6G5Z2107: Web Design and Development

# Lab 04 – Databases and Remote Storage

Stuart Cunningham

# Objectives

1. Read and understand PHP code that creates simple MySQL database tables
2. Read and understand PHP code that interacts with database tables to generate page content
3. Extend existing PHP code to create new pages making use of new database interactions within a multi-page login-driven site

The aim of today is to reinforce the concepts covered in this week’s lecture and prepare for our use of database tables and multi-user login capabilities in 2CWK50.

# Exercise 1: One Database, Multiple Scripts

This week we looked at a multi-page login-driven website about Favourite Films, where user data is stored in database tables. You can access and test the site via Mudfoot:

* <http://mudfoot.doc.stu.mmu.ac.uk/students/stuart_student/pd3/>

Your first task this week is to read the code and understand how each script is working. This process will take some time, but it’s very important: 2CWK50 and the later tasks in this lab draw heavily on the ideas and techniques in these scripts. **Please complete the questions on the following page by** **typing your answers into this Word document and saving it to your H: drive so that your tutor can discuss your answers with you.**

Here are some tips for getting the most out of reading the code:

* Read the scripts in the order suggested by the questions below
* Read each script in full to identify what you do understand as well as the bits you don’t
* Run and test the script to see how it behaves (either on Mudfoot, or locally on Xampp)
* Look for where the HTML you are seeing in your browser is echoed out by the script
* Read the comments in the code carefully
* Refer back to lecture slides as you read
* Look up new or unfamiliar syntax with an Internet search (and prioritise links to good resources like the online PHP manual and w3schools)
* Refer back to code listings from previous weeks to check familiar concepts you’re still a bit unsure about
* Discuss the code with a neighbour
* Ask your tutor for help

## **create\_data.php** – read the script then attempt the following questions

|  |  |
| --- | --- |
| Question: What are the values stored inside the following 3 variables: $dbhost, $dbuser, $dbpass   |  | | --- | | Answer: | |
| Question: What is the name of the database created by this script?   |  | | --- | | Answer: | |
| Question: What is the name of the first table created by this script?   |  | | --- | | Answer: Members Table | |
| Question: How many rows are inserted into the first table?   |  | | --- | | Answer: 6 | |
| Question: What is the name of the second table created by this script?   |  | | --- | | Answer: Favourites Table | |
| Question: What data type is used to store the contents of the "title" column?   |  | | --- | | Answer: Array | |
| Question: How many rows are inserted into the second table?   |  | | --- | | Answer: 6 | |

## **sign\_in.php** – read the script then attempt the following questions

|  |  |
| --- | --- |
| Question: We have to call session\_start() before we can access session data through the $\_SESSION superglobal array. Why is there no call to session\_start()in this script? (hint: skip to the next question if you get stuck, then come back to this one)   |  | | --- | | Answer: | |
| Question: Give a brief description (2 sentences max) of what the header.php script does   |  | | --- | | Answer: Shows a ‘favourite film’ as a title and a header. Also, allows the user (if logged in) the logged in menu, if not it shows the logged out menu. | |
| Question: How does sign\_in.php decide if the current user is signed in?   |  | | --- | | Answer: It decides by taking the ‘username’ and ‘password’ data that the user entered and check if it matches with the database in order to see if it matches. | |
| Question: Find the SQL statement executed by this file and copy it in below   |  | | --- | | Answer: = "SELECT \* FROM members WHERE username='$username' AND password='$password'"; | |
| Question: Describe in words what the above statement does   |  | | --- | | Answer: It tries to match what he user entered in the username and password bar with the saved data that is in the database. | |
| Question: Why is it impossible for the statement to return more than 1 row?   |  | | --- | | Answer: | |
| Question: Give a brief description (1 sentence max) of what the footer.php script does   |  | | --- | | Answer: | |

## **sign\_up.php** – read the script then attempt the following questions

|  |  |
| --- | --- |
| Question: Think back to the two tables that were created in create\_data.php (remind yourself by looking at the code if necessary). In words, what would be the basic operation that a sign\_up.php script would need to perform?   |  | | --- | | Answer: | |
| Question: Find the SQL statement executed by this script and copy it below   |  | | --- | | Answer: "INSERT INTO members (username, password) VALUES ('$username', '$password');"; | |
| Question: Would this statement return any data?   |  | | --- | | Answer: No | |
| Question: Based on your previous answer, what are the possible values returned by the call to mysqli\_query() (stored in $result)?   |  | | --- | | Answer: | |

