PHP5.3

Fortgeschrittene Techniken der Web-Programmierung

Andreas Dittfurth

1. Ausgabe, April 2010

GPHP53F

HERDT

3 Verwaltung von MySQL-Datenbanken mit PHP

In diesem Kapitel erfahren Sie

- wie Sie Datenbanken mit PHP steuern und verwalten
- wie Sie Datensätze anzeigen, ändern, hinzufügen und löschen
- wie Ihre Anwender Daten individuell über Formulare verwalten
- Tipps und Tricks zur Bearbeitung von Daten

Voraussetzungen

- SQL-Grundkenntnisse

3.1 Steuerung über PHP-Skripte

Im letzten Kapitel haben Sie phpMyAdmin als administratives Werkzeug für MySQL-Datenbanken kennengelernt und eine umfangreiche Beispieltabelle in eine Datenbank importiert. Darüber hinaus konnten Sie SQL-Grundlagenkenntnisse erwerben bzw. auffrischen und praktisch in phpMyAdmin anwenden.

Der Schwerpunkt dieses Kapitels liegt in der Steuerung von MySQL-Datenbanken mithilfe von PHP. Sie haben eine funktionstüchtige Datenbank und benötigen keine weiteren Tools außer Ihren PHP-Skripten zur Bearbeitung und Verwaltung Ihrer Daten.

3.2 Verbindungsaufnahme mit MySQL

Die Verbindungsaufnahme von PHP zur Verwaltung einer MySQL-Datenbank erfolgt in zwei Schritten. Zuerst wird eine Verbindung zum MySQL-Server hergestellt, danach die gewünschte Datenbank ausgewählt. Voraussetzung sind entsprechende Zugriffsberechtigungen, die der Benutzer haben muss.

PHP mit dem MySQL-Server verbinden

Eine Verbindung zum MySQL-Server erfolgt direkt aus dem PHP-Skript. Folgende Funktionen werden zum Aufbau und Beenden einer Verbindung zu einem MySQL-Server verwendet:

`mysql_connect();`	Verbindung zum MySQL-Server aufnehmen
`mysql_close();`	Verbindung zum verbundenen MySQL-Server beenden

Syntax der Funktion `mysql_connect()`

```
mysql_connect ([Server[:Port][, Benutzername[, Passwort[, neue Verbindung]]]]);
```

- Zur Herstellung einer Verbindung geben Sie den Namen des gewünschten Datenbank-Servers an, auf den Sie zugreifen wollen. Optional ist die Angabe eines `Ports` zur Verbindung mit dem Server. Standardmäßig wird Port 3306 verwendet. Die Parameter `Benutzername` und das dazugehörige `Passwort` authentifizieren den Benutzer am Datenbank-Server. Der Parameter `neue Verbindung` kann die Werte `TRUE` oder `FALSE` annehmen und steuert, ob ein zweiter Aufruf von `mysql_connect()` mit identischen Parametern eine neue Verbindung aufbauen darf oder nicht.
- Alle Angaben sind optional. Werden keine Parameter angegeben, wird als Server `localhost` angenommen, als Benutzername der Name des Benutzers, dem der Server-Prozess gehört (Standard nach XAMPP-Installation ist *ODBC*), sowie ein leeres Passwort.

- Als Rückgabewert erhalten Sie eine Verbindungskennung, die Sie in einer Variablen zwischenspeichern können. Konnte keine Verbindung hergestellt werden, wird der Wert `FALSE` zurückgeliefert. Die Verbindungskennung (sogenanntes Handle oder Datenbankhandle) benötigen Sie vor allem dann, wenn Sie mehrere Datenbankverbindungen in einem PHP-Skript aufbauen. Weitere Datenbanksteuerungsbefehle müssen dann eindeutig den einzelnen Verbindungen zugewiesen werden.

Schlägt `mysql_connect()` fehl, wird nur eine Warnung ausgegeben, das Skript wird weiter ausgeführt. Dies ist häufig nicht gewünscht, da jeder weitere Befehl zur Steuerung von Datenbanken damit fehlschlägt. Es empfiehlt sich der Einsatz der Funktion `mysql_connect()` in Kombination mit `or die("Meldung")`.

```
mysql_connect()
     or die("Meldung");
```

Dieser `or`-Zweig wird dann ausgeführt, wenn die Datenbankverbindung fehlgeschlagen ist. Die Ausführung des Skripts wird nach der Ausgabe des Parameters `Meldung` abgebrochen.

```
Beispiel: mysql_connect("localhost", "root", "")
          or die("Verbindung konnte nicht hergestellt werden.");
```

