Chapter

12

Accessing MySQL Databases Using PHP

Introduction

In the previous chapter we considered how to create databases using MySQL. While this is useful, it does not enable us to utilize the information stored within the database through a Web page. In this chapter we shall examine how to access, change and delete information stored in a MySQL database by using the PHP language.

A Simple PHP Script to Display Database Records

We shall begin by creating a script to extract a single record from the clients table in the exampledb database created in the previous chapter. Consider the following PHP script:

```
<html>
  <head>
    <title>chapt12-1.php Reading Database</title>
  </head>
  <body>
    <?php
    $db = mysql_connect("localhost", "root", "");
    mysql_select_db("exampledb",$db);
    $result = mysql_query("SELECT * FROM clients",$db);
    echo("First Name: " .
mysql_result($result,0,"firstname") .
      "<br>\n");
    echo("Last Name: " . mysql_result($result,0,"lastname")
.
      "<br>\n");
    echo("User Id: " . mysql_result($result,0,"userid") .
"<br>\n");
    echo("Password: " . mysql_result($result,0,"password") .
      "<br>\n");
    ?>
  </body>
</html>
```

Let us examine this script more closely as it introduces some unfamiliar PHP instructions. The first line of importance is:

```
$db = mysql_connect("localhost", "root", "");
```

`mysql_connect()` is a function which opens a link to a MySQL server on the specified host computer. The specified host is "`localhost`", but if the MySQL server is not located on the same machine on which you are developing this would have the Web address of that computer. The second parameter is the username, in this case "`root`". The username must be specified if the database has been created with security restrictions. The final parameter, which is blank in this case, is where you would supply a password. A variable `$db` is assigned as pointing to this database connection.

The next line is:

```
mysql_select_db("exampledb",$db);
```

This is a function that selects the database you wish to access. Two parameters are supplied, the first provides the name of the database, in this case "`exampledb`" and the second the variable which points to where the database is located. In this example it is `$db`. This function allows us to have multiple connections to different databases. This can be achieved through the inclusion of multiple `mysql_select_db` functions.

The next line:

```
$result = mysql_query("SELECT * FROM clients",$db);
```

is an instruction to the MySQL database to extract some database records. The `mysql_query()` function has two parameters. The first "`SELECT * FROM clients`" is a SQL query to select all the records from the clients table. The second parameter is the name of the variable pointing to the database. The whole database record is stored in a variable called `$result`.

The final four lines:

```
echo("First Name: " . mysql_result($result,0,"firstname") .
"<br>\n");
echo("Last Name: " . mysql_result($result,0,"lastname") .
"<br>\n");
echo("User Id: " . mysql_result($result,0,"userid") .
"<br>\n");
echo("Password: " . mysql_result($result,0,"password") .
"<br>\n");
```

display the contents of each of the database record fields. The function used to display the record fields is mysql_result(). This function consists of three parameters. The first is the name of the variable where the records which have been returned by the mysql_query function are stored ($result). The second is a variable which indicates which record to display (0 refers to the first record). Finally, the third field contains the name of the database field to display.

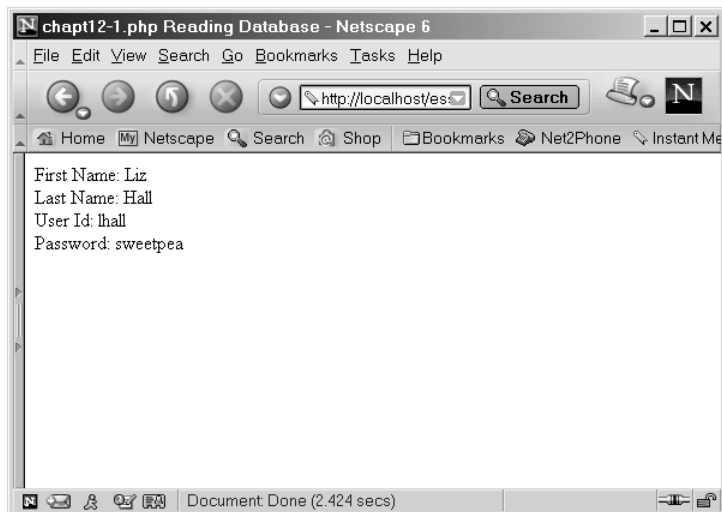The output produced from this script is illustrated in Figure 12.1.

Figure 12.1 A single database record.

Of course this is still not all that useful an example as we are only displaying one of the records. Lets create a new example that includes a loop which will allow us to display all the database records.

