeichen ist für die *GET*-Methode vorgesehen. Wir können immer nur eine Methode beim Versand der Formulardaten benutzen.
Alternativ könnten wir natürlich genrell auf die GET-Methode umstellen, dann würden aber alle Formulardaten an die URL angehängt werden und sichtbar sein – das ist nicht unbedingt sehr schön. Daher verstecken wir die Variable *$id* als unsichtbares Formularfeld, das zusammen mit den anderen Feldern mit Hilfe der *POST*-Methode übertragen wird.
Klickt der Internet-Surfer auf den Schaltknopf *Abschicken*, wird die Datei *reply_entry.php* aufgerufen, die Antwort in die Datenbank geschrie-

ben und der Forumsbeitrag, auf den sich die Antwort bezieht, aktualisiert. Die Datei *replay_entry.php* sieht so aus:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><body>
<?php
   $id=$HTTP_POST_VARS['forums_id'];
   $user=$HTTP_POST_VARS['user'];
   $mail=$HTTP_POST_VARS['mail'];
   $betreff=$HTTP_POST_VARS['betreff'];
   $forumsbeitrag=$HTTP_POST_VARS['forumsbeitrag'];
   //Zuerst Datum und Uhrzeit generieren:
   $punkt=".";
   $datum=date("d");
   $datum.=$punkt;
   $datum.=date("m");
   $datum.=$punkt;
   $datum.=date("Y");
   $zeit=date("G");
   $zeit.=$punkt;
   $zeit.=date("i");
   $zeit.=$punkt;
   $zeit.=date("s");
   //Sonderzeichen beachten:
   $user=htmlspecialchars($user);
   $user=htmlentities($user);
   $betreff=htmlspecialchars($betreff);
   $betreff=htmlentities($betreff);
   $forumsbeitrag=htmlspecialchars($forumsbeitrag);
   $forumsbeitrag=htmlentities($forumsbeitrag);
   $forumsbeitrag=nl2br($forumsbeitrag);
   //Antwort in die DB schreiben:
   $db=mysql_connect("localhost","root","")
      or die("<b>Kein Connect zum
                Datenbankserver!</b>");
   mysql_select_db("manitu")
      or die("<b>Datenbank konnte nicht angesprochen
                werden</b>");
```



```
   $anfrage="INSERT INTO forum VALUES ('0','0','";
   $anfrage.=$user;
   $anfrage.="', '";
   $anfrage.=$mail;
   $anfrage.="', '";
   $anfrage.=$datum;
   $anfrage.="', '";
   $anfrage.=$zeit;
   $anfrage.="', '";
   $anfrage.=$betreff;
   $anfrage.="', '";
   $anfrage.=$forumsbeitrag;
   $anfrage.="')";
   mysql_query($anfrage)
      or die("<b>Fehler bei der Datenbankanfrage</b>");
   //Jetzt die neue beitrags_id
   //der soeben eingetragenen
   //Antwort ermitteln:
   $antwort_id=mysql_insert_id($db);
   //Usprungsbeitrag, auf dem die Antwort
   //geschrieben wurde, aktualisieren:
   $anfrage="SELECT * FROM forum WHERE beitrags_id LIKE '";
   $anfrage.=$id;
   $anfrage.="'";
   $ergebnis=mysql_query($anfrage);
   $zeile=mysql_fetch_row($ergebnis);
   $update="UPDATE forum SET bezugs_id='";
   $update.=$antwort_id;
   $update.="' WHERE beitrags_id='";
   $update.=$id;
   $update.="'";
   mysql_query($update)
      or die(mysql_errno().": ".mysql_error());
   mysql_close($db);
   print("<p>Vielen Dank f&uuml;r Ihren
            Beitrag!</p>");
   print("<a href='index.php'>Zur&uuml;ck zum
               Forums&uuml;berblick</a>");
```

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

Der Code enthält einige bekannte Elemente. Zuerst ermitteln wir wieder das Datum und die Uhrzeit, anschließend lesen wir wieder die Variablen aus den Formularfeldern aus:

