

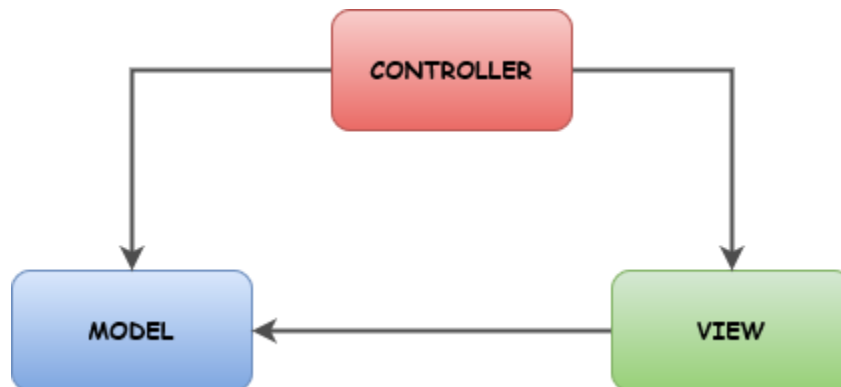


SCS2208 - Rapid Application Development

MVC framework with PHP Tutorial

In this tutorial we will try to create a simple MVC framework using PHP from scratch.

Model-View-Controller(MVC) architecture is a design pattern that separates the business logic and the presentation layer from each other. It categorises or separates the application into three components namely **Model**, **View** and **Controller**. It is a popular design pattern in the development of web applications and mobile applications.



- **Model:** corresponds to all data related logic that the user works with.
- **View:** corresponds to UI related logic. Presents data to the user and handles interaction.
- **Controller:** An interface between the model and view components.

Requirements for this tutorial:

- PHP
- My SQL

Let's implement a basic MVC framework with PHP together.

Objective: Developing a simple data retrieval application with PHP as MVC framework.

Environment Setup

- **Install Apache, MySQL, PHP in Windows**

- <https://codebriefly.com/how-to-setup-apache-php-mysql-on-windows-10/>
- <https://www.sitepoint.com/how-to-install-php-on-windows/>
- Install Xampp: <https://www.apachefriends.org/index.html>

Or Install Wamp: <https://www.wampserver.com/en/>

- **Install Apache, MySQL, PHP in Ubuntu 18.04**

- <https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-ubuntu-18-04>
- Install Apache
 - \$ sudo apt-get update
 - \$ sudo apt-get install apache2
- Install MySQL
 - \$ sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql
- Install PHP
 - \$ sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
- Install phpMyAdmin
 - \$ sudo apt-get install phpmyadmin

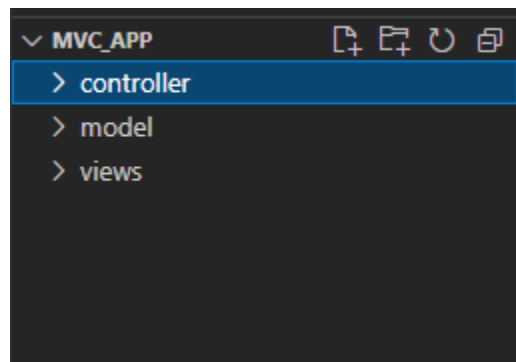
Let's begin ...

1. Create your project folder in htdocs (xampp) or www (wamp) folder

Example: Create a folder named mvc_app in htdocs folder (if you're using xampp)

2. Create 3 folders namely controller, model, and views within your application (mvc_app) folder

Directory structure will be as follows:



Now that you have created the necessary folders with which you want to develop a simple MVC.

3. First of all, let us create a simple Hello world MVC with php.
4. For this, create a controller file first.

Move into the Application/controller folder and create a file named hello.php with the code as follows.

```
1  <?php
2
3  class Hello extends Controller {
4
5      function __construct() {
6          parent::__construct();
7      }
8
9      function index() {
10
11
12          $this->view->render('hello/index');
13
14      }
15  ?>
```

Controller takes the user input and updates the model when required. Controllers are not necessary when there is no user interaction. But in this hello world example even though there is no user interaction, we are creating a controller file just to make you familiar with controllers.