Accessing Multiple Records by Record Number

The following script illustrates a modified version of our previous example that will display all of the records in the database:

```
<html>
  <head>
    <title>chapt12-2.php Reading and Displaying Multiple
         Records</title>
  </head>
  <body>
    <?php
    $db = mysql_connect("localhost", "root", "");
    mysql_select_db("exampledb",$db);
    $result = mysql_query("SELECT * FROM clients",$db);
    while ($rec = mysql_fetch_row($result)) {
       echo("Record Number: " . $rec[0] . "<br>\n");
       echo("First Name: " . $rec[1] . "<br>\n");
       echo("Last Name: " . $rec[2] . "<br>\n");
       echo("UserId: " . $rec[3] . "<br>\n");
       echo("Password: " . $rec[4] . "<br><br>\n");
       }
    ?>
</body>
</html>
```

This script is similar to the previous example, but introduces a loop instruction:

```
while ($rec = mysql_fetch_row($result)) {
```

Notice, that within the loop is a new function, `mysql_fetch_row()`. This function returns a single record, stored in the `$result` variable. The record is stored in an array called `$rec`. The following `echo` statements display the contents of the record within the loop:

```
echo("Record Number: " . $rec[0] . "<br>\n");
echo("First Name: " . $rec[1] . "<br>\n");
```

```
echo("Last Name: " . $rec[2] . "<br>\n");
echo("UserId: " . $rec[3] . "<br>\n");
echo("Password: " . $rec[4] . "<br><br>\n");
```

Notice that the echo statements display each of the elements of the $rec[] array. The array contains the value of each of the record's fields. The array element [0] contains the record id, while [1] contains the client's first name etc.

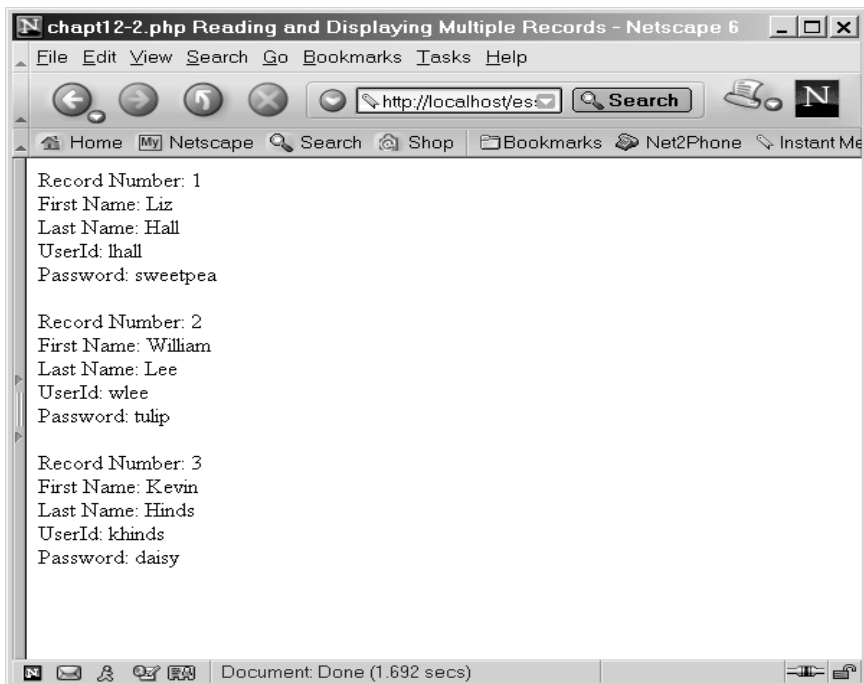The output produced by this script is illustrated in Figure 12.2.

Figure 12.2 Multiple database records.

Accessing Multiple Records by Column Name

In the previous example the mysql_fetch_row() function was used to access and display the database records. While this works okay, the person producing the PHP script has to

refer to the separate fields using array numbers. This does not make the code very easy to read and mistakes can be introduced. The following script provides an alternative way of accomplishing this:

```
<html>
  <head>
    <title>chapt12-3.php Reading and Displaying Multiple
Records
         2</title>
  </head>
  <body>
    <?php
    $db = mysql_connect("localhost", "root", "");
    mysql_select_db("exampledb",$db);
    $result = mysql_query("SELECT * FROM clients",$db);
    while ($rec = mysql_fetch_array($result)) {
      echo("Record Number: " . $rec["id"] . "<br>\n");
      echo("First Name: " . $rec["firstname"] . "<br>\n");
      echo("Last Name: " . $rec["lastname"] . "<br>\n");
      echo("UserId: " . $rec["userid"] . "<br>\n");
      echo("Password: " . $rec["password"] . "<br><br>\n");
      }
    ?>
  </body>
</html>
```

This script employs the `mysql_fetch_array()` function instead of using the `mysql_fetch_row()` function in the previous example. This is introduced in the line:

```
while ($rec = mysql_fetch_array($result)) {
```

This function does the same task as the `mysql_fetch_row()` function, except it allows us to refer to the database fields by their name. This can be seen in the lines:

```
echo("Record Number: " . $rec["id"] . "<br>\n");
echo("First Name: " . $rec["firstname"] . "<br>\n");
echo("Last Name: " . $rec["lastname"] . "<br>\n");
echo("UserId: " . $rec["userid"] . "<br>\n");
echo("Password: " . $rec["password"] . "<br><br>\n");
```

The variable array `$rec[]` is used to access the different data fields.