Syntax der Funktion `mysql_close()`

```
mysql_close([Verbindungskennung]);
```

- Diese Funktion beendet eine bestehende Verbindung mit einem MySQL-Server. Wenn Sie bei mehreren geöffneten Verbindungen eine bestimmte Verbindung beenden möchten, geben Sie den optionalen Parameter `Verbindungskennung` an. Wenn Sie diesen Parameter nicht angeben, wird automatisch die zuletzt geöffnete Verbindung geschlossen.
- PHP schließt automatisch mit Beendigung des Skripts die Server-Verbindung. Sie sollten dennoch eine Verbindung immer selbst schließen, um genutzte Ressourcen sicher - und möglichst frühzeitig - freizugeben.

Die gewünschte Datenbank auswählen

Nach einer erfolgreichen Verbindung zum MySQL-Server bestimmen Sie mithilfe von `mysql_select_db()`, welche Datenbank genutzt werden soll.

mysql_select_db();	Auswahl der Datenbank

Syntax der Funktion `mysql_select_db()`

```
mysql_select_db(Datenbankname[, Verbindungskennung]);
```

- Die Auswahl der zu nutzenden Datenbank erfolgt über deren Namen als Parameterangabe `Datenbankname`.
- Bestehen über `mysql_connect()` mehrere Verbindungen zu einer oder mehreren Datenbanken, ist die entsprechende Verbindungskennung anzugeben. Die Angabe der Verbindungskennung ist nicht nötig, wenn Sie mit nur einer Datenbankverbindung arbeiten.
- Der Rückgabewert ist bei Erfolg `TRUE`, ansonsten `FALSE`.

Beispiel: *db.inc.php*

Die Verbindungsaufnahme zum lokalen MySQL-Server und die Auswahl der gewünschten Datenbank erfolgen in einem Schritt. Die Zugangsdaten sollen nicht in jedem Skript gespeichert werden, das auf die Datenbank zugreifen will. Es wird eine wiederverwendbare Datei erstellt, die in jedes beliebige Skript eingebunden werden kann. Durch Definition einer allgemeinen Funktion kann die Datei für den Zugriff mit beliebigen Zugangsdaten verwendet werden.

```
<?php
     //Funktion zur Verbindungsaufnahme mit einer MySQL-Datenbank
①    function verbindung_mysql($datenbank, $server = "localhost",
                               $user = "root", $pass = "")
          {
②              global $verbindung;
               //Verbindungsaufnahme MySQL-Server
③              $verbindung = @mysql_connect($server, $user, $pass)
                    or die("Keine Verbindung zum Server... <br>
                              Abbruch des Skripts.");
               //Wechsel in angegebene Datenbank
④              @mysql_select_db($datenbank)
                    or die("Fehler beim Zugriff auf die Datenbank. <br>
                              MySQL-Fehler: " .mysql_error() .".<br>
                              Abbruch des Skripts.");
          }
?>
```

① Es wird eine Funktion `verbindung_mysql()` definiert, die die Verbindungsaufnahme zur gewünschten Datenbank vornehmen soll. Der Parameter `$datenbank` ist bei jedem Aufruf anzugeben. Die weiteren Parameter `$server`, `$user` und `$pass` sind mit Vorgabewerten belegt, sodass sie in der Testumgebung dieses Buches weggelassen werden können. Sind andere Werte gewünscht, z. B. zur Arbeit mit Ihren Daten im Internet, ändern Sie die Werte der Parameter in der Funktion entsprechend ab oder geben die Daten beim Aufruf der Funktion an.

② In der Funktion wird eine Variable `$verbindung`, die die Verbindungskennung beinhaltet (siehe ③), als globale Variable deklariert. Die Variable steht damit auch außerhalb der Funktion zur Verfügung. Somit kann im aufrufenden Skript die Variable als Parameter der Funktion `mysql_close($verbindung)` zur Beendigung der Verbindung mit der Datenbank verwendet werden.

③ Die Verbindungskennung (Handle), die bei der erfolgreichen Verbindung zurückgeliefert wird, wird in der Variablen `$verbindung` gespeichert. Das Zeichen @ sorgt dafür, dass im Fehlerfall keine Meldung am Bildschirm erscheint. Sind die Anmeldedaten nicht korrekt bzw. ist der MySQL-Server nicht gestartet, kann keine Verbindung hergestellt werden. In diesem Fall wird der Rückgabewert `FALSE` geliefert, der den mit `or` eingeleiteten Zweig auslöst. Dort wird über die Funktion `die()` eine Fehlermeldung ausgegeben und das aktuelle Skript beendet.