Line 5: function __construct()

A constructor allows you to initialize an object's properties upon creation of the object. PHP will automatically call this constructor function when you create an object from a class.

Line 12: \$this->view->render('hello/index');

Within the controller, render() function renders a named view. Here in our example, it renders the index view

5. Then create a view file that shows a message as 'Hello world'.

Get into the Application/views folder. Create a folder with the name hello and within that folder create a file named index.php to display a welcome message.

Add the following code into the index.php file and save it.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>Hello World</title>
5  </head>
6  <body>
7  |
8  |   <p>Hello World !!!</p>
9  | </body>
10 </html> |
```

The view contains all the display logic. It will be the part of the application which generates the HTML in the case of PHP. View has direct access to the model and can query the model to get its data. The View can create callbacks to its controller as well.

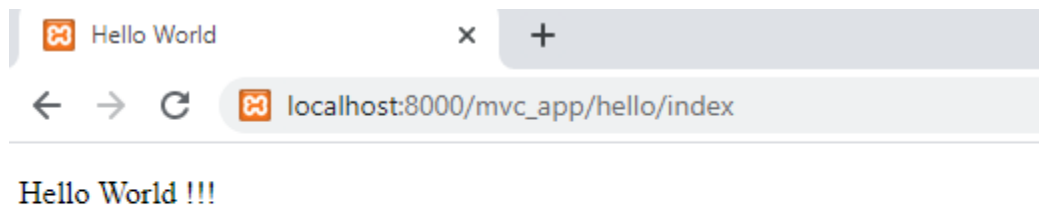
- Check whether the basic hello world example is working. For this purpose, start the apache and mysql options in the Xampp window.

- In your browser go to localhost and give your address correctly and view the page on the web browser.

`http://localhost/mvc_app/hello/index`

Or

`http://localhost:<port_number>/mvc_app/hello/index`



We have successfully done a Hello world example as an MVC architecture in PHP. Here we didn't use the model since there was no data logic in this example.

Now let us build a simple data retrieval MVC application with php.

Here we will build a simple mvc web application where we can search the users with their name. For this purpose we must have a database with the names of users. Then only we can retrieve data through our application from that database.

1. Go to the phpmyadmin page on your browser.

`http://localhost/phpmyadmin`

Or <http://localhost:8000/phpmyadmin>

2. Create a database with a preferred name (You can name it as mvc).

<https://dev.mysql.com/doc/refman/8.0/en/creating-database.html>

3. Within that database, create a table with the name 'users' and insert some dummy data.

