PHP 101 (part 4) − Look, Ma...It's Alive!

By Vikram Vaswani and Harish Kamath

Table of Contents

The Marriage Of PHP And mySQL

One of the most important factors driving PHP's popularity over the last couple of years has been its support for a variety of databases, including mySQL, mSQL, Oracle and Microsoft Access. By simplifying and streamlining database access, PHP allows developers to build complex data−driven Web applications while enjoying short development cycles because of the simplicity and flexibility of the language.

One of the most powerful combinations in the open source arena is the PHP/mySQL combination. For those of you new to open−source technology − where have you been, you galoots?! − mySQL is a fast, reliable, open−source database management system. By using PHP and mySQL, developers can provide customers with huge savings on the licensing costs of other commercially−licensed software, and also benefit from the tremendous amount of thought that PHP and mySQL developers have put into making sure that the two packages work together seamlessly and smoothly.

OK. Enough of the marketing talk. Let's get down to business.

In this issue of PHP 101, we're going to show you how to use PHP to extract data from a database, and use that data to build a dynamic Web page. All you need are the usual pre−requisites: a sense of humour, and a willingness to try something new. Some knowledge of SQL (Structured Query Language, the language used to interact with a database server) would be helpful, though not essential.

If you're planning on trying out some of the examples below, you'll also need to download and install the mySQL database server, available at http://www.mysql.com. We'll be assuming that you've installed and configured mySQL, and have the appropriate permissions to create and edit database tables.

If you're using a database other than mySQL, fear not − PHP supports all major databases, and you can use the techniques described over the next few pages to talk to other databases too. Your PHP manual will help you locate corresponding functions for other databases.

Dumped!

If you're familiar with SQL, you know that there are four basic types of operations possible with a database:

```
SELECT a record; INSERT a record; UPDATE a record; DELETE a
record.
```

In order to demonstrate these operations, we're going to build a little application that allows users to maintain a list of their three favourite Web sites in a database, and share this list with the rest of the Internet community.

The first step that goes into building such an application involves designing the database tables − in our case, each user will be identified by a unique login id, which will map to the list of three Web sites. We've put together a "dump file", which lets you create the database tables and initial set of records quickly − we suggest that you import this data into your mySQL database server, as we'll be using it throughout this article.

To import the data, download the dump file and use this command at your mySQL prompt:

```
mysql> mysql −u username −p database < dumpfile
```

Or you could insert the contents manually − here is what you 'll need:

```
# # Table structure for table 'url_list' # CREATE TABLE
url_list ( uid varchar(8) NOT NULL, title1 varchar(30) NOT
NULL, url1 text NOT NULL, title2 varchar(30) NOT NULL, url2
text NOT NULL, title3 varchar(30) NOT NULL, url3 text NOT
NULL, PRIMARY KEY (uid) ); # # Dumping data for table
'url_list' # INSERT INTO url_list VALUES( 'john', 'Melonfire',
'http://www.melonfire.com', 'Devshed',
'http://www.devshed.com', 'PHP.Net', 'http://www.php.net');
INSERT INTO url_list VALUES( 'bill', 'Yahoo',
'http://www.yahoo.com', 'Slashdot', 'http://www.slashdot.org',
'32Bit.com', 'http://www.32bit.com');
```

This will create a table named "url_list" with columns for usernames, Web site titles and Web site URLs for two mythical users, "bill" and "john".

Now check whether or not the data has been successfully imported with a SELECT query (the SELECT SQL statement is used to retrieve information from a database.) Enter this at your mySQL command prompt:

```
mysql> select count(*) from url_list;
```

which, in English, means "count all the records in the table url_list and give me the total", and you should see the number "2" as a result, indicating that there are two records in the table.

Hello Database!

