PHP/MySQL Tutorial

Part 1 – Introduction

1.1 Introduction

For many people, the main reson for learning a scripting language like PHP is because of the interaction with databases it can offer. In this tutorial I will show you how to use PHP and the MySQL database to store information on the web and include it into your website. Before you read this tutorial you should have at least a basic knowledge of how to use PHP. If you do not yet know PHP, I suggest that you read our PHP tutorial before continuing.

1.2 Why Would I Want A Database?

It is actually surprising how useful a database can be when used with a website. There are a huge variety of things you can do when you interact the two, from displaying simple lists to running a complete website from a database. Some examples of PHP and MySQL being used together are:

- Banner Rotation. On this site, where each banner is, a PHP script is called. This opens a database and picks a random banner from it to show the visitor. It also counts the number of times the banner has been viewed and could, with a few changes, track clicks too. To add, change or edit the banners all I have to do is change the database and the script will pick the correct banners for all the pages on the site.
- Forums. Hundreds of forums (message boards) on the internet are run using PHP and MySQL. These are much more efficent than other systems that create a page for each message and offer a wide variety of options. All the pages in the forum can be updated by changing one script.
- Databases. One quite obvious example is sites which get all there information from a database. For example Script Avenue is run by a few scripts, which gain all their information from a large database. All the different script categories can be accessed in one script by just changing the URL to access a different part of the database.
- Websites. If you have a large website and you want to change the design it can take a very long time to update and upload all the pages. With PHP and MySQL your whole website could be just one or two PHP scripts. These would access a MySQL database to get the information for the pages. To update the website's design you would just have to change one page.

1.3 What Do I Need To Begin?

You only really need three things to run PHP scripts which access MySQL databases. Firstly, you will, of course, need a webserver. This can either be on a computer of your own or on a web host. Any web server software should work with PHP and MySQL but the best to use is Apache, which is free.

PHP also needs to be installed on the server. If it is not already installed you can install it (or ask your web host to install it). It can be downloaded from PHP.net and is also free. If you are not sure if you have PHP installed I will show you a way to check it later.

Finally, you will also require MySQL. This is the actual database software. You can also use most other types of database (SQL, Oracle etc.) but as this is a PHP/MySQL tutorial I will deal just now with the MySQL database (although the commands used here will also work with SQL databases). As with the other software you need, MySQL is free and can be downloaded from the MySQL homepage. If you are not sure if you have MySQL installed, I will show you how to check later.

If you cannot install (or your web host won't allow) PHP and MySQL you can still use another web host. Freedom2Surf are a free (banner supported) web host and support PHP and have MySQL installed. HostRocket are an excellent web host and can offer you 300MB of space with PHP, MySQL and loads of other extras for under $10 a month.

1.4 Testing For PHP and MySQL

There is a simple test for both PHP and MySQL. Open a text editor and type in the following:

```
<?
phpinfo();
?>
```

and save it as phpinfo.php

Now upload this to your webspace and go to it in your browser. If you have PHP installed you will see a huge page with all the details of your PHP installation on it. Next, scroll down through all this information. If you find a section about MySQL then you will know that MySQL is installed.

1.5 Managing Databases

Although all the database administrative options can be done through PHP scripts, I strongly suggest installing a copy of PHPMyAdmin on your server. It is an excellent free set of scripts that will provide you with an administrative interface for your MySQL database(s). You can add, remove, edit, backup and view your databases using this and it is especially useful when troubleshooting your databases.

1.6 What You Will Learn in Doing This Tutorial

This tutorial will showing you some of the basics of using PHP and MySQL together, using the same example all the way through. As you use this tutorial, you will learn:

- How to create a web based contact management program. It will allow you to store names with their addresses, e-mail and phone numbers.
- How to update records and search the database.
- How to send an e-mail out to some or all the people in the database
- To be able to go on and create nearly any type of database enabled site you want to.

Part 2 - Setting Up The Database

2.1 Introduction

Before you actually start building your database scripts, you must have a database to place information into and read it from. In this section I will show you how to create a database in MySQL and prepare it for the data. I will also begin to show you how to create the contacts management database.

2.2 Database Construction

MySQL databases have a standard setup. They are made up of a database, in which is contained tables. Each of these tables is quite separate and can have different fields etc. even though it is part of one database. Each table contains records which are made up of fields.

