Introduction to Using PHP with MySQL

by Joey Mingrone


Comment on this article

PHP is a scripting language that deserves the honor of being the most popular add-on module for the most popular web server today. It's fast, stable, constantly updated, well documented, and best of all, it's free.

PHP is in it's element when it is running on a web server, grabbing data from a database and generating dynamic HTML. In this regard, it is very similar to ASP, Cold Fusion, JSP or even CGI. Running as a CGI program it can be used with just about any web server, but for better performance, it can run as an ISAPI filter, and for best performance, it can be run as an Apache module. As for database support, PHP supports a very large number of DBMSs including dBase, Informix, Oracle, PostgreSQL, MS SQL Server, MySQL, Sybase, Solid, as well as ODBC.

The purpose of this article is to give a programmer, new to PHP, an introduction to the language. Additionally, rudimentary database functionality will be covered using MySQL. If you're a C, C++, Java or PERL programmer you will be slightly ahead of the game because in many ways, PHP borrows its syntax from these languages. Even if you've never programmed in any of these languages before, this article is still basic enough for you. The only assumptions made are:

- you have some kind of a programming background
- you have PHP 4.x and MySQL working (it doesn't matter what OS you are running)

If you need installation instructions for PHP head here. Here's MySQL installation instructions.

Here's another good article on DevShed describing the whole setup on a Unix or Unix clone system.

OK, every introduction to a new language does it, so we better not break the tradition. Let's start with a "Hello World!" example.

```
<!------- FILE: helloWorld.php ------->
<html>
   <body>

      <?php
         $aVar = "hello world!";
         echo($aVar);
      ?>

   </body>
</html>
<!------------------------------------>
```

That's it! Put this page in a web shared directory on your web server and open it in a browser and you should see **hello world!** Everything within the **<?php** and **?>** should be interpreted by the PHP engine. If the directive short_open_tag is set to **On** in your php.ini file, you can use **<?** instead of **<?php** to open a php tag (By default this directive is set to **On** in PHP 4.x). Take note of the following in the above example:

- Like PERL, all variables start with a $ and do not have to be declared
- Commands are separated with a semi-colon,
- To assign a value to a variable use: $var = *value*;

- **echo** is used to insert text into the html document

So let's look at how this page is displayed. First, you request the page by typing the URL in your browser. The web server sees that it has a .php extension, so it sends it off to the PHP engine. Everything within the **<?php** and **?>** is interpreted by PHP and plain HTML is sent back to the web server. Finally, the page is sent to your browser. If you looked at the source of this page, you would see:

```
<!------- FILE: helloWorld.php ------->
<html>
   <body>

      hello world!

   </body>
</html>
<!------------------------------------->
```

Now let's get MySQL into the mix. Here's an example that connects to a MySQL database.
Note - you must have an account with the MySQL server to be able to connect.

```
<!------- FILE: example1.php ------->
<?php

if( !mysql_connect("DATABASEHOST", "USERNAME", "PASSWORD") )
{
   echo("connection failed");
}
else
{
   echo("successfully connected");
}

?>
<!------------------------------------->
```

Let's examine this example. **if(!mysql_connect("DATABASEHOST", "USERNAME", "PASSWORD"))** is saying, try to connect to the mysql server called "DATABASEHOST" with the username "USERNAME" and the password "PASSWORD". If the connection fails display the message "connection failed" otherwise display the message "successfully connected". The function mysql_connect returns a **true** value when the connection is successful and a **false** value when the connection fails. The exclamation point means **logical NOT**. So this example is saying **if not successfully connected** display **connection failed** otherwise display **successfully connected**.

```
<!------- FILE: example2.php ------->
<?php

if( !($dbLink = mysql_connect("DATABASEHOST", "USERNAME", "PASSWORD")) )
{
   echo("connection failed<br>");
   exit();
}
else
{
   echo("successfully connected<br>");
   if( !mysql_create_db("ExampleDB", $dbLink) )
   {
      echo("error creating database<br>");
      exit();
   }
   mysql_select_db("ExampleDB", $dbLink);
   $createTableString = "CREATE TABLE ExampleTable";
   $createTableString .= "(ID SMALLINT ZEROFILL NOT NULL ";
   $createTableString .= "AUTO_INCREMENT PRIMARY KEY, Name VARCHAR(50))";
   if( !mysql_query($createTableString, $dbLink) )
   {
      echo("error creating table<br>");
      exit();
   }
   echo("success<br>");
}

?>
```

```
<!------------------------------------>
```

If you see the message:
successfully connected
success
then you're in good shape. If you see anything else, first make sure your code was copied correctly. If you are sill getting an error, perhaps you don't have the appropriate permissions to create a database or table on the MySQL server.