Checking for the Existence of Records

In our previous example we displayed all of the records returned from the database. But what happens if there are no records to display? Well in the case of our previous example the script works correctly, but nothing would appear. The problem is that the user is not informed that there are no records to display and so, from a user's point of view, this is bad design. We should really determine if there are no records to display and generate a message to this effect. The following script provides an example of doing exactly this:

```
<html>
  <head>
    <title>chapt12-4.php Checking Existence of
Records</title>
  </head>
  <body>
    <?php
    $db = mysql_connect("localhost", "root", "");
    mysql_select_db("exampledb",$db);
    $result = mysql_query("SELECT * FROM clients",$db);
    if (!$rec = mysql_fetch_array($result)) {
      echo ("Sorry, no records found");
      }
    else {
      do {
        echo("Record Number: " . $rec["id"] . "<br>\n");
        echo("First Name: " . $rec["firstname"] .
"<br>\n");
        echo("Last Name: " . $rec["lastname"] . "<br>\n");
        echo("UserId: " . $rec["userid"] . "<br>\n");
        echo("Password: " . $rec["password"] .
"<br><br>\n");
      }
      while ($rec = mysql_fetch_array($result));
    }
    ?>
</body>
</html>
```

This script is similar to the previous example, but includes the following if statement:

```
if (!$rec = mysql_fetch_array($result)) {
  echo ("Sorry, no records found");
  }
```

The if statement invokes the mysql_fetch_array() function. If no records are returned the function returns a value of 0 and a message is displayed informing the user of this. If records are found the else part of the if statement is invoked:

```
else {
  do {
    echo("Record Number: " . $rec["id"] . "<br>\n");
    echo("First Name: " . $rec["firstname"] . "<br>\n");
    echo("Last Name: " . $rec["lastname"] . "<br>\n");
    echo("UserId: " . $rec["userid"] . "<br>\n");
    echo("Password: " . $rec["password"] . "<br><br>\n");
  }
  while ($rec = mysql_fetch_array($result));
```

Within the else statement is a do while loop which displays the database records. A do while loop has been used because the previous if statement contained a call to the mysql_fetch_array() function. We do not want to invoke this function again until we have displayed the first database record that it retrieved. If we used a while loop the function would be invoked again and the first record returned from the database would not be displayed.

Unfortunately, the program doesn't display the message that no records were found because our database does contain records.

Searching for Specific Records

Instead of displaying all the records in a database we may wish to search for and display specific ones. In doing this we may search for a record which does not exist and so the need to display a message that no records were found becomes more important. The following script is a modification of our previous example:

```
<html>
  <head>
    <title>chapt12-5.php Search for particular
Records</title>
  </head>
  <body>
    <?php
    $db = mysql_connect("localhost", "root", "");
    mysql_select_db("exampledb",$db);
    $result = mysql_query("SELECT * FROM clients WHERE
id=2",$db);
    if (!$rec = mysql_fetch_array($result)) {
      echo ("Sorry, no records found");
      }
    else {
      do {
        echo("Record Number: " . $rec["id"] . "<br>\n");
        echo("First Name: " . $rec["firstname"] . "<br>\n");
        echo("Last Name: " . $rec["lastname"] . "<br>\n");
        echo("UserId: " . $rec["userid"] . "<br>\n");
        echo("Password: " . $rec["password"] . "<br><br>\n");
      }
      while ($rec = mysql_fetch_array($result));
    }
    ?>
</body>
</html>
```

The only change in this example from the previous one is:

```
$result = mysql_query("SELECT * FROM clients WHERE id=2",$db);
```

This line invokes the function `mysql_query()` passing the SQL command `SELECT * FROM clients WHERE id=2`. This command only searches for records which have an id of 2. As there is only one record with an id of 2, only one record is displayed.

If you change the value of id to 4 you should see the message "`Sorry, no records found`".

The output from this script is illustrated in Figure 12.3.

Of course, this is not all that useful at the moment, as we need to modify the script each time we wish to find a different record. What we need is a way of interacting with the script to select which records to search for.
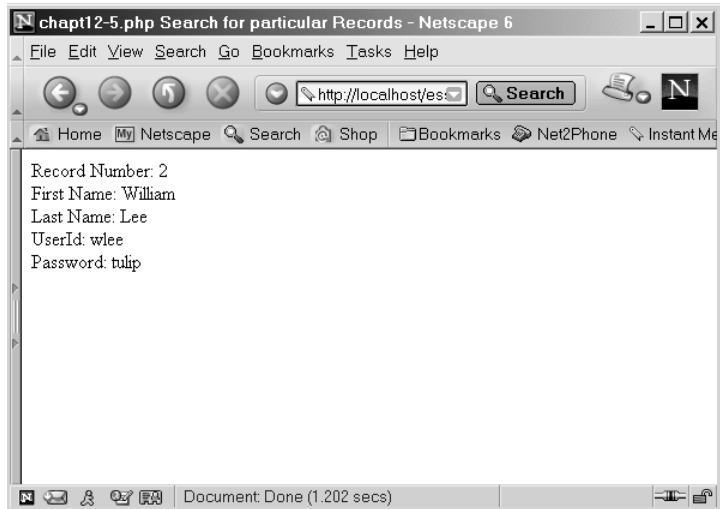
Figure 12.3 Displaying a particular record.