2.3 Databases And Logins

The process of setting up a MySQL database varies from host to host, you will however end up with a database name, a user name and a password. This information will be required to log in to the database.

If you have PHPMyAdmin (or a similar program) installed you can just go to it to log in with your user name and password. If not you must do all your database administration using PHP scripts.

2.4 Creating Your First Table

Before you can do anything with your database, you must create a table. A table is a section of the database for storing related information. In a table you will set up the different fields which will be used in that table. Because of this construction, nearly all of a site's database needs can be satisfied using just one database.

Creating a table in PHPMyAdmin is simple, just type the name, select the number of fields and click the button. You will then be taken to a setup screen where you must create the fields for the database. If you are using a PHP script to create your database, the whole creation and setup will be done in one command.

2.5 Building Your First Fields With MYSQL

There are a wide variety of fields and attributes available in MySQL. We will go over just a few of these here:

Field Type	Description
TINYINT	Small Integer Number
SMALLINT	Small Integer Number
MEDIUMINT	Integer Number
INT	Integer Number
VARCHAR	Text (maximum 256 characters)
TEXT	Text

Table 1: A Few of the MYSQL Field Types

These are just a few of the fields which are available. A search on the internet will provide lists of all the field types allowed.

2.6 Creating A Table With PHP

To create a table in PHP is slightly more difficult than with MySQL. It takes the following format:

```
CREATE TABLE tablename {
Fields
}
```

The fields are defined as follows:

fieldname type(length) extra info,

The final field entered should not have a comma after it. We will have a full example of using these later in the section.

2.7 The Contacts Database

The contacts database will contain all the conact information for the people you enter and the information will be able to be edited and viewed on the internet. The following fields will be used in the database:

Name	Type	Length	Description
id	INT	6	A unique identifier for each record
first	VARCHAR	15	The person's first name
last	VARCHAR	15	The person's last name
phone	VARCHAR	20	The person's phone number
mobile	VARCHAR	20	The person's mobile number
fax	VARCHAR	20	The person's fax number
email	VARCHAR	30	The person's e-mail address
web	VARCHAR	30	The person's web address

Table 2: MYSQL 'contacts' table field definitions

We used VARCHAR fields for the phone/fax numbers even though they are made up of digits. You could use INT fields but using VARCHAR it will allow dashes and spaces in the number, as well as textual numbers (like 1-800-COMPANY). Since we will not be initiating phone calls from the web it is not a problem.
The id field will also be set as PRIMARY, INDEX, UNIQUE and will be set to auto_increment (found under Extra in PHPMyAdmin). The reason for this is that this will be the field identifier (primary and index) and so must be unique. The auto increment setting means that whenever you add a record, as long as you don't specify an id, it will be given the next number.

If you are using PHPMyAdmin or a management program you can now create this in a table called contacts.

2.8 Creating The Contacts Table In PHP

The following code should be used to create this table in PHP. Some of the code has not been covered yet but will be explained as we go through this tutorial.

```
<?
$user="username";
$password="password";
$database="database";
mysql_connect(localhost,$user,$password);
@mysql_select_db($database) or die( "Unable to select database");
$query="CREATE TABLE contacts (id int(6) NOT NULL auto_increment,first
varchar(15) NOT NULL,last varchar(15) NOT NULL,phone varchar(20) NOT
NULL,mobile varchar(20) NOT NULL,fax varchar(20) NOT NULL,email
varchar(30) NOT NULL,web varchar(30) NOT NULL,PRIMARY KEY
(id),UNIQUE id (id),KEY id_2 (id))";
mysql_query($query);
mysql_close();
?>
```

Enter your database, MySQL username and MySQL password in the appropriate positions on the first three lines above.

Part 3 - Inserting Information

3.1 Introduction

So far, we have covered what you need to get started with PHP and MYSQL and have created a “contacts” table with MYSQL. Now that we have a table, we will next insert some information into your database so that it is more useful.

3.2 Connecting To The Database

The first thing you must do before you can do any work at all is to connect to the MySQL database. This is an extremely important step as, if you are not connected, your commands to the database will fail.