```
$id=$HTTP_POST_VARS['forums_id'];
$user=$HTTP_POST_VARS['user'];
$mail=$HTTP_POST_VARS['mail'];
$betreff=$HTTP_POST_VARS['betreff'];
$forumsbeitrag=$HTTP_POST_VARS['forumsbeitrag'];
```

Dann konvertieren wir eventuell vorhandene Sonderzeichen und schreiben die Antwort über den SQL-Befehl *INSERT INTO* in die Datenbank. Dieser Abschnitt ist identisch mit dem Schreiben eines neuen Forumsbeitrags. Wir müssen noch die Zahl des Primärschlüssels der Antwort kennen, damit im Ursprungsbeitrag diese Zahl für die Antwort eingetragen werden kann. Soeben haben wir ja einen neuen Datensatz in die Datenbank geschrieben, und dieser hat einen neuen Wert für die erste Tabellenspalte *beitrags_id* erhalten. Diese ermitteln wir über eine einfache PHP-Funktion:

```
$antwort_id=mysql_insert_id($db);
```

mysql_insert_id() gibt die Zahl des Primärschlüssels zurück, die zuletzt geschrieben wurde. Dabei muss die Funktion sich auf eine *INSERT-INTO*-Operation beziehen. Diese Zahl speichern wir in *$antwort_id*. Dann müssen wir den Datensatz aktualisieren, der den Ursprungsbeitrag enthält, auf den sich die Antwort bezieht. Dazu müssen wir zuerst den Ursprungsbeitrag ermitteln. Wir haben ja dessen Primärschlüssel, also können wir diesen über eine *SELECT*-Abfrage erhalten:

```
$anfrage="SELECT * FROM forum WHERE beitrags_id LIKE '";
$anfrage.=$id;
$anfrage.="'";
$ergebnis=mysql_query($anfrage);
$zeile=mysql_fetch_row($ergebnis);
```


In *$zeile* liegt jetzt der gesamte Datensatz vor. Wir müssen nur die zweite Spalte aktualisieren. Hier muss der Primärschlüssel der Antwort (die liegt ja in *$antwort_id* vor) eingetragen werden. Dazu greifen wir zum SQL-Befehl *UPDATE*, da ja der Datensatz nur aktualisiert werden muss:

```
$update="UPDATE forum SET bezugs_id='";
$update.=$antwort_id;
$update.="' WHERE beitrags_id='";
$update.=$id;
$update.="'";
mysql_query($update)
   or die(mysql_errno().": ".mysql_error());
```

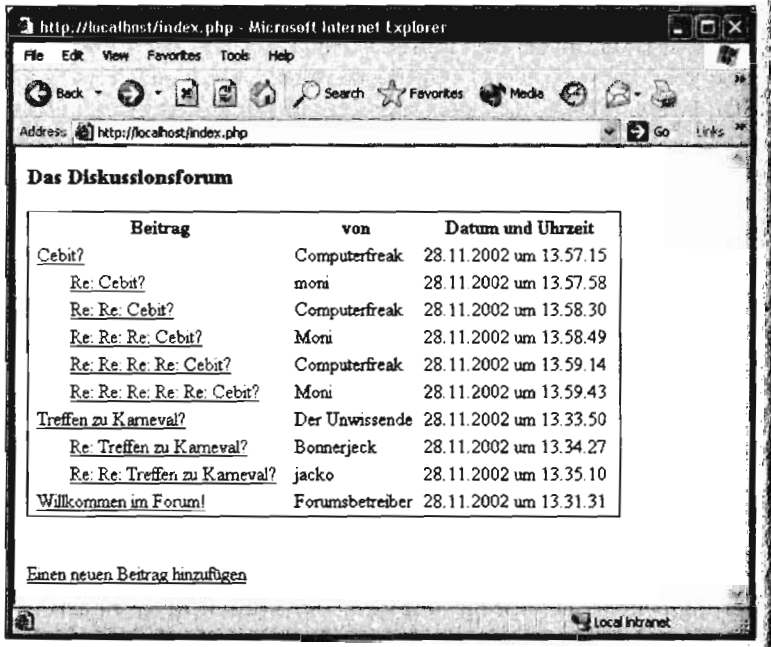
Der *UPDATE*-Befehl ist kurz, denn wir geben über *SET* nur die Spalte an, die im Datensatz aktualisiert werden muss. Wir müssen nicht alle sieben Spalten des Datensatzes aktualisieren. Das könnte man zwar auch machen, ist aber nicht sinnvoll, da sich ja nur die zweite Spalte ändert. Um den Datensatz auszuwählen, verwenden wir bei *UPDATE* das Statement *WHERE*. Das Suchkriterium ist ja der Primärschlüssel des Ursprungsbeitrags, den wir hier angeben *($id)*. Dann wird das Update des Datensatzes ausgeführt.

Geben Sie jetzt einmal mehrere Beiträge ein, schreiben Sie Antworten auf die Beiträge (auch z. B. Antworten auf Antworten). Sehen Sie sich die Tabelle dann einmal in phpMyAdmin an (siehe Abbildung am Ende von Kapitel 11.2). Sie erkennen deutlich, dass die Beiträge, die keine Antwort besitzen, in der zweiten Spalte *bezugs_id* eine 0 tragen, während alle Beiträge mit Antworten einen anderen Wert enthalten. Dies werden wir im nächsten Schritt ausnutzen.

11.3.2 Die Antworten im Forumsüberblick anzeigen

Bisher hatten wir uns noch nicht um eine Ausgabe der Antworten im Forumsüberblick *(index.php)* gekümmert. Alle Datensätze werden nacheinander ausgegeben. Die Ausgabe muss so umgestellt werden, dass sich eine Antwort unterhalb des Ursprungsbeitrags befindet. Die nachfolgende Abbildung zeigt eine derartige korrekte Ausgabe im Browser.

Unterhalb eines Ursprungsbeitrags befinden sich eine oder **mehrere** Antworten. Auf jede Antwort kann auch wieder ein Antwort **geschrieben** werden. Die Antworten werden immer unterhalb des Ursprungs-



beitrags angezeigt. Die Antworten werden auch gleichzeitig nach rechts eingerückt, damit sofort erkennbar ist, auf welchen Forumsbeitrag sich die Antworten beziehen.

Wir müssen dazu den Forumsüberblick in *index.php* überarbeiten. Den Code müssen wir komplett umstellen. Der generelle «Trick» hierbei: Die Ausgabe eines Forumsbeitrags befindet sich in einer separaten Funktion. Innerhalb dieser Funktion wird geprüft, ob eine Antwort auf den Beitrag vorliegt. Ist dies der Fall, wird jetzt unterhalb des Beitrags die Antwort ermittelt und durch einen erneuten Aufruf dieser Funktion die Anwort ausgegeben. Hier wird abermals geprüft, ob eine Antwort vorliegt. Ist dies der Fall, wird die Antwort auf die Antwort ermittelt und erneut die Funktion zur Ausgabe aufgerufen. Wir müssen hier rekursiv arbeiten, d.h., die Funktion muss sich immer dann wieder selbst aufrufen, wenn eine Antwort auf einen Beitrag vorliegt. Der Code zu *index.php* sieht jetzt so aus:


```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><body>
<h3>Das Diskussionsforum</h3>
<table style="border:black solid;border-width: 1px;border-
style:inset" width="500">
<th>Beitrag</th>
<th>von</th>
<th>Datum und Uhrzeit</th>
<?php
   $db=mysql_connect("localhost","root","");
   mysql_select_db("manitu");
   $anfrage="SELECT * FROM forum";
   $ergebnis=mysql_query($anfrage);
   $anz=mysql_num_rows($ergebnis);
   for($a=$anz-1;$a>-1; $a--) {
      mysql_data_seek($ergebnis,$a);
      $zeile=mysql_fetch_row($ergebnis);
      $antw=strstr($zeile[6],"Re:");
      if($antw==false) {
         ausgabe($zeile);
      }
   }
   print("</table>");
   mysql_close($db);
   function ausgabe($datensatz) {
      print("<tr align='left'>");
      print("<td>");
      $antw=strstr($datensatz[6],"Re:");
      if($antw==false) {
         print(" ");
      }
      else {
         print("    ");
         print("    ");
      }
      print("<a href='read.php?forums_id=");
      print($datensatz[0]);
      print("'>");
      print($datensatz[6]);
```