Using HyperLinks to Select Records

The following PHP script uses hyperlinks to access different database records:

```
<html>
  <head>
    <title>chapt12-6.php Using hyperlinks to access different
           records</title>
  </head>
  <body>
    <?php
    $db = mysql_connect("localhost", "root", "");
    mysql_select_db("exampledb",$db);
    if ($id) {
      $result = mysql_query("SELECT * FROM clients WHERE " .
            "id=$id",$db);
      if (!$rec = mysql_fetch_array($result)) {
        echo ("Sorry, no records found");
      }
```

```
     else {
        do {
          echo("Record Number: " . $rec["id"] . "<br>\n");
          echo("First Name: " . $rec["firstname"] .
"<br>\n");
          echo("Last Name: " . $rec["lastname"] . "<br>\n");
          echo("UserId: " . $rec["userid"] . "<br>\n");
          echo("Password: " . $rec["password"] .
"<br><br>\n");
        }
        while ($rec = mysql_fetch_array($result));
      }
    }
    echo ("<br><a href=\"$PHP_SELF?id=1\">Click for record
1</a>");
    echo ("<br><a href=\"$PHP_SELF?id=2\">Click for record
2</a>");
    echo ("<br><a href=\"$PHP_SELF?id=3\">Click for record
3</a>");
    ?>
  </body>
</html>
```

This example contains a new if statement which surrounds the majority of the script:

```
if ($id) {
  $result = mysql_query("SELECT * FROM clients WHERE
id=$id",$db);
```

The if statement checks to see if a variable $id exists. The first time through the script this variable does not exist and the mysql_query() function and all other code within the if statement is not processed. Below the if statement are three echo statements which display the hyperlinks:

```
echo ("<br><a href=\"$PHP_SELF?id=1\">Click for record 1</a>");
echo ("<br><a href=\"$PHP_SELF?id=2\">Click for record 2</a>");
echo ("<br><a href=\"$PHP_SELF?id=3\">Click for record 3</a>");
```

Each echo statement creates a link to this same script using the $PHP_SELF variable, which provides the name of the script. The ?id= part of the hyperlink passes the value assigned to the $id variable back to the script, when the

hyperlink is clicked. Depending on which hyperlink is selected a value of 1, 2 or 3 is assigned to the variable $id. This is then checked by the if ($id) statement at the start of the script. The mysql_query() function then accesses the database for all records which contain an id equal to the value in $id:

```
$result = mysql_query("SELECT * FROM clients WHERE
id=$id",$db);
```

Of course this example is not a brilliant design as we need to create a separate hyperlink for each record we wish to access. It would be better if we were able to interact with the script using a form to select the record for which to search.

Using a Form to Select Records

The following example illustrates using a form to access a database record:

```
<html>
  <head>
    <title>chapt12-7.php Using a form to access different
        records</title>
  </head>
  <body>
    <?php
    $db = mysql_connect("localhost", "root", "");
    mysql_select_db("exampledb",$db);
    if ($id) {
      $result = mysql_query("SELECT * FROM clients WHERE " .
            "id=$id",$db);
      if (!$rec = mysql_fetch_array($result)) {
        echo ("Sorry, no records found");
      }
      else {
        do {
          echo("Record Number: " . $rec["id"] . "<br>\n");
          echo("First Name: " . $rec["firstname"] .
"<br>\n");
          echo("Last Name: " . $rec["lastname"] . "<br>\n");
```

```
        echo("UserId: " . $rec["userid"] . "<br>\n");
        echo("Password: " . $rec["password"] .
"<br><br>\n");
      }
      while ($rec = mysql_fetch_array($result));
    }
  }
  ?>
  <form action="<?php echo $PHP_SELF?>" method="post">
  Id: <input type="text" name="id">
  <input type="submit" value="Search Database">
  </form>
</body>
</html>
```

This script replaces the hyperlinks with a simple form:

```
<form action="<?php echo $PHP_SELF?>" method="post">
Id: <input type="text" name="id">
<input type="submit" value="Search Database">
</form>
```

The form is written in HTML, but uses a short PHP script to provide the name of the script: action="<?php echo $PHP_SELF?>". The form consists of a single input field to allow the user to enter the id of the record they wish to search for. A submit button is included to allow the form data to be submitted to the script.

So far our script only allows us to search for records on the id number, but we should be able to search for records on any of the database fields.

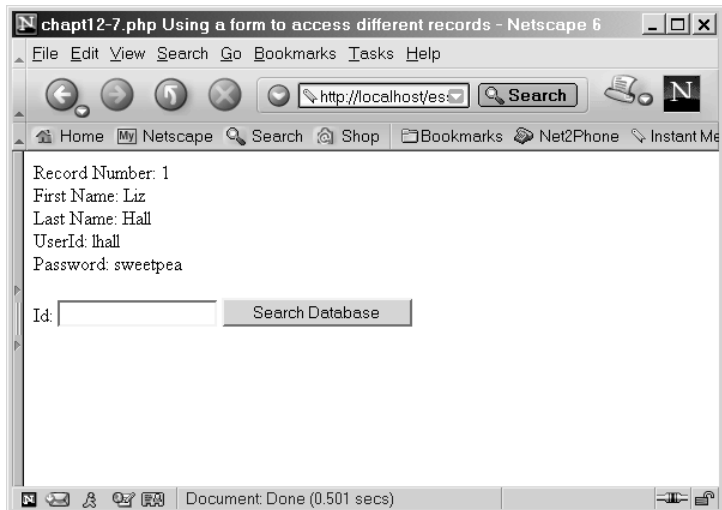The output produced by this script is illustrated in Figure 12.4.