Good practice for using databases is to specify the username, password and database name first so that if you change any of them at a later date you will only have to change one line:

```
$username="username";
$password="password";
$database="your_database";
```

At this point you may be wondering if it is a security risk, keeping your password in the file. You don't need to worry, though, because the PHP scource code is processed by the server before being sent to the browser so it is impossible for the user to see the script's source.

Next, you will need to issue the command to start a database connection:

```
mysql_connect(localhost,$username,$password);
```

This line tells PHP to connect to the MySQL database server at 'localhost' (localhost means the server that the site is running one. Unless you web host tells you otherwise you should use localhost. If you are given a server address (such as sql.myserver.com you should replace localhost with "sql.myserver.com" (including the quotes)) using the username stored in $username and the password in $password.

Before I show you how to work with the database, I will show you one more command:

```
mysql_close();
```

This command closes the connection to the database server. Your script will still run if you do not include this command but too many open MySQL connections can cause problems for a web host. It is good practice to always include this line once you have issued all your commands to the database, to keep the server running well.

3.3 Selecting The Database

After you have connected to the database server you must then select the database you wish to use. This must be a database to which your username has access. The following command:

```
@mysql_select_db($database) or die( "Unable to select database");
```

is used to do this. This tells PHP to select the database stored in the variable $database (which you set earlier). If it cannot connect it will stop executing the script and output the text: Unable to select database

This extra 'or die' part is good to leave in as it provides a little error control but it is not essential.

3.4 Executing Commands

Now you have connected to the server and selected the database you want to work with you can begin executing commands on the server.

There are two ways of executing a command. One is to just enter the command in PHP. This way is used if there will be no results from the operation.

The other way is to define the command as a variable. This will set the variable with the results of the operation.

In this part of the tutorial we will use the first way as we are not expecting a response from the database. The command will look like this:

```
mysql_query($query);
```

The useful thing about using this form of the command is that you can just repeat the same command over and over again without learning new ones. All you need to do is to change the variable.

3.5 Inserting Data

We will now add our first information to the Contacts database:

First: John
Last: Smith
Phone: 1-212-013-4354
Mobile: 1-307-460-7754
Fax: 1-212-013-4557
E-mail: johnsmith@jjjsmith.com
Web: http://jjjsmith.com

This will all be put in with one command:

```
$query = "INSERT INTO contacts VALUES ('','John','Smith','1-212-013-4354','1-307-460-7754','1-307-460-754','johnsmith@jjjsmith.com','http://www.jjjsmith.com')";
```

This may look a little confusing at first so I will explain what it all means.

Firstly $query= is there because we are assigning this to the variable $query (see the section above). The next part:

```
INSERT INTO contacts VALUES
```

tells PHP to insert into the contacts table the values in the brackets which follow.

In the PHP command to create the table, $query="CREATE TABLE contacts (id int(6) NOT NULL auto_increment, we are using the first field as an “id” or “index” field. No two records in the database will have the same ID. This is because we set up the database we set ID to 'auto_increment'. This means that if you assign it no value it will take the next number in the series. This means that this first record will have an ID value of 1.

The part in the brackets contains all the information to add. It uses all the fields in order and inserts the information from between the quotes. In the command, VALUES ('','John','Smith', the value John will be inserted into the 2nd field, and Smith will be inserted into the third field of the contacts table. Similarly, the 3 phone values, the email address, and the web site address will be inserted into the remaining fields.

Part 4 - Displaying Data

4.1 Introduction

So far in this tutorial, you have created a database and put information into it. In this part, we will create an input page for your database, and enable the display of the whole contents of the database.

4.2 HTML Input – A simple Form and Using Variables

Inputting the data using HTML pages is almost identical to inserting it using a PHP script. The benefit, though, is that you do not need to change the script for each piece of data you want to input and you can also allow your users to input their own data. The following form code will show an HTML page with textboxes to enter the appropriate details. You enter the form in your HTML code and when the form executes, it presents a box with field names and places to enter each of the data items you want to enter.

The first part of each line, such as “First Name:” is what the form will show you as a field name. Then on the same line, the item name=”first” defines the database variable which will hold the information you entered on the form.

