PHP AND MYSQL WITH PDO

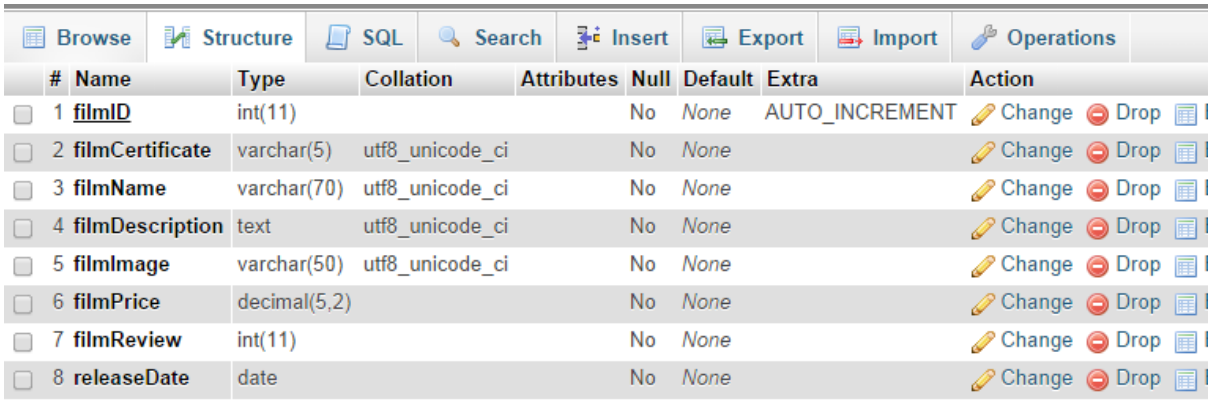
INTRODUCTION

If you haven't already done so create a MySQL user account. Access the phpmyadmin application at:

http://homepages.shu.ac.uk/mysql

PHPMYADMIN

In phpMyAdmin create yourself a table called 'movies'. Add fields to the table as follows:



Browse | Structure | SQL | Search | Insert | Export | Import | Operations

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	filmID	int(11)			No	None	AUTO_INCREMENT	Change Drop
2	filmCertificate	varchar(5)	utf8_unicode_ci		No	None		Change Drop
3	filmName	varchar(70)	utf8_unicode_ci		No	None		Change Drop
4	filmDescription	text	utf8_unicode_ci		No	None		Change Drop
5	filmImage	varchar(50)	utf8_unicode_ci		No	None		Change Drop
6	filmPrice	decimal(5,2)			No	None		Change Drop
7	filmReview	int(11)			No	None		Change Drop
8	releaseDate	date			No	None		Change Drop

Download the files from Blackboard or Github . Populate your *movies* table with data from the zip in the *data/movies.csv* file via the 'import' tab.

Once you have the data in MySQL use the export tab to create an SQL file backup of your data.

Take time to look at the data. Knowing your data is extremely useful when building an application.

CONNECTING TO MYSQL

Open the PHP file in the 'includes' folder called *conn.inc.php*. You will need to edit this file to match your own database name, username and password. For the purposes of this module you should only need to create this file once and then reference it in each of the pages on `public_html` that requires database access.

DEBUGGING

The homepages.shu.ac.uk server has error handling switched off. By adding the following to the top of any PHP script we can switch on error messages that will help when debugging the application.

```
ini_set('display_errors', 1);
```

CREATING A SIMPLE QUERY - SINGLE RECORD

Open the file *select-single.php*. Use this to build a simple query of your data to extract one unique value. First you need to include the connection file.

```
<?php
require('includes/conn.inc.php');
?>
```

As the include file contains the login credentials for the database this should be moved above the public visible directory thus the `../includes` path above.

- Build a SQL query.
- Use PDO to send your query to MySQL.
- Create an array to hold the results of your query.

You code should now appear:

```
<?php
require('includes/conn.inc.php');
$queryFilms = "SELECT filmName, filmDescription, filmImage FROM movies
WHERE filmID = 53";
$stmt = $pdo->query($queryFilms);
$row =$stmt->fetchObject();
?>
```

These pages use Bootstrap to style the page. In the body of the HTML file use `echo` to output the value of *filmName* within the `<div class="col-md-8">` element.

```
<?php
echo "<h2>{$row->filmName}</h2>";
?>
```

Amend the above to output the *filmDescription* field in a `<p>` element underneath the *filmName*.

