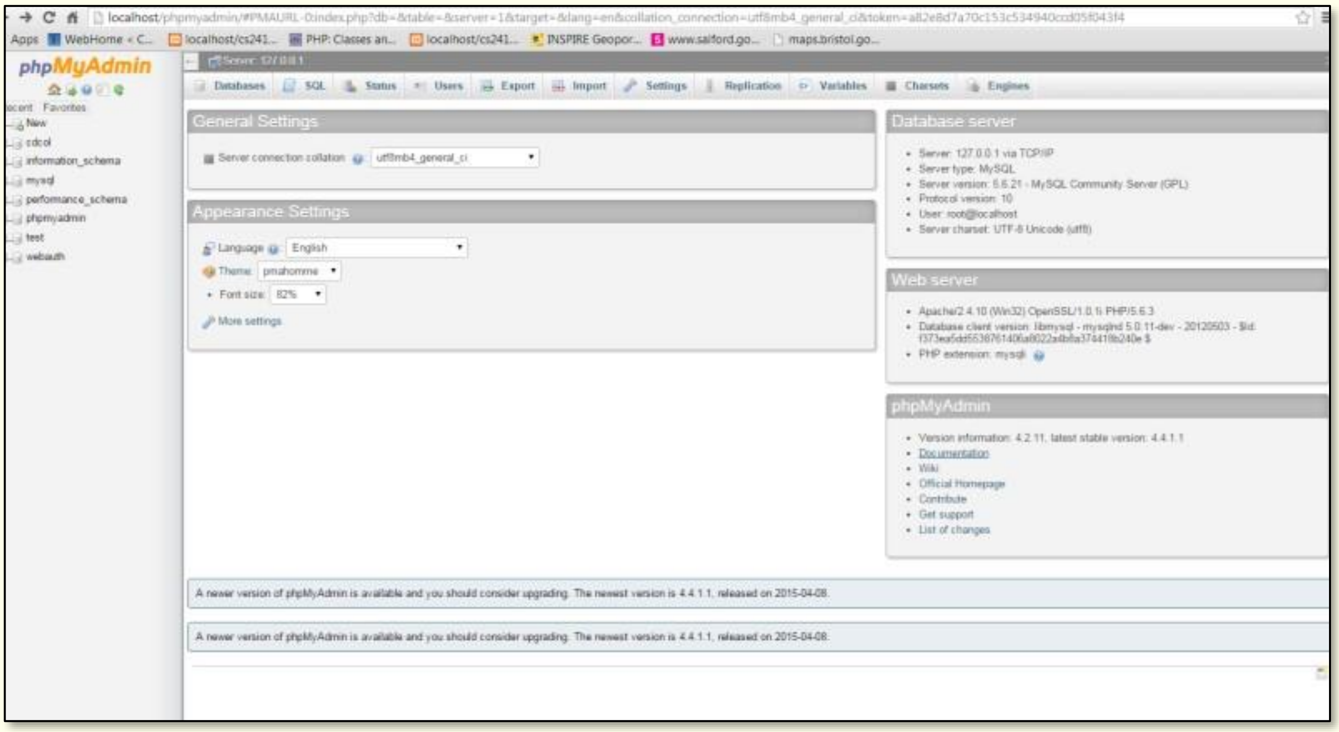
Unit 4: Handling Forms and Databases

In this lab session, we will write PHP scripts for form handling and database operations. You will create a MySQL database through a GUI-based interface – phpMyAdmin – and then use PHP to run SQL queries. You need to handle errors and consider security issues in these tasks. You should create a **unit4** folder in the document root folder and you should save all files of the lab in this location.

Before working on the tasks, remember to start your Apache and also MySQL server. Normally, when you start your XAMPP (or WAMP/MAMP/LAMP) server, MySQL will be started.

To check if your MySQL server is running on your local computer or not, you can go to *http://localhost/phpmyadmin*. If you see a screen like below, your MySQL server is running fine. We will create and use a MySQL database after the first two tasks.



