Create dynamic sites with PHP & MySQL

Presented by developerWorks, your source for great tutorials

ibm.com/developerWorks

Table of Contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

Section 1. About this tutorial

Should I take this tutorial?

This tutorial shows you how to use two open source, cross-platform tools for creating a dynamic Web site: PHP and MySQL. When we are finished, you will know how dynamic sites work and how they serve the content, and you will be ready to serve your own dynamic content from your site.

About the author

For technical questions about the content of this tutorial, contact the author, Md. Ashraful Anam, at *russell@bangla.net* .

Md. Ashraful Anam works as an independent Web developer. Having conquered the Windows platform, he recently changed his interest to Linux and immediately fell in love with it.

In his spare time he can be seen wandering the virtual avenues of the net, testing open source software, and trying to promote his country, Bangladesh, in the international IT market. He can be reached at *russell@bangla.net* .

Section 2. Introduction and installation

The need for dynamic content

The Web is no longer static; it's dynamic. As the information content of the Web grows, so does the need to make Web sites more dynamic. Think of an e-shop that has 1,000 products. The owner has to create 1,000 Web pages (one for each product), and whenever anything changes, the owner has to change all those pages. Ouch!!! Wouldn't it be easier to have only one page that created and served the content on the fly from the information about the products stored in a database, depending on the client request?

Nowadays sites have to change constantly and provide up-to-date news, information, stock prices, and customized pages. PHP and SQL are two ways to make your site dynamic.

PHP PHP is a robust, server-side, open source scripting language that is extremely flexible and actually fun to learn. PHP is also cross platform, which means your PHP scripts will run on Unix, Linux, or an NT server.

MySQL SQL is the standard query language for interacting with databases. MySQL is an open source, SQL database server that is more or less free and extremely fast. MySQL is also cross platform.

Installing Apache server routines

First we will install the Apache server routines in the Linux environment. To install these packages you will need root access to your server. If someone else is hosting your site, ask the administrator to install them for you.

Installing Apache is relatively simple. First download the Apache archive, apache_x.x.xx.tar.gz (the latest I downloaded was apache_1.3.14.tar.gz) from the *Apache site* and save it in /tmp/src directory. Go to that directory:

```
# cd /tmp/src/
```

Extract the files with the command:

```
# gunzip -dc apache_x.x.xx.tar.gz | tar xv
```

replacing those xs with your version number. Change to the directory that has been created:

```
# cd apache_x.x.xx
```

Now to configure and install apache, type the commands:

```
# ./configure --prefix=/usr/local/apache --enable-module=so
# make
# make install
```

This will install Apache in the directory /usr/local/apache. If you want to install Apache to a different directory, replace /usr/local/apache with your directory in the prefix. That's it! Apache is installed.

You might want to change the default server name to something of real value. To do this, open the httpd.conf file (located at /usr/local/apache/conf) and find the line starting with `ServerName`. Change it to `ServerName localhost`.

To test your install, start up your Apache HTTP server by running:

```
# /usr/local/apache/bin/apachectl start
```

You should see a message like "httpd started". Open your Web browser and type "http://localhost/" in the location bar (replace localhost with your ServerName if you set it differently). You should see a nice welcome page.

Installing MySQL

Next comes MySQL. We will follow the same procedure (replacing those xs again with our version number). Download the source from the *MySQL site* and save it in /tmp/src. The latest version I found was mysql-3.22.32.tar.gz.

```
# cd /tmp/src/
# gunzip -dc mysql-x.xx.xx.tar.gz | tar xv
# cd mysql-x.xx.xx
# ./configure --prefix=/usr/local/mysql
# make
# make install
```

MySQL is installed. Now you need to create the grant tables:

```
# scripts/mysql_install_db
```

Then start the MySQL server:

```
# /usr/local/bin/safe_mysqld &
```

And test your installation by typing:

```
mysql -uroot -p
```

At the password prompt, just press Enter. You should see something like:

```
Welcome to MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 5 to server version 3.22.34

Type 'help' for help.

mysql>
```

If you see this, you have MySQL running properly. If you don't, try installing MySQL again. Type `status` to see the MySQL server status. Type `quit` to exit the prompt.

