**PHP MVC Framework Tutorial: CodeIgniter Example**

**What is PHP MVC framework?**

PHP MVC is an application design pattern that separates the application data and business logic (model) from the presentation (view). MVC stands for Model, View & Controller.

The controller mediates between the models and views.

Think of the MVC design pattern as a car and the driver.

The car has the windscreens (view) which the driver (controller) uses to monitor traffic ahead then speed or brake (model) depending on what he sees ahead.

**Why use PHP MVC Framework?**

* PHP MVC Frameworks simplify working with complex technologies by;
  + Hiding all the complex implementation details
  + Providing standard methods that we can use to build our applications.
  + Increased developer productivity, this is because the base implementation of activities such as connecting to the database, sanitizing user input etc. are already partially implemented.
  + Adherence to professional coding standards

**In this tutorial, you will learn-**

* [PHP MVC Design Pattern](https://www.guru99.com/php-mvc-frameworks.html#3)
* [Types of PHP MVC framework](https://www.guru99.com/php-mvc-frameworks.html#4)
* [Porting the opinion poll application to CodeIgniter](https://www.guru99.com/php-mvc-frameworks.html#5)
* [Database configuration settings](https://www.guru99.com/php-mvc-frameworks.html#6)
* [Creating Our Model](https://www.guru99.com/php-mvc-frameworks.html#7)
* [Creating Our Views](https://www.guru99.com/php-mvc-frameworks.html#8)
* [Testing our application](https://www.guru99.com/php-mvc-frameworks.html#9)

**PHP MVC Design Pattern**

Let’s now briefly discuss each component of the MVC design pattern.

**Model** – this part is concerned with the business logic and the application data. It can be used to perform data validations, process data and store it. The data can come from;

* flat file
* database
* XML document
* Other valid data sources.

**Controller** – this is the part deals with the users’ requests for resources from the server.

As an example, when the users requests for the URL *…/index.php?products=list*, the controller will load the products model to retrieve the products data then output the results in the list view.

In a nutshell, the controller links the models and views together depending on the requested resources.

**Views** – this part deals with presenting the data to the user. This is usually in form of HTML pages.

**Types of PHP MVC framework**

Selecting the best PHP framework is a challenge.

You don’t have to write your own framework to benefit from the advantages of MVC.

You should only attempt to create your own MVC related application design for understanding how MVC frameworks work.

Once you are comfortable with the way MVC frameworks work, you should move on to the mature and already tested frameworks.

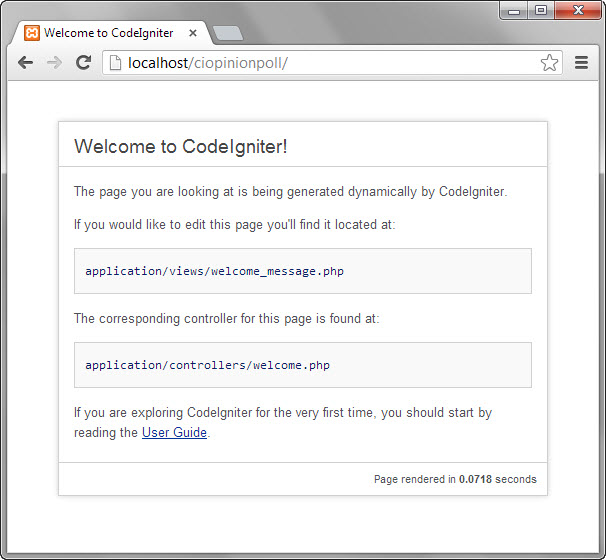
The table below briefly describes some of the popular php frameworks and the features that each framework offers.

