PHP/MySQL Lab 1 – The obligatory "Hello World!" (And something to do for a further 119 minutes…)

Introduction

The purpose of the next 4 labs is to create a working web application to interact with the database you have designed for this module.
You will be developing this web application using PHP to select, add, edit and delete entries as well carry out some processing on the data as it comes and goes from the database.
Each lab session will last for two hours and during this you can work at your own pace and ask questions as required.
It is assumed you will be using the student web server for these labs, but you can use your own server as long as it supports PHP and MySQL.
Details for accessing the student webserver can be found at http://mayar.abertay.ac.uk – Filezilla

Task 1- Obligatory Hello World.

Unlike running an application on your PC the PHP script is executed on the server, and the plain HTML result is sent back to the browser.

Syntax

We talk about PHP's syntax, let us first define what syntax is referring to.
* Syntax - The rules that must be followed to write properly structured code.

PHP's syntax and semantics are similar to most other programming languages (C, Java, Perl) with the addition that all PHP code is contained with a tag, of sorts, since it is designed to be embedded within HTML pages. All PHP code must be contained within the following:

```
<?php
?>
```

There is also a shorthand version of this PHP tag, which requires shorthand support to be enabled on your server:

```
<?
?>
```

When writing PHP, we suggest that you use the standard form (which includes the ?php) rather than the shorthand form. This will ensure that your scripts will work on any server with PHP support, whatever the settings.

How to save your PHP pages

If you have PHP inserted into your HTML and want the web browser to interpret it correctly, then you must save the file with a .php extension – if your file has any other extension, it will probably be sent to the browser as-is, without running any of your commands!

Example - simple HTML & PHP page

Below is an example of one of the easiest PHP and HTML page that you can create and still follow web standards.

PHP and HTML Code:

```
<html>
<head>
<title>My First PHP Page</title>
</head>
<body>
<?php
echo "Hello World!";
?>
</body>
</html>
```

Display:

Hello World!

If you save this file (e.g. helloworld.php) and place it on PHP enabled server and load it up in your web browser, then you should see "Hello World!" displayed. If not, please check that you followed our example correctly.

We used the PHP command echo to write "Hello World!" and we will be talking in greater depth about how echo is special later on in this tutorial.

The semicolon!

As you may or may not have noticed in the above example, there was a semicolon after the line of PHP code. The semicolon signifies the end of a PHP statement and should never be forgotten. For example, if we repeated our "Hello World!" code several times, then we would need to place a semicolon at the end of each statement.

PHP and HTML Code:

```
<html>
<head>
<title>My First PHP Page</title>
</head>
<body>
<?php
echo "Hello World! ";
echo "Hello World! ";
echo "Hello World! ";
echo "Hello World! ";
echo "Hello World! ";
?>
</body>
</html>
```

Display:

Hello World! Hello World! Hello World! Hello World! Hello World!

White space

As with HTML, whitespace is ignored between PHP statements. This means it is OK to have one line of PHP code, then 20 lines of blank space before the next line of PHP code. You can also press tab to indent your code and the PHP interpreter will ignore those spaces as well.

PHP and HTML Code:

```
<html>
<head>
<title>My First PHP Page</title>
</head>
<body>
<?php
echo "Hello World!";


        echo "Hello World!";


?>
</body>
</html>
```

Display:

Hello World!Hello World!

Variables

If you have never had any programming, algebra, or scripting experience, then the concept of *variables* might be a new concept to you. A detailed explanation of variables is beyond the scope of this tutorial, but we've included a refresher crash course to guide you.

A variable is a means of storing a value, such as text string "Hello World!" or the integer value 4. A variable can then be reused throughout your code, instead of having to type out the actual value over and over again. In PHP you define a variable with the following form:

- $variable_name = Value;

If you forget that dollar sign at the beginning, it will not work. This is a common mistake for new PHP programmers!

Note: Also, variable names are case-sensitive, so use the exact same capitalization when using a variable. The variables **$a_number** and **$A_number** are different variables in PHP's eyes.

A quick variable example

Say that we wanted to store the values that we talked about in the above paragraph. How would we go about doing this? We would first want to make a variable name and then set that equal to the value we want. See our example below for the correct way to do this.