All working? Good. Now, let's use PHP to do exactly the same thing − fire a SELECT query at the database, and display the results in an HTML page.

```
<html> <head> </head> <body> <?php // set up some variables //
server name $server = "localhost"; // username $user = "test";
// password $pass = "test"; // database to query $db =
"php101"; // open a connection to the database $connection =
mysql_connect($server, $user, $pass); // formulate the SQL
query − same as above $query = "select count(*) from
url_list"; // run the query on the database // assume the
database is named "php101" $result = mysql_db_query($db,$query
,$connection); // assign the result to a variable $counter =
mysql_result($result,0); // and display it echo "There are
$counter records in the database"; // free up used memory
mysql_free_result($result); ?> </body> </html>
```

And you'll see something like this:

```
There are 2 records in the database
```

As you can see, using PHP to get data from a database involves several steps, each of which is actually a pre−defined PHP function. Let's dissect each step:

1. The first thing to do is specify some important information needed to establish a connection to the database server. This information includes the server name, the username and password required to gain access to it, and the name of the database to query. These values are all set up in regular PHP variables.

2. In order to begin communication with the mySQL database server, you first need to open a connection to the server. All communication between PHP and the database server takes place through this connection.

In order to initialize this connection, PHP offers the mysql_connect() function.

```
$connection = mysql_connect($server, $user, $pass);
```

The function requires three parameters: the name of the server, and the mySQL username and password. If the database server and the Web server are both running on the same physical machine, the server name is usually "localhost"

This function then returns a "link identifier", which is stored in the variable $connection; this identifier is used throughout the script when communicating with the database.

3. Now that you have a connection to the database, it's time to send it a query via the mysql_db_query() function. This function also needs three parameters: the database name, the query string and the link identifier

for the connection.

```
$query = "select count(*) from url_list"; $result =
mysql_db_query($db, $query, $connection);
```

The result set returned by the function above is stored in the variable $result. This result set may contain, depending on your query, one or more rows or columns of data. You then need to retrieve specific sections or subsets of the result set with different PHP functions − the one we've used is the mysql_result() function, which uses the result variable and the row number to return the value you need. You can optionally add the column name as well to get to a specific value.

```
$counter = mysql_result($result,0);
```

There are several other efficient alternatives to this function, which will be explained a little further down.

4. Finally, each result set returned after a query occupies some amount of memory − and if your system is likely to experience heavy load, it's a good idea to use the mysql_free_result() function to free up the used memory.

Different Strokes...

What you just saw was a very basic example. Our next example will query the database, return the list of users with their URL list, and display it all in a neat

...For Different Folks

You can use PHP's mysql_fetch_row() function to obtain a simple array of values, and then use these values according to the array index − a slightly different variation of the technique used above. Take a look:

```
<html> <head> </head> <body> <?php // set up some variables //
server name $server = "localhost"; // username $user = "test";
// password $pass = "test"; // database to query $db =
"php101"; // open a connection to the database $connection =
mysql_connect($server, $user, $pass) or die("Invalid server or
user"); // formulate the SQL query $query = "select * from
url_list" or die("Error in query"); // run the query on the
database $result = mysql_db_query($db, $query, $connection) or
die("Error in query"); // display the result echo "<table
width=450 border=1 cellspacing=0 cellpadding=0>"; echo
"<tr><td width=150 align=left>User</td><td width=100
align=left>Site #1</td><td width=100 align=left>Site
#2</td><td width=100 align=left>Site #3</td></tr>"; // with a
while loop // this loop will iterate as many times as there
are records // and return an array named $myrow while ($myrow
= mysql_fetch_row($result)) { echo "<tr><td width=150
align=left>$myrow[0]</td><td width=100 align=left><a
href=$myrow[2]>$myrow[1]</a></td><td width=100 align=left><a
href=$myrow[4] >$myrow[3]</a></td><td width=100 align=left><a
href=$myrow[6]> $myrow[5]</a></td></tr>"; } // memory flush
mysql_free_result($result); ?> </body> </html>
```