General Settings

Server connection collation: utf8mb4_general_ci

Appearance Settings

Language: English
Theme: pmahomme
Font size: 82%
More settings

Database server

- Server: 127.0.0.1 via TCP/IP
- Server type: MySQL
- Server version: 5.6.21 - MySQL Community Server (GPL)
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.6.3
- Database client version: libmysql - mysqlnd 5.0.11-dev - 20120503 - $Id: f373ea5dd5538761406a8022a4b8a374418b240e $
- PHP extension: mysqli

phpMyAdmin

- Version information: 4.2.11, latest stable version: 4.4.1.1
- Documentation
- Wiki
- Official Homepage
- Contribute
- Get support
- List of changes

A newer version of phpMyAdmin is available and you should consider upgrading. The newest version is 4.4.1.1, released on 2015-04-08.

A newer version of phpMyAdmin is available and you should consider upgrading. The newest version is 4.4.1.1, released on 2015-04-08.

Task 1 Basic Form design

Use your favourite editor to create a HTML file **calculator.html.** This HTML file should include a basic form where users can input two integers and submit for calculations. The form **method** could use **GET**. The form **action** URL should be set to the **calculator.php** file which you will create in the next task.

You could check your form by visiting: *http://localhost/unit4/calculator.html* in a browser if you have put the file **calculator.html** inside the **unit4** folder in your document root. Of course, there will be errors when clicking the submit button at the moment since we have not written the **calculator.php** file yet.

Hint: *To get an integer input in your form, you could use* `<input type="number"` …..../>

Task 2 Basic form handling

Write the form handling PHP script **calculator.php** which will do the addition, subtraction, multiplication and division for the two inputs.

1) Get the two numbers from the form (using $_GET or $_REQUEST array);
2) Do some necessary form validation. For example, if a user enters 0 for the second number, this can cause an overflow for division, you should show an error message rather than do the division;
3) Write the suitable PHP code to output the input and calculation results.

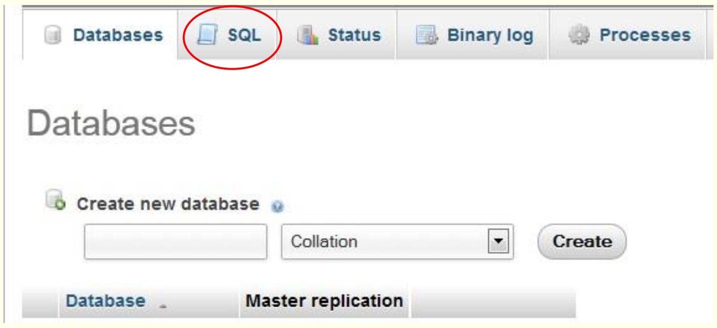
   A sample output would be like:

```
Your first input: 2
Your second input: 8
Addition Result: 10
Deduction Result: -6
Multiplication Result: 16
Division Result: 0.25
```

Now you could check your work by visiting: *http://localhost/unit4/calculator.html*. Input two numbers and click the submit button, you should be able to see the right calculation results if you have written your PHP code properly.

Task 3. Create the 'carent' database using phpMyAdmin

phpMyAdmin is one of the most popular applications for MySQL database management. To create a database called '**carent**', go to http://localhost/phpmyadmin and select the 'Databases' tab as shown below. Input the new database name as '**carent**', click 'create' and you will see the new '**carent**' database listed on the left menu.



Now select the 'carent' database and we will use the SQL tab (in the red circle as above) to create a table called **vehicles** by running the following SQL statement**:**

```
CREATE TABLE 'vehicles' (
  'reg_no' varchar(8) PRIMARY KEY NOT NULL,
  'category' enum('car','truck') NOT NULL DEFAULT 'car',
  'brand' varchar(30) DEFAULT NULL,
  'description' varchar(256) DEFAULT NULL,
  'dailyrate' decimal(6,2) NOT NULL DEFAULT '10.00'
)
```

You could also insert a few records by running the following SQL statement**:**

```
INSERT INTO 'vehicles'
('reg_no', 'category', 'brand', 'description', 'dailyrate') VALUES
('BA5923W', 'car', 'Toyota', 'black 4 door 2.6 engine ', '9.99'), ('BA6611A', 'car',
'NISSAN ', '4 Door Saloon, Automatic', '9.99'),
('BM1245a', 'car', 'Golf', NULL, '9.00'),
('GA5955E', 'truck', 'NISSAN CABSTAR 3.0', 'Lorry, Manual ', '18.99')
```