PHP Code:

```
<?php
$hello = "Hello World!";
$a_number = 4;
$anotherNumber = 8;
?>
```

Note for programmers: PHP does not require variables to be declared before being initialized.

Variable naming conventions

There are a few rules that you need to follow when choosing a name for your PHP variables.

- PHP variables must start with a letter or underscore "_".
- PHP variables may only be comprised of alpha-numeric characters and underscores. a-z, A-Z, 0-9, or _ .
- Variables with more than one word should be separated with underscores. $my_variable
- Variables with more than one word can also be distinguished with capitalization. $myVariable

This lab assumes you have a basic grasp of programming conventions and have an ability to learn for yourself. Good sources of reference material are:

- http://www.w3schools.com/php
- http://www.tizag.com/phpT/

The remainder of this lab should be spent carrying out the following task to connect to your MySQL Database

Task 2 – Connecting to MySQL

Open a Connection to the MySQL Server

Before we can access data in a database, we must open a connection to the MySQL server. In PHP, this is done with the mysqli_connect() function.

Syntax

```
mysqli_connect(host,username,password,dbname);
```

Parameter	Description	If using Lochnagar
host	Optional. Either a host name or an IP address	lochnagar.abertay.ac.uk
username	Optional. The MySQL user name	sql123456789 (from e-mail)
password	Optional. The password to log in with	(from e-mail)
dbname	Optional. The default database to be used when performing queries	Sql123456789 (same as username)

Note: There are more available parameters, but the ones listed above are the most important. Although technically "optional", you need to specify all four to connect to Lochnagar, the Abertay database server – you'll be told the username and password to use in an e-mail by Abertay's IS department.

In the following example we store the connection in a variable ($con) for later use in the script:

```
<?php
// Create connection
$con=mysqli_connect("example.com","peter","abc123","
my_db");

// Check connection
if (mysqli_connect_errno($con))
  {
  echo "Failed to connect to MySQL: " .
mysqli_connect_error();
  }
?>
```

Close a Connection

The connection will be closed automatically when the script ends. To close the connection before, use the mysqli_close() function:

```
<?php
$con=mysqli_connect("example.com","peter","abc123","
my_db");

// Check connection
if (mysqli_connect_errno($con))
  {
  echo "Failed to connect to MySQL: " .
mysqli_connect_error();
  }

mysqli_close($con);
?>
```

Task 3 – Create a Table

The CREATE TABLE statement is used to create a table in MySQL.
We must add the CREATE TABLE statement to the mysqli_query() function to execute the command.
The following example creates a table named "Persons", with three columns: "FirstName", "LastName" and "Age":

```
<?php
$con=mysqli_connect("example.com","peter","abc123","
my_db");
// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " .
mysqli_connect_error();
  }

// Create table
$sql="CREATE TABLE Persons(FirstName
CHAR(30),LastName CHAR(30),Age INT)";

// Execute query
if (mysqli_query($con,$sql))
  {
  echo "Table persons created successfully";
  }
else
  {
  echo "Error creating table: " .
mysqli_error($con);
  }
?>
```

Note: When you create a field of type CHAR, you must specify the maximum length of the field, e.g. CHAR(50).
The data type specifies what type of data the column can hold.
If using Lochnagar, you'll be able to check the results of this via PHPMyAdmin: https://phpmyadmin.abertay.ac.uk/phpmyadmin/

Primary Keys and Auto Increment Fields

Each table in a database should have a primary key field.

A primary key is used to uniquely identify the rows in a table. Each primary key value must be unique within the table. Furthermore, the primary key field cannot be null because the database engine requires a value to locate the record.

The following example sets the PID field as the primary key field. The primary key field is often an ID number, and is often used with the AUTO_INCREMENT setting. AUTO_INCREMENT automatically increases the value of the field by 1 each time a new record is added. To ensure that the primary key field cannot be null, we must add the NOT NULL setting to the field:

```
$sql = "CREATE TABLE Persons
(
PID INT NOT NULL AUTO_INCREMENT,
PRIMARY KEY(PID),
FirstName CHAR(15),
LastName CHAR(15),
Age INT
)";
```

If you've already created the database, running CREATE TABLE a second time will give you an error message – you need to delete ("DROP" in SQL) the existing table before creating it with new contents.

