Project: Part 1

# Objectives

* Load a new database
* Create a small website
* Use version control

# Assignment

This is the beginning of the project for the term. We will build a small database-driven website with a user login and registration system. The theme of the project is a bowling league. It’s called 10 Pin.

* Create a folder named bowling
  + Create a file named config.php
* Create folders named
  + styles
  + views
  + functions

## Version Control

Even though we haven’t really started coding, it is time to start with version control. Open your gitBash (Windows) or Terminal (Mac) and navigate to your web182\bowling folder then run the following commands.

$ git init

$ git add .

$ git commit -m “Initial commit for the term project”

$ git status (check where you are at)

$ git

### Branch it

We don’t want to work on our Master branch. We will use it as our “production” branch; meaning that whenever we are ready to commit to production we will come back to the Master branch before pushing our code up to the server. Create a new branch called new-feature. Technically you may name it whatever you wish.

$ git checkout -b new-feature

$ git branch (so you can see that you have moved to the new branch)

Now that is set up it is time to create our database

## Database

First, we will load up the database. Download and run the bowlers.sql file. This is the same process as we did with the text book’s test.sql file.

Create users

Open the SQL tab in PHPMyAdmin and type in the following commands to create a new user and associate it with this database. To keep it simple, we will use the database name, bowling for the user and the password

CREATE USER ‘bowling’@’localhost’ IDENTIFIED BY ‘bowling’;

GRANT ALL ON bowling.\* TO ‘bowling’@’localhost’

## Test your connection

* Open your **index.php** file and write the code that
* Connects to the database
* Runs a query that displays the bowlers first name and last name in ascending alphabetical order

## Version control

Once your code is running correctly it is time to commit and merge it. Open your Terminal or gitBash window

$ git status

$ git add .

$ git commit -m “bowling database connection is complete.”

Now you will need to go back to your Master branch

$ git checkout Master

$ git branch (see that you are on the Master)

$ git merge feature

### Screenshot of git log

Run your git log command and take a screenshot of it. Name the screenshot, **gitlog.png**

$ git log

Next class we will learn how to push our code to GitHub.

## Push your code to your webhost

Now that everything is working, it is time to push your code to your webhost. Remember that you will need to create the database on your host, create a user and associate that user with the database. You will also need to change your **config.php** information on the host so that it matches your host’s naming convention.

# Submit you work

All you need to submit it the gitLog.png file. I will go to your webhost to see that you have the bowlers displayed.

Part 2: Create a login and logout

# Objective

* Create a 3 page website
* Use PHP sessions to login
* Use PHP sessions to logout
* Continue working with version control

# Introduction

As we continue to build the project, we will create a “proof of concept” where a user from the 10Pin Bowling League can

* log in where the program will create a session
* log out where the project will destroy the session.

In the next part we will start to build a nicer looking site using Bootstrap and refactor our code. As always, git will become part of the workflow.

## Reading and code

Review the section in chapter 6 that contains the information on creating and destroying sessions.

Read the code in the **samples\sessions** folder. All of the code follows a similar pattern. The math-quiz.php and math-quit.php files provide a nice example. You may want to read through the gold hunter example as well. Be sure to run the code and see how it works. In addition you will want to run the Web Development toolbar and examine the cookies.

# Assignment

## Branch

Open gitBash (Windows) or the Terminal (Mac and Linux). Mac users may want to download another Terminal called iTerm2. It is a terminal window like the one that comes with the Mac OS but it includes a lot of nice features that makes using it easier (maybe even fun!).

If you are on your Master branch, then commit with the following before moving to your branch. I have added in extra git status commands so you can see what is happening while you work.

$ git status (just to check your status)

$ git add .

$ git commit -m “Preparing code for bootstrap and moving to new-feature branch”

$ git status

$ git branch (see what branch you are on and what branches are available)

Create a new branch

$ git checkout -b dev-login

$ git checkout dev-login

$ git status

Now we are ready to go!

## Note about password

Currently our database is missing a password field and any security for that matter. We will work on that in an upcoming assignment.

## Save the connection code

In Part 1 we created some code to connect to the database and display the **first\_name** and **last\_name** from the bowlers database. We will reuse a lot of this code so let’s save it

* Copy **index.php** and save it as **connection-example.php**
* Delete all of the code in **index.php**

In the file manager, copy the folder from the previous assignment and name the new folder **bowling02**.

In the previous assignment we created a test to make sure we could connect to a database and display the data. We want to save that code. Copy the **index.php** file and name it **connection-code.php**.

### Build the index.php

* Delete all of the code in the index.php file
* replace it with an HTML5 template.
* Create a form that accepts the user’s first and last name. The form will also need a submit button. Don’t worry that the form looks ugly. We will fix that in an upcoming assignment.
* Name the action for the form to **login.php**
* No need to set a session on this page.

