Facilitator Meeting

Instructions 10 – PHP and MySQL Database Operations

Aims:

- The aim of this exercise is to practice the coding of SQL by creating and manipulating a database table on the MySQL database server **feenix-mariadb.swin.edu.au**
- Write PHP scripts to connect to the database, query tables, and display data in HTML

The tasks are due in two weeks time on the day of your scheduled facilitator meetings. Email the link of your web page running on the Mercury server to your facilitator before the due date to be marked off. Tasks will not be marked if the email is not received.

Task 1: Connecting to a database from PHP

In this task we will connect to a MySQL database from PHP and retrieve records from the table created in the previous tasks.

Step 1: Connection Info

First we will create a 'settings' file that stores the login information and can be reused whenever we want to connect to our database.

Create a file **settings.php** that contains variable declaration and initialisation of the database host, user name and password. It should not contain any HTML.

```
<?php
    $host = "feenix-mariadb.swin.edu.au";
    $user = "s1234567890"; // your user name
    $pwd = "ddmmyy"; // your password (date of birth ddmmyy unless changed)
    $sql_db = "s1234567890_db"; // your database
?>
```

Change these to your mariadb username, password, database

Step 2: Connection and displaying records in a table

Use the **@mysqli_connect** command to create a connection to the database from PHP using the parameters that were set in the **settings.php** file.

Note that the @ symbol is not part of the command.
It is an *operator* used to suppress any error messages that may be generated by mysqli_connect.

```
$conn = @mysqli_connect($host,
            $user,
            $pwd,
            $sql_db
        );
```

HUPD

As shown in the code on the following page, create a file **cars_display.php** that selects only make, model and price from the database table cars, created in earlier tasks, and then displays them neatly in a well coded html table. ***(The code for this is on the next page).***

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="description" content="Creating Web Applications Lab 10" />
<meta name="keywords" content="PHP, MySql" />
<title>Retrieving records to HTML</title>
</head>
<body>
<h1>Creating Web Applications - Lab10</h1>
<?php
     require_once ("settings.php");   //connection info

     $conn = @mysqli_connect($host,
        $user,
        $pwd,
        $sql_db
     );
     // Checks if connection is successful
     if (!$conn) {
        // Displays an error message
        echo "<p>Database connection failure</p>"; // not in production script
     } else {
        // Upon successful connection

        $sql_table="cars";

        // Set up the SQL command to query or add data into the table
        $query = "select make, model, price FROM cars ORDER BY make, model";

        // execute the query and store result into the result pointer
        $result = mysqli_query($conn, $query);

        // checks if the execution was successful
        if(!$result) {
           echo "<p>Something is wrong with ", $query, "</p>";
        } else {
           // Display the retrieved records
           echo "<table border=\"1\">\n";
           echo "<tr>\n "
                ."<th scope=\"col\">Make</th>\n "
                ."<th scope=\"col\">Model</th>\n "
                ."<th scope=\"col\">Price</th>\n "
                ."</tr>\n ";
           // retrieve current record pointed by the result pointer

           while ($row = mysqli_fetch_assoc($result)){
              echo "<tr>\n ";
              echo "<td>",$row["make"],"</td>\n ";
              echo "<td>",$row["model"],"</td>\n ";
              echo "<td>",$row["price"],"</td>\n ";
              echo "</tr>\n ";
        }
        echo "</table>\n ";
        // Frees up the memory, after using the result pointer
        mysqli_free_result($result);
        } // if successful query operation