| **Framework** | **Description** |
| --- | --- |
| [Learn PHP MVC Frameworks with CodeIgniter Example](https://www.guru99.com/images/2013/04/code_igniter.gif)  CodeIgniter  <http://ellislab.com/codeigniter> | It is one of the most popular PHP MVC frameworks. It’s lightweight and has a short learning curve. It has a rich set of libraries that help build websites and applications rapidly. Users with limited knowledge of OOP programming can also use it. CodeIgniter powered applications include;   * <https://www.pyrocms.com/> * <http://www.shopigniter.com/> |
| [Learn PHP MVC Frameworks with CodeIgniter Example](https://www.guru99.com/images/2013/04/kohana.png) Kohana  [http://kohanaframework.org](http://kohanaframework.org/) | It’s a Hierarchical Model View Controller HMVC secure and lightweight framework. It has a rich set of components for developing applications rapidly. Companies that use Kohana include;   * [www.wepay.com](http://www.wepay.com/) * <http://kids.nationalgeographic.com/kids/> * <http://www.sittercity.com/> |
| [Learn PHP MVC Frameworks with CodeIgniter Example](https://www.guru99.com/images/2013/04/cake.png)  CakePHP  [www.cakephp.org](http://www.cakephp.org/) | It is modeled after Ruby on rails. It’s known for concepts such as software design patterns, convention over configuration, ActiveRecord etc. CakePHP powered applications include;   * <http://invoicemachine.com/> * <http://www.fmylife.com/> |
| [Learn PHP MVC Frameworks with CodeIgniter Example](https://www.guru99.com/images/2013/04/zend.png)  [www.framework.zend.com](http://www.framework.zend.com/)  Zend | It is a powerful framework that is;   * Secure, reliable, fast, and scalable * Supports Web 2.0 and creation of web services.   It features APIs from vendors like Amazon, Google, Flickr, Yahoo etc. It’s ideal for developing business applications. Zend powered applications include;   * Pimcore CMS, * DotKernel.   Companies using the Zend framework include;   * BBC * Cisco * Webex * Offers.com |

**Porting the opinion poll application to CodeIgniter**

In this [tutorial](https://www.guru99.com/case-study-opinion-poll-app.html), we created a PHP poll application. Here, we will port that code to CodeIgniter

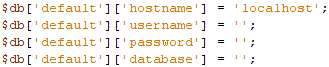
* [Download](http://ellislab.com/codeigniter) the latest version of CodeIgniter from their [website.](http://ellislab.com/codeigniter)
* Extract the contents of the zipped file to your development directory in your web server directory. We will use ciopinionpoll as the folder name in this lesson.
* Browse to the URL **http://localhost/ciopinionpoll/**

[](https://www.guru99.com/images/2013/04/ci_home.jpg)

We are now going to port our opinion poll application to CodeIgniter. Recall that our application was divided into three major components namely the;

* Front controller – this is the part that responds to URL requests and returns the requested page. This code will go into the controller
* Model – this is the code that responds to data requested and returns the requested data. This code will go into the model
* Views – this is the code responsible for formatting and displaying the data. This code will go into the view
  + Browse to ciopinionpoll folder
  + Open the *database.php* file located in application/config directory.
  + Locate the following lines of code

**Database configuration settings**

[](https://www.guru99.com/images/2013/04/ci_config.png)

* Set the username to root
* Set the password to your localhost root  password
* Database name to opinion\_poll. Note we will be using the database created in the previous lesson.
* Save the changes and close the file.

**Creating Our Model**

Next we are going to create our model that will extend the CI\_Model. The CI\_Model is part of the CodeIgniter libraries. The model will be located in application/models *opinion\_poll\_model.php*

<?php

class Opinion\_poll\_model extends CI\_Model

{

public function \_\_construct()

{

$this->load->database();

}

public function total\_votes()

{

$query = $this->db->select('COUNT(choice) as choices\_count')->get('js\_libraries');

return $query->row()->choices\_count;

}

public function get\_results()

{

$libraries = array("", "JQuery", "MooTools", "YUI Library", "Glow");

$table\_rows = '';

for ($i = 1; $i < 5; $i++)

{

$sql\_stmt = "SELECT COUNT(choice) choices\_count FROM js\_libraries WHERE choice = $i;";

$result = $model->

select($sql\_stmt); $table\_rows .= "<tr><td>" . $ libraries [$i] . " Got:</td><td><b>" . $result[0] . "</b> votes</td></tr>";

}

public function add\_vote($choice)

{

$ts = date("Y-m-d H:i:s"); $data = array('choice' => $choice, 'ts' => '$ts'); $this->db->insert('js\_libraries', $data);

}

}

?>

  HERE,

* “class Opinion\_poll\_model extends CI\_Model…” is our model that extends the CI\_Model
* “…parent:: \_\_construct();” calls the CI\_Model constructor
* “$this->load->database();” loads the database library so that our application can interact with the database
* “$this->db->” is CodeIgniter’s active record. Check this [link](http://ellislab.com/codeigniter/user-guide/database/active_record.html) for more information on the active record.

Creating Our Controller Let’s now create the controller. We will use the default CodeIgniter controller located in application/controllers/welcome.php. Replace its source codes with the following code.