Installing PHP

We will follow a similar procedure to install PHP. Download and save the source from the *PHP site* to /tmp/src:

```
# cd /tmp/src/
# gunzip -dc php-x.x.xx.tar.gz | tar xv
# cd php-x.x.xx
# ./configure --with-mysql=/usr/local/mysql --with-apxs=/usr/local/apache/bin/apxs
# make
# make install
```

Copy the ini file to the proper directory:

```
# cp php.ini-dist /usr/local/lib/php.ini
```

Open httpd.conf in your text editor (probably located in /usr/local/apache/conf directory), and find a section that looks like the following:

```
# And for PHP 4.x, use:
#
#AddType application/x-httpd-php .php
```

```
#AddType application/x-httpd-php-source .phps
```

Just remove those #s before the AddType line so that it looks like:

```
# And for PHP 4.x, use:
#
AddType application/x-httpd-php .php .phtml
AddType application/x-httpd-php-source .phps
```

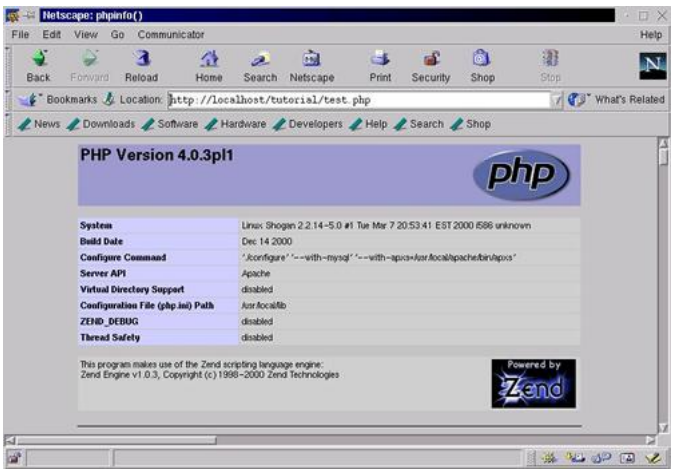
Save your file and restart apache:

```
# /usr/local/apache/bin/apachectl stop
# /usr/local/apache/bin/apachectl start
```

Then test whether you have PHP installed properly. Type the following code in a text editor and save it as test.php in a directory accessible by your Web server:

```
<HTML>
<?php
phpinfo();
?>
</HTML>
```

Set the permission of the file to executable by typing at console `chmod 775 test.php`, and then view it with your browser. You should see a detailed description of the environment variables in PHP similar to the image below. If you don't, then PHP was not installed properly. Try reinstalling it. Make sure there is a section "MySQL" in the php info; if not, MySQL connectivity will not work.



Section 3. Start coding

Your first script

Following tradition, we will begin coding with a "hello world" example. Fire up your text editor and type the following code:

```
<HTML>
<?php
echo "Hello World";
?>
</HTML>
```

Save the file as first.php and view it in the browser (remember to set the permission to `chmod 775` first). The page shows "Hello World". View the HTML source of this page through your browser. You will only see the text `Hello World`. This happened because PHP processed the code, and the code told PHP to output the string "Hello World". Notice the <?php and ?>. These are delimiters and enclose a block of PHP code. <?php tells PHP to process all the lines following this as PHP code and ?> tells PHP to stop processing. All lines beyond this scope are passed as HTML to the browser.

Your first database

Now that we have PHP running properly and have created our first script, let's create our first database and see what we can do with it. Drop to console and type in the following command:

`mysqladmin -uroot create learndb`
This creates a database named "learndb" for us to use. Here we have assumed that you are root user. If you are logged in as another user, just use the command `mysqladmin -uusername -pyourpassword create learndb`, replacing username and yourpassword with your username and password respectively. If you are hosting your site through a hosting company, you probably don't have permission to run `mysqladmin`. In this case, you have to ask your server administrator to create the database for you.