        // close the database connection
        mysqli_close($conn);
     } // if successful database connection
?>
</body>
</html>
```

\n is optional.
Creates tidy served code.

Test in the browser. A table of cars should be displayed.

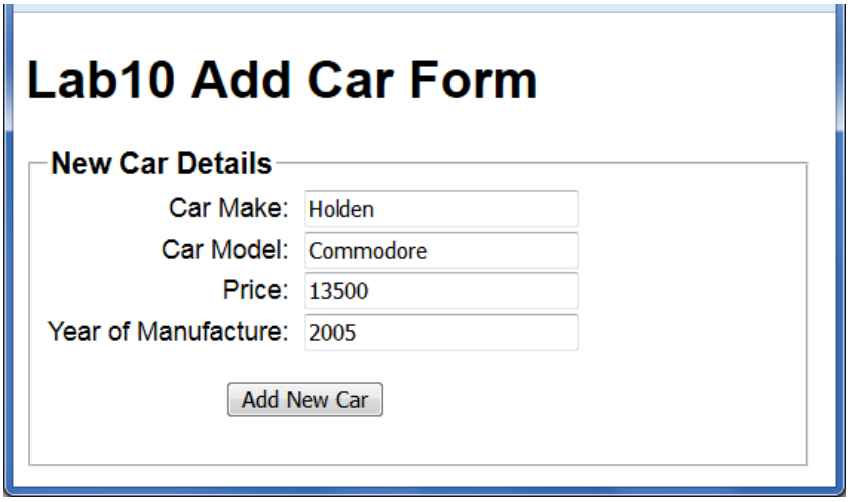
Check that the page is valid HTML5.

Task 2: Creating a form to post data to SQL queries

In this task we will create a form to add new records to the database table.

Step 1: Create a form to add records

Create an HTML page called **addcar.html** with a form that allows the user to enter data that will be sent to a server-side script that will then connect to the database and add records.
(A basic html and CSS are provided on Canvas)



Note: As discussed in the lecture we are *adding* information to the server so we will use HTTP **POST** method rather than **GET**

Have the form **post** name-value pairs to a file called **cars_add.php**

```
<form method="post" action="cars_add.php">
```

Step 2: Create a PHP file to talk to the database
Create the **cars_add.php** that creates a connection to the database. See Task 1 for a template. (Don't forget to close the connection when you have finished). This time if the connection is successful we will assign the values from the 'post' into the variables.

```
$make = trim($_POST["carmake"]);
$model      = trim($_POST["carmodel"]);
$price      = trim($_POST["price"]);
$yom  = trim($_POST["yom"]);
```

Then in the query, instead of making a SELECT query as we did in Task 1, we will create an INSERT SQL query.

```
$query = "insert into $sql_table (make, model, price, yom) values ('$make', '$model', '$price',
'$yom')";
// execute the query  -we should really check to see if the database exists first.
            $result = mysqli_query($conn, $query);
            // checks if the execution was successful
            if(!$result) {
                  echo "<p class=\"wrong\">Something is wrong with ",       $query, "</p>";
              //Would not show in a production script
            } else {
                  // display an operation successful message
                  echo "<p class=\"ok\">Successfully added New Car record</p>";
            } // if successful query operation

            // close the database connection
            mysqli_close($conn);
      }  // if successful database connection
```

Step 3: Deploy and Test

Load to Mercury and test in the browser. Use MySQL Monitor command line or the phpMyAdmin interface to check that the new record was added to the cars table.

Run **cars_display.php** to check that all the records are still there. ☺

Check that the delivered page is valid HTML5.
Note that there are alternative outputs to check.

Step 4: Add some 'security'

Note: Now that we are using a form, we have created a user interface to our php code and to our mysql database. Never trust a user – NEVER. At present we have no 'checks' on the data that users can enter. For example, users could put html code in a form input, or even worse, users could put php or sql into a form input.

We do not want to be distracted *here* by writing php server-side data checking, but we can use some easy 'blockers' by simply converting any special characters entered
by using the php function **htmlspecialchars(**$_POST["name"]**)**
or within mysqli using **mysqli_escape_string($conn,** $_POST["name"]**)**

Task 3: Create a search form.

In this task, using Task 2 as a model, create a form to search for a car, based on criteria entered by a user, and then display the results in a table, as done in Task 1.

Step 1: Create a form to search for records

Using Task 2 as a model, create an HTML page called **searchcar.html** with a form that allow users to enter search values, that will be sent to a new php script **cars_search.php** that will connect to the database, query the table for records, and display any records found, or a message if none are found.

Step 2: Create the PHP script that will action the search

Using Task 1 and 2 as a model, create the **cars_search.php**
Note for the query use ‘like’ and add the wildcard character ‘%’
eg. **select * from cars where make like 'hol%'**
Remember that you can test your SQL either using MySQL Monitor, or phpMyAdmin SQL interface.

Step 3: Deploy and Test

Load to Mercury and test in the browser.
Check that the delivered page is valid HTML5.
Check all alternative outputs.