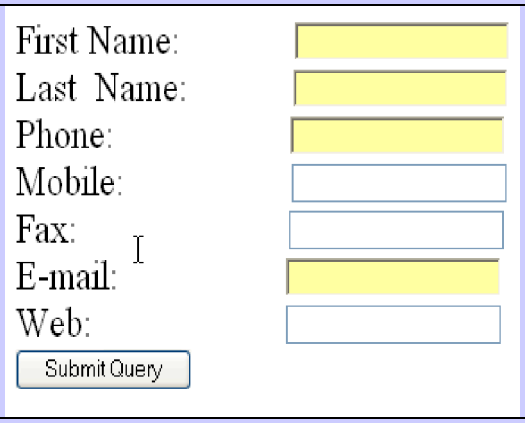
Form Code for “insert.php”	What the Form Looks Like
<form action="insert.php" method="post"> First Name: <input type="text" name="first"> Last Name: <input type="text" name="last"> Phone: <input type="text" name="phone"> Mobile: <input type="text" name="mobile"> Fax: <input type="text" name="fax"> E-mail: <input type="text" name="email"> Web: <input type="text" name="web"> <input type="Submit"> </form>	

Table 3: Code for “insert.php” Code and Associated Form

This page could, of course, be formatted and have other changes made to it. It is just a basic form to get you started. Once the form has been entered, your web browser will display a screen similar to the following, so that you can enter data.

Next, we edit edit the script to directly enter information, so that we can enter it using the form. Here, we again use the variables defined in the contacts table.

```
<?
$username="username";
$password="password";
$database="your_database";
$first=$_POST['first'];
$last=$_POST['last'];
$phone=$_POST['phone'];
$mobile=$_POST['mobile'];
$fax=$_POST['fax'];
$email=$_POST['email'];
$web=$_POST['web'];
mysql_connect(localhost,$username,$password);
@mysql_select_db($database) or die( "Unable to select database");
$query = "INSERT INTO contacts VALUES
('','$first','$last','$phone','$mobile','$fax','$email','$web')";
mysql_query($query);
mysql_close();
?>
```

This script should then be saved as insert.php so that it can be called by the HTML form. Instead of entering the data being locally, as we did in Section 3.5, it is being entered into the form and stored in variables which are then passed to PHP for processing, which will then enter the values into the contacts table.

You could also add to this script a message confirming the data input. This is basic PHP, though, and you should read our PHP tutorial if you do not know how to do this.

4.3 Outputting Data

Now you have at least one record, if not many more, in your database you will be wanting to know how you can output this data using PHP. Before beginning, though you should be familiar with loops in PHP (you can find out about them in the tutorial on Free Webmaster Help) as they are used for this way of outputting data.

The first command you will need to use is a MySQL query made up like this:

```
SELECT * FROM contacts
```

This is a basic MySQL command which will tell the script to select all the records in the contacts table. Because there will be output from this command it must be executed with the results being assigned to a variable:

```
$query="SELECT * FROM contacts";
$result=mysql_query($query);
```

In this case the whole contents of the database are now contained in a special array with the name $result. Before you can output this data you must change each piece into a separate variable. There are two stages to this.

Counting Rows

Before you can go through the data in your result variable, you must know how many database rows there are. You could, of course, just type this into your code but it is not a very good solution as the whole script would need to be changed every time a new row was added. Instead you can use the command:

```
$num=mysql_numrows($result);
```

This will set the value of $num to be the number of rows stored in $result (the output you got from the database). This can then be used in a loop to get all the data and output it on the screen.

Setting Up The Loop

You must now set up a loop to take each row of the result and print out the data held there. By using $num, which you created above, you can loop through all the rows quite easily. In the code below, $i is the number of times the loop has run and is used to make sure the loop stops at the end of the results so there are no errors.

PHP Code	Explanation of this Code
$i=0; while ($i < $num) {	Setup the variable $i and the WHILE loop
CODE;	Put your code in
$i++; }	Increment $i then end of loop

Assigning The Data To Variables

The final part of this output script is to assign each piece of data to its own variable. The following code is used to do this:

```
$variable=mysql_result($result,$i,"fieldname");
```

So to take each individual piece of data in our database we would use the following:

```
$first=mysql_result($result,$i,"first");
$last=mysql_result($result,$i,"last");
$phone=mysql_result($result,$i,"phone");
$mobile=mysql_result($result,$i,"mobile");
$fax=mysql_result($result,$i,"fax");
$email=mysql_result($result,$i,"email");
$web=mysql_result($result,$i,"web");
```

We do not need to get the ID field (although we could have done) because we have no use for it in the current output page.