④ Im letzten Schritt wird die gewünschte Datenbank ausgewählt. Auch hier erfolgt im Fehlerfall ein Abbruch des Skriptes mit einer Fehlermeldung.

Beispiel: *verbindung.php*

Die vorbereitete Datei *db.inc.php* wird in einem Skript verwendet. Danach ist im Erfolgsfall die Datenbank über PHP ansprechbar.

```
<?php
①    require_once("db.inc.php");
②    verbindung_mysql("bankleitzahlen");
     //weitere Befehle...
③    mysql_close($verbindung);
?>
```

① Die Datei *db.inc.php* wird eingebunden. Damit steht im Skript die dort definierte Funktion `verbindung_mysql()` zur Verfügung.

② Mithilfe der Funktion `verbindung_mysql()` erfolgt die Verbindungsaufnahme zum lokalen MySQL-Server. Dort wird die Datenbank *bankleitzahlen* ausgewählt. Möchten Sie mit denselben Anmeldedaten eine andere Datenbank auswählen, geben Sie den gewünschten Namen der Datenbank als Parameter an. Wollen Sie sich mit anderen Daten oder an einem anderen Server anmelden, verwenden Sie die Funktion mit allen vier definierten Parametern in der vorgesehenen Reihenfolge der Parameter: `$datenbank`, `$server`, `$user` und `$pass`.

③ Die Verbindung zur Datenbank wird mithilfe der Funktion `mysql_close()` beendet.

3.3 MySQL-Abfragen

Abfrage senden

Zur Abfrage der Tabellendaten einer bestimmten Datenbank sind die nachfolgenden Befehle notwendig.

mysql_query();	Senden einer MySQL-Anfrage zur aktiven Datenbankverbindung

Syntax der Funktion `mysql_query()`

```
mysql_query(SQL-Abfrage[, Verbindungskennung]);
```

- ✔ Eine Anfrage an MySQL realisieren Sie über die Funktion `mysql_query()`.
- ✔ Die entsprechende Verbindungskennung ist anzugeben, wenn mehrere Verbindungen zu einem MySQL-Server bestehen. Geben Sie diesen Parameter nicht an, wird die zuletzt geöffnete Verbindung genutzt.
- ✔ Als Rückgabewert erhalten Sie keine Informationen zur Anzahl der zurückgelieferten Daten, sondern nur, ob die Ausführung des Befehls erfolgreich war (`TRUE` z. B. beim `insert`-Befehl bzw. Ressourcenkennung bei `select`-Befehlen) oder nicht (`FALSE`). Die Rückgabe von `FALSE` erfolgt, wenn z. B. ein Fehler in der SQL-Syntax aufgetreten ist, eine angeforderte Tabelle nicht vorhanden ist oder der Zugriff auf eine Tabelle verwehrt wird.

Beispiel: *abfrage.php*

Am Beispiel der Tabelle `de` in der Datenbank `bankleitzahlen` wird eine Verbindung zur Datenbank aufgenommen und eine einfache SQL-Anweisung ausgeführt. Zur Veranschaulichung wird jede Aktion als Meldung im Browser ausgegeben.

```
<?php
①     require_once("db.inc.php");
      verbindung_mysql("bankleitzahlen");
②     $sql = "SELECT * FROM de WHERE ort = 'Berlin'";
③     $abfrage = mysql_query($sql);
④     if ($abfrage)
            {
                  echo "<p>Erfolg! " . mysql_num_rows($abfrage) ." Treffer.</p>";
            }
      else
            {
                  echo "<p>Die SQL-Anweisung ist fehlgeschlagen...</p>";
            }
⑤     mysql_close($verbindung);
?>
```

① Das Skript *db.inc.php* wird eingebunden. Die Funktion `verbindung_mysql()` steht zur Verfügung und wird für die Verbindungsaufnahme verwendet.

② Eine SQL-Abfrage zur Auswahl aller Banken aus Berlin wird in der Variablen `$sql` abgelegt.

③ Die Abfrage wird mithilfe der Funktion `mysql_query()` ausgeführt, der Rückgabewert in der Variablen `$abfrage` gespeichert. Im Erfolgsfall ist dies die Ressourcenkennung, ansonsten `FALSE`.

④ Je nachdem, ob die SQL-Anweisung erfolgreich ausgeführt werden konnte oder nicht, wird eine entsprechende Meldung angezeigt.

⑤ Zum Schluss wird die Verbindung zum MySQL-Server wieder geschlossen.