Next we will create tables in this database and enter some information. Go to the console. Type:

`mysql`
You should see something like:

```
Welcome to MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 5 to server version 3.22.34

Type 'help' for help.
```

Type:

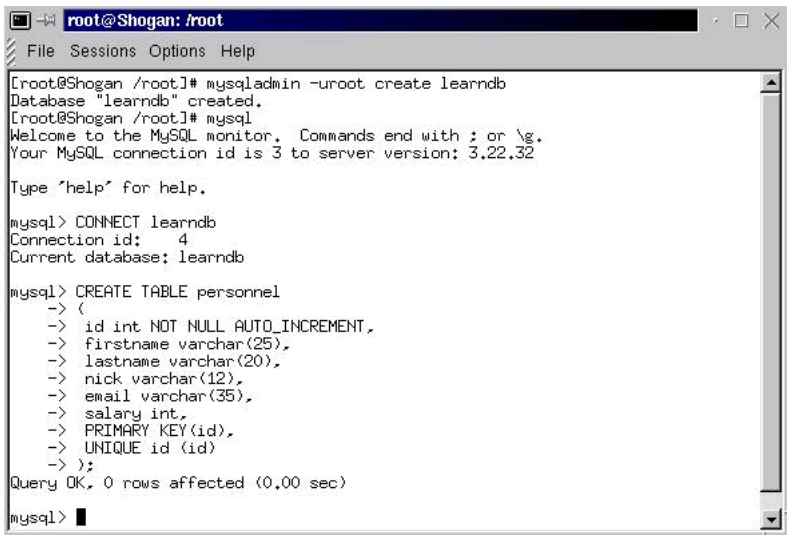
```
CONNECT learndb

CREATE TABLE personnel
(
  id int NOT NULL AUTO_INCREMENT,
  firstname varchar(25),
  lastname varchar(20),
```

```
  nick varchar(12),
  email varchar(35),
  salary int,
  PRIMARY KEY (id),
  UNIQUE id (id)
);

INSERT INTO personnel VALUES ('1','John','Lever','John', 'john@everywhere.net','75000');
INSERT INTO personnel VALUES ('2','Camilla','Anderson','Rose', 'rose@flower.com','66000');
```

This creates a table with 5 fields and puts some information in it.



Where's my view?

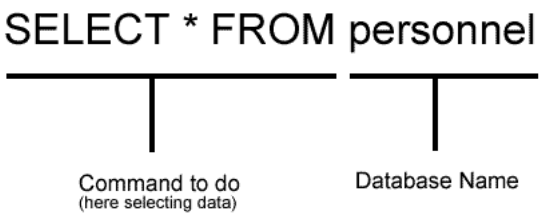
Now that we have a database with some information with it, let's see if we can view it with PHP. Save the following text as viewdb.php:

```
<HTML>
<?php
$db = mysql_connect("localhost", "root", "");
mysql_select_db("learndb",$db);
$result = mysql_query("SELECT * FROM personnel",$db);
echo "<TABLE>";
echo"<TR><TD><B>Full Name</B><TD><B>Nick Name</B><TD><B>Salary</B></TR>";
while ($myrow = mysql_fetch_array($result))
   {
   echo "<TR><TD>";
   echo $myrow["firstname"];
   echo " ";
   echo $myrow["lastname"];
   echo "<TD>";
   echo $myrow["nick"];
   echo "<TD>";
   echo $myrow["salary"];
   }
echo "</TABLE>";
?>
</HTML>
```

Run it through your browser and you will see a personnel database. But what is this code doing and how is it generated? Let's examine the code. First we declare a variable $db. In

PHP we declare a variable by putting the '$' sign before it. The string after $ is the name of that variable. We assign value to it by coding:`$variable_name=somevalue; (example: $count=4;)`
Remember to put ';' after all the lines that are executable in PHP. So we declare the variable $db and create a connection to the mysql database with the statement "mysql_connect("localhost", "root", "")". In plain English, it means connect to MySQL database in localhost server with the username root and password "". Replace them with your own username and password if they are different.