```
1 CREATE TABLE `users` (  
2   `id` int(11) NOT NULL,  
3   `name` varchar(255) NOT NULL  
4 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
5  
6 ALTER TABLE `users`  
7   ADD PRIMARY KEY (`id`),  
8   ADD KEY `name` (`name`);  
9  
10 INSERT INTO `users` (`id`, `name`) VALUES  
11 (24, 'Abby Ward'),  
12 (21, 'Aleksandra Devine'),  
13 (43, 'Aston Simmonds'),  
14 (47, 'Aston Simmonds'),  
15 (15, 'Beth Skywalker'),  
16 (26, 'Bridget Wooten'),  
17 (10, 'Coby Kelleigh');
```

You can save the above code snippet as a single file (users.sql) and import it to create a table in your database. Or manually create a table and enter the users' data.

<https://dev.mysql.com/doc/refman/8.0/en/creating-tables.html>

<https://www.tutorialspoint.com/mysql/mysql-create-tables.htm>

In our code snippet,

Line 1: Creates a table named users

Within which you can define your attributes (id and name in this example)

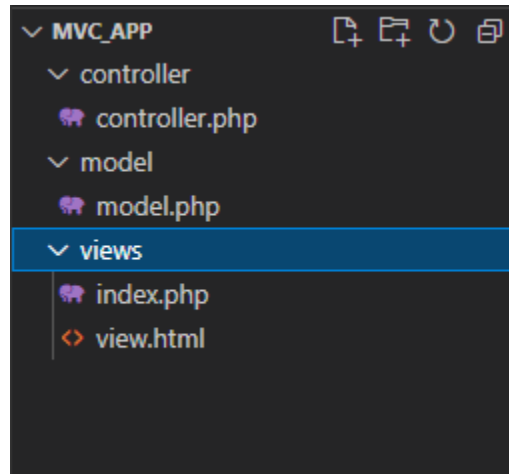
Line 6: Alters/ edits the table named users

Line 10: 'INSERT INTO' is used to add dummy data to the table we created.

With that, our table with user data is ready.

Let us build our MVC web application now.

Keep the same directory structure discussed in the previous example. But create different files as shown below.



4. Create a model file named model.php within the model folder. It will include the database settings, parameters to connect to the database and management of sql queries

```
1  <?php
2  class DB {
3      // (A) CONNECT TO DATABASE
4      public $error = "";
5      private $pdo = null;
6      private $stmt = null;
7      function __construct () {
8          try {
9              $this->pdo = new PDO(
10                 "mysql:host=".DB_HOST.";dbname=".DB_NAME.";charset=".DB_CHARSET,
11                 DB_USER, DB_PASSWORD, [
12                     PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
13                     PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
14                 ]
15             );
16         } catch (Exception $ex) { die($ex->getMessage()); }
17     }
18
19     // (B) CLOSE CONNECTION
20     function __destruct(){
21         if ($this->stmt!=null) { $this->stmt = null; }
22         if ($this->pdo!=null) { $this->pdo = null; }
23     }
24 }
```

```

25 // (C) RUN A SELECT QUERY
26 ✓ function select($sql, $cond=null){
27     $result = false;
28     try {
29         $this->stmt = $this->pdo->prepare($sql);
30         $this->stmt->execute($cond);
31         $result = $this->stmt->fetchAll();
32         return $result;
33     } catch (Exception $ex) {
34         $this->error = $ex->getMessage();
35         return false;
36     }
37 }
38 }
39
40 // (D) DATABASE SETTINGS - CHANGE TO YOUR OWN!
41 define('DB_HOST', 'localhost');
42 define('DB_NAME', 'mvc');
43 define('DB_CHARSET', 'utf8');
44 define('DB_USER', 'root');
45 define('DB_PASSWORD', '');

```

<https://stackoverflow.com/questions/37005255/how-to-connect-to-database-using-mvc-in-php/37005531>

In our code snippet,

Line 2 - Line 17: Creates connection to database

Line 20 - Line 23: Closes connection to database

Line 26 - Line 38: Creates a select query function

Line 41 - Line 45: Defines the database settings

Line 41: Defines the host of your database (default is 'localhost')

Line 42: Defines the name of the database (which you gave while creating the database in phpMyAdmin)

Line 43: Defines the database character set (default is 'utf8')

Line 44: Defines the username of the database (default is 'root')

Line 45: Defines the database password

5. Create a controller file named controller.php within the controller folder. Use the following code in the controller file.

