CodeIgniter

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

CodeIgniter is a powerful PHP framework with a very small footprint, built for developers who need a simple and elegant toolkit to create full-featured web applications. CodeIgniter was created by EllisLab, and is now a project of the British Columbia Institute of Technology.

Audience

This tutorial has been prepared for developers who would like to learn the art of developing websites using CodeIgniter. It provides a complete understanding of this framework.

Prerequisites

Before you start proceeding with this tutorial, we assume that you are already exposed to HTML, Core PHP, and Advance PHP. We have used CodeIgniter version 3.0.1 in all the examples.

Copyright & Disclaimer

Table of Contents

1. CODEIGNITER – OVERVIEW

CodeIgniter is an application development framework, which can be used to develop websites, using PHP. It is an Open Source framework. It has a very rich set of functionality, which will increase the speed of website development work.

If you know PHP well, then CodeIgniter will make your task easier. It has a very rich set of libraries and helpers. By using CodeIgniter, you will save a lot of time, if you are developing a website from scratch. Not only that, a website built in CodeIgniter is secure too, as it has the ability to prevent various attacks that take place through websites.

CodeIgniter Features

Some of the important features of CodeIgniter are listed below:

- Model-View-Controller Based System
- Extremely Light Weight
- Full Featured database classes with support for several platforms.
- Query Builder Database Support
- Form and Data Validation
- Security and XSS Filtering
- Session Management
- Email Sending Class. Supports Attachments, HTML/Text email, multiple protocols (sendmail, SMTP, and Mail) and more.
- Image Manipulation Library (cropping, resizing, rotating, etc.). Supports GD, ImageMagick, and NetPBM
- File Uploading Class
- FTP Class
- Localization
- Pagination
- Data Encryption
- Benchmarking
- Full Page Caching
- Error Logging
- Application Profiling

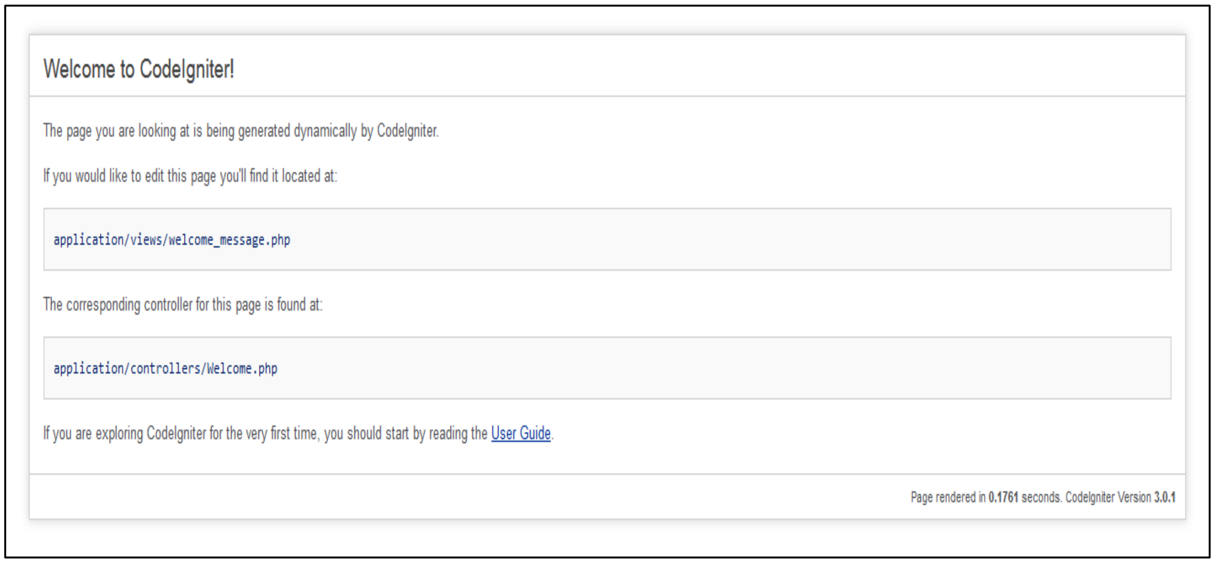
- Calendaring Class
- User Agent Class
- Zip Encoding Class
- Template Engine Class
- Trackback Class
- XML-RPC Library
- Unit Testing Class
- Search-engine Friendly URLs
- Flexible URI Routing
- Support for Hooks and Class Extensions
- Large library of "helper" functions