### Build the login.php

login.php will

* Start a session
* Accept the first and last name values
* Test to see that they are not empty
* If either is empty then display a message to the user and provide a back link to get them to the **index.php** page
* If both fields are filled in, then the program will run a database query to see if there is a match in the bowlers table. Refer to the **SELECT** query example in your book.
* If there is no math, then the program will display an error message and provide a link back to the **index.php** page
* If there is a match, the program will set a session with the user’s first name and provide a link that allows them to look at all of the bowlers in the league. This is the same query that you wrote for the previous assignment.
* When a user clicks on the display bowlers link, it will take them to a new page named **display-bowlers.php**.

### Create the display-bowlers.php

* The **display-bowlers.php** page will check that the first name session exists and display that user’s first name at the top of the page with the message, **Logout {first\_name}.** Notice that we are able to pass the user’s first name to the **display-bowlers.php** page without using an HTML form! This is where the power of sessions come into play.
* Create a **logout** link, that will destroy the session and return the user back to the index.php page.

## Back to version control

Once you have everything running be sure to run the following git commands

$ git status

$ git add .

$ git commit -m “finished writing the first login and logout pages”

$ git checkout master

$ git status (if you forgot to commit previously the do so now)

$ git merge dev-login

## Upload to GitHub

We will go over this in class.

## Upload to your Webhost

Make sure you test your code to see that it works on your site. I will see that the code runs on your host.

## What to turn in

Your GitHub account URL. I will go out the GitHub and pull down your code and run it.

Part 3: Registration

# Objectives

* Write a registration page that validates and enters a new user into the database.
* Continue working with git and gitHub

# Introduction

You may either use your own code that you have been working on up till this point or download the code in Moodle as a starting point.

# Hashing a password

We will be using the [password\_hash()](https://secure.php.net/manual/en/function.password-hash.php) function to encrypt the password. To test how it works, try out the following in your sandbox. Create a folder named sandbox in **htdocs\web182** if you haven’t done so. A sandbox is a place to “play” with code.

create a file named **hash.php**

Paste in the following code and run it to see how the hash works

$pass = 'Hello';

echo password\_hash($pass, PASSWORD\_DEFAULT);

You will be saving the hashed password in your database. This is a one-way encryption.

# Version Control

Talk with you instructor if you have any questions.

Always work on a development branch and not your Master branch. Only use the master branch when your work on the development branch needs to be merged.

This assumes that you are working from the code provided in Moodle

Check your status

$ git status

Check what branch you are on

$ git branch

Make sure you are working on a clean branch. In this case the master branch

$ git add .

$ git commit -m “login complete. Starting to build the registration”

Create a new branch (you may also continue working on your previous development branch)

$ git checkout -b dev/registration

This will switch you to the branch.

# Files

Create two files and store them in your web182\bowling directory

registration-form.php

registration.php

## Modify index.php

Add a link to your index.php file. The link is named registration and it should point to a file named registration-form.php. Even though this is a relational database and a bowler should be a member of a team, we will start by getting getting the basics of the registration complete before addressing that.

### registration-form.php

This file allows the user to enter the following information

* firstName
* lastName
* email
* password

The password should use the HTML password field instead of text to cover the data with bullets.

All of the text fields should use the HTML5 required keyword. This will give us a little bit of validation on the client side; however, we will still need to write client side validation in PHP later.

After clicking on the Submit button, the data entered in the form is sent to the registration.php file.

## Algorithm for registration.php

Receives first name, lastName, email, password and teamNum from registration-form.php

### Client-side Validation

If any of the fields are empty, then display a message to the user that all fields need to be entered.

Test for database connection. If the database cannot connect, then display a message to the user.

Test for a SQL error. If the SQL statement produces and error, the display a message to the user

Test that the INSERT query ran correctly. Here is a link to the [mysqli\_affected\_rows](https://www.w3schools.com/php/func_mysqli_affected_rows.asp) function. Read through this example from w3schools. Run the code in your sandbox if you need to. This is the way to start building and testing small programs to see if they behave as required before moving on.

### Insert query with password hashing algorithm

Here is some [quick reading on the INSERT statement](https://www.w3schools.com/sql/sql_insert.asp)

You already have an example of using an INSERT query from the text book. We will now build it up by adding a password hash. If you are having difficulty with this part, then get the query to run first without the password hash and add it later. Again, you could write this little bit of code in your sandbox.

Here is some code to get you started with the INSERT query and an encrypted password

include 'config.php';

$firstName = "Jean";

$lastName = "Connor";

$email = "jean@me.com";

$pass = "hello"; // not a good password -- for example only

include('config.php');

$connect = mysqli\_connect(SERVER, USER, PW, DB);

if(!$connect)

{

exit("Error could not connect to the database.");

}

// hello is the password in this example

$pass = password\_hash('hello', PASSWORD\_DEFAULT);

$query = "INSERT INTO bowlers (email, pass, first\_name, last\_name) VALUES ( '$email', '$pass', '$firstName', '$lastName')";

$result = mysqli\_query($connect, $query);

Test that your code ran correctly by looking in your database using PHPMyAdmin (or the terminal). You may need to refresh the data.