Combining The Script

We can now write a full script to output the data. In this script the data is not formatted when it is output:

```
<?    //  Start PHP, define the user, password, and database to open

$username="username";
$password="password";
$database="your_database";
//  Connect to the database and setup the array

mysql_connect(localhost,$username,$password);
@mysql_select_db($database) or die( "Unable to select database");
$query="SELECT * FROM contacts";
$result=mysql_query($query);
$num=mysql_numrows($result);
mysql_close();
//  Now for the code to move through the database, record by record

echo "<b><center>Database Output</center></b><br><br>";
$i=0;
while ($i < $num) {
$first=mysql_result($result,$i,"first");
$last=mysql_result($result,$i,"last");
$phone=mysql_result($result,$i,"phone");
$mobile=mysql_result($result,$i,"mobile");
$fax=mysql_result($result,$i,"fax");
$email=mysql_result($result,$i,"email");
$web=mysql_result($result,$i,"web");
//  Now, output the record you have just extracted from the database
echo "<b>$first $last</b><br>Phone: $phone<br>Mobile: $mobile<br>Fax:
$fax<br>E-mail: $email<br>Web: $web<br><hr><br>";
$i++;  //increment $i
}   // end of loop
?>  // shut down php
```

Part 5 - More Outputs

5.1 Introduction

So far in this tutorial, we have created a database and a table, inserted information and displayed the database information. Now we will show more ways to display and output information in the database.

5.2 Formatting Output

In the last part of the tutorial we output a list of all the people stored in the database. This basic output is not particularly useful for a working website. Instead, it would be better if we could format it into a table and display it like this.

Doing this formatting is not particularly complicated. All you need to do is use PHP to output HTML and include your variables in the correct spaces. The easiest way to do this is by closing your PHP tag and entering the HTML normally. When you reach a variable position, include it as follows: `<? echo $variablename; ?>` in the correct position in your code.
You can also use the PHP loop to repeat the appropriate code and include it as part of a larger table. For example, using a section of the code from part 4 which looped to output the database you can format it to display it in one large table:

```
<table border="0" cellspacing="2" cellpadding="2">
<tr>
<th><font face="Arial, Helvetica, sans-serif">Name</font></th>
<th><font face="Arial, Helvetica, sans-serif">Phone</font></th>
<th><font face="Arial, Helvetica, sans-serif">Mobile</font></th>
<th><font face="Arial, Helvetica, sans-serif">Fax</font></th>
<th><font face="Arial, Helvetica, sans-serif">E-mail</font></th>
<th><font face="Arial, Helvetica, sans-serif">Website</font></th>
</tr>
<?
$i=0;
while ($i < $num) {
$first=mysql_result($result,$i,"first");
$last=mysql_result($result,$i,"last");
$phone=mysql_result($result,$i,"phone");
$mobile=mysql_result($result,$i,"mobile");
$fax=mysql_result($result,$i,"fax");
$email=mysql_result($result,$i,"email");
$web=mysql_result($result,$i,"web");
?>
<tr>
<td><font face="Arial, Helvetica, sans-serif"><? echo $first." ".$last; ?></font></td>
<td><font face="Arial, Helvetica, sans-serif"><? echo $phone; ?></font></td>
<td><font face="Arial, Helvetica, sans-serif"><? echo $mobile; ?></font></td>
<td><font face="Arial, Helvetica, sans-serif"><? echo $fax; ?></font></td>
<td><font face="Arial, Helvetica, sans-serif"><a href="mailto:<? echo $email; ?>">E-
mail</a></font></td>
<td><font face="Arial, Helvetica, sans-serif"><a href="<? echo $web;
?>">Website</a></font></td>
</tr>
<?
$i++;
}
echo "</table>";
```

Table 4: A PHP Script for Outputting Information

This code will print out table headers, then add an extra row for each record in the database, formatting the data as it is output.

