11.2 Ganz einfach: Textdatenbanken

PHP bietet mehrere Möglichkeiten, mit einfachen Textdatenbanken umzugehen. Im Gegensatz zu einem »richtigen« Datenbankmanagementsystem muss hier in der Regel nichts installiert werden. Freilich werden Sie ein entsprechendes Modul in PHP benötigen, was je nach Provider und Konfiguration unterschiedlich ausfällt. In vielen Fällen wird GDBM angeboten, was schon einiges an Leistungsfähigkeit bietet.

11.2.1 GDBM: Minidatenbank inklusive

Darauf bezieht sich die folgende Beschreibung. Sie können Ihr System daraufhin überprüfen, indem Sie die entsprechende Sektion in der Ausgabe der Funktion `phpinfo` konsultieren.

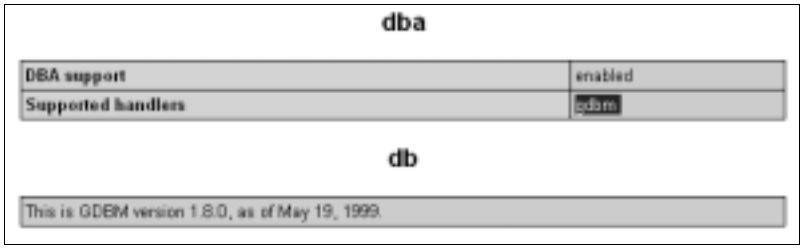

Abbildung 11.1: Ausgabe, wenn GDBM verfügbar ist

Finden Sie dagegen die in der folgenden Abbildung gezeigte Ausgabe, stehen nur die etwas primitiveren DBM-Funktionen zur Verfügung. In diesem Fall gibt es einige weitere Einschränkungen in der Verwendbarkeit.

Abbildung 11.2: Pech : hier ist GDBM nicht installiert

GDBM ist der GNU-Datenbankmanager, der als Open Source vorliegt und von der folgenden Adresse beschafft werden kann, wenn Sie sich für den Quellcode (in C) interessieren:

```
http://www.gnu.org/software/gdbm/gdbm.html
```

11.2.2 Daten mit GDBM speichern

Das erste Beispiel umfasst alle Funktionen, die für die Datenspeicherung notwendig sind. Wenn Sie es problemlos ausführen können, lassen sich alle anderen Skripte auch nutzen. Wenn es nicht

funktioniert, beispielsweise weil die Module nicht installiert sind, versuchen Sie es mit den `dbm`-Funktionen.

Listing 11.1: Test für gdbm-Daten (dba-Funktionen)

```
<?php
$id = @dba_open ("data/test.db", "n", "gdbm");
if (!$id)
{
   echo "Datenbank konnte nicht angelegt werden<br>";
} else {
   echo 'Datenbank "gdbm" erfolgreich geöffnet<br>";
   dba_replace ("dat", "Meine ersten Daten", $id);
   if (dba_exists ("dat", $id))
   {
      $value = dba_fetch ("dat", $id);
      dba_delete ("dat", $id);
   }
   echo $value;
   dba_close ($id);
}
?>
```

Jede Operation mit Datenbanken verlangt einen bestimmten Ablauf beim Zugriff auf die Funktionen. Dies wiederholt sich auch später bei MySQL und anderen DBMS:

- Öffnen der Verbindung zur Datenbank

  Diese Operation gibt ein Handle auf die Verbindung zurück, mit der anderen Funktionen der Zugriff ermöglicht werden kann. Schlägt der Verbindungsversuch fehl, wird `FALSE` zurückgegeben.

- Operation mit der Datenbank

  Alle Funktionen benötigen eine Verbindung zur Datenbank, die über ein Verbindungshandle zur Verfügung gestellt wird.

- Schließen der Verbindung

  Das Handle wird gelöscht und der Datenbank wird mitgeteilt, dass die Verbindung nicht mehr besteht.

Im Skript aus Listing 11.1 ist die folgende Zeile für das Öffnen der Verbindung zuständig:

```
$id = @dba_open ("data/test.db", "n", "gdbm");
```

Die Funktion `dba_open` verlangt drei Parameter: Zuerst der Name der Datei, in der die Daten physisch abgelegt werden. In diesem Fall wird die Datei »test.db« in einem Unterverzeichnis mit dem Namen »data« abgelegt. Das Verzeichnis muss existieren und es müssen uneingeschränkte Schreibrechte darauf vorhanden sein.