Then we assign a pointer to this database to $db; in other words, $db points to our database server `localhost`. Next we select the database with which we want to interact with the lines "mysql_select_db("learndb",$db);" which means we wish to use the database "learndb" located by the pointer variable $db. But we want information from the database, so we query the database with the lines "$result = mysql_query("SELECT * FROM personnel",$db);" The part "SELECT * FROM personnel" is an SQL statement (in case you don't know SQL), which means select all the stuff from the database personnel.



We run this query with the PHP command mysql_query() and save the result returned by the database to the variable $result. Now we can access the different data in the different rows of the database from the $result variable. We use the function mysql_fetch_array() to extract each row from $result and assign them to variable $myrow. So $myrow contains information about each row as opposed to all the rows in $result.

Then we output the data contained in each row. `"echo $myrow["firstname"];"` means send to output the value contained in the field "firstname" of the row contained in $myrow; in other words, we access different fields of the row with `$myrow["fieldname"]`.

We have used the while() loop here, which means as long as or while there are data to be extracted from $result, execute the lines within those brackets {}. Thus we get nicely formatted output in our browser. Viewing the PHP code and the HTML source from the browser side-by-side may help you easily understand the procedure. Congratulations! You have created your first dynamic page.

Section 4. Add new records

Creating an HTML form

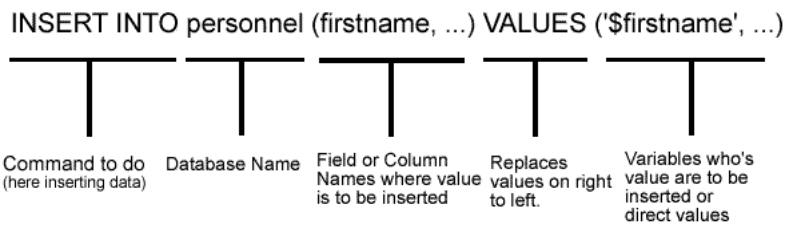
So now you can view records stored in your MySQL database and display them in your browser using PHP. But you want to add new record. Assuming that you know about HTML forms, let's code a page that will do just that. First we'll create a static form, datain.html:

```
<HTML>
<BODY>
<form method="post" action="datain.php">
First name:<input type="Text" name="first"><br>
Last name:<input type="Text" name="last"><br>
Nick Name:<input type="Text" name="nickname"><br>
E-mail:<input type="Text" name="email"><br>
Salary:<input type="Text" name="salary"><br>
<input type="Submit" name="submit" value="Enter information">
</form>
</HTML>
```

Now we have a form that will post the information to a page "datain.php". We must code this page so that it is able to process the posted data and send it to our MySQL database. The following listing of datain.php will do that:

```
<HTML>
<?php
$db = mysql_connect("localhost", "root","");
mysql_select_db("learndb",$db);
$sql = "INSERT INTO personnel (firstname, lastname, nick, email, salary) VALUES ('$first','$last','$nickname
$result = mysql_query($sql);
echo "Thank you! Information entered.\n";
?>
</HTML>
```

The first 3 lines are same as before, only we use the SQL command `"INSERT INTO"`, which means insert into the database into the columns specified (here firstname, lastname, nick, email) the data contained in the variable '$first','$last','$nickname','$email' respectively.



But where did these variables come from? Well, PHP has a wonderful way of creating the variables automatically from the data posted to it. So the text box with name "first" created the variable $first and it contained the text typed in that textbox.