As long as you are familiar with PHP and HTML the code is probably pretty self explanatory but I will just point out the last two lines in the table, for example:

```
<a href="mailto:<? echo $email; ?>">E-mail</a>
```

This shows one of the useful features of using PHP to include MySQL data as you can use it to output parts of your code and make pages fully dynamic.

5.3 Selecting Pieces of Data

PHP can be used to select individual records, or records which match certain criteria. To display the whole table we used the query:

Select Statement for Database	Explanation
SELECT * FROM contacts	Select all the records
SELECT * FROM contacts WHERE first='john'	Select only those records whose first name field is john

Table 5: Selecting portions of the Database

As with other MySQL queries, it is almost like plain English. In the same way you could select records based on any field in the database. You can also select ones with more than one field by adding more:

```
field='value'
```

sections onto the query.

Although I won't go into great depth about it in this section, you can also use variables to give the database criteria. For example, if you had a search form you could get the last name people wanted to search for and store it in a variable called $searchlast. Then you could execute the following piece of code:

```
$query="SELECT * FROM contacts WHERE last='$searchlast'";
$result=mysql_query($query);
```

Please note that at the end of the first line there is a ' followed by a " before the semicolon.

5.4 Security

You must be very careful in using the technique given above. Without correct security measures, it would be very easy for someone to access data on your server, or even make changes to the database. This can occur if the user sets the variable to a value which edits the SQL string being generated in such a way that it can be used for their own purposes. Many websites which give full details by such queries. Go to Google and search for 'sql injection attack'.

This security hole is easy to plug with a bit of work. Always check input data for invalid chanracters and use PHP's built in functions to remove control characters and HTML code etc. Again, there are many websites which go into this in depth.

Part 6 - Single Records & Error Trapping

6.1 Introduction

In the past two parts of this tutorial, we learned to take data out of the database and display it on screen. In this part, we concentrate on selecting one piece of data, displaying it, and stopping errors from happening when you output data.

6.2 Error Trapping

By outputting all the information from the database, it is quite unlikely that there will be no data, but if you allow updating and deleting of records, it is certainly a possibility. After connecting to the database, and pulling the database into the variable $result, you can use the command `$num=mysql_numrows($result);` to determine the number of rows in the database table. We can use this value to make a simple error trap using an IF statement:

```
if ($num==0) {
echo "The database contains no contacts yet";
} else {
Output Loop
}
```

You can expand on this more by making it more user friendly (for example by providing a link to the Add Data page if no contacts exist).

6.3 Ordering Data

Not only can you output data based on the contents of a field, but you can also order the output based on a field (for example placing users in alphabetical order). By default, the output from your queries will be in order of the id field, going from 1 upwards. You can sort it on any field, though.
For example, a useful sort would be to place all the users in alphabetical order based on their last name. For those not familiar with standard databases, this would be in Ascending order as it goes from A to Z. (Ascending order is also for 1-10 etc. and descending order provides Z to A and 10-1). To do this you would use the following query:

```
SELECT * FROM contacts ORDER BY last ASC
```

You could also use DESC to order the data in Descending order.

6.4 More Uses Of mysql_numrows and Sorting

The value you have assigned to $num is very important for both error trapping and loops, but it also has many other uses. An example of this would be to print out only the last 5 records added to a database. Firstly, they would need to be placed into order based on the id field (as the one with the latest ID would have been added last. This would require them to be in Descending order.

Now you have them in order of newest to oldest but this does not restrict the script to only showing the first 5. To do this, you would need to set your loop to run to 5 instead of $num (as this would only run the loop 5 times so only 5 records would be output). Of course, before doing this, it would be important to check that $num was greater than 5, as if you ran the loop 5 times and there were only 3 rows you would get an error. This is easy to do though and the following code is an example of the sort of thing you would want to have:

```
if ($num>5) {
$to=5;
}else{
$to=$num;
}
$i=0;
while ($i < $to) {
REST OF CODE
```

This code would check if there were more than 5 rows in the database. If there were, the loop would be set to run 5 times. If there were less than 5 rows the loop would run the correct number of times to output the whole database.

The ID Field

If you remember back to creating the database for the contacts at the beginning of this tutorial, you will remember that we included a numerical field called id. This field was set as auto_increment as well as being the primary field. I have already explained how this field is unique for every single record in the database, but I will now take this a stage further by explaining how this can be used to select an individual record from a database.