3.4 Rückgabe aus MySQL-Abfrage auswerten

Datensätze einer MySQL-Tabelle mithilfe von PHP anzeigen

Nachdem die Verbindung zur MySQL-Datenbank *bankleitzahlen* hergestellt wurde, können Sie z. B. die Anzahl der Datensätze oder alle Datensätze der Tabelle *de* anzeigen. Wenn Sie wissen möchten, wie viele Datensätze die SQL-Abfrage zurückliefert, verwenden Sie die Funktion `mysql_num_rows()`. Diese Funktion war bereits Bestandteil des letzten Beispiels.

Syntax der Funktion `mysql_num_rows()`

```
mysql_num_rows(Abfrageergebnis);
```

- ✔ Diese Anweisung liefert Ihnen die Anzahl der Datensätze, die mit dem SQL-Befehl `SELECT` an das Skript zurückgeliefert werden.
- ✔ Der Parameter `Abfrageergebnis` ist die Ergebniskennung der aktuellen SQL-Abfrage. Geben Sie an dieser Stelle die Funktion `mysql_query()` an, wird erst eine Abfrage ausgeführt und dann die Anzahl der Datensätze der Ergebnismenge ermittelt.

Beispiel:
```
$ergebnis = mysql_query("SELECT * FROM de");
echo mysql_num_rows($ergebnis);
```

Speichern der Datensätze als Feld mit der Funktion `mysql_fetch_array()`

Mit dieser Funktion erhalten Sie jeweils einen Datensatz einer SQL-Abfrage als Feld.

Syntax der Funktion `mysql_fetch_array()`

```
mysql_fetch_array(Abfrageergebnis[, Ergebnistyp]);
```

- ✔ Der anzugebende Parameter `Abfrageergebnis` stellt den Handle der aktuellen SQL-Abfrage dar.
- ✔ Mit der optionalen Angabe `Ergebnistyp` wandeln Sie die Daten des Ergebnisses in einen bestimmten Feldtyp um. Die möglichen Parameter sind `MYSQL_ASSOC`, `MYSQL_NUM` und `MYSQL_BOTH`. Hiermit bestimmen Sie, wie Sie die Elemente des Feldes ansprechen möchten.
  - ✔ Bei MYSQL_ASSOC verwenden Sie den Feldnamen als Schlüssel (z. B. $zeile["ort"]).
  - ✔ Bei MYSQL_NUM arbeiten Sie über die Angabe des numerischen Indexes (z. B. $zeile[3]).
  - ✔ Wenn Sie die Option MYSQL_BOTH verwenden, sind beide Varianten zum Ansprechen eines Elements möglich.
- ✔ Wenn Sie keinen `Ergebnistyp` angeben, wird standardmäßig die Option `MYSQL_BOTH` verwendet.

Beispiel: *auslesen.php*

In diesem Beispiel werden Datensätze aus einer Tabelle ausgelesen und geringfügig formatiert auch in Tabellenform angezeigt.

```
<?php
      require_once("db.inc.php");
      verbindung_mysql("bankleitzahlen");
①     $sql = "SELECT * FROM de WHERE ort = 'Rostock' AND bic <> '' ORDER BY plz";
      $abfrage = mysql_query($sql);
②     if(!$abfrage)
            {
                  echo "<p>Die SQL-Anweisung ist fehlgeschlagen...</p>";
            }
      else
            {
③                 $anzahl = mysql_num_rows($abfrage);
                  echo "<p>Die Abfrage hat $anzahl Datensätze gefunden:</p>";
④                 echo "<table width='100%' border='1'><tr><th>BLZ</th>
                        <th>Name</th><th>PLZ</th><th>Ort</th><th>BIC</th></tr>";
⑤                 while ($zeile = mysql_fetch_array($abfrage))
                        {
                              echo "<tr><td>" .$zeile["bankleitzahl"] ."</td><td>"
                                    .$zeile["bezeichnung"] ."</td><td>"
                                    .$zeile["plz"] ."</td><td>"
                                    .$zeile["ort"] ."</td><td>"
                                    .$zeile["bic"] ."</td></tr>";
                        }
                  echo "</table>";
            }
      mysql_close($verbindung);
?>
```