Putting it together

Now let's merge the code into one file. We will call it input.php

```
<HTML>
<?php
if($submit)
        {
        $db = mysql_connect("localhost", "root","");
        mysql_select_db("learndb",$db);
        $sql = "INSERT INTO personnel (firstname, lastname, nick, email, salary) VALUES ('$first','$last','$
        $result = mysql_query($sql);
        echo "Thank you! Information entered.\n";
        }
else
        {
        ?>
        <form method="post" action="input.php">
        First name:<input type="Text" name="first"><br>
        Last name:<input type="Text" name="last"><br>
        Nick Name:<input type="Text" name="nickname"><br>
        E-mail:<input type="Text" name="email"><br>
        Salary:<input type="Text" name="salary"><br>
        <input type="Submit" name="submit" value="Enter information"></form>
        <?
        }
?>
</HTML>
```

This creates a script that shows the form when there is no input or otherwise enters the information into the database. How does the script understand when to do what? We have already learned that PHP automatically creates variable with information posted to it. So it will create the variable $submit if the form is posted. The script determines whether there exists the variable $submit. If it exists and contains a value then we have to enter the posted values into the database; otherwise, we have to show the form.

So now we can enter information to our database and view it. Try inserting some new data into the database and check to see if it really works by viewing them with viewdb.php.

Section 5. Get a better view

Passing variables

Let's take a different view now and consider how information can be passed to another PHP page. One method is by using forms as we have done already; another is by using query strings. What are query strings? Change the line `method="post"` to `method="get"` in our script input.php. Now try submitting new data into the database with it. After clicking submit you will see our familiar "Thank you! Information entered" in the browser. But look at the URL. It looks something like:

`http://yourhost/input.php?first=Rick&last=Denver&nickname=Mike&email=j@xyz.com&s`
Look closely. Now the information is passed as a string in the URL instead of posting directly. The sentence after the ? is the query string, and as you can see it contains the name of the variable and its values. When PHP receives a query string like ?first=John it automatically creates a variable named $first and assigns the value from the query string to it. So it is equivalent to $first="John"; When more than one variable is present, the variables are separated by an ampersand (&).

Viewing individual rows

So now we will create a script that will display the information of a particular row in our database defined by the variable $id. Save the following code as view.php. Try viewing it through your Web server as http://yourhost/view.php?id=2 (here we have passed the variable $id=2 through the query string). The page should show information corresponding to the id 2 in the MySQL database.

```
<HTML>
<?php
$db = mysql_connect("localhost", "root", "");
mysql_select_db("learndb",$db);
$result = mysql_query("SELECT * FROM personnel WHERE id=$id",$db);
$myrow = mysql_fetch_array($result);
echo "First Name: ".$myrow["firstname"];
echo "<br>Last Name: ".$myrow["lastname"];
echo "<br>Nick Name: ".$myrow["nick"];
echo "<br>Email address: ".$myrow["email"];
echo "<br>Salary: ".$myrow["salary"];
?>
</HTML>
```

Here the SQL command has changed and it tells the database to search for the row that has the value $id. But can't multiple rows contain the same values of id? Generally a column can contain any value, the same or different. But in our database two rows can never have the same value of id, as we have defined id as UNIQUE when we created our database.

We immediately modify our previous viewdb.php to viewdb2.php so that it can call view.php with the proper query string.

```
<HTML>
<?php
$db = mysql_connect("localhost", "root", "");
mysql_select_db("learndb",$db);
$result = mysql_query("SELECT * FROM personnel",$db);
echo "<TABLE BORDER=2>";
```

```
echo"<TR><TD><B>Full Name</B><TD><B>Nick Name</B><TD><B>Options</B></TR>";
while ($myrow = mysql_fetch_array($result))
        {
        echo "<TR><TD>".$myrow["firstname"]." ".$myrow["lastname"]."<TD>".$myrow["nick"];
        echo "<TD><a href=\"view.php?id=".$myrow[id]."\">View</a>";
        }
echo "</TABLE>";
?>
</HTML>
```

Viewing this page will show a list of names and corresponding nicknames. Look at the third column with a hyperlink view. Take your mouse over the hyperlink and look what it points to. The link should be something like http://yourhost/view.php?id=3 and the links in each row will be different. Click on one of the links. It will bring up our previously coded view.php showing the detailed information of that person. How is this achieved?