In the column to the left `<div class="col-md-4">` add code to output the image associated with this film.

```
<?php
echo "<p><img src=\"images/{$row->filmImage}\"></p>";
?>
```

Note the use of backslash to escape the quotes that are needed around the `src` attribute value.

CREATING A SIMPLE QUERY - MULTIPLE RECORDS

Open the file *select-multiple.php*. Use this to build a simple query of your data to extract multiple records. First you need to include the connection file.

```
<?php
require('includes/conn.inc.php');
?>
```

- Build a SQL query.
- Use the mysqli extension to send your query to MySQL.

You code should now appear:

```
<?php
require('includes/conn.inc.php');
$queryFilms = "SELECT * FROM movies";
$stmt = $pdo->query($queryFilms);
?>
```

Find the `<div class="col-md-12">` element that spans across the width of the page. Add a `while` loop to `echo` around each value of *filmName* from the associate array produced by `fetch_assoc()`;

```
<ul>
<?php
while($row =$stmt->fetchObject()){
     echo "<li>{$row->filmName} - {$row->filmCertificate}</li>";
}?>
</ul>
```

THINGS TO TRY

Can you display the images for each film found by the loop?

You could use Bootstrap features to output the images in a grid by setting a column value such as `col-md-3.`

CREATING A SEARCH/RESULTS PAGE - SINGLE RECORD

Open the file *prepare-single.php*. Here you will find a HTML form. In this file build a prepared statement to query of your data based on user input from a form.

- Add your connection include
- Build a `prepare` statement
- Bind a `$_GET` value to the query
- Execute the query and bind the results
- Fetch the results and close the query.

Your code should now appear:

```
<?php
require('includes/conn.inc.php');
$sql= "SELECT * FROM movies WHERE filmID = :filmID";
$stmt = $pdo->prepare($sql);
$stmt->bindParam(':filmID', $_GET['filmID'], PDO::PARAM_INT);
$stmt->execute();
$row = $stmt->fetchObject();
?>
```

This code queries the database by the *filmID* field which is the primary key in the *movies* table.

In the body of the HTML file use conditional logic to see if the form has been submitted and then output the results of the query.

```
<?php
if(isset($_GET['filmID'])){
      echo "<ul>";
      echo "<li>{$row->filmName}</li>";
      echo "</ul>";
}
?>
```

CALCULATING THE NUMBER OF RETURNED RECORDS

If no record is found we want to display a message. To do so we will use the `rowCount()` method of the statement object.

Amend the PHP code that performs the query as follows:

```
<?php
require('includes/conn.inc.php');
$sql= "SELECT * FROM movies WHERE filmID = :filmID";
$stmt = $pdo->prepare($sql);
$stmt->bindParam(':filmID', $_GET['filmID'], PDO::PARAM_INT);
$stmt->execute();
$totalnoFilms = $stmt->rowCount();
$row = $stmt->fetchObject();
?>
```

The value of the `num_rows` object has been stored in the variable `$numRows`.

Amend your code to use conditional logic to display a message if no records match the primary key used.

DISPLAYING DATES

When the $releaseDate field is output it will appear in the format YYYY-MM-DD.

To output the date in a more user friendly format use the PHP `date()` method.

```
if(isset($_GET['filmID'])){
	$timestampDate = strtotime($row->releaseDate);
	$displayDate = date("D d M Y", $timestampDate);
	echo "<ul>";
	echo "<li>{$row->filmName} - {$displayDate}</li>";
	echo "</ul>";
}
```

CREATING A SEARCH/RESULTS PAGE - MULTIPLE RECORDS

Open the file *prepare-multiple.php*. Here you will again find a HTML form. In this file build a prepared statement to query the database based on user input from a form. The user expects to be able to search by film name. As they are unlikely to know the exact name of the film you will need to use wildcards in the SQL.

- Add your connection include.
- Build a prepare statement.
- Create a variable that concatenates the SQL wildcard characters to the value received from the form.
- Bind your newly created value to the query using the `PDO::PARAM_STR` data type.
- Execute the query and bind the results.

In the body of the HTML file use conditional logic to see if the form has been submitted. Then use a while loop to retrieve and echo the results of the query. Output the film's name and release date, formatting the release date using the PHP `date()` function.