<?php

if (!defined('BASEPATH'))  exit('No direct script access allowed');

class Welcome extends CI\_Controller {

    public function \_\_construct() {

        parent::\_\_construct();

        $this->load->model('opinion\_poll\_model');

    }

    public function index() {

        if ($this->input->post('submitbutton') && !$this->input->post('vote')) {

            echo "<script>alert('You did not vote!');</script>";

        }

        if ($this->input->post('vote')) {

            $this->opinion\_poll\_model->add\_vote($this->input->post('vote'));

            $data['total\_votes'] = $this->opinion\_poll\_model->total\_votes();

            $data['rows'] = $this->opinion\_poll\_model->get\_results();

            $this->load->view('results', $data);

        } else {

            $this->load->view('opinion\_poll\_form');

        }

    }

}

/\* End of file welcome.php \*/

/\* Location: ./application/controllers/welcome.php \*/

?>

    HERE,

* “if (!defined('BASEPATH'))  exit('No direct script access allowed');” ensures that users do not directly access the controller class
* “class Welcome extends CI\_Controller…” our controller extends the CI\_Controller class
* “public function \_\_construct()” calls CI\_Controller’s class contructor method and loads our Opinion\_poll\_model model
* “public function index()…” is the function that maps to index.php. it uses CodeIgniter’s input class to check if a vote has been submitted, add it to the database then display the results. If the post array of the input class is empty, it loads the voting page.
* “$this->input->post('…')” is the CodeIgniter input class that grabs the contents of the $\_POST global variable.
* “$this->opinion\_poll\_model->add\_vote($this->input->post('vote'))” calls the model’s add\_vote method to add the vote into the database.

**Creating Our Views**

Recall from the previous example that we had two HTML pages, one for voting and the other for results. We will use the same HTML code with minimal modifications to create our views. Create the following files in application/views directory

opinion\_poll\_form.php

<html>

<head>

<title>

JavaScript Libraries - Opinion Poll

</title>

</head>

<body>

<h2>JavaScript Libraries - Opinion Poll</h2>

<p><b>What is your favorite JavaScript Library?</b></p>

<form method="POST" action="index.php">

<p>

<input type="radio" name="vote" value="1" /> JQuery

<br />

<input type="radio" name="vote" value="2" /> MooTools

<br />

<input type="radio" name="vote" value="3" /> YUI Library

<br />

<input type="radio" name="vote" value="4" /> Glow </p>

<p>

<input type="submit" name="submitbutton" value="OK" />

</p>

</form>

</body>

</html>

  Let’s now create the results page results.php

<html>

    <head>

        <title>JavaScript Libraries - Opinion Poll Results</title>

    </head>

    <body>

        <h2>JavaScript Libraries - Opinion Poll Results</h2>

        <p><b>What is your favorite JavaScript Library?</b></p>

        <p><b><?php echo $total\_votes; ?></b> people have thus far taken part in this poll:</p>

        <p><table><tr><td>

            <?php print($rows); ?>

        </tr></td></table></p>

        <p><a href="">Return to voting page</a></p>

</body>

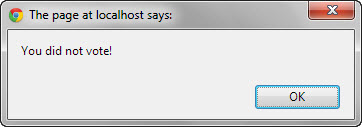
</html>

**Testing our application**

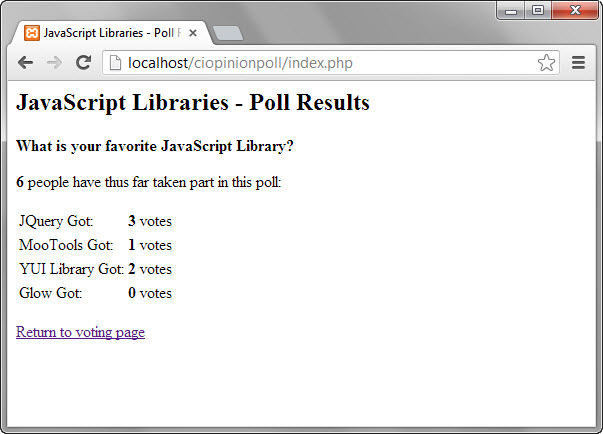
Assuming the root directory of your application is ciopinion, browse to <http://localhost/ciopionpoll/>

[](https://www.guru99.com/images/2013/04/polls_home.jpg)

Click on OK button, you will see the following alert message

[](https://www.guru99.com/images/2013/04/validation_msg.jpg)

Vote for your favorite candidate then click on OK You will see the following results page

[](https://www.guru99.com/images/2013/04/poll_results.jpg)

**Conclusion**

CodeIgniter is an easy to learn and use PHP MVC framework that can greatly reduce the time spent developing applications.

**Summary**

* A framework is a set of libraries that provide partial implementation of common tasks.
* PHP has a number of open source mature and tested MVC frameworks.
* A good development approach separates the data from the presentation and encourages the use of single entry point into an application.
* Traditional PHP applications that follow application design best practices can be ported to MVC frameworks with minimal modifications.