## **sign\_out.php** – try to answer this question **before** you read the script

|  |  |
| --- | --- |
| Question: The site uses session data to keep track of whether individual users are logged in. Thinking back to our lecture on sessions, what would you expect to happen inside this script?   |  | | --- | | Answer: Have a session data that would take the user to the logged out page. | |

## **all\_favourites.php** – read the script then attempt the following questions

|  |  |
| --- | --- |
| Question: Find the SQL statement executed by this file and copy it in below   |  | | --- | | Answer: "SELECT title, year, rating, genre FROM favourites ORDER BY year DESC"; | |
| Question: Describe in words what the above statement does   |  | | --- | | Answer: It selects the data that holds title,year,rating,genre from ‘favourites’ in the database and orders them in a descending order. | |
| Question: Would this statement return any data?   |  | | --- | | Answer: yes | |
| Question: Based on your previous answer, what will be returned by the call to mysqli\_query() (stored in $result)?   |  | | --- | | Answer: | |
| Question: What is returned by a single call to mysqli\_fetch\_assoc()?   |  | | --- | | Answer: | |

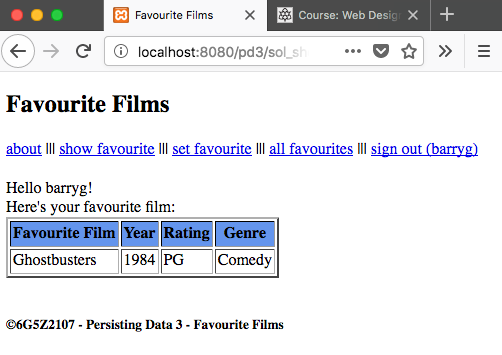
## General

|  |  |
| --- | --- |
| Question: Three mysqli\_ functions are called by the following files: sign\_in.php, sign\_up.php, all\_favourites.php. These functions control: i) opening the connection to the MySQL server; ii) executing a query; iii) closing the connection to the MySQL server. Name these 3 functions:   |  | | --- | | Answer: | |

# Exercise 2: Implementing show\_favourite.php

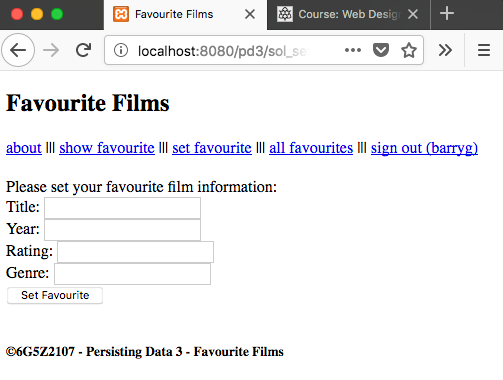
In this second exercise, you’re going to implement the show\_favourite.php script. The job of this script is to show a signed-in user their favourite film (or prompt them to set one up if they don’t have one). You can see how the script should work by testing it on Mudfoot.

Download the exercise files for this week from Moodle, and copy the them into your xampp/htdocs/ folder. Run the create\_data.php script and check that you can get the site working locally. You will notice that show\_favourite.php (and set\_favourite.php, see also the next exercise) haven’t been implemented. Your task is to add code into the show\_favourite.php script so that it works properly (see image, and test on Mudfoot).

You should consider the following when implementing your show\_favourite.php script:

1. How to check the user is logged in and what to do if they aren’t.
2. Which table you need to interact with (and how) in order to get the information that you need.
3. The three mysqli calls you will need in order to:
   * Connect to the MySQL server
   * Execute your query
   * Close the connection
4. What will be returned by your query and how you can make use of this value in your script.
5. What will happen for users who are logged in but haven’t yet set up a favourite film.

# Exercise 3: Implementing set\_favourite.php

In this exercise, you will implement the missing set\_favourite.php script. The job of this script is to let a signed in user change their favourite film (or set it up for the first time). You can see how the script should work by testing it on Mudfoot.