```
1  <?php
2  // (A) DATABASE CONNECTION
3  require "../model/model.php";
4  $DB = new DB();
5
6  // (B) SEARCH FOR USERS
7  $results = $DB->select(
8      "SELECT * FROM `users` WHERE `name` LIKE ?",
9      ["%{$_POST['search']}%"]
10 );
11
12 // (C) OUTPUT RESULTS
13 echo json_encode(count($results)==0 ? null : $results);
```

In line 3, “../model/model1.php” is used since this controller file is in a folder named controller and model file is in a separate folder named model. If we don’t put these files in separate folders, we can directly use the model file name only in the controller file as “model1.php”.

In this example controller acts as an endpoint that accepts POST search requests from clients and uses the model to search and return the results.

Line 7 - Line 10: Implies the select query function that we defined in our model file. Query is given as to select all from users table when the name is like the name gained through post request from the form included in the view file.

<https://stackoverflow.com/questions/1737868/an-example-of-an-mvc-controller/1737903>

6. Then create the view file. It is just the user interface to be displayed. In our example only HTML, CSS and Javascript are used in this view file.

You can name this file as view.php or since we are just using HTML and CSS here, you can name it as view.html also.

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>MVC (User Interface)</title>
5    </head>
6    <body>
7      <!-- (A) SEARCH JAVASCRIPT -->
8      <script>
9        function doSearch () {
10          // (A1) GET SEARCH TERM
11          var data = new FormData();
12          data.append("search", document.getElementById("search").value);
13
14          // (A2) AJAX - USE HTTP:// NOT FILE://
15          var xhr = new XMLHttpRequest();
16          xhr.open("POST", "../controller/controller.php");
17          xhr.onload = function(){
18            let results = document.getElementById("results"),
19              search = JSON.parse(this.response);
20            results.innerHTML = "";
21            if (search !== null) { for (let s of search) {
22              results.innerHTML += `<div>${s.id} - ${s.name}</div>`;
23            }}
24          };
25          xhr.send(data);
26          return false;
27        }
28      </script>
29
30
31    <!-- (B) SEARCH FORM -->
32    <form onsubmit="return doSearch()">
33      <input type="text" id="search" required/>
34      <input type="submit" value="Search"/>
35    </form>
36
37    <!-- (C) SEARCH RESULTS -->
38    <div id="results"></div>
39  </body>
40 </html>

```

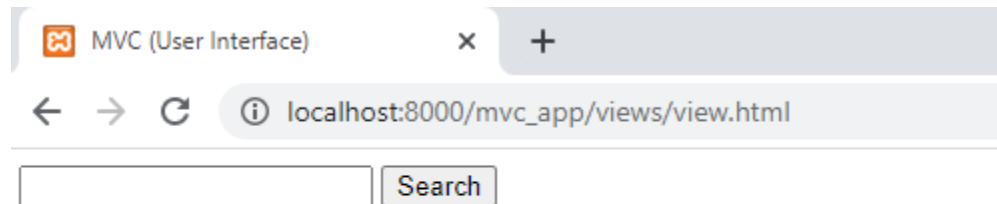
A simple search form is included in this view file. Comments are given in the code snippet attached above, so that you can understand the functionality of those particular code blocks.

Line 9 - Line 29: Javascript is used to get the search term and to send it as a POST request to our controller. And it gets the result to be displayed.

Line 32 - Line 35: Simple search form is created. It returns the search function(doSearch) that we initialized in the scripting section (Line 9 - Line 29) when the user clicks the 'Search' (Submit) button.

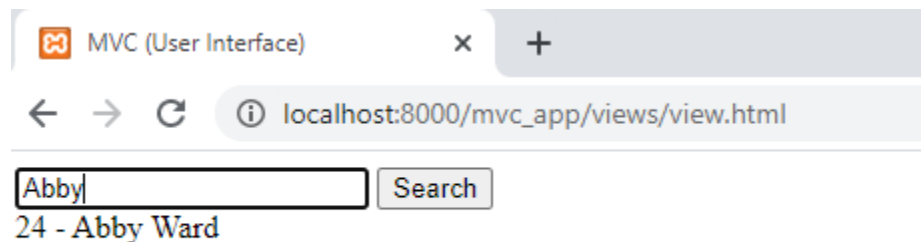
Line 38: 'results' is the id used to search the term.

