Project 2: Internet Movie Database

Overview

In this project, you will create a fully functioning Movie Database system accessed by users exclusively through a Web site. The Web site will be implemented with PHP running inside an Apache Web server whose data is managed by MySQL.

Development Environment

Project 2 must be completed within the "mysql-apache" container that you set up for Project 1. Remember that you can start the container with the following command:

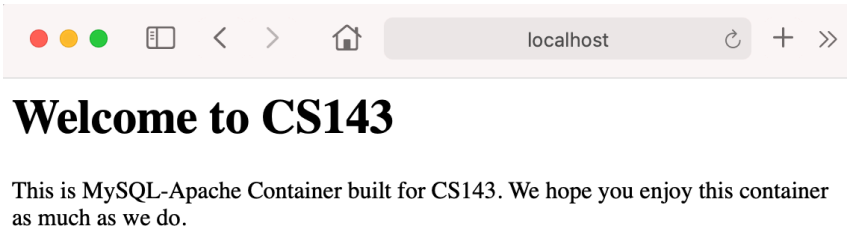
```
$ docker start -i mysql-apache
 * Starting MariaDB database server mysqld                              [ OK ]
 * Starting Apache httpd web server apache2
 *
cs143@f3339ccac015:~$
```

If you want to start from scratch by recreating the container, you can use the following sequence of commands:

```
$ docker container rm mysql-apache
$ docker run -it -v {your_shared_dir}:/home/cs143/shared -p 8888:80 --name mysql-a
```

Learn PHP and MySQL connectivity

Since you need to implement this project with PHP, learn the basics of the PHP language by reviewing the W3CSchools PHP tutorial first, if you are not familiar with PHP. Please read at least up to PHP Form Handling page of the tutorial. You can test the examples in the tutorial by creating PHP pages in the `/home/cs143/www/` directory of our container. All files in `/home/cs143/www/` are served by the Apache Web server and are accessible at http://localhost:8888/ from your host browser. In fact, the following "Welcome page" that you saw when you first set up the container is produced by the `index.php` file in the directory.



For information on how you can interact with MySQL from PHP (e.g., issuing a query to MySQL and retrieving results), see our brief tutorial on MySQL and PHP. PHP Manual on MySQL Improved Extension section has the full list of MySQL functions available in PHP.

Project Requirements

The functionality of the Movie Database system that you need to build is quite flexible, although we expect all students to implement the following four baseline pages:

1. **Actor page `actor.php`**: A page that shows an actor information.
    - This page must take the actor id as the `id` parameter of the request URL and display the corresponding actor's information, including their name and the movies that they were in. For example, the URL http://localhost:8888/actor.php?id=4033 must display the information on Ms. Drew Barrymore.

      Note: Any `name=value` pair appearing after `?` in the URL (e.g., `id=4033` in http://localhost:8888/actor.php?id=4033) is available as `$_GET['name']` (e.g., `_GET['id']`) in your PHP code.
    - For every movie that the actor wwas in, the actor page must include a hyperlink to the corresponding "movie page" described next.
2. **Movie page `movie.php`**: A page that shows a movie information based on the movie id provided as the `id` parameter of the URL. For example, the URL http://localhost:8888/movie.php?id=705 must display the information on the movie "Charlies Angels." This page must
    - Show hyperlinks to the actor pages for each actor that was in this movie.
    - Show the average score of the movie based on the user feedback.

- Show all user comments.
- Contain an "add Comment" link/button, which links to the movie's review page described below.

3. **Search page `search.php`**: A page that lets users search for an actor/movie through a keyword search interface.

   - If no parameter is given in the URL, the page must display one or more search boxes to let the user search for a movie and for an actor. (For an actor/actress, you should examine first/last name, and for a movie, you should examine title.)
   - If (a set of) keyword(s) are provided as the `actor` parameter of the URL, the page must return the list of actors whose first or last name contains the keyword(s). Clicking on each actor must lead to the corresponding actor page.
   - If (a set of) keyword(s) are provided as the `movie` parameter of the URL, the page must return the list of movies whose title contains the keyword(s). Clicking on each actor must lead to the corresponding movie page.
   - The search page **should** support **multi-word** search, such as "Tom Hanks," and be **case-insensitive**. For multi-word search, interpret space as "AND". That is, return all actors that contain "Tom" **AND** "Hanks" in their first or last name columns. To support case-insensitive search, you can apply the `LOWER()` function to both the column (to be searched in) and the string (that you search for).

4. **Review page `review.php`**: A page that lets users add a review to a movie whose id is the `id` parameter of the URL.

   - When the request URL contains only the `id` parameter, the page must include input boxes to let the user to input their name, the rating of the movie and their comment on the movie.
   - When the request contains `id`, `name`, `rating` and `comment` parameters, you must insert a row to the `Review` table with the provided values. The value for the `time` column should be the time when the review is written. The returned page from this URL must display "confirmation text" saying that the review has been successfully added.

Make sure that ***all page names and parameters are exactly as specified, including their cases***.

A demo site is available here (The link is password protected. Use username "project" and password "demo" to access the page.) This page is available strictly to give you an idea of the basic requirements and is *not* meant to guide your choice of style or user interface in any way.