Figure 12.4 Searching a database using a form.

Using a Form to Search on Different Fields

The following script extends the form in the previous script so that we can search the records on any field value:

```
<html>
  <head>
    <title>chapt12-8.php Using a form to access different
records
          </ title>
  </head>
  <body>
    <?php
    $db = mysql_connect("localhost", "root", "");
    mysql_select_db("exampledb",$db);
    if ($form) {
      $result = mysql_query("SELECT * FROM clients",$db);
      while ($rec = mysql_fetch_array($result)) {
        $ok = 0;
        if ($id && $rec["id"] == $id)
          $ok = 1;
```

```
        if ($firstname && $rec["firstname"] == $firstname)
          $ok = 1;
        if ($lastname && $rec["lastname"] == $lastname)
          $ok = 1;
        if ($userid && $rec["userid"] == $userid)
          $ok = 1;
        if ($password && $rec["password"] == $password)
          $ok = 1;
        if ($ok) {
          echo("Record Number: " . $rec["id"] . "<br>\n");
          echo("First Name: " . $rec["firstname"] .
"<br>\n");
          echo("Last Name: " . $rec["lastname"] .
"<br>\n");
          echo("UserId: " . $rec["userid"] . "<br>\n");
          echo("Password: " . $rec["password"] .
"<br><br>\n");
        }
      }
    }
    ?>
    <form action="<?php echo $PHP_SELF?>" method="post">
    Id: <input type="text" name="id"><br>
    First Name: <input type="text" name="firstname"><br>
    Last Name: <input type="text" name="lastname"><br>
    User Id: <input type="text" name="userid"><br>
    Password: <input type="text" name="password"><br>
    <input type="hidden" name="form" value="1">
    <input type="submit" value="Search Database">
    </form>
</body>
</html>
```

This example includes a larger form than in previous examples:

```
<form action="<?php echo $PHP_SELF?>" method="post">
Id: <input type="text" name="id"><br>
First Name: <input type="text" name="firstname"><br>
Last Name: <input type="text" name="lastname"><br>
User Id: <input type="text" name="userid"><br>
Password: <input type="text" name="password"><br>
<input type="hidden" name="form" value="1">
<input type="submit" value="Search Database">
</form>
```

The form contains input fields for each of the record fields. An additional hidden field has been included to send a variable $form to the script. The value of $form is checked to see if the form has been sent. The line:

```
$result = mysql_query("SELECT * FROM clients",$db);
```

selects all records from the database. A while loop iterates through each record:

```
while ($rec = mysql_fetch_array($result)) {
```

A variable $ok is set to 0 and then each of the form variables is checked in turn to see if they match the $rec[] field variable values. If so, then the value of $ok is set to 1:

```
$ok = 0;
if ($id && $rec["id"] == $id)
  $ok = 1;
if ($firstname && $rec["firstname"] == $firstname)
  $ok = 1;
if ($lastname && $rec["lastname"] == $lastname)
  $ok = 1;
if ($userid && $rec["userid"] == $userid)
  $ok = 1;
if ($password && $rec["password"] == $password)
  $ok = 1;
```

Finally, if the value of $ok is equal to 1 then the record is printed:

```
if ($ok) {
  echo("Record Number: " . $rec["id"] . "<br>\n");
  echo("First Name: " . $rec["firstname"] . "<br>\n");
  echo("Last Name: " . $rec["lastname"] . "<br>\n");
  echo("UserId: " . $rec["userid"] . "<br>\n");
  echo("Password: " . $rec["password"] . "<br><br>\n");
```

The output produced by this script is illustrated in Figure 12.5.
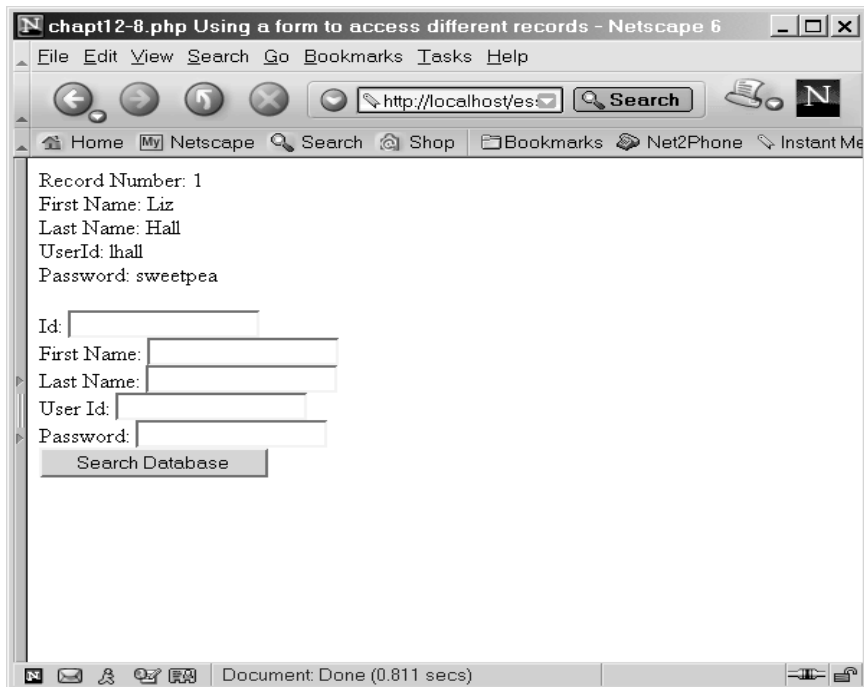
Figure 12.5 Searching on different fields using a form.