2. INSTALLING CODEIGNITER

It is very easy to install CodeIgniter. Just follow the steps given below:

- **Step-1:** Download the CodeIgniter from the link http://www.codeigniter.com/download
- **Step-2:** Unzip the folder.
- **Step-3:** Upload all files and folders to your server.
- **Step-4:** After uploading all the files to your server, visit the URL of your server, e.g., www.domain-name.com.

On visiting the URL, you will see the following screen:



Welcome to CodeIgniter!

The page you are looking at is being generated dynamically by CodeIgniter.

If you would like to edit this page you'll find it located at:

```
application/views/welcome_message.php
```

The corresponding controller for this page is found at:

```
application/controllers/Welcome.php
```

If you are exploring CodeIgniter for the very first time, you should start by reading the User Guide.

Page rendered in 0.1761 seconds. CodeIgniter Version 3.0.1

3. APPLICATION ARCHITECTURE

The architecture of CodeIgniter application is shown below.

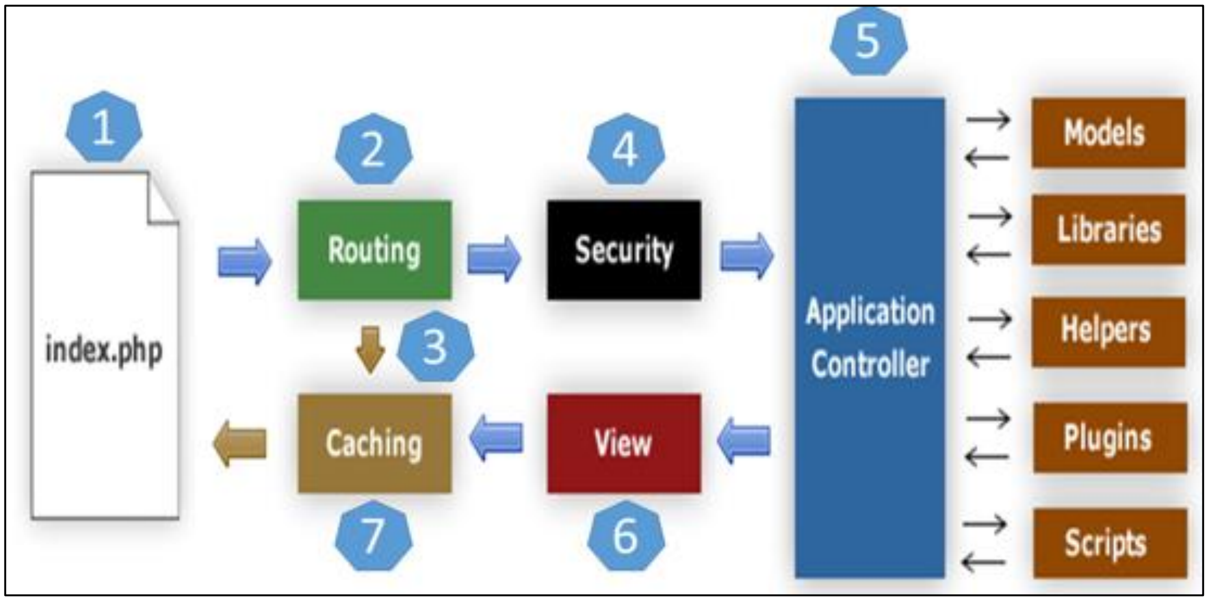


Figure: CodeIgniter Application Flowchart

- As shown in the figure, whenever a request comes to CodeIgniter, it will first go to **index.php** page.
-
- In the second step, **Routing** will decide whether to pass this request to step-3 for caching or to pass this request to step-4 for security check.
-
- If the requested page is already in **Caching**, then **Routing** will pass the request to step-3 and the response will go back to the user.
-
- If the requested page does not exist in **Caching**, then **Routing** will pass the requested page to step-4 for **Security** checks.
-
- Before passing the request to **Application Controller**, the **Security** of the submitted data is checked. After the **Security** check, the **Application Controller** loads necessary **Models, Libraries, Helpers, Plugins** and **Scripts** and pass it on to **View**.
-
- The **View** will render the page with available data and pass it on for **Caching**. As the requested page was not cached before so this time it will be cached in **Caching,** to process this page quickly for future requests.

Directory Structure

The image given below shows the directory structure of the CodeIgniter.

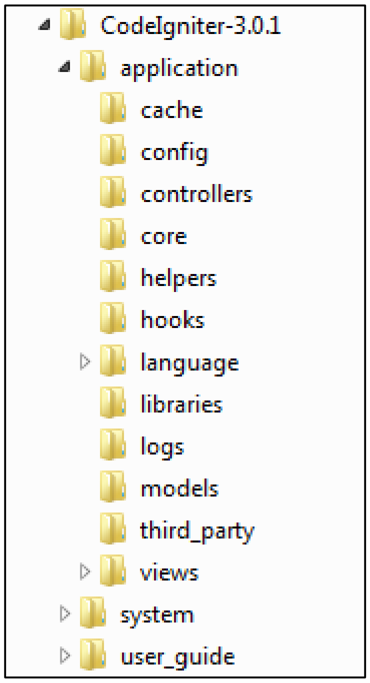


Figure: Directory Structure

CodeIgniter directory structure is divided into 3 folders:

- Application
- System
- User_guide

Application

As the name indicates the Application folder contains all the code of your application that you are building. This is the folder where you will develop your project. The Application folder contains several other folders, which are explained below:

- **Cache:** This folder contains all the cached pages of your application. These cached pages will increase the overall speed of accessing the pages.

- **Config:** This folder contains various files to configure the application. With the help of **config.php** file, user can configure the application. Using **database.php** file, user can configure the database of the application.

- **Controllers:** This folder holds the controllers of your application. It is the basic part of your application.

- **Core:** This folder will contain base class of your application.

- **Helpers:** In this folder, you can put helper class of your application.
- **Hooks:** The files in this folder provide a means to tap into and modify the inner workings of the framework without hacking the core files.

- **Language:** This folder contains language related files.

- **Libraries:** This folder contains files of the libraries developed for your application.

- **Logs:** This folder contains files related to the log of the system.

- **Models:** The database login will be placed in this folder.

- **Third_party:** In this folder, you can place any plugins, which will be used for your application.

- **Views:** Application's HTML files will be placed in this folder.

System

This folder contains CodeIgniter core codes, libraries, helpers and other files, which help make the coding easy. These libraries and helpers are loaded and used in web app development.

This folder contains all the CodeIgniter code of consequence, organized into various folders:

- **Core:** This folder contains CodeIgniter's core class. Do not modify anything here. All of your work will take place in the application folder. Even if your intent is to extend the CodeIgniter core, you have to do it with hooks, and hooks live in the application folder.
-
- **Database:** The database folder contains core database drivers and other database utilities.
-
- **Fonts:** The fonts folder contains font related information and utilities.
-
- **Helpers:** The helpers folder contains standard CodeIgniter helpers (such as date, cookie, and URL helpers).
-
- **Language:** The language folder contains language files. You can ignore it for now.
-
- **Libraries:** The libraries folder contains standard CodeIgniter libraries (to help you with e-mail, calendars, file uploads, and more). You can create your own libraries or extend (and even replace) standard ones, but those will be saved in the **application/libraries** directory to keep them separate from the standard CodeIgniter libraries saved in this particular folder.