Let's take a look at our code viewdb2.php. Look at line 11, where all the real stuff takes place. The only unfamiliar thing here should be those odd dots (.) all around the line. The dot (.) is a concatenating operator in PHP, which means it concatenates the two strings on its two sides, which in turn means that if we write echo "Hello"."World", the output will actually be "HelloWorld". In our example we use the concatenate operator to generate a line like:

`<TR><TD>Camilla Anderson<TD>Rose<TD><a href="view.php?id=2">View</a>` for the browser.

Section 6. Delete, edit, and search data

Deleting rows

So far we have only entered new information in our database and viewed it. Where's the fun if we can't trash some of those data, at least the useless ones? Our delete.php will do just that. It works exactly like view.php. The only difference is the SQL command `"DELETE FROM personnel WHERE id=$id"`, which tell MySQL to delete the row that contains the id corresponding to the variable $id. Generally, the SQL command for deleting a row is `DELETE FROM database_name WHERE field_name=somevalue`

```
<HTML>
<?php
$db = mysql_connect("localhost", "root", "");
mysql_select_db("learndb",$db);
mysql_query("DELETE FROM personnel WHERE id=$id",$db);
echo "Information Deleted";
?>
</HTML>
```

Once again we modify our previous viewdb2.php script to viewdb3.php to add this new feature. The additions should be obvious.

```
<HTML>
<?php
$db = mysql_connect("localhost", "root", "");
mysql_select_db("learndb",$db);
$result = mysql_query("SELECT * FROM personnel",$db);
echo "<TABLE BORDER=2>";
echo"<TR><TD><B>Full Name</B><TD><B>Nick Name</B><TD><B>Options</B></TR>";
while ($myrow = mysql_fetch_array($result))
        {
        echo "<TR><TD>".$myrow["firstname"]."
".$myrow["nick"];
        echo "<TD><a href=\"view.php?id=".$myrow[id]."\">View</a>  ";
        echo "<a href=\"delete.php?id=".$myrow[id]."\">Delete</a>";
        }
echo "</TABLE>";
?>
</HTML>
```

Try clicking on delete and then view the database again with viewdb3.php to verify that the row was really deleted. You may have to refresh your browser.

Editing data

So far we have viewed and deleted database content. But sometimes we need to edit database content. For this we will modify our previously coded input.php file. By now you are familiar with the concept of passing variables by URL. We will call this modified script addedit.php:

```
<HTML>
<?php
if($submit)
        {
        $db = mysql_connect("localhost", "root","");
        mysql_select_db("learndb",$db);
```

```
        $sql = "INSERT INTO personnel (firstname, lastname, nick, email, salary)
        VALUES ('$first','$last','$nickname','$email','$salary')";
         $result = mysql_query($sql);
        echo "Thank you! Information entered.\n";
        }
 else if($update)
        {
        $db = mysql_connect("localhost", "root","");
        mysql_select_db("learndb",$db);
        $sql = "UPDATE personnel SET firstname='$first',lastname='$last',nick='$nickname',email='$email',
        salary='$salary' WHERE id=$id";
        $result = mysql_query($sql);
        echo "Thank you! Information updated.\n";
        }
 else if($id)
        {
        $db = mysql_connect("localhost", "root", "");
        mysql_select_db("learndb",$db);
        $result = mysql_query("SELECT * FROM personnel WHERE id=$id",$db);
        $myrow = mysql_fetch_array($result);
        ?>
        <form method="post" action="<?php echo $PHP_SELF?>">
        <input type="hidden" name="id" value="<?php echo $myrow["id"]?>">
        First name:<input type="Text" name="first" value="<?php echo
br>
        Last name:<input type="Text" name="last" value="<?php echo $myrow["lastname"]?>"><br>
        Nick Name:<input type="Text" name="nickname" value="<?php echo $myrow["nick"]?>"><br>
        E-mail:<input type="Text" name="email" value="<?php echo $myrow["email"]?>"><br>
        Salary:<input type="Text" name="salary" value="<?php echo $myrow["salary"]?>"><br>
        <input type="Submit" name="update" value="Update information"></form>
        <?

        }
 else
        {
        ?>
        <form method="post" action="<?php echo $PHP_SELF?>">
        First name:<input type="Text" name="first"><br>
        Last name:<input type="Text" name="last"><br>
        Nick Name:<input type="Text" name="nickname"><br>
        E-mail:<input type="Text" name="email"><br>
        Salary:<input type="Text" name="salary"><br>
        <input type="Submit" name="submit" value="Enter information"></form>
        <?
        }
 ?>
 </HTML>
```