Adding Records

So far we have written scripts to allow us to extract data from the database. It would be beneficial if we were able to add new records to the database. The following illustrates a simple example allowing us to add database records:

```
<html>
  <head>
    <title>chapt12-9.php Adding records</title>
  </head>
  <body>
    <?php
    $db = mysql_connect("localhost", "root", "");
    mysql_select_db("exampledb",$db);
    $result = mysql_query("SELECT * FROM clients",$db);
    while ($rec = mysql_fetch_array($result)) {
```

```
      echo("Record Number: " . $rec["id"] . "<br>\n");
      echo("First Name: " . $rec["firstname"] . "<br>\n");
      echo("Last Name: " . $rec["lastname"] . "<br>\n");
      echo("UserId: " . $rec["userid"] . "<br>\n");
      echo("Password: " . $rec["password"] . "<br><br>\n");
    }
    if ($firstname) {
      $result = mysql_query("INSERT INTO clients " .
        " (firstname, lastname, userid, password) " .
        "VALUES ('$firstname', '$lastname', '$userid', " .
            " '$password')",$db);
      echo ("<br>Record Added");
    }
    ?>
    <form action="<?php echo $PHP_SELF?>" method="post">
    First Name: <input type="text" name="firstname"><br>
    Last Name: <input type="text" name="lastname"><br>
    User Id: <input type="text" name="userid"><br>
    Password: <input type="text" name="password"><br>
    <input type="submit" value="Add Record">
    </form>
  </body>
</html>
```

Breaking this example down into smaller sections we see that the following fragment of script:

```
$result = mysql_query("SELECT * FROM clients",$db);
while ($rec = mysql_fetch_array($result)) {
  echo("Record Number: " . $rec["id"] . "<br>\n");
  echo("First Name: " . $rec["firstname"] . "<br>\n");
  echo("Last Name: " . $rec["lastname"] . "<br>\n");
  echo("UserId: " . $rec["userid"] . "<br>\n");
  echo("Password: " . $rec["password"] . "<br><br>\n");
}
```

selects all the records from the database and uses a `while` loop displays them. The next section of code:

```
if ($firstname) {
  $result = mysql_query("INSERT INTO clients " .
    " (firstname, lastname, userid, password) " .
    "VALUES ('$firstname', '$lastname', '$userid', " .
        " '$password')",$db);
  echo ("<br>Record Added");
}
```

checks that the variable $firstname contains a value. This indicates that the form has been submitted. If so a mysql_query() function is invoked with the SQL command "INSERT INTO clients (firstname, lastname, userid, password) VALUES ('$firstname', '$lastname', '$userid', '$password')". This SQL command inserts a new database record into the database table.

The form script does not include an id field as it is generated automatically when the record is added to the database:

```
<form action="<?php echo $PHP_SELF?>" method="post">
First Name: <input type="text" name="firstname"><br>
Last Name: <input type="text" name="lastname"><br>
User Id: <input type="text" name="userid"><br>
Password: <input type="text" name="password"><br>
<input type="submit" value="Add Record">
</form>
```
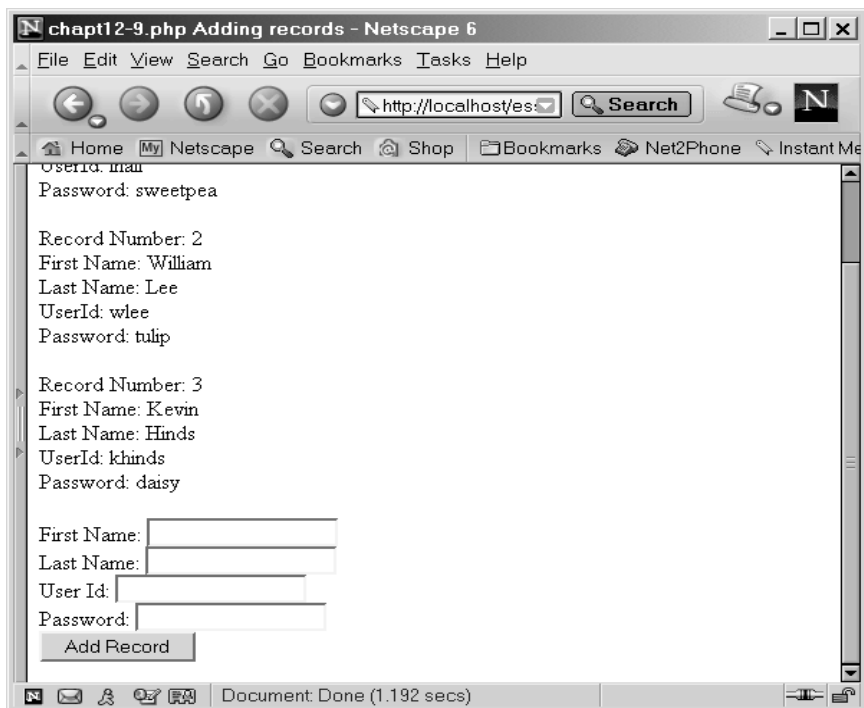
The output from this script is illustrated in Figure 12.6.