Der zweite Parameter "n" führt dazu, dass die Datei neu angelegt und zum Schreiben geöffnet wird. Andere Optionen sind: "r" für Lesezugriff, "w" für Lese- und Schreibzugriff und "c" für Lese- und Schreibzugriff. Der dritte Parameter ist der Name des Datenbankhandlers – also des tatsächlich angesprochenen Datenbanksystems. Welche Module unterstützt werden, zeigt phpinfo an (siehe Abbildung 11.1). Auf dem Server, der auch die Skripte für dieses Buch enthält, wird GDBM unterstützt.

War die Verbindung erfolgreich, können nun Daten gespeichert werden. Einfache Datenbanken erlauben die Abspeicherung von Werte-/Schlüsselpaaren:

```
dba_replace ("dat", "Meine ersten Daten", $id);
```

Jetzt ist in der Datenbank der Wert »Meine ersten Daten« unter dem Schlüssel »dat« gespeichert. Beachten Sie die Variable *$id*, mit der auf die Datenbankverbindung zugegriffen wird.

Mit der folgenden Zeile wird geprüft, ob zu einem spezifischen Schlüssel Daten existieren:

```
   if (dba_exists ("dat", $id)) {
```

Ist das der Fall, können Sie die Angaben wieder zurückholen:

```
    $value = dba_fetch ("dat", $id);
```

Falls erforderlich, kann man so natürlich die Daten auch wieder löschen. Auch wenn die Datei leer ist, wird sie nicht entfernt:

```
    dba_delete ("dat", $id);
```

Sind alle Operationen abgeschlossen, sollten Sie die Verbindung selbst wieder schließen.

```
   dba_close ($id);
```

Abbildung 11.3: Ausgabe des Skripts aus Listing 11.1

Schließen der Verbindung oder nicht - ausnahmsweise keine philosophische Frage

PHP schließt zwar alle Verbindungen am Ende des Skripts automatisch, wenn das Skript aber abstürzt oder abgebrochen wird, könnten offene Verbindungen zurückbleiben und die Systemleistung reduzieren. Denn bei Textdateien kann PHP nicht satzweise Daten sperren. Um Konflikte mit anderen Benutzern zu vermeiden, muss die gesamte Datei gesperrt werden. Andere, parallel ablaufende Skripte warten also beim Aufruf von `dba_open`, bis die Sperre wieder aufgehoben wurde. Laufen andere Skripte länger, ohne das noch Zugriffe erfolgen, ergeben sich so unnütze Verzögerungen.

Dem ersten Test sollte der praktische Einsatz folgen. Im nächsten Abschnitt wird ein kleiner Buchladen vorgestellt, basierend auf dem Amazon-Partnerprogramm.

Sie müssen erst Mitglied im Partnerprogramm werden und einen entsprechenden Vertrag mit Amazon schließen, der Ihnen die Verwendung der Bilder und Links erlaubt. Mehr Informationen dazu finden Sie unter *http://www.amazon.de/partner*.

11.2.3 Verkaufshilfe mit dba-Funktionen

Maximieren Sie Ihre Nebeneinnahmen

Jeder Besitzer einer halbwegs attraktiven Webseite kann ein paar Mark nebenbei einnehmen, wenn er Bücher anbietet und Mitglied im Amazon-Partnerprogramm wird. Pro direkt verlinktem und verkauftem Buch gibt es immerhin bis zu 15% des Nettoverkaufspreises. Davon wird man zwar mit Sicherheit nicht reich, ein paar Skripte lohnt es aber dafür anzulegen.

Die folgende kleine Applikation nutzt die DBA-Funktionen, um die so einfach wie möglich zu bewerkstelligen. Der Benutzer der Seite kann dabei aus einer Anzahl vorbereiteter Suchanfragen auswählen. Der Trick besteht letztlich darin, dass die Ergebnisse so aufbereitet werden, dass die Links zu der maximal möglichen Provision führen.

Nachteile vermeiden

Nachteil einer statischen Verlinkung ist die mangelnde Resistenz gegen Änderungen der Site bei Amazon. Wenn der Partnerlink geändert werden muss, kann das bei einigen Dutzend Büchern mühevoll sein. Mit der gezeigten Lösung werden die Links dynamisch erzeugt – nur an einer einzigen Stelle ist eine Änderung nötig.