Hmmm...the code looks quite complex. But really it isn't. Previously input.php had two features: it could add information to the database or could show the form. We'll add two more features to it: the ability to show the same form but with values of a particular person already there and the ability to update records for that person. The SQL commands for entering new information and updating existing information are different, so we can't use our previous code for entering information.

The script searches for the $submit variable. If it contains some value, then someone submitted new data and the information is entered into the database. If $submit does not contain any value, then someone might have just posted their updated information, so we check $update. If it contains a value, then we update that person's record with the SQL statement "UPDATE personnel SET fieldname1='$variablename1',fieldname2='$variablename2' ..... WHERE id=$id";". Otherwise, if someone provided the id in the query string, we show that person's information, but this time in a form so he may change it. If all these are not the case, we simply have to show the old form.

Experiment with the script. Open it with your browser to see what comes up. Then call it

providing query string ?id=1. Change the information and click update. Verify whether the database is updated by viewing the database with viewdb3.php.

Another new element was just introduced. It is the global PHP variable $PHP_SELF. This variable always contains the name of the script it is in and its location. We have used this variable in a '`form action`' so no matter what you name this file, this script will always post information to itself.

Once again we modify our viewing script incorporating this feature. Here's the listing for viewdb4.php:

```
<HTML>
<?php
$db = mysql_connect("localhost", "root", "");
mysql_select_db("learndb",$db);
$result = mysql_query("SELECT * FROM personnel",$db);
echo "<TABLE BORDER=2>";
echo"<TR><TD><B>Full Name</B><TD><B>Nick Name</B><TD><B>Options</B></TR>";
while ($myrow = mysql_fetch_array($result))
        {
        echo "<TR><TD>".$myrow["firstname"]." ".$myrow["lastname"]."</a><TD>".$myrow["nick"];
        echo "<TD><a href=\"view.php?id=".$myrow[id]."\">View</a>  ";
        echo "<a href=\"delete.php?id=".$myrow[id]."\">Delete</a>  ";
        echo "<a href=\"addedit.php?id=".$myrow[id]."\">Edit</a>";
        }
echo "</TABLE>";
?>
</HTML>
```

Searching our data

Information is useless if you can't find the data you require from a wealth of information. We need a way to search our database, so let's implement a search function. The page will show a static form initially and will show the search result when we have something submitted.

```
<HTML>
<?php
if ($searchstring)
        {
        $sql="SELECT * FROM personnel WHERE $searchtype LIKE '%$searchstring%' ORDER BY firstname ASC";
        $db = mysql_connect("localhost", "root", "");
        mysql_select_db("learndb",$db);
        $result = mysql_query($sql,$db);
        echo "<TABLE BORDER=2>";
        echo"<TR><TD><B>Full Name</B><TD><B>Nick Name</B><TD><B>Options</B></TR>";
        while ($myrow = mysql_fetch_array($result))
                {
                echo "<TR><TD>".$myrow["firstname"]."
".$myrow["nick"];
                echo "<TD><a href=\"view.php?id=".$myrow["id"]."\">View</a>";
                }
        echo "</TABLE>";
        }
else
        {
        ?>
        <form method="POST" action="<?php $PHP_SELF ?>">
        <table border="2" cellspacing="2">
        <tr><td>Insert you search string here</td>
        <td>Search type</td></tr>
        <tr>
        <td><input type="text" name="searchstring" size="28"></td>
        <td><select size="1" name="searchtype">
                <option selected value="firstname">First Name</option>
```

```
                <option value="lastname">Last Name</option>
                <option value="nick">Nick Name</option>
                <option value="email">Email</option>
        </select></td>
        </tr>
        </table>
        <p><input type="submit" value="Submit" name="B1"><input type="reset" value="Reset"
></p>
        </form>
        <?php
        }
 ?>
 </HTML>
```