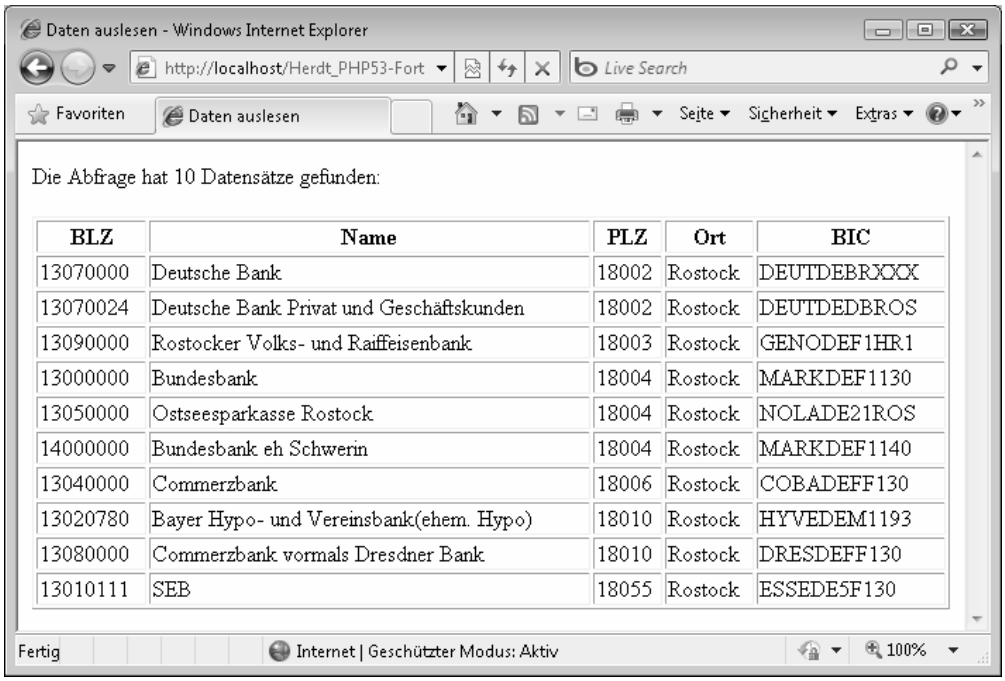
① Um die Anzahl der ausgewählten Datensätze für die Beispielausgabe gering zu halten, wird eine SQL-Anweisung formuliert, die nur alle Banken aus Rostock auswählt. Zusätzlich muss das Feld `bic` (BIC ist der sogenannte "bank identifier code") bei diesen Banken einen Wert enthalten. Es soll aufsteigend nach der eingetragenen Postleitzahl sortiert werden. Die Abfrage wird ausgeführt.

② Es wird nachgefragt, ob die Abfrage an die Datenbank fehlgeschlagen ist. In diesem Fall wird eine Meldung ausgegeben, im Erfolgsfall werden die Anweisungen des `else`-Zweigs ausgeführt.

③ Die Anzahl der Datensätze ermitteln Sie mit der Funktion `mysql_num_rows()`.

④ Die Daten sollen in Tabellenform ausgegeben werden. Am einfachsten ist es, den HTML-Tabellenkopf vor und den Abschluss der Tabelle nach der `while`-Schleife zu platzieren. In der Schleife geben Sie dann pro Datensatz eine komplette Tabellenzeile aus.

⑤ Mithilfe der Funktion `mysql_fetch_array()` ermitteln Sie ein Feld, das dem aktuellen Datensatz der Tabelle entspricht. Gleichzeitig wird durch diese Funktion der sogenannte Datensatzzeiger auf den nächsten Datensatz des Ergebnisses gesetzt. Durch die `while`-Schleife werden somit nacheinander alle Datensätze der Tabelle an ein Feld `($zeile)` übergeben. Die Schlüssel dieses Feldes sind die Feldnamen der MySQL-Tabelle. Wenn keine weiteren Datensätze vorliegen, wird `FALSE` zurückgegeben und damit die Schleife beendet.



BLZ	Name	PLZ	Ort	BIC
13070000	Deutsche Bank	18002	Rostock	DEUTDEBRXXX
13070024	Deutsche Bank Privat und Geschäftskunden	18002	Rostock	DEUTDEDBROS
13090000	Rostocker Volks- und Raiffeisenbank	18003	Rostock	GENODEF1HR1
13000000	Bundesbank	18004	Rostock	MARKDEF1130
13050000	Ostseesparkasse Rostock	18004	Rostock	NOLADE21ROS
14000000	Bundesbank eh Schwerin	18004	Rostock	MARKDEF1140
13040000	Commerzbank	18006	Rostock	COBADEFF130
13020780	Bayer Hypo- und Vereinsbank(ehem. Hypo)	18010	Rostock	HYVEDEM1193
13080000	Commerzbank vormals Dresdner Bank	18010	Rostock	DRESDEFF130
13010111	SEB	18055	Rostock	ESSEDE5F130

Anzeige der Beispieldatei "auslesen.php"