Selecting A Single Record

At the end of the last part of this tutorial, we showed you how to select records from the database based on the contents of particular fields using:

```
SELECT * FROM contacts WHERE field='value'
```

Now, by using the unique ID field we can select any record from our database using:

```
SELECT * FROM contacts WHERE id='$id'
```

Where $id is a variable holding a number of a record. This may seem to be a little worthless as it is, but you can use this very effectively in a number of different ways. For example, if you wanted to have a dynamically generated site run through a database and a single PHP script, you could write the script to include the database data into the design. Then, using the id field, you could select each individual page and put it into the output. You can even use the page's URL to specify the record you want e.g.

```
http://www.yoursite.com/news/items.php?item=7393
```

And then have the PHP script look up the record with the id corresponding to $item, which in this case would be 7393.

Links For Single Records

Using this method of choosing a record using the URL to select the record can be expanded further by generating the URL dynamically. Even though this sounds a bit

complicated, it is not so. The “contacts” script we are writing will show you how to create an Update page where the user can update the contact details.

To do this, another column will be included in the output column, with an Update link in it. This update link will point to a page allowing the user to update the record. To select the record in this page, we will put: ?id=$id

By getting the id of the record along with the other information when we are outputting the information from the database, this code will create a link which has each record's ID number in it. Then, on the update page, there can be code to just select this item.

Part 7 - Updating & Deleting

7.1 Introduction

So far we have put information into your MySQL database, viewed the information in it, and selected which information you would like to view. Now we will show you how to do the two final actions, updating your database and deleting records from it.

7.2 Updating your Database

In Section 6, we saw how to create a link for each record to point to your update script. By using the $id variable you output links which would pass the correct ID to the script so that it can update the database. Using this you can then create the update script, which will actually have two sections to it.

Displaying The Update Page

The first part of the update script uses the single record selection from Section 6, but adds a little HTML to it to make it more useful. First of all, we connect to the database and select the appropriate record.

```
$id=$_GET['id'];
$username="username";
$password="password";
$database="your_database";
mysql_connect(localhost,$username,$password);

$query=" SELECT * FROM contacts WHERE id='$id'";
```

```
$result=mysql_query($query);
$num=mysql_numrows($result);
mysql_close();

$i=0;
while ($i < $num) {
$first=mysql_result($result,$i,"first");
$last=mysql_result($result,$i,"last");
$phone=mysql_result($result,$i,"phone");
$mobile=mysql_result($result,$i,"mobile");
$fax=mysql_result($result,$i,"fax");
$email=mysql_result($result,$i,"email");
$web=mysql_result($result,$i,"web");

Space For Code
++$i;
}
```

Where 'Space For Code' is in this script is where the code for the update page will go. This is, in fact, just HTML formatting for the output:

```
<form action="updated.php" method="post">
<input type="hidden" name="ud_id" value="<? echo $id; ?>">
First Name: <input type="text" name="ud_first" value="<? echo $first; ?>"><br>
Last Name: <input type="text" name="ud_last" value="<? echo $last; ?>"><br>
Phone Number: <input type="text" name="ud_phone" value="<? echo $phone;
?>"><br>
Mobile Number: <input type="text" name="ud_mobile" value="<? echo $mobile;
?>"><br>
Fax Number: <input type="text" name="ud_fax" value="<? echo $fax; ?>"><br>
E-mail Address: <input type="text" name="ud_email" value="<? echo $email;
?>"><br>
Web Address: <input type="text" name="ud_web" value="<? echo $web; ?>"><br>
<input type="Submit" value="Update">
</form>
```

This code will output a standard form, but instead of having blank boxes like on the form for inserting a new record, this one already has the current information from the database inserted into it. This makes it much more effective for an update script.

Updating The Database

The next stage of this script is to actually update the database. This is a simple operation and just involves a new query for the database:

```
$query = "UPDATE contacts SET first = '$ud_first', last = '$ud_last', phone = '$ud_phone', mobile = '$ud_mobile', fax = '$ud_fax', email = '$ud_email', web = '$ud_web' WHERE id = '$ud_id'";
```

This query tells the database to update the contacts table where the ID is the same as the value stored in $ud_id (which as you can see from the form on the previous page was set as the id of the record we are updating) and to set the following fields to the specified values (which were set using the form on the previous page).