Task 4 – Insert into a Database

The INSERT INTO statement is used to add new records to a database table.

Syntax

It is possible to write the INSERT INTO statement in two forms.

The first form doesn't specify the column names where the data will be inserted, only their values:

INSERT INTO table_name
VALUES (value1, value2, value3,...)

The second form specifies both the column names and the values to be inserted:

INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)

To get PHP to execute the statements above we must use the mysqli_query() function. This function is used to send a query or command to a MySQL connection.

Example

In the previous chapter we created a table named "Persons", with three columns; "FirstName", "LastName" and "Age". We will use the same table in this example. The following example adds two new records to the "Persons" table:

```
<?php
$con=mysqli_connect("example.com","peter","abc123","
my_db");
// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " .
mysqli_connect_error();
  }

mysqli_query($con,"INSERT INTO Persons (FirstName,
LastName, Age)
VALUES ('Peter', 'Griffin',35)");

mysqli_query($con,"INSERT INTO Persons (FirstName,
LastName, Age)
VALUES ('Glenn', 'Quagmire',33)");

mysqli_close($con);
?>
```

Task 5 - Insert Data From a Form Into a Database

Now we will create an HTML form that can be used to add new records to the "Persons" table.

Here is the HTML form:

```
<html>
<body>

<form action="insert.php" method="post">
Firstname: <input type="text" name="firstname">
Lastname: <input type="text" name="lastname">
Age: <input type="text" name="age">
<input type="submit">
</form>

</body>
</html>
```

When a user clicks the submit button in the HTML form, in the example above, the form data is sent to "insert.php".

The "insert.php" file connects to a database, and retrieves the values from the form with the PHP $_POST variables.

Then, the mysqli_query() function executes the INSERT INTO statement, and a new record will be added to the "Persons" table.

Here is the "insert.php" page:

```
<?php
$con=mysqli_connect("example.com","peter","abc123","
my_db");
// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " .
mysqli_connect_error();
  }

$sql="INSERT INTO Persons (FirstName, LastName, Age)
VALUES
('$_POST[firstname]','$_POST[lastname]','$_POST[age]
')";

if (!mysqli_query($con,$sql))
  {
  die('Error: ' . mysqli_error($con));
  }
echo "1 record added";

mysqli_close($con);
?>
```

Task 6 – Select Data From a Database Table

The SELECT statement is used to select data from a database.

Syntax

SELECT column_name(s)
FROM table_name

To get PHP to execute the statement above we must use the mysqli_query() function. This function is used to send a query or command to a MySQL connection.

Example

The following example selects all the data stored in the "Persons" table (The * character selects all the data in the table):

```
<?php
$con=mysqli_connect("example.com","peter","abc123","
my_db");
// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " .
mysqli_connect_error();
  }

$result = mysqli_query($con,"SELECT * FROM
Persons");

while($row = mysqli_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br>";
  }

mysqli_close($con);
?>
```

The example above stores the data returned by the mysqli_query() function in the $result variable.

Next, we use the mysqli_fetch_array() function to return the first row from the recordset as an array. Each call to mysqli_fetch_array() returns the next row in the recordset. The while loop loops through all the records in the recordset. To print the value of each row, we use the PHP $row variable ($row['FirstName'] and $row['LastName']).

The output of the code above will be:

```
Peter Griffin
Glenn Quagmire
```

Display the Result in an HTML Table

The following example selects the same data as the example above, but will display the data in an HTML table:

```
<?php
$con=mysqli_connect("example.com","peter","abc123","
my_db");
// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " .
mysqli_connect_error();
  }

$result = mysqli_query($con,"SELECT * FROM
Persons");

echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>";

while($row = mysqli_fetch_array($result))
  {
  echo "<tr>";
  echo "<td>" . $row['FirstName'] . "</td>";
  echo "<td>" . $row['LastName'] . "</td>";
  echo "</tr>";
  }
echo "</table>";

mysqli_close($con);
?>
```

The output of the code above will be:

Firstname	Lastname
Glenn	Quagmire
Peter	Griffin