Charlie Wallin
April 15, 2017

Bowling: Adding and Deleting Records

Objectives

- Using an HTML hidden field to pass a value to another file
- Display records in a table
- Delete a record from a database using PHP and MySQL
- Add a record from a database using PHP and MySQL

Introduction

We will continue to use and modify the code from chapter 11 and the previous assignment in order to allow a user to add and remove data from the bowling database.

Database updated

Running bowling.sql locally

There was an error in the previous database so I have updated the bowlers.sql file. The SQL file is in Moodle. It is written so you can run the script without any modifications to your database. Since we didn't modify any records to the previous version of the database, this script will not overwrite any data that we use for this class.

Running bowling.sql on your webhost

X

Teams

- Start by creating a new folder named **proj02-teams** in your **my-code** folder.
- Copy the contents from proj01-teams and paste them into proj02-teams-view
- Start XAMPP and MySQL
- Make sure the program still works as before.

Refactoring

Refactoring means to go back into your code and rework it. Refactoring can be something as simple as renaming a variable or it can be a bit more involved such as creating a new folder structure to make your code more readable. This is part of what we will be doing during this phase of the project.

Create a View

- Create a new folder named **views** inside your **proj02-teams** folder
- Create a new file named **teams.view.php** and save it in your **views** folder.

In this section we are going to start to break up our code to continue designing our code into an MVC (Model View Controller) pattern. The MVC pattern breaks down as follows.

- Model: connecting to and communicating with the database
- View: any code that is displayed in the browser
- Controller: handles the logic of the program. It also acts as a traffic cop, redirecting calls from the Model and the view.

Modify your controller

In an MVC pattern the index.php file is the controller. Modify the **proj02-teams\index.php** file so that the first lines look like

```
<?php

echo '<a href="teams.view.php">Teams</a>';

?>
```

Then comment out the remaining lines for now.

Run the program. It should just show a link to Teams. At this point nothing will happen if you click on the link.

Add some HTML

For this part of the project, I'm going to concentrate on getting the pieces working then go back later and add Bootstrap to make it look nicer and responsive. For the time being you can use your own HTML5 boilerplate or copy and paste this code into the **teams.view.php** file. I left in a couple of new meta tags that came with the boilerplate from my editor.

As a side note, I'm trying the new editor from Microsoft for this project. It is called Visual Studio Code. I've been a big fan of Sublime Text and PHPStorm but this editor is winning me over and it works great on a Mac!

```
<!DOCTYPE html>

   <head>

       <meta charset="utf-8">

       <title>Bowling</title>

       <meta name="description" content="Bowling database for WEB 182">

   </head>

   <body>



   </body>

</html>
```

Now we need to add the PHP code that will show the teams in the bowling league.

In between the opening and closing `<body> … </body>` tags, paste the code that you commented earlier from your index.php file. Uncommented, the code should look like

```
<?php

// We need to put this 'include' line here temporarily to

// get the program to work. Eventually we will

// move all of our database code to the databases folder



include '../database/db-conn.php';

// create a new database object

$db = dbConnect();

// select all from teams

$query = "SELECT * from teams";

     $stmt = $db->prepare($query);

     $stmt->execute();

     echo '<h1>Teams</h1>';

     echo '<ul>';

     // display each returned row

     while($result = $stmt->fetch(PDO::FETCH_OBJ)) {

       echo "<li>" . $result->team_name . '</li>';

     }
```

```
    echo '</ul>';

?>
```

Test it

Test the code by starting on your **proj02-teams/index.php** file and click on the **Teams** link. This will redirect you to your view. It’s not terribly exciting at this point (it should look the same as your original index.php file) but there is more work to do.

Display as a table

Modify the view so the teams are listed in a table. We will also create a couple of new links for deleting and adding data. The PHP section of your **teams.view.php** code should now look like

```
 <?php

// create a new database object



// We need to put this 'include' line here temporarily to

// get the program to work. Eventually we will

// move all of our database code to the databases folder



include '../database/db-conn.php';

$db = dbConnect();

// query is "select all from teams"

$query = "SELECT * from teams";

     $stmt = $db->prepare($query);

     $stmt->execute();



     echo '<h1>Teams</h1>';

     echo '<table border="1" width="50%">';



     // display each returned row
     while($result = $stmt->fetch(PDO::FETCH_OBJ)) {

       echo "<tr><td>" . $result->team_name . '</td> <td><a
href="../index.php?add-team">Add</a></td> <td><a
href="../index.php?delete-team">Delete</a></td></tr>';

     }

     echo '</table>';

?>
```

We have modified the view so the output is in table format. The important part is the HTML anchor tags that add a bit of code so when they are clicked not only will they return to the main index.php page (the controller) they will tell the controller what you clicked. Those line of code are

```
<a href="../index.php?add-team">Add</a>
```

And

```
<a href="../index.php?delete-team">Delete</a>
```

Save the file and return to your **index.php** (controller) to start on the main page, Click on Teams, then click on one of the links to **Add** or **Delete**. This will return you to the controller. Notice in your URL that the address will now look like (your 'localhost' may look different)

```
http://localhost:8888/index.php?delete-team
```

The content `?delete-team` is an associative array in PHP and can be accessed using the `$_GET[]` superglobal array.

It doesn't do anything yet other than give us a way to tell the controller what to do.

Test the ?delete-team and ?add-team

To test that the ?delete-team=true or ?add-team=true let’s modify the original index.php with the following code

```
<?php

/*

*  Need to test if the Add or Delete has been clicked.

*  if they haven't, then display the default page.

*/



// Report all errors except E_NOTICE

error_reporting(E_ALL & ~E_NOTICE);



if ($_GET['delete-team'] == true)

{

   $action = 'delete-team';

   echo "<p>Test to see action: $action</p>";



}

elseif ($_GET['add-team'] == true )

{

   $action = 'add-team';
   echo "<p>Test to see action: $action</p>";

}



else

{

   echo '<a href="views/teams.view.php">Teams</a>';

}

?>
```

Because GET had not been set the first time the page loads, I used the `ini_set()` function to show all error except for Notices. After you get the code to run try uncommenting the `ini_set()` function and see what happens. This is kind of a hack but it works for now.

Bowlers

Now that we have been through setting up the teams. You job is to set up the same scenario for the bowlers. Here are the steps.

- Add a bowlers link to the main index.php page
- Create a view named bowlers.view.php
- The bowlers.view.php file will display just the bowler’s first name, last name and email address. It will also have links to Add and Delete a bowler.