Important Notes

- While the functionality of your Movie Database system is quite open-ended, the interface itself is *extremely* open-ended. This class is not a user interface class, and **you will get full credit for a bare-bones functioning system** with simple input boxes, menus, and/or radio buttons, and simple HTML output tables as long as your web site is reasonably intuitive to use. That is, the user should be able to navigate through your web site without too much difficulty. You are welcome to make your web site fancy, but there won't be any extra credit given to beautifully designed interfaces.
- It is OK to embed some javascript or CSS to make your web site look good, but please make sure your work is **completely self-contained**. Namely, all javascript/CSS files related to your web site should be incorporated in your submission. Make sure that the size of your entire web site files (excluding our movie database) is less than 20MB.
- You may assume that the users are not malicious. They do not intentionally perform anything bad, such as an SQL injection. However, any PHP/database errors due to simple data entry errors or bad input values should be managed gracefully, i.e., it should be possible for users to continue interacting with the system, and the database should not get corrupted.
- Note that your Project 2 is dependent on your Project 1. Please make sure your Project 1 is working correctly and fix any existing bugs. As part of Project 2 submission, you will have to submit the relevant files of your (revised) Project 1 again.
- You need to use the same set of tables (and only them) that you created in Project 1. You should use the database `cs143` running on `localhost` via username "cs143" and empty password.
- In order to make your site portable, **all URLs embedded in your pages should be *relative*, not absolute**. For example, if you want to include a link to "search.php" located in the same directory as the current page, please use a relative URL like `<a href="search.php">` or `<a href="./search.php">`, not an absolute URL like `<a href="http://localhost:8888/search.php">`.
- Although the project interface is very open-ended, we have to make the basic features of your system accessible in a typical browser environment. More specifically, you must make sure your system works with the most recent version of Google Chrome browser without any additional plugins or extensions.

Hints on Implementation

- Note that your PHP pages can be accessed directly through a URL like http://localhost:8888/myPHPpage.php?param1=value1&param2=value2&param3=value3 where myPHPpage.php is the name of your PHP page, and the string after the question mark contains parameters passed to the PHP page just like in a "GET" method. So you can *embed a clickable link to the page* generated by your PHP code using hyperlinks of the above format, with appropriate parameters that instruct your PHP code to execute the right query.
- If you are not sure how you can use HTML input boxes to take input from the user (for implementing a search box, for instance), read our brief tutorial on Handling User Input in PHP. For information on how you can interact with MySQL from PHP, again, see our tutorial on MySQL and PHP.
- In adding user reviews to a movie, you may need to obtain the current timestamp. You can do it either in your PHP code or in MySQL itself. The MySQL date and time functions page explains how you can obtain the current timestamp in MySQL.

Submit Your Project

Your project must be submitted electronically before thYour project must be submitted electronically before the deadline through GradeScope. You can submit your work an unlimited number of times. In case of multiple submissions, the grade from the latest submission will be used.

What to Submit

For this project, you will have to submit one zip file `project2.zip` that must be created using the packaging script provided below. This file will contain all source codes that you wrote for Project 2 and the database creation and load script from Project 1.

Creating `project2.zip`

To submit the code for a functional web site, you will need to create a zip file that includes all files necessary for your site, including the `create.sql` and `load.sql` script that you created in Project 1. Roughly, the zip file that you submit should have the following structure:

```
project2.zip
 +- actor.php
 +- movie.php
 +- search.php
 +- review.php
 +- ... other files for your site
 +- sql
    +- create.sql
    +- load.sql
 +- README.txt (optional)
```

Make sure that your zip file contains the four required PHP pages at the root. Your two SQL scripts, `create.sql` and `load.sql`, should be placed in the `sql/` subdirectory of the zip file. If you used a third-party library, such as the Bootstrap CSS library or React JS library, please make sure to include them in the zip file as well. `README.txt` is optional, where you can include any information that you want to communicate to us.

To help you package your submission zip file, you can download and use our packaging script p2_package. After downloading the script in the root directory of your PHP site and setting its executable permission, do the following:

1. Clean up the directory, removing any files that are not necessary for your site.
2. Create a subdirectory `sql/` and place the two SQL script from Project 1, `create.sql` and `load.sql` in the `sql/` directory
3. Run `p2_package` to create the `project2.zip` file.

When executed, our packaging script will collect all files located in the same directory as the script and create the `project2.zip` file according to our specification like the following:

```
$ ./p2_package
  adding: search.php (stored 0%)
  adding: actor.php (stored 0%)
  adding: movie.php (stored 0%)
  adding: review.php (stored 0%)
  adding: sql/ (stored 0%)
  adding: sql/load.sql (deflated 81%)
  adding: sql/create.sql (deflated 63%)
  adding: README.txt (stored 0%)
[SUCCESS] Created '/home/cs143/www/project2.zip'
```

Testing Your Zip File

Things can go wrong for a number of unexpected reasons during grading. In order to minimize any surprises, we are providing a simplified version of the "grading script" p2_test that we will use to set up your web site on the grading machine. In essence, the grading script unzips your submission to a temporary directory and executes the SQL files to load the database and make your site files available in the `grading/` subdirectory of `/home/cs143/www`.

Important Note: Our grading script will drop all existing data from `cs143` database and delete any files that may exist in the `/home/cs143/www/grading/` directory. If you want to keep any data/files in them, please make sure that you **back them up before you run our grading script.**

When executed, you will see an output similar to the following from the grading script:

```
$ ./p2_test project2.zip
Removing all files in /home/cs143/www/grading/
Dropping existing tables in the cs143 database
Downloading and unzipping data.zip file to load database...
Running your create.sql script...
Running your load.sql script...
Linking your submission to the grading/ subdirectory
All done!

Please ensure that you have a fully functional Web site available
at http://localhost:8888/grading/
```

After setting up your web site using the submission zip file, please access your site at http://localhost:8888/grading/ with a browser from your host machine and make sure that your site works fine without any issues.

Submitting Your Zip File

Visit GradeScope to submit your zip file electronically by the deadline. In order to accommodate the last minute snafu during submission, you will have 1-hour window after the deadline to finish your submission process. That is, as long as you start your submission before the deadline and complete within 1 hour after the deadline, you are OK.