### Display a message to the user

If the query runs correctly then display a message to the user and provide a link for the user to return to the main page to login. Ideally there are better ways to handle this, but we aren’t quite there yet.

### Test your new entry

Now that your registration works be sure to test it with a login.

# Version Control

You have been developing your code on a git branch and it works correctly. It is time to commit it, merge with the master branch and push it to your GitHub account.

While you are on the dev-registration branch

$ git add .

$ git commit -m “completed the first round of registration. Going to merge it with the master branch”

Change to the master branch

$ git checkout master

$ git status (see where you are at. Commit if you need to)

$ git merge dev/registration

$ git push -u origin master

# Upload to your web host

Upload your code to **web182\bowling**.

\*Be sure to update your database fields to include the email and password fields. It is important that these fields are spelled exactly the same as your local database.

There is no need to upload your code to Moodle. I will pull it down from GitHub to look at it.

Part 4: Organizing views and bootstrap

# Objectives

* Complete registration and login using sessions
* Refactor code
* Continue with “separation of concerns”
* See where MVC breaks down without routing
* Use bootstrap and/or CSS to make the site look more presentable
* Continue working with version control

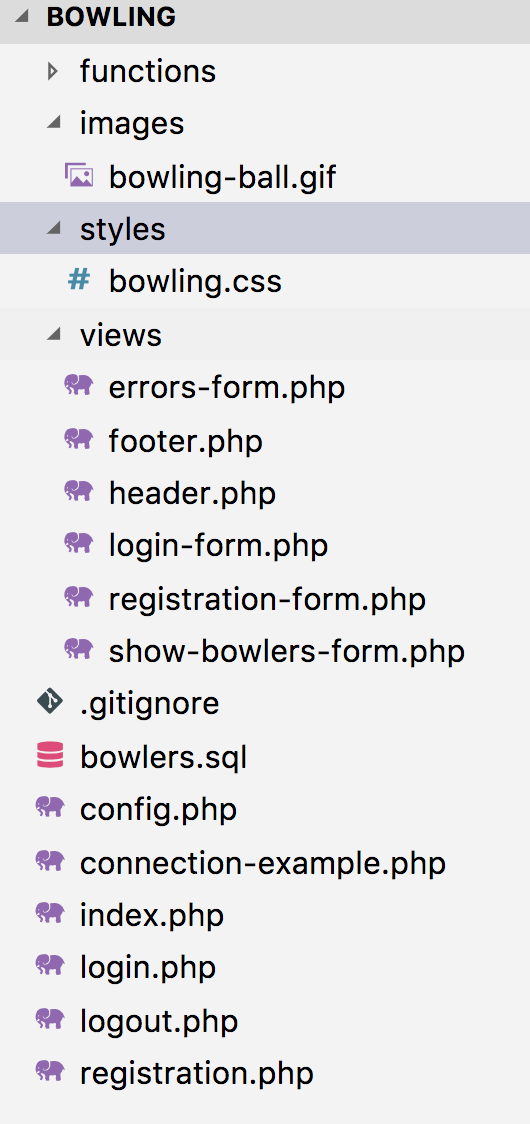
# Instructions

You can view the project at <http://charliewallin.com/web182/bowling/>.

Use this as a visual guide. You may take liberties and dress up the site more if you like.

## File structure

We are continuing to work toward an MVC (Model View Controller) architecture. In particular, we want to continue with the concept of “separation of concerns”. At this point it means we are separating our forms from the logic of our code. Here is a screenshot of the file structure.



Note that the errors-form is optional at this point.

We will go over the problems with this current structure and how the code can continue to be refactored so it is easier to worth with.

Notice that all forms (pages that only display data) are in a views folder. In addition, the logic of the program, index.php, login.php, logout.php and registration.php are in the root of the program. Ideally we will want to put them in their own folder. In MVC, these files would be considered controllers. They control whether we need to contact a database, display data or something else entirely. For now they are fine where they are at.

# Version control

Here is a very brief way of emulating part of the git workflow

Navigate to your working folder

If you have been working on a branch other than master then use git status to see if you need to make a commit.

$ git add .

$ git commit -m “some message describing the status”

Create a branch named develop

$ git checkout -b develop

This will automatically move you to the development branch

Create a new branch named feature\_registration

$ git checkout -b feature\_registration

You are now on the feature\_registration branch and you can start coding.

Commit your code on the feature branch once you are done

Go back to the develop branch

$ git checkout develop

Merge with the feature\_registration

$ git merge feature\_registration

Create a new branch named release/0.1.0

This will automatically put you on this branch. Go back to the develop branch to merge

$ git checkout develop

Push your release code to GitHub

$ git push -u origin release/0.1.0

Visit GitHub to see that your code is present.

# Push your code to your web host

Now that you have all of that finished, push all of your code to your web host so you can see that it works. Remember that you may need to reconfigure your config.php file or just make sure you don’t overwrite it.