Notes: You could interactively use phpMyAdmin to create new databases/tables, insert new records, run SQL and do many other database operations. However, this lab does not intend to give all details about how to use phpMyAdmin. You could reference the phpMyAdmin quick guidance document in this lab's folder on Blackboard. There are also many tutorials on the web about how to use phpMyAdmin. For example, a good link to look through if you're stuck: http://www.siteground.com/tutorials/phpmyadmin/.

You could of course create MySQL database using command lines through a terminal. The tutorial in https://dev.mysql.com/doc/refman/5.7/en/tutorial.html has the detailed introduction.

Task 4. Connect the 'carent' database

Before working on the following database tasks, we set up the database connection using PHP. Write a '**connectdb.php**' script to connect to your MySQL database '**carent**' using PDO**.** Your script should create a PDO object and be able to catch the PDO exception.

Note: *You normally need to use the database connection in different scripts. It is a good idea to put the database connection script into a separate PHP file (e.g. connectdb.php here) which can be included in all your other php files which require the database connection.*

Task 5. Display all the records in the vehicles table

Create a **listvehicles.php** file and write PHP script to display a table listing all records in your **vehicles** table.

1) Include the **connectdb.php** file you produced in Task 4;
2) Write the suitable SQL to query all records in the **vehicles** table (using PDO method); 3) If successful, display all the records in a HTML table with headings for columns including

   "Reg_no", "Category", "Brand", "Dailyrate", and "Description".

Hint: *you need to loop through your query result set and display each row in a table. The basic HTML table syntax:*

```
<table>
 <tr>
    <th>Head A</th> <th>Head B</th>
 </tr>
 <tr>
    <td>Cell A1</td> <td>Cell B1</td>
 </tr>
 <tr>
    <td>Cell A2</td> <td>Cell B2</td>
  </tr>
</table>
```

After finish the coding, you could check the result by visiting: *http://localhost/unit4/listvehicles.php* if you have put the **listvehicles.php** file inside the **unit4** folder in your document root folder.

Task 6 Create a HTML Form for a vehicle record

Create a web form in order for users to add a new vehicle to the '**vehicles**' table**.** You could design the form yourself and name the file as **addvehicles.php**. The **method** for your form should be POST. The form **action** should be set to point to itself i.e. the **addvehilces.php** file. You will write the PHP script part in Task 7.

In general, the form should contain the following elements:

1) A text input field for the vehicle's reg_no;
2) A text input field for the vehicle's brand. Or you could use a dropdown list containing some popular brands of cars and trucks;
3) A input field for daily rate;
4) A radio button for 'category' containing the 'car' and 'truck' items;
5) Add a hidden field in your form. This hidden field will be used to check whether the form has been submitted or not. This could ensure that you only run the PHP part of the **addvehicles.php** file when the form is submitted;

   Hint*: use* `<input type="hidden" name="submitted" value="true" />`

6) A submit button and a reset button

Task 7 Form handling - Adding a new record to the Vehicles table

Write the form handling PHP script in the same **addvehicles.php** file**.** The PHP code will get the form input about a vehicle and insert a record into the **vehicles** table.

1) Connect to the database by including the "**connectdb.php**";
2) Get the form input data using $_POST or $_REQUEST array since you use the POST method in your form;
3) Do some necessary form validation i.e. if a user does not enter certain information (e.g., reg_no) or not in the right format (e.g., numeric data needed for daily rate), a relevant error message(s) should be displayed;

   Hint: *functions* `isset()`, `empty()` *and* `is_numeric()` *etc. could be used for validation purpose.*

4) Write the suitable insert SQL and use PDO to insert the form data into the **vehicles** table; 5) Your script should consider SQL injection problems and report suitable error messages; 6) If successful, display a proper message.

Run your code and add a few records to your vehicles table. You could check the **vehicles** table through running your **listvehicles.php** or using the phpmyadmin GUI.