Figure 12.6 Adding records.

The script here is far from being perfect. For example, there is no check to ensure that the entire form has been completed. You can submit the form and create a record if you insert text into only the first name field. Also, although a record is added and a message displayed to this effect, the latest record is not displayed until the next submission of the form. This is due to that fact that the records are displayed before the new record is added. Furthermore, you must submit a firstname or no record will be added.

Deleting Records

Now that we can view and add records to the database the next step is to allow us to delete records. The following script allows us to delete database entries:

```
<html>
  <head>
    <title>chapt12-10.php Deleting records</title>
  </head>
  <body>
    <?php
    $db = mysql_connect("localhost", "root", "");
    mysql_select_db("exampledb",$db);
    if ($id) {
      $result = mysql_query("DELETE FROM clients " .
                 "WHERE id=$id",$db);
      echo ("<br>Record Deleted<br>");
    }
    $result = mysql_query("SELECT * FROM clients",$db);
    while ($rec = mysql_fetch_array($result)) {
      echo("Record Number: " . $rec["id"] . "<br>\n");
      echo("First Name: " . $rec["firstname"] . "<br>\n");
      echo("Last Name: " . $rec["lastname"] . "<br>\n");
      echo("UserId: " . $rec["userid"] . "<br>\n");
      echo("Password: " . $rec["password"] . "<br><br>\n");
    }
    ?>
    <form action="<?php echo $PHP_SELF?>" method="post">
    Id: <input type="text" name="id"><br>
    <input type="submit" value="Delete Record">
```

```
     </form>
   </body>
</html>
```

This example contains an if statement which checks the value of `$id`, indicating whether the form has been submitted. `$id` is used to select the record to be deleted:

```
if ($id) {
   $result = mysql_query("DELETE FROM clients WHERE
id=$id",$db);
   echo ("<br>Record Deleted<br>");
}
```

If the variable `$id` contains a value the `mysql_query()` function is invoked with the SQL "`DELETE FROM clients WHERE id=$id`". This deletes the record in the database with the id equal to `$id`. The form only contains a single input field as only the unique id is required to locate and delete a record:

```
<form action="<?php echo $PHP_SELF?>" method="post">
Id: <input type="text" name="id"><br>
<input type="submit" value="Delete Record">
</form>
```
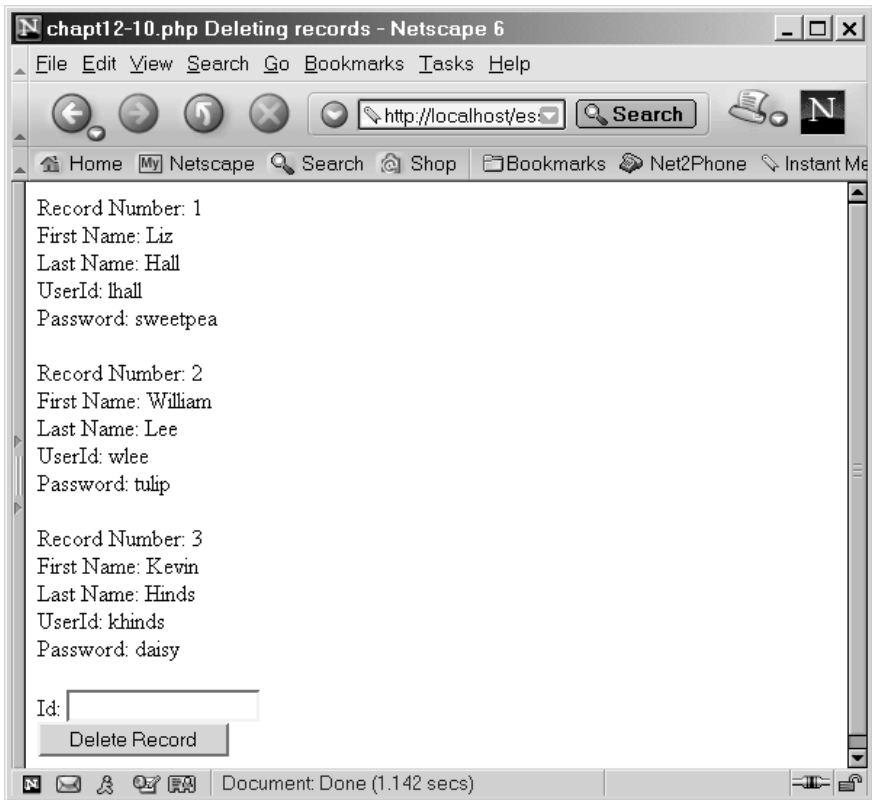
The output from this script is illustrated in Figure 12.7.