Example 2 adds on to example 1. **if(!mysql_connect("DATABASEHOST", "USERNAME", "PASSWORD"))** is still connecting us to the database server, only this time we are saving the link to the database in the variable $dbLink. **mysql_create_db("ExampleDB", $dbLink)** is using the link to the server to create a new database called "ExampleDB". **mysql_select_db("ExampleDB", $dbLink);** means that we want to work with this database. Finally, **mysql_query($createTableString, $dbLink)** is creating a new table in the ExampleDB database called "ExampleTable".

```
<!------- FILE: example3.php ------->
<?php

if( !($dbLink = mysql_connect("DATABASEHOST", "USERNAME", "PASSWORD")) )
{
   echo("connection failed<br>");
   exit();
}
else
{
   echo("successfully connected<br>");
   mysql_select_db("ExampleDB", $dbLink);

   $query = "INSERT INTO ExampleTable (ID, Name)";
   $query .= " VALUES (NULL, 'Hello World!')";
   if( !mysql_query($query, $dbLink) )
   {
      echo("error inserting into table<br>");
      exit();
   }
   if( !($queryID = mysql_query("SELECT * FROM ExampleTable", $dbLink)) )
   {
      echo("error selecting form the table<br>");
      exit();
   }
   $queryRow = mysql_fetch_array($queryID);
   echo($queryRow[ID] . " " . $queryRow[Name]);
}

?>
<!------------------------------------>
```

This example also builds upon previous examples. After connecting to the server and selecting the ExampleDB database, we now insert and select a row from ExampleTable. **mysql_query("INSERT INTO ExampleTable (ID, Name) VALUES (NULL, 'Hello World!')", $dbLink)** uses our link to the server ($dblink) to add a row to the ExampleTable. We insert NULL in the first column because it was declared as an auto_increment field, which means the database automatically decides a value (In the next example, we'll see how to access this value). **$queryID = mysql_query("SELECT * FROM ExampleTable", $dbLink)** selects all columns of all rows (only 1 row in our case) from the ExampleDB table. We save a link to this query in the variable $queryID. Finally we use the function **mysql_fetch_array** to save one row as an associative array in the variable $queryRow, and we print out what we selected from the database. Take a look at the last line in this example. **echo($queryRow[ID] . " " . $queryRow[Name]);** means display the ID field then concatenate (that's what the periods mean, concatenate a string) a space, then concatenate the Name field.

```
<!------- FILE: example4.php ------->
<?php

if( !($dbLink = mysql_connect("DATABASEHOST", "USERNAME", "PASSWORD")) )
{
   echo("connection failed<br>");
   exit();
}
else
{
   echo("successfully connected<br>");
   mysql_select_db("ExampleDB", $dbLink);

   $query = "INSERT INTO ExampleTable (ID, Name) VALUES ";
```

```
   $query .= "(NULL, 'The moon in June is like a big balloon!') ";
   if( !mysql_query($query, $dbLink) )
   {
      echo("error inserting into table<br>");
      exit();
   }

   echo("The last auto_increment id inserted into the table was ");
   echo(mysql_insert_id($dbLink) . "<br>");

   if( !($queryID = mysql_query("SELECT * FROM ExampleTable", $dbLink)) )
   {
      echo("error selecting form the table<br>");
      exit();
   }

   while($queryRow = mysql_fetch_array($queryID))
      echo($queryRow[ID] . " " . $queryRow[Name] . "<br>");

   echo("<p>\n");

   mysql_data_seek($queryID, 0);
   while($queryRow = mysql_fetch_array($queryID))
   {
      echo($queryRow[ID] . " " . $queryRow[Name] . "<br>");
   }
}

?>
<!------------------------------------>
```

In this final example we introduce three new concepts. **echo("The last auto_increment id inserted into the table was " . mysql_insert_id($dbLink) . "
);** shows a very important new function **mysql_insert_id**. As you've probably figured out from the string displayed, **mysql_insert_id** returns the value of the last inserted auto_increment for the database link $dbLink.

```
   while($queryRow = mysql_fetch_array($queryID))
      echo($queryRow[ID] . " " . $queryRow[Name] . "<br>");
```

This shows how to loop through multiple rows of data. To do this we used a while loop. One thing to notice about this while loop is that the statement within is not surrounded by curly brackets { }. When the curly brackets are not present, the loop will continue until it sees the first semi-colon then it will loop back. With the curly brackets, the while loop will continue until it sees its matching bracket then loop back. Did you also notice that the second time through the loop the second row was automatically printed out? That's because **mysql_fetch_array** automatically increments the row pointer to the next row.

This brings us to **mysql_data_seek($queryID, 0);**. This function simply sets the row pointer back to the beginning. If we didn't call this function before the second

```
   while($queryRow = mysql_fetch_array($queryID))
   {
      echo($queryRow[ID] . " " . $queryRow[Name] . "<br>");
   }
```

nothing would be displayed because the **mysql_fetch_array()** function would return false immediately.

Conclusion

After completing this introduction, you should be able to:

- understand the basics of a PHP page
- be able to connect to a MySQL database
- create a MySQL table
- insert data into a MYSQL table
- select data from a MySQL table

References

PHP Manual
MySQL Documentation

Developed Under:

Apache 1.3.12
PHP Version 4.0.1
MySQL Version 9.38 Distribution 3.22.32 for FreeBSD 4.0