```
        print("</a>");
        print("</td>");
        print("<td>");
        print($datensatz[2]);
        print("</td>");
        print("<td>");
        print($datensatz[4]);
        print(" um ");
        print($datensatz[5]);
        print("</td>");
        print("</tr>");
        if($datensatz[1]>0) {
           antwort_holen($datensatz[1]);
        }
    }
    function antwort_holen($id) {
    $anf="SELECT * FROM forum WHERE beitrags_id='";
    $anf.=$id;
    $anf.="'";
    $er=mysql_query($anf);
    $z=mysql_fetch_row($er);
    ausgabe($z);
}
?>
<br><br>
<a href="new.html">Einen neuen Beitrag hinzuf&uuml;gen</a>
</body></html>
```

Der Anfang des Codes sieht noch ähnlich wie der Code des letzten Abschnitts aus. Die *for*-Schleife ist verändert, denn die Ausgabe eines Beitrags befindet sich in einer Funktion:

```
for($a=$anz-1;$a>-1;$a--) {
   mysql_data_seek($ergebnis,$a);
   $zeile=mysql_fetch_row($ergebnis);
   $antw=strstr($zeile[6],"Re:");
   if($antw==false) {
      ausgabe($zeile);
   }
}
```


Neu in der *for*-Schleife ist der Aufruf der Funktion *strstr()*. Hier prüfen wir, ob in der Betreff-Zeile die Zeichenkette *Re:* enthalten ist. Damit stellen wir fest, ob der gerade ausgelesene Beitrag eine Antwort ist oder nicht. Wir prüfen in der *if*-Abfrage auf den Rückgabewert. Ist dieser *false*, wurde die Zeichenkette nicht gefunden, und es handelt sich um einen Ursprungsbeitrag. Damit können wir den Forumsbeitrag ausgeben, in dem wir die selbst geschriebene Funktion *ausgabe()* aufrufen. An die Funktion übergeben wir den Parameter *$zeile*, d. h. den aktuellen Datensatz.
Handelt es sich bei dem ausgelesenen Datensatz um eine Antwort, befindet sich in der Betreff-Zeile die Zeichenkette *Re:*. Damit darf der Beitrag nicht an dieser Stelle ausgeben werden. Die Ausgabe einer Antwort erfolgt am Ende der Funktion *ausgabe()*.
Die Funktion *ausgabe()* enthält jetzt im Prinzip alles das, was sich im letzten Abschnitt in der *for*-Schleife befand. Hier wird der konkrete Forumsbeitrag als HTML-Tabellenzeile ausgegeben. Bei der Ausgabe der Betreff-Zeile müssen wir jedoch noch einmal darauf prüfen, ob der Beitrag, der ausgebenen wird, ein Ursprungsbeitrag oder eine Antwort ist:

```
$antw=strstr($datensatz[6],"Re:");
if($antw==false) {
   print(" ");
}
else {
   print("    ");
   print("    ");
}
```

An dieser Stelle wird nämlich eine Antwort nach rechts eingerückt. Dazu prüfen wir wie vorher über *strpos()*, ob die Zeichenkette *Re:* in der Betreff-Zeile vorhanden ist. Ist dies nicht der Fall, liegt ein Ursprungsbeitrag vor, und wir geben ein geschütztes Leerzeichen (* *) aus. Liegt jedoch eine Antwort vor, wird der *else*-Block ausgeführt und werden dort mehrere geschützte Leerzeichen ausgegeben. Dadurch wird die Betreff-Zeile eingerückt.
Der weitere Code der Funktion *ausgabe()* ist bekannt. Erst am Ende der Funktion *ausgabe()* müssen wir jetzt feststellen, ob zu dem soeben ausgegebenen Beitrag eine Antwort vorliegt:

```
if($datensatz[1]>0) {
   antwort_holen($datensatz[1]);
}
```

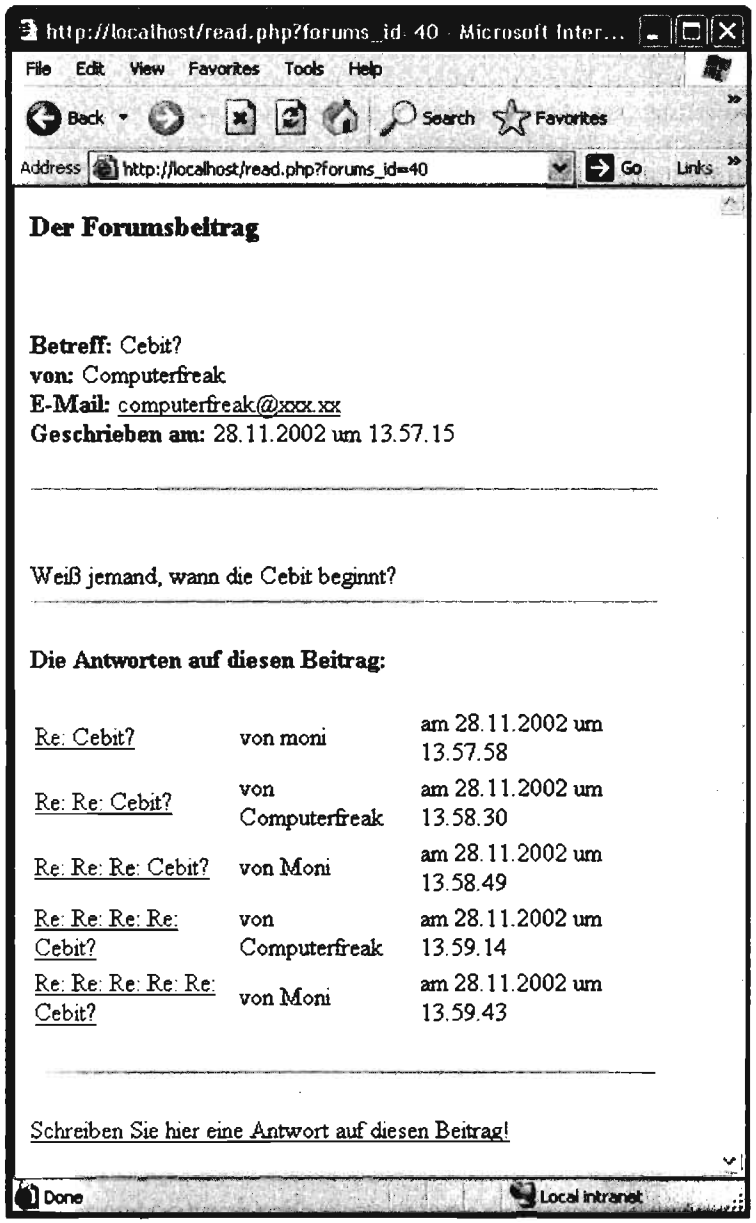
Dazu lesen wir den Inhalt in der zweiten Spalte aus. Liegt eine Antwort auf den Beitrag vor, muss hier der Primärschlüssel eingetragen sein. Ist der Wert 0, liegt keine Antwort vor. Das bedeutet, dass wir nur auf einen Wert größer 0 prüfen müssen. Ist dies der Fall, liegt eine Antwort vor, und wir rufen die Funktion *antwort_holen()* auf. An die Funktion übergeben wir noch einen Parameter, und zwar die Zahl des Primäschlüssels aus der zweiten Spalte. Dies ist ja die *beitrags_id* der Antwort. Die Funktion *antwort_holen()* liest jetzt die Antwort aus der Datenbank aus:

```
function antwort_holen($id) {
   $anf="SELECT * FROM forum WHERE beitrags_id='";
   $anf.=$id;
   $anf.="'";
   $er=mysql_query($anf);
   $z=mysql_fetch_row($er);
   ausgabe($z);
}
```

Die Funktion macht nichts anderes, als eine weitere Datenbankabfrage durchzuführen – und zwar ermittelt sie über den *SELECT*-Befehl die Antwort. Dazu wird als Suchkriterium die übergebene Zahl des Primärschlüssels benutzt, die ja in der Spalte *beitrags_id* angegeben ist. In *$z* liegt das Ergebnis vor. Es kann sich nur um einen Datensatz handeln. Wir müssen jetzt diesen Datensatz nur ausgeben. Dazu rufen wir wieder die Funktion *ausgabe()* auf und übergeben *$z* an die Funktion. Am Ende der Funktion *ausgabe()* wird geprüft, ob eine Antwort auf diese Antwort vorliegt. Ist dies der Fall, wird *antwort_holen()* erneut aufgerufen. Liegt keine Antwort vor, wird der nächste Beitrag ausgelesen und ausgegeben.

11.3.3 Die Antworten auf einen Beitrag auflisten

Auf der Basis des Codes des letzten Abschnitts können wir beim Lesen eines Forumsbeitrags auch die einzelnen Antworten auflisten. Dies hat den Vorteil, dass der Internet-Surfer gleich die weiteren Antworten le-



sen kann, indem er diese anklickt (siehe Abbildung S. 337). Er braucht nicht mehr auf den Forumsüberblick zurückzukehren.
Unterhalb des Beitrags werden wie im Forumsüberblick die Antworten aufgelistet. Um dies umzusetzen, müssen wir nur die Datei *read.php* überarbeiten. Das Prinzip ist das gleiche wie bei dem Forumsüberblick in *index.php*. Wir müssen am Ende des angezeigten kompletten Beitrags prüfen, ob eine Antwort vorliegt, diese dann auslesen und dort erneut feststellen, ob eine Antwort auf die Antwort vorliegt. Auch hier müssen wir rekursiv vorgehen. Dazu können wir die Funktionen *antwort_holen()* und *ausgabe()* aus dem Forumsüberblick fast ohne Änderung weiterverwenden. Die Datei *read.php* sieht so aus:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><body>
<h3>Der Forumsbeitrag</h3>
<div style="width: 400px">
<?php
   $id=$HTTP_GET_VARS['forums_id'];
   $db=mysql_connect("localhost","root","");
   mysql_select_db("manitu");
   $anfrage="SELECT * FROM forum WHERE beitrags_id
             LIKE '";
   $anfrage.=$id;
   $anfrage.="'";
   $ergebnis=mysql_query($anfrage);
   $zeile=mysql_fetch_row($ergebnis);
   print("<br><br><b>Betreff: </b>");
   print($zeile[6]);
   print("<br><b>von: </b>");
   print($zeile[2]);
   print("<br><b>E-Mail: </b>");
   print("<a href='mailto:");
   print($zeile[3]);
   print("'>");
   print($zeile[3]);
   print("</a><br><b>Geschrieben am: </b>");
   print($zeile[4]);
   print(" um ");
   print($zeile[5]);
```



```
   print("<br><br><hr><br><br>");
   print($zeile[7]);
   if($zeile[1]>0) {
      print("<hr>");
      print("<br><b>Die Antworten auf diesen
                Beitrag:</b><br>");
      print("<table>");
      antwort_holen($zeile[1]);
      print("</table>");
   }
   mysql_close($db);
   print("<br><hr><br>");
   print("<a href='replay.php?forums_id=");
   print($id);
   print("'>Schreiben Sie hier eine Antwort auf
            diesen Beitrag!</a>");
   function antwort_holen($id) {
      $anf="SELECT * FROM forum WHERE
               beitrags_id='";
      $anf.=$id;
      $anf.="'";
      $er=mysql_query($anf);
      $z=mysql_fetch_row($er);
      ausgabe($z);
   }
   function ausgabe($datensatz) {
      print("<tr align='left'>");
      print("<td>");
      print("<a href='read.php?forums_id=");
      print($datensatz[0]);
      print("'>");
      print($datensatz[6]);
      print("</a>");
      print("</td>");
      print("<td> von ");
      print($datensatz[2]);
      print("</td>");
      print("<td> am ");
      print($datensatz[4]);
```

```
        print(" um ");
        print($datensatz[5]);
        print("</td>");
        print("</tr>");
        if($datensatz[1]>0) {
            antwort_holen($datensatz[1]);
        }
    }
?>
</div></body></html>
```

Bevor die Verbindung zur Datenbank beendet wird, prüfen wir, ob auf den angezeigten Beitrag eine Antwort vorliegt. Dazu lesen wir wieder die zweite Spalte des aktuell angezeigten Datensatzes aus:

```
if($zeile[1]>0) {
   print("<hr>");
   print("<br><b>Die Antworten auf diesen
            Beitrag:</b><br>");
   print("<table>");
   antwort_holen($zeile[1]);
   print("</table>");
}
```