**[**If you haven’t attempted Exercise 2: download the exercise files from Moodle, and copy the them into your xampp/htdocs/ folder. Run the create\_data.php script and check that you can get the site working locally. You will notice that set\_favourite.php (and show\_favourite.php, see also the previous exercise) have not been implemented.**]** Your task is to add code into the set\_favourite.php script so that it works properly (see image, and test on Mudfoot).

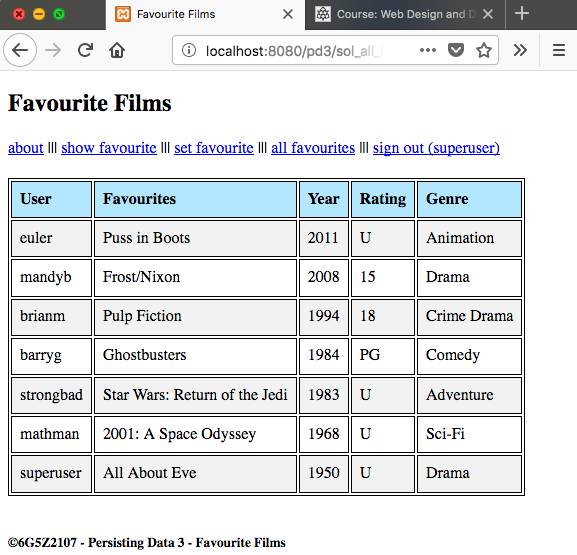
You should consider the following when implementing your set\_favourite script:

1. How to check the user is logged in and what to do if they aren’t.
2. Which table you need to interact with (and how) in order to store the new film:
   * What query will you need if the user already has a favourite film?
   * What query will you need if they don’t?
   * What query will let you decide between the previous two options?
3. What will be returned by your queries and how you can make use of these values in your script.

# Extension Exercise 4: ‘superuser’ Account

In this final exercise you should extend the all\_favourites.php and create\_data.php scripts. Specifically, you should create a new user for the site that has the username superuser and a password of AAA. The details of superuser’s favourite movie are:

|  |  |  |  |
| --- | --- | --- | --- |
| **Title** | **Year** | **Rating** | **Genre** |
| All About Eve | 1950 | U | Drama |

When logged-in, superuser should be able to use the site like anyone else, but when they access the all\_favourites.php script they should also be able to see the name of each user in the table of favourite films (see image, and test on Mudfoot).

You should consider the following when implementing your all\_favourites.php script:

1. What modifications will you have to make to create\_data.php before adjusting the allfavourites.php script?
2. How to check the user is logged in as superuser and not anyone else.
3. How to modify the existing PHP/MySQL to get the additional username field in the favourites table.
4. How to format and present the data in the table.

# Help! I don’t know what to do!

Don’t panic!

There are plenty of resources available, including the following, which are all great starting points:

* The lecture slides (available on Moodle) which recap what we covered in the lecture
* w3schools MySQL tutorial (we are using the procedural mysqli interface): <http://www.w3schools.com/php/php_mysql_intro.asp>
* The PHP manual’s pages on the mysqli extension (we are using the procedural interface): <http://php.net/manual/en/book.mysqli.php>
* Chapter 8 and 9 of “Learning PHP, MySQL and JavaScript” by Robin Nixon (available as an e-book through the library webpages) and chapter 10 for examples of using the object-oriented mysqli interface

Now might also be a good time to look back at your notes from last year’s “Introduction to Web Design and Development” unit (and the “Information Systems” unit if you took it).

# I fancy a challenge, what do you recommend?

In the lecture, we talked about extending the Favourite Films site to include user profiles and a global poll where site visitors can vote for their favourite film from the all\_favourites.php list, including discussing suitable table designs. Try extending the existing code to incorporate this new functionality. [Be strategic: you should only attempt this extra challenge if you’re already making good progress with 2CWK50 and the skeleton code…] Start by extending create\_data.php to generate the new tables and insert suitable test data. Then create a favourite\_poll.php script that lets users vote via a form, and view\_profiles.php and view\_profile.php scripts that let users browse and view member profiles (including their own).