Erfassung der Daten

Die Daten über die gesuchten Bücher werden unter Nutzung der Suchmaschine der Amazon-Seite erfasst. Die Techniken wurden als Anregung bereits in Abschnitt 8.3 *Verbindungen zu Servern* ab Seite 182 behandelt. Außerdem kommen reguläre Ausdrücke zum Einsatz, beschrieben in Abschnitt 13.3 *Suchexperten: Reguläre Ausdrücke* ab Seite 327. Nach dem Vorliegen einer passenden Liste suchen Sie sich die besten Bücher heraus und geben die Daten in ein Formular ein. Das Prinzip eignet sich auch für jede andere Art der Datenerfassung.

Listing 11.2: Erfassung der Titeldaten

```
<html>
<head>
   <title>Erfassung der Buchtitel</title>
</head>

<body>
<?php
$id = dba_open('data/amazon.db', 'c', 'gdbm')
      or die('Datenbank konnte nicht angelegt
              werden</body></html>');
if ((strlen($new) > 1) and (strlen($isbn) == 10))
{
   $arrData['isbn'] = $isbn;
   $arrData['titel'] = $titel;
   $arrData['autor'] = $autor;
   $arrData['preis'] = $preis;
   $strData = serialize($arrData);
   dba_replace ($isbn, $strData, $id);
} elseif (strlen($delete) > 1) {
   if (is_array($delisbn))
   {
      foreach($delisbn as $disbn)
     {
        dba_delete($disbn, $id);
     }
   }
}
?>
<h2>Erfassung neuer Titel</h2>
Bereits vorhandene Titel werden überschrieben, wenn
die ISBN identisch ist.
<form method="post" action="<?php echo $PHP_SELF; ?>">
<table border="0" cellpadding="2">
   <tr>
      <td>ISBN</td>
      <td><input type="text" size="10"
                 name="isbn"></td>
   <tr>
```

```
   </tr>
      <td>Titel</td>
      <td><input type="text" size="40"
                 name="titel"></td>
   <tr>
   </tr>
      <td>Autor</td>
      <td><input type="text" size="30"
                 name="autor"></td>
   <tr>
   </tr>
      <td>Preis</td>
      <td><input type="text" size="10"
                 name="preis"></td>
   </tr>
   <tr>
      <td></td>
      <td>
         <input type="submit" value="Eingeben"
                name="new">
      </td>
   </tr>
</table>
</form>
<h2>Aktuelle Liste</h2>
Markiere Titel, die zu l&ouml;schen sind.
<form method="post" action="<?php echo $PHP_SELF ?>">
<table>
<tr bgcolor="Silver">
   <th><input type="Submit"
              value="Markierte Titel löschen"
              name="delete">
   </th>
   <th>Titel</th>
   <th>ISBN</th>
   <th>Autor</th>
   <th>Preis</th>
</tr>
<?php
if ($id)
{
   $isbn = dba_firstkey($id);
   while ($isbn != '')
   {
     $strData = dba_fetch($isbn, $id);
     $arrData = unserialize($strData);
     $isbn = $arrData['isbn'];
     echo "<tr>\n";
     echo "<td><input type=\"checkbox\"
            name=\"delisbn[]\" value=\"$isbn\"></td>";
```

```
      echo '<td>' . $arrData['titel'] . '</td>';
      echo '<td>' . $arrData['isbn'] . '</td>';
      echo '<td>' . $arrData['autor'] . '</td>';
      echo '<td>' . $arrData['preis'] . '</td>';
      echo "</tr>\n";
      $isbn = dba_nextkey($id);
    }
    dba_close ($id);
}
?>
</form>
</table>
</body>
</html>
```

Das Skript besteht aus drei Teilen. Im ersten Teil wird die Datenbank geöffnet und eventuell vorhandene Formulardaten werden abgelegt. Da die GDBM-Datenbank nur einfache Kombinationen aus Schlüsseln und Werten speichern kann, zu einer ISBN aber mehrere Informationen gehören, wird hier auf einen kleinen Trick zurückgegriffen. Wann immer Sie Daten in Form von einfachen Datentypen speichern müssen – das sind Zahlen oder Zeichenketten – bietet sich die Serialisierung an. Das erledigt im Beispiel die Funktion `serialize`. Sie stellt das erfasste Array als Zeichenkette dar. Die Zeichenkette sieht dann folgendermaßen aus:

```
a:4:{s:4:"isbn";s:10:"3446215468";s:5:"titel";s:34:"PH
P 4 - Grundlagen und
Profiwissen";s:5:"autor";s:11:"Jörg
Krause";s:5:"preis";s:5:"99,80";}
```

Warum werden Daten serialisiert?

