CSCU9B3 Practicals 6 & 7: Introduction to PHP and MySQL

Computing Science, University of Stirling

So far we have been using PHPMyAdmin as an interface to MySQL. For some things, this is fine, but for many other applications it is of no use at all: you may need an interface that is specific to the database and may run queries without human intervention. For example, generating payslips from an employee database. For this you would need a computer programme that interacts directly with the database to carry out required queries and format the results. Database updates could also be done this way.

mysqli API

In this practical we will start to access a MySQL database directly from **PHP** code, using the **mysqli** Application Programming Interface (API). This can be used in either a procedural or object-oriented way. Examples of both are provided in the lecture slides. We will start with the procedural version. A useful reference for mysqli functions (procedures) can be found at:

https://www.w3schools.com/php/php_ref_mysqli.asp

What to Do in this Practical

To get started, do the following:

1. In a Windows Explorer (file browser) window, open your web folder on our PHP server by typing the appropriate address in the address bar:

   \\wamp0.cs.stir.ac.uk\www\xxx

2. Download the template PHP file, **dbexample.php** from the module's Canvas Practicals page to this folder.
3. Open this file in your favourite editor or IDE (a simple text editor like TextPad is fine).

Connecting to the database

Now let's create a connection to your database:

4. In the PHP code, you should see a number of variables that define how to access your database. $servername is already set up correctly, but the other three variables need your user-specific information. Change these variables as needed.
5. Save this file and try it out by entering the web address in a web browser:

   http://wamp0.cs.stir.ac.uk/xxx/dbexample.php

6. Hopefully you will see a "connection successful" message displayed in the browser window. If not, then check the variable values you specified. If you think these are correct, then please ask for help. Otherwise, try again.

Running queries

Given that you can connect to your database, now let's try running a query on your existing tables (from previous practicals):

7. The code is currently set up to run a query to extract `Employee number` from the table `employees` if a match is found in `employees`.`name`.
8. If necessary, edit your PHP file to change the field (column) and table names to match ones you have in your database from the first two practical sheets. Reload the file in the web browser.
9. Try this query out by typing suitable text (part of a name) in the form text box in the web page. Try names that should and should not find a match in your table.
10. You should note that the code does not handle not finding a match very well, just showing blank for the results. Add an extra test to the code so that a suitable message is displayed when no results are returned.
11. Also try inputs to test the "data sanitising" code that escapes special characters (such as quotes) and strips out HTML tags.

Now let's construct a completely new query to return information about projects. This will test both your PHP and MySQL skills:

12. After the code for the name search above, but before the line that closes the database connection, add new lines of PHP to extract from the database these details about all the projects: project name, project description, manager name (not the employee number of the manager).
13. Write code to display these results neatly in the web page: you can just use echo or print commands and use the HTML
 tag to create new lines; but if you know some HTML you might like to come up with a more fancy display e.g. a table.
14. EXTRA: Adjust your code so that each row of results is returned as an associative array, using the mysqli_fetch_assoc() function, with results displayed accordingly.

CHECKPOINT 7: show your working code for the "projects" query (either enumerated or associative array version) and its web browser output.

Using object-oriented mysqli

Mysqli can also be used with objects. Some examples are given in the lectures and in the W3 Schools PHP mysqli information.

15. Download a new, clean copy of **dbexample.php**, giving it a new name when you save it.
16. Edit this file and firstly change the <form> statement so that it calls this file as it's action (otherwise it will call your previous version that you created above!)
17. Now recode the PHP so that it uses the object version of mysqli. Note you need only complete the original, form-based query and not add the new projects query developed above.

CHECKPOINT 8: show your working code for your object version and its web browser output.

Building Queries using Prepared Statements

It is often convenient and efficient to prepare SQL statements in advance, before all the parameter values for e.g. searches or insertions, are known.

1. Create a copy of your object-oriented code (give it a new file name and again, remember to changes the file name in the <form> tag).
2. Change the code so that it uses a prepared statement to set up the query before it is executed with values obtained from the form. Note the following:
3. The SQL for your search should have a question mark (**?**) in place of the value you wish to look up, so in this case, it should replace the employee name, like:

```
$stmt = $conn->prepare("SELECT Name, `Employee Number` FROM
employees WHERE Name LIKE ?");
```

Note that ? replaces %$name% in the original query, so you will need to adjust the value of $name returned from the form to include the two % symbols before executing the query.