This query could then be integrated into a simple script:

```
$ud_id=$_POST['ud_id'];
$ud_first=$_POST['ud_first'];
$ud_last=$_POST['ud_last'];
$ud_phone=$_POST['ud_phone'];
$ud_mobile=$_POST['ud_mobile'];
$ud_fax=$_POST['ud_fax'];
$ud_email=$_POST['ud_email'];
$ud_web=$_POST['ud_web'];

$username="username";
$password="password";
$database="your_database";
mysql_connect(localhost,$username,$password);
$query="UPDATE contacts SET first='$ud_first', last='$ud_last',
phone='$ud_phone', mobile='$ud_mobile', fax='$ud_fax', email='$ud_email',
web='$ud_web' WHERE id='$ud_id'";
mysql_query($query);
echo "Record Updated";
mysql_close();
```

This code would update the database and give the user a confirmation.

Deleting Records

The final part of the contacts database which needs to be created is a page to delete records. As with the Update page, this should have a record ID sent to it in the URL e.g.: delete.php?id=9
The code to do this is the same as to update the database, except with a slightly different MySQL query. Instead of the UPDATE query you should use:

```
DELETE FROM contacts WHERE id='$id'
```

This would then be used with the connection and confirmation code as above.

Loops

Besides using a loop to get information from a database, you can also use loops to execute queries. For example, if you wanted to change all the records in the database with the last name Smith to have the website www.smith.com:

Standard Database Connection Code

```
$query=" SELECT * FROM contacts WHERE last='Smith'";
$result=mysql_query($query);
$num=mysql_numrows($result);
$i=0;
while ($i < $num) {
$id=mysql_result($result,$i,"id");
$query1="UPDATE contacts SET web='http://www.smith.com'
WHERE id='$id'";
mysql_query($query);
++$i;
}
mysql_close();
```

Of course, this could have been archived far easier and quicker using:

```
$query1="UPDATE contacts SET web='http://www.smith.com'
WHERE last='Smith'";
```

and no loop.

Part 8 - Finishing The Script

8.1 Introduction

Throughout this tutorial, we have showed you how to use PHP to interact with a MySQL database and how to use the most common commands available. We have also shown you how to create a basic contacts management system to illustrate some of the options you can use. We will now show you some final MySQL tips and will give you a final version of the script.

8.2 Saving Time

When creating complex scripts using databases you will find that the most common thing you are doing is connecting to a database. Because of this, you can actually save time by creating either a username/password file or a connection file. For example, for a username/password file you would create a file called:

```
dbinfo.inc.php
```

and put the following in it:

```
<?
$username="databaseusername";
$password="databasepassword";
$database="databasename";
?>
```

Replacing the appropriate sections. Then in your php files use the following code:

```
include("dbinfo.inc.php");
```

or

```
include("/full/path/to/file/dbinfo.inc.php");
```

at the beginning. Then, you can use the variables $username, $password and $database throughout your scripts without having to define them every time. Also, if you ever change this information, for example if you move to another web host, there is only one file to change.

You can use the same principle to connect to the database, by putting the connection code in the file, but you must always be sure to close the connection in each file or you may have problems with your MySQL server.

8.3 Searching

A limited form of searching can also be performed on your database using a built in MySQL function. This is by using the LIKE function as follows:

```
SELECT * FROM tablename WHERE fieldname LIKE '%$string%'
```

To explain further, LIKE tells the database to perform its 'searching' feature. The % signs mean that any other data could appear in their place and $string would hold your search string. In this place could be a word or number as well e.g.:

```
LIKE '%piano%'
```

which would output any rows with piano in the specified field.

Similarly, you can leave out one of the % signs so that you can specify the position of the string e.g.:

```
LIKE 'piano%'
```

Will only output rows where the specified field begins with piano, so:

"The piano is next to the table." would not show up.

8.4 Conclusion

From this tutorial you should now know the basics of using PHP and MySQL together to create database-enabled websites and programs. Using databases with the web opens up a huge new selection of things you can do and can make a simple website much more powerful, saving time updating the site, allowing user interaction and feedback and much more.