You can also use PHP's mysql_fetch_row() and list() functions to obtain a simple array of values, and then assign these values to different variables − another variation of the technique we've shown you above. Take a look − only the "while" loop will change:

```
<html> <head> </head> <body> <?php // set up some variables //
server name $server = "localhost"; // username $user = "test";
// password $pass = "test"; // database to query $db =
"php101"; // open a connection to the database $connection =
mysql_connect($server, $user, $pass) or die("Invalid server or
user"); // formulate the SQL query $query = "select * from
url_list" or die("Error in query"); // run the query on the
database $result = mysql_db_query($db, $query, $connection) or
die("Error in query"); // display the result echo "<table
width=450 border=1 cellspacing=0 cellpadding=0>"; echo
"<tr><td width=150 align=left>User</td><td width=100
align=left>Site #1</td><td width=100 align=left>Site
#2</td><td width=100 align=left>Site #3</td></tr>"; // with a
while loop // this loop will iterate as many times as there
are records // and the values returned will be assigned to
different // variables via list() while (list($user, $title1,
```

```
$url1, $title2, $url2, $title3, $url3) =
mysql_fetch_row($result)) { echo "<tr><td width=150
align=left>$user</td><td width=100 align=left><a
href=$url1>$title1</a></td><td width=100 align=left><a
href=$url2>$title2</a></td><td width=100 align=left><a
href=$url3>$title3</a></td></tr>"; } // memory flush
mysql_free_result($result); ?> </body> </html>
```

In this case, the list() function is used to assign different elements of the result set t o PHP variables, which are then used when rendering the page.

What's In A Name?

Now that you've got the basics straight, let's evolve the application a little further. Our next example will ask the user to enter a user name. If it's a valid user name, the script will connect to the database and display the corresponding list of URLs; if not, it'll return an error.

We'll be using a single page for the entire operation − the $submit variable [you remember this technique, don't you?] is used to decide whether to display the initial form or the result page. Take a look:

```
<?php // check if the form has been submitted if($submit) { //
initialize database connection $conn =
mysql_connect("localhost", "test", "test");
mysql_select_db("php101", $conn); $result =
mysql_query("select title1,url1,title2,url2,title3,url3 from
url_list where uid = '$username'", $conn); // check if the
user name is valid $num_rows = mysql_num_rows($result); // if
no rows are returned then the username is invalid
if(!$num_rows) { ?> <html> <head> <basefont face="Arial">
</head> <body> <center> <font size="3">User name not
found!</font><br> <a href="list.php4">Click here to try
again.</a> </center> </body> </html> <?php } else { // else if
user name is valid // display list $result =
mysql_query("select title1,url1,title2,url2,title3,url3 from
url_list where uid = '$username'", $conn);
list($title1,$url1,$title2,$url2,$title3,$url3) =
mysql_fetch_row($result); ?> <html> <head> <basefont
face="Arial"> </head> <body> <center> <font size="3">Welcome,
<?php echo $username; ?>!</font><br> Pick your destination:
<p> <?php echo "<a href=$url1>$title1</a>"; ?>   <?php echo
"<a href=$url2>$title2</a>"; ?>   <?php echo "<a
href=$url3>$title3</a>"; ?> </center> </body> </html> <?php
mysql_free_result($result); } } else { // $submit not found //
so display a form ?> <html> <head> <basefont face="Arial">
</head> <body> <form action="list.php4" method="POST"> <table>
<tr> <td> Username: </td> <td> <input type="text"
name="username" length=10 maxlength="30"> </td> </tr> <tr> <td
colspan="2" align="center"> <input type="submit" name="submit"
value="Log in"> </td> </tr> </table> </form> </body> </html>
<?php } ?>
```

Log in as "bill" or "john" and watch as PHP connects to the database and retrieves that user's list of sites.

If you look closely, you'll see that the script above actually contains three HTML pages embedded within it, one for each of the three possible scenarios. The first time the user visits the page, the variable $submit will not be set, and so a simple HTML form is displayed; this forms asks for a user name, which is stored in the variable $username.

Once the form has been submitted, the same page is called again; however, this time around, the $submit variable will be found and so PHP will initiate a connection to the database to check if the user name is valid.

One simple technique that can be used for simple "yes/no" type queries [such as the one above] is the mysql_num_rows() function; this function tells you the number of result rows returned by the query. A value of zero implies that no rows were found, which in turn implies that the user was not found.

The result of mysql_num_rows() is stored in the variable $num_rows. If this variable exists, it implies that the user name is valid, and it's then possible to move ahead and fetch the Web addresses corresponding to that user name from the database. These addresses are then formatted and displayed on the page as active hyperlinks.

New Friends

Thus far, we've simply been using SELECT queries to pull information out of a database. But how about putting something in?