Die Antworten werden selbst wieder in einer kleinen HTML-Tabelle ausgegeben. Die Antwort wird über die Funktion *antwort_holen()* ermittelt. Dazu übergeben wir an die Funktion die Zahl des Primärschlüssels der Antwort.
Die Funktion *antwort_holen()* ist identisch mit der Funktion des letzten Abschnitts. Hier haben wir keine Änderungen durchgeführt. Am Ende der Funktion wird die Funktion *ausgabe()* aufgerufen, indem die Antwort als Datensatz an *ausgabe()* übergeben wird. Die Funktion *ausgabe()* ist ebenfalls identisch mit dem letzten Code. Am Ende der Funktion *ausgabe()* prüfen wir, ob eine Antwort auf die Antwort vorliegt. Entsprechend wird wieder die Funktion *antwort_holen()* aufgerufen.

11.3.4 Verbesserungen

Das Diskussionsforum hat noch den Nachteil, dass bei dem Auflisten der Antworten auf die Zeichenkette *Re:* in der Betreffzeile geprüft wird.

Es tritt aber häufiger der Fall auf, dass zwar eine Antwort auf einen Beitrag geschrieben, die Betreff-Zeile aber geändert wird. Die Folge: Die Antwort mit geändertem Betreff würde nicht als Antwort unterhalb des Ursprungsbeitrags aufgelistet werden.
Dieses Problem lässt sich jetzt nicht so ohne weiteres umgehen. Auf den Inhalt der Spalte *bezugs_id* können wir nicht prüfen. Denn Beiträge ohne Antwort enthalten dort immer eine 0 – dies gilt gleichermaßen für Ursprungsbeiträge ohne Antworten und Antworten, auf die wiederum keine Antworten geschrieben wurden. Wie wollen wir das voneinander differenzieren? Die Lösung: Wir verwenden eine weitere Spalte, in der wir eintragen, ob es sich um eine Antwort handelt oder nicht. Es geht nicht darum, ob ein Beitrag eine Antwort hat, sondern ob der Beitrag (egal ob es ein Ursprungsbeitrag oder ein Antwort ist) selbst eine Antwort ist oder nicht.
Zu diesem Zweck müssen wir unsere Tabelle *forum* um eine Spalte erweitern. Dazu verwenden wir wieder den SQL-Befehl *ALTER TABLE*:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><body>
<p>Spalte hinzuf&uuml;gen</p>
<?
   $db=mysql_connect("localhost","root","");
   mysql_select_db("manitu");
   $anfrage="ALTER TABLE forum ADD antwort
             VARCHAR(10)";
   mysql_query($anfrage);
   mysql_close($db);
?>
</body></html>
```

Die neue Spalte *antwort* wird über *ADD* hinzugefügt. Der Spaltentyp von *antwort* ist *VARCHAR*. Wir tragen je nach Inhalt des Beitrags entweder *true* (es handelt sich um eine Antwort) oder *false* (es ist ein Ursprungsbeitrag) ein. Wir müssen jetzt unsere Codes weiter überarbeiten, denn wir müssen ja den Wert *true* oder *false* bei den Datenbankzugriffen hinzufügen. Wird ein komplett neuer Beitrag geschrieben, muss der Beitrag den Wert *false* haben. Wir müssen also die Datei *new.php* erweitern und am Ende des *INSERT-INTO*-Befehls explizit *false* hinzuschreiben:

```
$anfrage="INSERT INTO forum VALUES ('";
$anfrage.="0', '0', '";
$anfrage.=$user;
$anfrage.="', '";
$anfrage.=$mail;
$anfrage.="', '";
$anfrage.=$datum;
$anfrage.="', '";
$anfrage.=$zeit;
$anfrage.="', '";
$anfrage.=$betreff;
$anfrage.="', '";
$anfrage.=$forumsbeitrag;
$anfrage.="','false')";
```

Wird eine Antwort geschrieben, müssen wir in *replay_entry.php* das Gleiche durchführen, jedoch muss hier dann *true* hineingeschrieben werden:

```
$anfrage="INSERT INTO forum VALUES ('0','0','";
$anfrage.=$user;
$anfrage.="', '";
$anfrage.=$mail;
$anfrage.="', '";
$anfrage.=$datum;
$anfrage.="', '";
$anfrage.=$zeit;
$anfrage.="', '";
$anfrage.=$betreff;
$anfrage.="', '";
$anfrage.=$forumsbeitrag;
$anfrage.="','true')";
```

Im nächsten Schritt müssen wir noch den Forumsüberblick in *index.php* überarbeiten. Wir müssen explizit auf *true* oder *false* prüfen. In der ersten *for*-Schleife müssen wir die *if*-Abfrage

```
$antw=strstr($zeile[6],"Re:");
if($antw==false) {
```



```
   ausgabe ($zeile);
}
```

durch

```
if($zeile[8]=="false") {
   ausgabe($zeile);
}
```

ersetzen. In der Funktion *ausgabe()* ist ebenfalls die erste *if-else*-Abfrage duch diese Abfrage zu ersetzen:

```
if($datensatz[8]=="false") {
   printf(" ");
}
else {
   printf("    ");
   printf("    ");
}
```

Führen Sie diese Änderungen durch und sehen Sie sich das Forum an, ergibt sich bei dem Anzeigen im Browser keine Änderung gegenüber der letzten Fassung. Der einzige Unterschied besteht jetzt darin, dass Sie die Betreff-Zeile ändern können, wenn Sie eine Antwort schreiben und diese auch als Antwort angesehen wird.

11.3.5 Erweiterungen des Forums

Das Diskussionsforum lässt sich jetzt noch um weitere Elemente bereichern. Denkbar ist zum Beispiel, dass über eine Schaltfläche immer zur nächsten Antwort auf den Beitrag geschaltet werden kann (oder auch zur vorhergehenden Antwort). Denkbar ist auch bei dem Ursprungsbeitrag die Anzahl der Antworten anzuzeigen. Dazu muss nur eine globale Variable bei der Ermittlung der Antworten hochgezählt werden, die dann ausgegeben wird. Ist ein bestimmter Wert erreicht oder sogar überschritten (z. B. 10 Antworten), wird eine Grafik angezeigt, die darauf hinweist, das es sich hier um ein «heißes» Thema handelt, zu dem es viele Antworten gibt. Eine zusätzliche Erweiterung kann darin bestehen, dass der Internet-Surfer, der einen Beitrag geschrieben hat und der

eine Antwort im Forum erhält, per E-Mail über die Antwort benachrichtigt wird. Eine Erweiterung könnte auch darin bestehen, dass das Forum nur registrierten Benutzern (mit Benutzername und Passwort) zugänglich ist.

Foren lassen sich in vielen Bereichen weiter verbessern und verschönern. Denken Sie sich neue Optionen aus und sehen Sie sich verschiedene Foren einmal an, wie diese gestaltet sind. Holen Sie sich bei diesen Diskussionsforen Anregungen:

http://board.webxsite.de/
http://www.woltlab.info/de/forum/
http://www.chip.de/community_events/chip-foren/uebersicht.html
http://www.dasboard.de
http://www.parsimony.de/

Es gibt auch viele vorbereitete PHP-Codes für Diskussionsforen. Sehen Sie sich diese Foren auch einmal an:

http://www.webmart.de/
http://www.tforum.de
http://www.blue-universe.de/blueboard.php
http://www.cynox.ch/cyphor/
http://phorum.org/
http://www.b1g-online.de/

11.4 Zusammenfassung

- MySQL kann einen so genannten Primärschlüssel erstellen und verwenden. Dies ist eine eindeutige Zahl, die in der entsprechenden Spalte nur einmal vorkommen kann. Über den Primärschlüssel kann ein Datensatz eindeutig identifiziert werden.
- Bei neuen Einträgen in die Datenbank wird der Primärschlüssel von der Datenbank automatisch aktualisiert.
- Die *GET*-Methode zum Übergeben von Variablen und deren Werte kann manuell über das Fragezeichen *(?)* eingesetzt werden, indem das Fragezeichen sowie die Variable und deren Wert an die URL angehängt wird.
- Die Funktion *mysql_data_seek()* bewegt den Zeiger nach einer Datenbankanfrage an eine bestimmte Position, um dann über *mysql_fetch_row()* den Datensatz an dieser Position zu ermitteln.