Figure 12.7 Deleting records.

Once again this script is not perfect as there is no check to ensure that the entered value of id is valid.

Updating Records

The next step is to allow us to amend a record. The following provides an example:

```
<html>
  <head>
    <title>chapt12-11.php Updating records</title>
  </head>
  <body>
```

```php
    <?php
    $db = mysql_connect("localhost", "root", "");
    mysql_select_db("exampledb",$db);
    if ($id) {
      $result = mysql_query("UPDATE clients " .
          "SET firstname='$firstname',
lastname='$lastname', " .
              "userid='$userid', password='$password' " .
          "WHERE id='$id'",$db);
      echo ("<br>Record Updated<br>");
    }
    $result = mysql_query("SELECT * FROM clients",$db);
    while ($rec = mysql_fetch_array($result)) {
      echo("Record Number: " . $rec["id"] . "<br>\n");
      echo("First Name: " . $rec["firstname"] . "<br>\n");
      echo("Last Name: " . $rec["lastname"] . "<br>\n");
      echo("UserId: " . $rec["userid"] . "<br>\n");
      echo("Password: " . $rec["password"] . "<br><br>\n");
    }
    ?>
    <form action="<?php echo $PHP_SELF?>" method="post">
    Id: <input type="text" name="id"><br>
    First Name: <input type="text" name="firstname"><br>
    Last Name: <input type="text" name="lastname"><br>
    User Id: <input type="text" name="userid"><br>
    Password: <input type="text" name="password"><br>
    <input type="submit" value="Update Record">
    </form>
  </body>
</html>
```

This script is similar to the previous example but includes a more complex form and includes the following script:

```php
if ($id) {
  $result = mysql_query("UPDATE clients SET
  firstname='$firstname', lastname='$lastname',
userid='$userid',
  password='$password' WHERE id='$id'",$db);
  echo ("<br>Record Updated<br>");
}
```

The `mysql_query()` function is used with the following SQL statement: "UPDATE clients SET firstname='$firstname', lastname='$lastname', userid='$userid', password=

'`$password`' `WHERE id='$id'`", which updates the record identified by `$id` with the supplied variables.
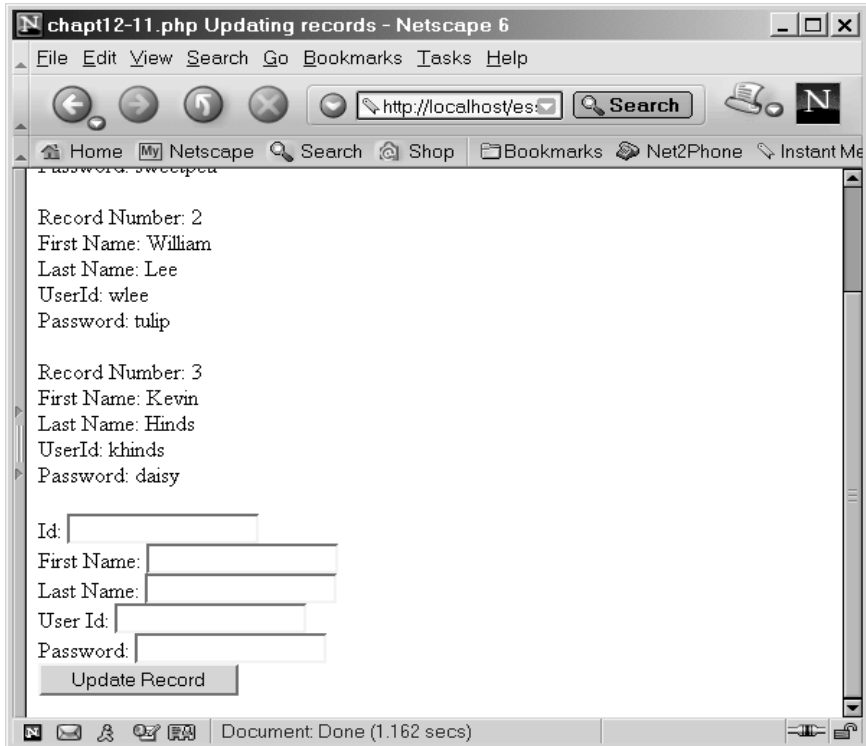

Figure 12.8 Updating records.

Summary

This chapter has illustrated how to interface PHP with a MySQL database and extract, add, delete and amend records within the database. Although the examples are simple, databases with more complex and multiple tables can be created and accessed in exactly the same way.

In the next chapter we will examine how to create user-defined functions.