7. Go to your browser and check the page

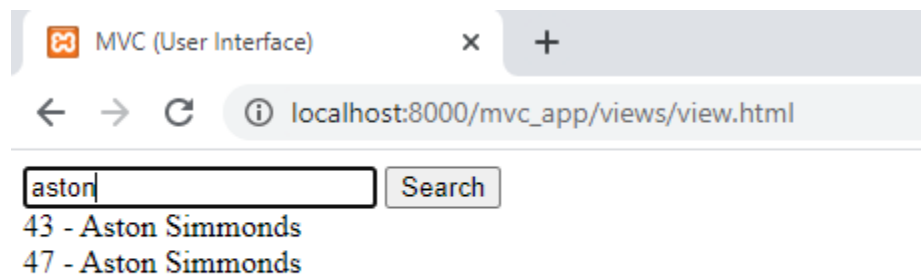


8. Search for a name you have in the database table.

For example, search Abby



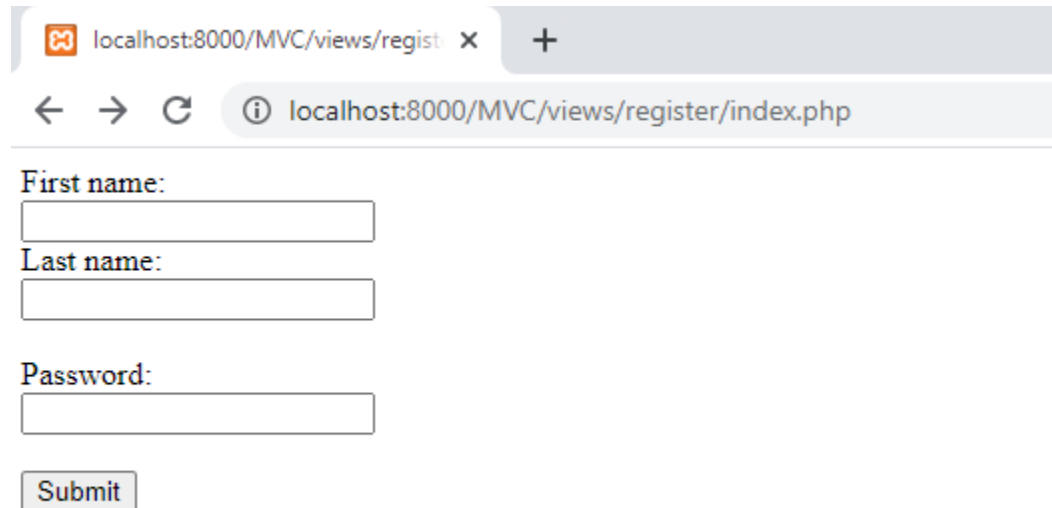
Search another name



We have successfully built an MVC application with php to retrieve data from the database. Likewise you can add, update and also delete data using this kind of an application.

Task

Develop a user registration form as shown below with the MVC architecture using php. You must be able to use this form and add user data to the database table.



The screenshot shows a web browser window with the address bar displaying 'localhost:8000/MVC/views/register/index.php'. The page contains a registration form with the following elements:

- First name:** A text input field.
- Last name:** A text input field.
- Password:** A text input field.
- Submit**: A button to submit the form.

Useful Resources:

- <https://www.youtube.com/watch?v=6ERdu4k62wI> (> 6 hrs)
- <https://www.youtube.com/watch?v=OsCTzGASImQ&list=PLfdiltiRHWGXVHXX09fxXDi-DqInchFD>
- <https://magemastery.net/courses/user-registration-with-php-mysql/form-validation>
- https://www.w3schools.com/howto/howto_js_password_validation.asp
- <https://www.cluemediator.com/how-to-validate-password-strength-in-php>
- <https://code-boxx.com/simple-php-mvc-example/>
- <https://stackoverflow.com/questions/37005255/how-to-connect-to-database-using-mvc-in-php/37005531>