SQL aficionados know that this happens via an INSERT query − and our next example demonstrates how you can use an HTML form to insert data into the database.

```
<? // if form has not been submitted, display form if
(!$submit) { ?> <html> <head> <basefont face=Arial> </head>
<body> <h3>Enter your bookmarks:</h3> <form method="POST"
action="push.php4"> <table> <tr> <td> Username </td> <td>
<input name="username" length="10" maxlength="30"> </td> </tr>
<tr> <td> Web site </td> <td> <input name="title1" length="30"
maxlength="30"> </td> </tr> <tr> <td> URL </td> <td> <input
name="url1" length="30"> </td> </tr> <tr> <td> Web site </td>
<td> <input name="title2" length="30" maxlength="30"> </td>
</tr> <tr> <td> URL </td> <td> <input name="url2" length="30">
</td> </tr> <tr> <td> Web site </td> <td> <input name="title3"
length="30" maxlength="30"> </td> </tr> <tr> <td> URL </td>
<td> <input name="url3" length="30"> </td> </tr> <tr> <td
colspan="2" align="center"> <input type="submit" name="submit"
value="Submit"> </td> </tr> </table> </form> </body> </html>
<? } // or process form input else { ?> <?php // connect to
database $connection = mysql_connect("localhost", "test",
"test") or die("Invalid server or user"); // select database
mysql_select_db("php101",$connection); // formulate and run
query $query = "insert into
url_list(uid,title1,url1,title2,url2,title3,url3)
values('$username','$title1','$url1','$title2','$url2','$tit
le3','$url3')"; $result = mysql_query($query,$connection) or
die("Error in query"); ?> <html> <head> <basefont face=Arial>
</head> <body> <center> <h3>Success!</43> <table> <tr> <td>
<?php echo $username; ?>'s bookmarks have been saved. </td>
</tr> </center> </body> </html> <? } ?>
```

This time around, we've demonstrated yet another PHP function − mysql_select_db(), which allows you to specify the database which will be used when connecting to the database. And since you've specified the database, you can use mysql_query() instead of mysql_db_query() − while the latter needs the database name as parameter [as you've seen in the examples above], the former simply needs the query string and the connection identifier.

As you can see, inserting a record into the database is very straightforward − simply fire the query and your INSERT statement will be executed. The only way to find out if the INSERT was successful is to check the value of the variable $result − if the variable isn't set, it implies that something didn't go as planned.

Today's Special

You should also give your users the option of updating their list of preferred Web sites − in SQL, this is accomplished via an UPDATE statement. The example below asks for a user name and then displays the list of URLs which can be edited − this list is obtained from the database.

Once you submit the form with your modifications, another PHP script, "modify.php4", is called; this script is responsible for generating the UPDATE query and modifying the items in the database.

[list.php4]

```
<?php // this script checks the username and then displays the
URLs // for the user as editable text boxes // check if the
form has been submitted if($submit) { // initialize database
connection $conn = mysql_connect("localhost", "test", "test");
mysql_select_db("php101", $conn); $result =
mysql_query("select title1,url1,title2,url2,title3,url3 from
url_list where uid = '$username'", $conn); // check if the
user name is valid $num_rows = mysql_num_rows($result); // if
no rows are returned then the username is invalid
if(!$num_rows) { ?> <html> <head> <basefont face="Arial">
</head> <body> <center> <font size="3">User name not
found!</font><br> <a href="list.php4">Click here to try
again.</a> </center> </body> </html> <?php } else { // else if
user name is valid // display list $result =
mysql_query("select title1,url1,title2,url2,title3,url3 from
url_list where uid = '$username'", $conn);
list($title1,$url1,$title2,$url2,$title3,$url3) =
mysql_fetch_row($result); ?> <html> <head> <basefont
face="Arial"> </head> <body> <center> <font size="3">Welcome,
<?php echo $username; ?>!</font><br> Select the sites you'd
like to modify: <p> <form action="modify.php4" method="post">
<table border=1> <tr> <td> <input type=text name=title1
value="<? echo $title1; ?>"> </td> <td> <input type=text
name=url1 value="<? echo $url1; ?>"> </td> </tr> <tr> <td>
<input type=text name=title2 value="<? echo $title2; ?>">
</td> <td> <input type=text name=url2 value="<? echo $url2;
?>"> </td> </tr> <tr> <td> <input type=text name=title3
value="<? echo $title3; ?>"> </td> <td> <input type=text
name=url3 value="<? echo $url3; ?>"> </td> </tr> <tr> <td
colspan=3 align=center> <input type=submit value="Modify!">
</td> </tr> </table> <input type=hidden name=username
value="<? echo $username; ?>"> </form> </center> </body>
</html> <?php mysql_free_result($result); } } else { //
$submit not found // so display a form ?> <html> <head>
<basefont face="Arial"> </head> <body> <form
action="list.php4" method="POST"> <table> <tr> <td> Username:
</td> <td> <input type="text" name="username" length=10
```

```
maxlength="30"> </td> </tr> <tr> <td colspan="2"
align="center"> <input type="submit" name="submit" value="Log
in"> </td> </tr> </table> </form> </body> </html> <?php } ?>
```