The script checks whether a search string exists. If $searchstring contains a value, then we have something to search; otherwise, we just show an HTML form. The part of code that searches is similar to our viewdb2.php. The SQL command deserves a bit of explanation here. Let's look at it closely. The SQL command is:

```
"SELECT * FROM personnel WHERE $searchtype LIKE '%$searchstring%' ORDER BY firstname ASC"
```

Two news things are introduced here, "LIKE" and "ORDER BY". LIKE simply means 'sounds like'. The '%' sign represents any possible combination of characters (numbers or letters). So to find people whose first name starts with 'J' we would use the SQL command

```
"SELECT * FROM personnel WHERE firstname LIKE 'J%'"
```

To find those people with a name ending with J we have to use '%J'. If we wish find people with 'J' anywhere in their name (first, middle, or last) we have to use '%J%'.

'ORDER BY' simply orders the records in ascending or descending order. The syntax is: "ORDER BY fieldname order_method" where order_method is ASC or DESC allowing the ordering to be done in ASCending or DESCending order.

Section 7. Next steps: tips and resources

Tips for common tasks

We have covered the basics. Where you go from here is up to you. You know enough now to implement some of these useful tasks:

* **User database**
  You could implement a user database. You can add a login feature to this.

* **News**
  You could code a section that always displays the latest news or maybe a "What's new" section that's automatically generated. The TABLE could be something like:

```
CREATE TABLE news
(
id INT NOT NULL AUTO_INCREMENT,
title VARCHAR(40),
newsbody TEXT,
news_date DATE,
PRIMARY KEY (id),
UNIQUE id (id)
);
```

  And assuming you want to automatically show the title of the latest five news items, the code could be something like:

```
<HTML>
<?php
$sql="SELECT * FROM news ORDER by news_date DESC";
$db = mysql_connect("localhost", "root", "");
mysql_select_db("newsdb",$db);
$result = mysql_query($sql,$db);
echo "Latest News:<br>";
$i=1;
while ($myrow = mysql_fetch_array($result))
        {
        echo "<a
/a><br>";
        $i=$i+1;
        if($i>5)
                break;
        }
?>
</HTML>
```

* **Product database**
  You could create a detailed database of your products. Clients could see all the products or search for particular product.

Resources

You'll find useful information for further study at these sites:

* *PHP site*
  At this official PHP site, you will find PHP source as well as compiled binaries for both Linux and Windows. You will also find documentation and some useful links to various PHP sites, including a list of hosting providers that support PHP.

* *MySQL site*
  Here you'll find news, downloads, training information, documentation, and also job information.

* *Apache Software Foundation site*
  The Apache Software Foundation has created some of the best open source software projects. One of them is the Apache Web Server, which is currently the most popular Web server on the net.

* *AbriaSoft site*
  AbriaSoft specializes in the setup of a PHP, MySQL development environment. Their AbriaSQL Lite, which is free, is probably the easiest solution for installing Apache, PHP3, and MySQL.

* *PHPBuilder site*
  This is a must-visit resource site for PHP developers. You will find code, tips, discussion forums, news, jobs, links, and all sorts of useful stuff.

Your feedback

Please let us know whether this tutorial was helpful to you and how we could make it better. We'd also like to hear about other tutorial topics you'd like to see covered. Thanks!

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The Toot-O-Matic tool is a short Java program that uses XSLT stylesheets to convert the XML source into a number of HTML pages, a zip file, JPEG heading graphics, and PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML.