**Setup:**

Ensure that you note the PORT for MySQL – here it is 3308



The port is specified in the database library include file: model/db.php

Change this if required, before you proceed.

1. Obtain solution code for this task as you will work on this today.

<https://github.com/leeagles/WBSSimplePHPMVC>

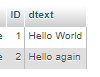
Extract the files into a directory called ‘Simple’ (Capital first letter) in the root of your USBWebServer or WAMPServer.

Once extracted you will see the following directories:

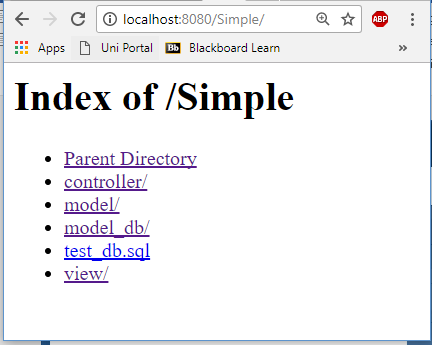
|  |  |
| --- | --- |
| **view** | Contains basic website that display data and separate views for each query |
| **controller** | Contains code that gathers data via the Web API |
| **db** | Contains the database interaction scripts |
| **webapi** | Script to gather data from the database and return it to a WEBAPI client in JSON format |

1. Ensure you have the following table in your database test\_db:

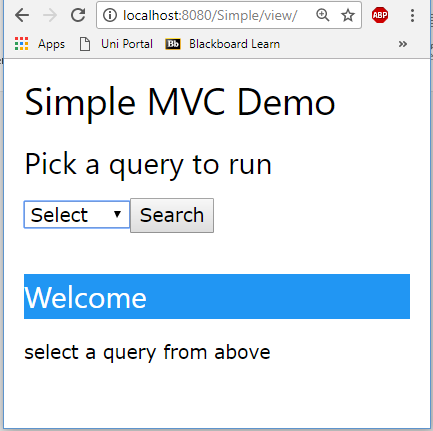
Table is: test\_table



Select Localhost from USBWebServer to reveal the directories available:

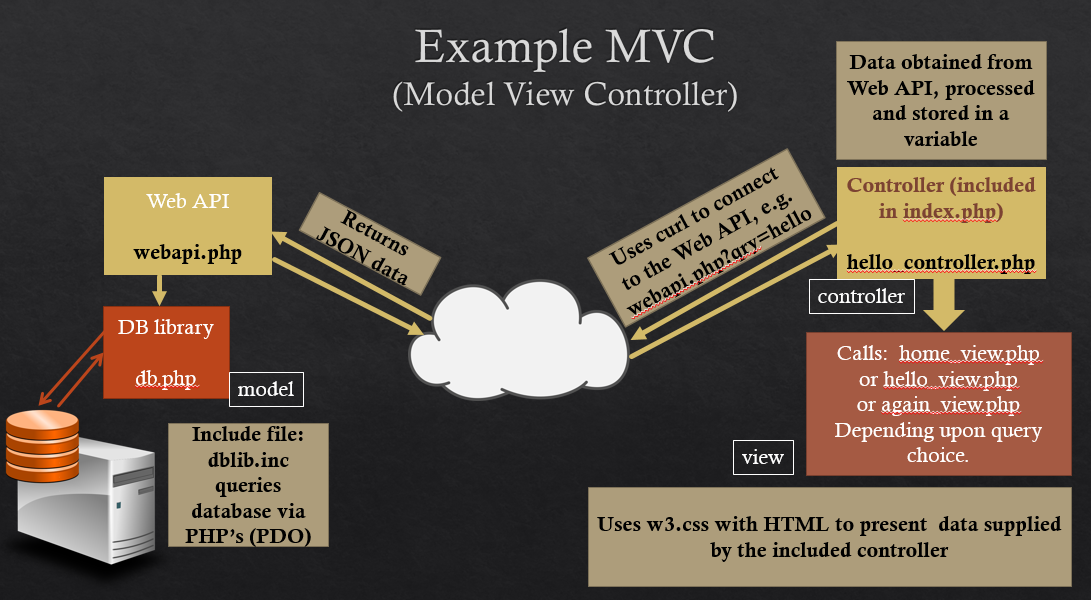


Select view to see the query drop down box:



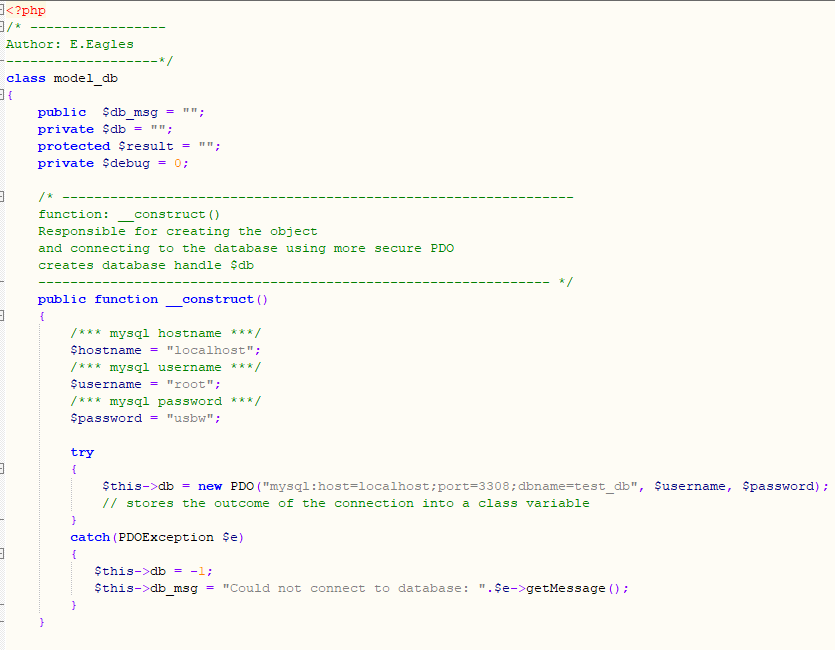
You will see that for each option you will see a different view.

Code explanation:

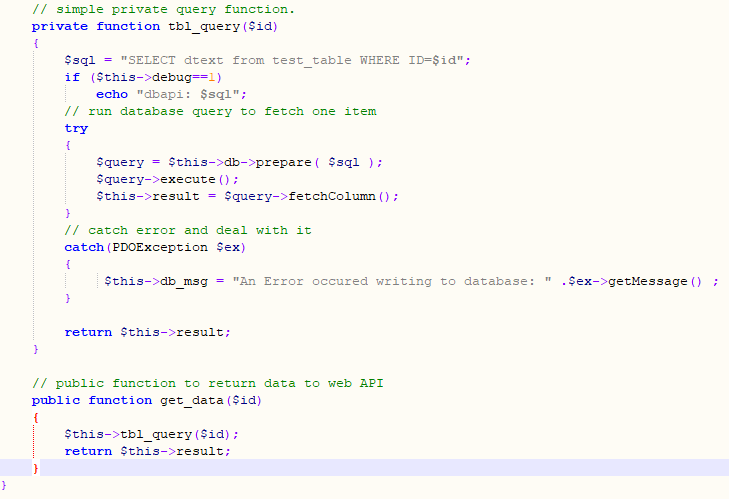


db.php contains a basic database connection script (using object-oriented techniques and PDO) and two query functions: a private function for internal use and a public function called by model/webapi.php. This file is stored in the model directory.

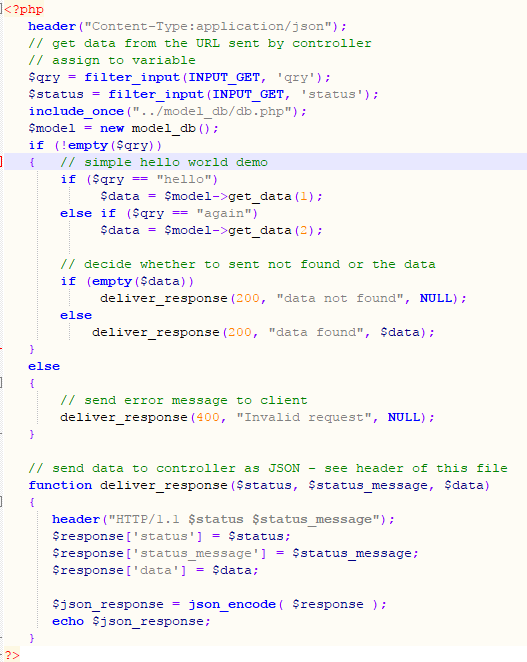
The following are the class variables and constructor which when called will connect to database: test\_db (port 3308).



Below is in the same file (db.php) and contains the two query functions:



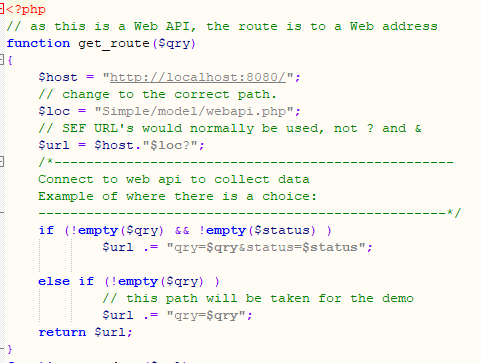
The public function is called from model/webapi.php, which is shown below:



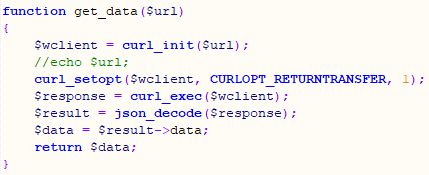
Deliver\_response sends the data generated by the web api script, which should contain either a “data not found”, an “invalid request” message or the data retrieved by the db.php function.

The web api is called by the controller script: hello\_controller.php:

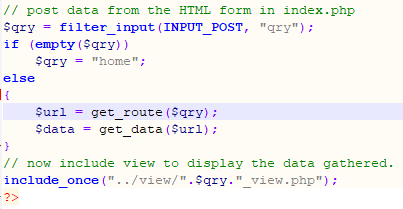
The first part of the controller script is a function that is used to decide which route to take: how to query the web api (of course, much of this would be passed via a config file):



Below this is the function to connect to the URL and obtain the Web API’s JSON data:

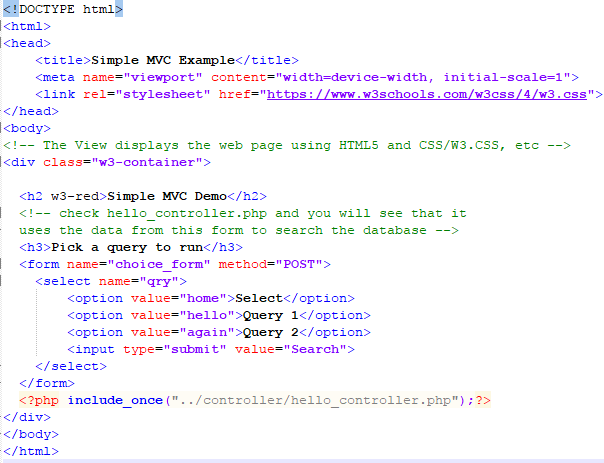


To run the above, the controller passes the HTML form data from index.php to get\_route(…), which constructs the URL for function get\_data(…):



Finally, as can be seen at the bottom of the script (procedurally written this time), a view files is then included, based on the query data passed from the form.

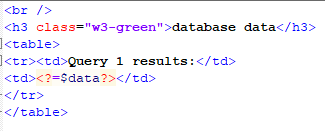
The file that brings the website together is index.php:



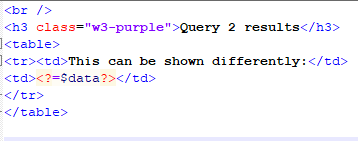
As can be seen, there is only one include file, meaning the view files can easily be changed.

As for the controller included view files, they are shown below (kept simple for the example):

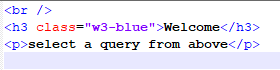
view/hello\_view.php (included if $qry = “hello”):



view/again\_view.php (included if $qry = “again”):



view/home\_view.php (included if $qry = “home”):



Experiment by adding more form items, further queries in the database and changing the display – use Bootstrap for the web page layout.