Um Plattform- und Systemunabhängig arbeiten zu können, muss man sich auf die Basisdatentypen, Zahlen und Zeichen, beschränken. Alle komplexeren Datentypen (dazu gehören Arrays, Objekte, Hashes und andere Konstrukte) setzen sich aus diesen Basistypen zusammen. Mit der Serialisierung – die für sich genommen wieder systemabhängig funktioniert, werden komplexe Daten in elementare überführt. Das Verfahren ist inzwischen auf der Basis von XML standardisiert worden und steht beispielsweise als WDDX auch in PHP zur Verfügung. Für das Beispiel bietet WDDX aber keine Vorteile und verkompliziert nur den Aufbau der Skripte .

Der Vorteil besteht in der leichten Weiterverarbeitung, denn beim erneuten Auslesen der Werte reicht der Aufruf der Funktion `unserialize`, um das leichter zu verarbeitende Array zu erhalten.

Im Skript besteht dieser Teil aus dem Abschnitt zum Aufbau des Arrays:

```
$arrData['isbn'] = $isbn;
$arrData['titel'] = $titel;
$arrData['autor'] = $autor;
$arrData['preis'] = $preis;
```

Nachfolgend wird das Array serialisiert:

```
$strData = serialize($arrData);
```

Anschließend wird die Kombination aus Schlüssel und serialisiertem Array wird gespeichert.

```
dba_replace ($isbn, $strData, $id);
```

Die ausgegebenen Werte können, zum einen zur Kontrolle, zum anderen zum Löschen genutzt werden. Die von diesem generierten Formular gesendeten Daten werden gleich zu Beginn ausgewertet.

Zuerst erfolgt eine Kontrolle, ob der Feldwert tatsächlich ein Array ist. Ohne Kontrolle könnte die `foreach`-Anweisung fehlschlagen.

```
if (is_array($delisbn))
{
```

Kontrollkästchen (`checkbox`), die unter demselben Namen in HTML programmiert werden und deren Namen mit dem Klammerpaar `[]` enden, erscheinen als Array. Entsprechend bietet sich `foreach` zur Auflösung an:

```
  foreach($delisbn as $disbn)
  {
```

Praktisch wird nun ein Aufruf mit der ISBN-Nummer in der Variablen *$disbn* für jedes aktivierte Kontrollkästchen ausgeführt. Das Löschen übernimmt dann die `dba_delete`-Funktion:

```
    dba_delete($disbn, $id);
```

Der mittlere Teil des Skripts ist lediglich ein Formular. Ebenso soll der Rest nicht betrachtet werden, da hier nur die nötigen HTML-Tags und schließende Klammern für PHP-Anweisungen zu finden sind.

Abbildung 11.4: Das Skript mit der Datenerfassung (Listing 11.2) in Aktion

Formatierte Ausgabe der Daten

Die so einfach erfassten Daten müssen nun in ansprechender Form ausgegeben werden. Die Darstellung ist überwiegend ein HTML-Problem. Wenn Sie tatsächlich Amazon ansprechen wollen, werden Sie mit dem Abschluss des Partnerprogramms auch die Links mitgeteilt bekommen, die eingesetzt werden müssen.

Direktlinks haben folgenden Aufbau:

```
http://www.amazon.de/exec/obidos/ASIN/<isbn>/<id>
```

Dabei muss *<isbn>* durch die ISBN des Titels und *<id>* durch den Partnercode ersetzt werden. Interessant ist natürlich immer auch die Anzeige eines Bildes. Dieses finden Sie als verkleinertes Abbild unter der folgenden Adresse:

```
http://images-
       eu.amazon.com/images/P/<isbn>.03.THUMBZZZ.jpg
```

Große Bilder werden dagegen folgendermaßen aufgerufen:

```
http://images-
       eu.amazon.com/images/P/<isbn>.03.LZZZZZZZ.jpg
```

Das folgende Skript zeigt alle gespeicherten Titel mit entsprechenden Links und den verkleinerten Bildern an.