[modify.php4]

```
<?php // this script takes all the information from list.php4
// and updates the database // connect $connection =
mysql_connect("localhost", "test", "test") or die("Invalid
server or user"); // select database
mysql_select_db("php101",$connection) or die("Invalid
database"); // generate query $query = "update url_list set
title1 = '$title1', url1 = '$url1', title2 = '$title2', url2 =
'$url2', title3 = '$title3', url3 = '$url3' where uid =
'$username'"; $result = mysql_query($query,$connection) or
die("Error in query"); ?> <html> <head> <basefont face=Arial>
</head> <body> <?php if($result) { echo
"<center><h3>Success!</h3><p>$username's bookmarks have been
modified.</center>"; } ?> </body> </html>
```

Nuking The People

Finally, now that you've added a million users, it's time to do a little routine maintenance. This example demonstrates how to use the DELETE statement to wipe out a user and his list of sites. Again, the basic principles remain the same, with only the query string changing.

```
<?php // has the form been submitted? if($submit) { // connect
to the database $conn = mysql_connect("localhost", "test",
"test"); mysql_select_db("php101",$conn); $result =
mysql_query("select title1,url1,title2,url2,title3,url3 from
url_list where uid = '$username'",$conn); // check username
$num_rows = mysql_num_rows($result); // if no rows are
returned, the username is invalid. if(!$num_rows) { ?> <html>
<head> <basefont face=Arial> </head> <body> <center> <font
size="3">User not found</font><br> <a href="nuke.php4">Click
here to try again</a> </center> </body> </html> <?php } else {
// if username is valid, delete the bookmarks. $result =
mysql_query("delete from url_list where uid =
'$username'",$conn); ?> <html> <head> <basefont face=Arial>
</head> <body> <center> <h3>Success!</h3> <?echo $username;
?>'s book marks have been deleted! </center> </body> </html>
<?php } } else { // if form has not been submitted ?> <html>
<body> <form action="nuke.php4" method="POST"> <table> <tr>
<td> Username </td> <td> <input type="text" name="username"
length=10 maxlength="30"> </td> </tr> <tr> <td colspan="2"
align="center"> <input type="submit" name="submit"
value="Delete Bookmarks"> </td> </tr> </table> </form> </body>
</html> <? } ?>
```

Oops!

All done? Nope, not quite yet − before you go out there and start building cool data−driven Web sites for your customers, you should be aware that PHP comes with some powerful error−tracking functions which can speed up development time. Take a look at the following example, which contains a deliberate error in the SELECT query string:

```
<?php // connect $connection = mysql_connect("localhost",
"test", "test") or die("Invalid server or user");
mysql_select_db("php101",$connection) or die("Invalid
database"); // query $query = "select from url_list"; //
result $result = mysql_query($query,$connection); if(!$result)
{ $error_number = mysql_errno(); $error_msg = mysql_error();
echo "MySQL error $error_number: $error_msg"; } ?>
```

And here's the output:

```
MySQL error 1064: You have an error in your SQL syntax near
'from url_list' at line 1
```

The mysql_errno() function displays the error code returned by mySQL if there's an error in your SQL statement, while the mysql_error() function returns the actual error message. Turn these both on, and you'll find that they can significantly reduce the time you spend fixing bugs.

And that's about all we have for this issue of PHP 101. Next week, we'll be wrapping things up with a look at PHP's file read/write capabilities, and we'll also show you how to define your own PHP functions. Don't miss it!