Listing 11.3: Anzeige der Buchempfehlungen mit Link zu Amazon

```
<html>
<head>
   <title>Unsere Empfehlungen</title>
</head>

<body>
<h2>Unsere Top-Titel</h2>
Die folgenden Titel haben wir für Sie begutachtet und
in unsere Buchempfehlungen mit aufgenommen:
<table>
<tr bgcolor="Silver">
   <th></th>
   <th>Titel</th><th>ISBN</th><th>Autor</th>
   <th>Preis</th>
</tr>
<?php
$id = dba_open('data/amazon.db', 'r', 'gdbm')
      or die('Datenbank konnte nicht geöffnet
              werden</body></html>');
if ($id)
{
   $pid = 'activeserverpaOe';
   $imgserver = 'http://images-eu.amazon.com/images/P/
                 <ISBN>.03.THUMBZZZ.jpg';
   $amzserver =
'http://www.amazon.de/exec/obidos/ASIN/
                 <ISBN>/<ID>';
   $isbn = dba_firstkey($id);
   while ($isbn != '')
   {
      $strData = dba_fetch($isbn, $id);
      $arrData = unserialize($strData);
      $img = str_replace('<ISBN>', $arrData['isbn']
                                 , $imgserver);
      $lnk = str_replace('<ISBN>', $arrData['isbn']
                                 , $amzserver);
      $lnk = str_replace('<ID>', $pid, $lnk);
      echo "<tr>\n";
      echo "<td><a target=\"_blank\" href=\"$lnk\">
            <img src=\"$img\" border=0></a></td>";
      echo "<td><a target=\"_blank\" href=\"$lnk\">"
            . $arrData['titel'] . '</a></td>';
      echo '<td>' . $arrData['isbn'] . '</td>';
      echo '<td>' . $arrData['autor'] . '</td>';
      echo '<td>' . $arrData['preis'] . '</td>';
      echo "</tr>\n";
      $isbn = dba_nextkey($id);
```

```
    }
    dba_close ($id);
}
?>
</table>
</body>
</html>
```

Praktisch entspricht das Skript der Ausgabe der Titel am Ende des Erfassungsskripts. Nur anstatt der Kontrollkästchen werden die Bilder und Links angezeigt. Das Ergebnis zeigt den gewünschten Effekt. Der Ersatz der Platzhalter `<ISBN>` und `<ID>` erfolgt mit Hilfe von `str_replace`.

Das gezeigte Prinzip mit einfachen Schlüssel-/Wertepaaren ist nicht für größere Projekte geeignet. Wenn Sie mehr programmieren möchten, sollten Sie sich mit SQL beschäftigen. Für den praktischen Einsatz dieser Datenbankabfragesprache kommt dann das Datenbankmanagementsystem MySQL zum Einsatz.

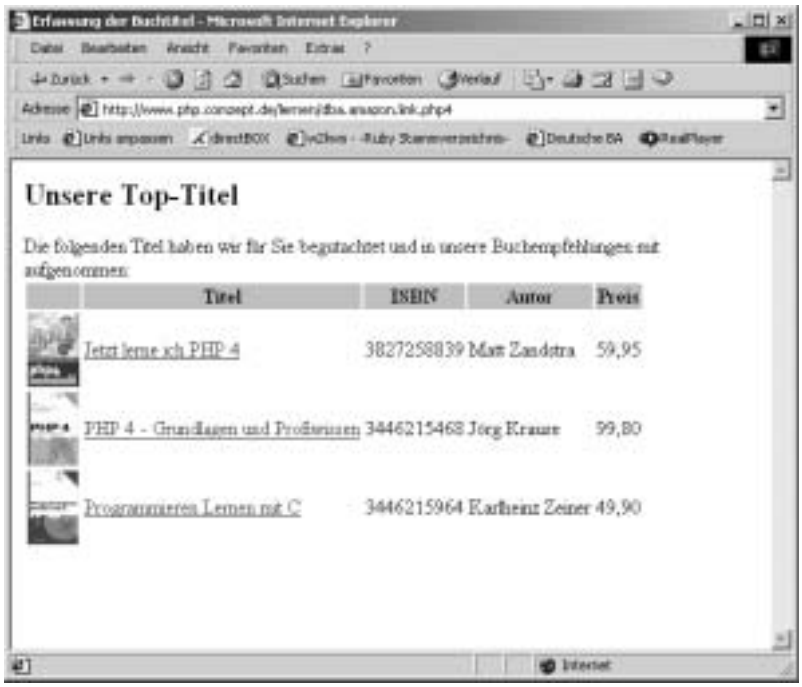

Abbildung 11.5: Bücherliste mit Partner-ID