User_guide

This is your user guide to CodeIgniter. It is basically, the offline version of user guide on CodeIgniter website. Using this, one can learn the functions of various libraries, helpers and classes. It is recommended to go through this user guide before building your first web app in CodeIgniter.

Beside these three folders, there is one more important file named "**index.php**". In this file, we can set the application environment and error level and we can define system and application folder name. It is recommended, not to edit these settings if you do not have enough knowledge about what you are going to do.

4. CODEIGNITER – MVC FRAMEWORK

CodeIgniter is based on the **Model-View-Controller (MVC) development pattern**. MVC is a software approach that separates application logic from presentation. In practice, it permits your web pages to contain minimal scripting since the presentation is separate from the PHP scripting.

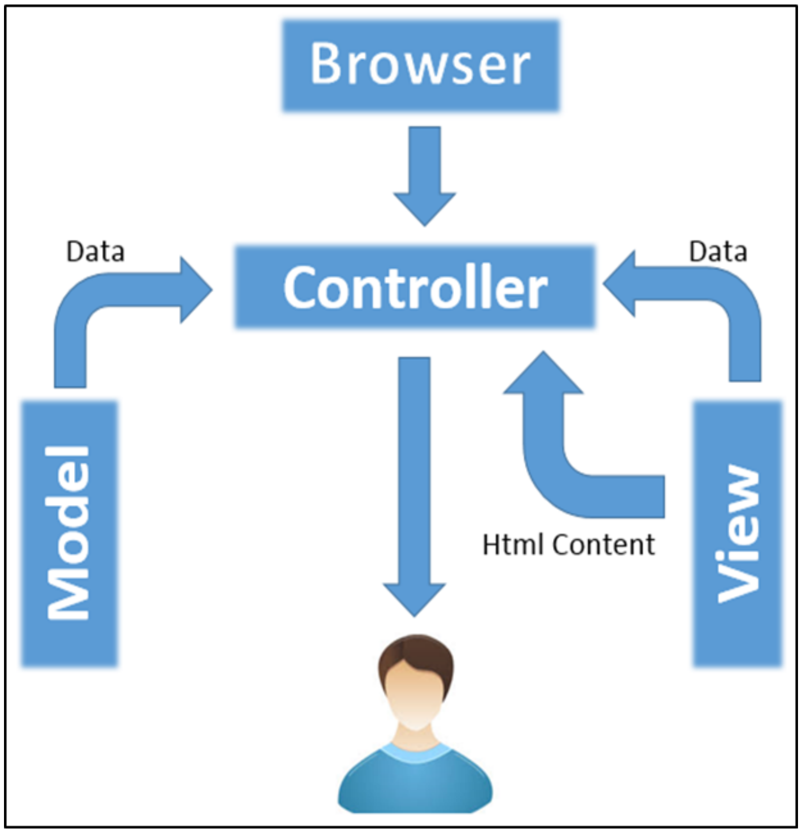


Figure: CodeIgniter – MVC Framework

-
- The **Model** represents your data structures. Typically, your model classes will contain functions that help you retrieve, insert and update information in your database.
-
- The **View** is information that is being presented to a user. A View will normally be a web page, but in CodeIgniter, a view can also be a page fragment like a header or footer. It can also be an RSS page, or any other type of "page".
-
  - The **Controller** serves as an intermediary between the Model, the View, and any other resources needed to process the HTTP request and generate a web page.

5. CODEIGNITER – BASIC CONCEPTS

Controllers

A controller is a simple class file. As the name suggests, it controls the whole application by URI.

Creating a Controller

First, go to **application/controllers** folder. You will find two files there, **index.html** and **Welcome.php**. These files come with the CodeIgniter.

Keep these files as they are. Create a new file under the same path named "**Test.php**". Write the following code in that file:

```
<?php
class Test extends CI_Controller {


	public function index()
	{
		echo "Hello World!";
	}
}
?>
```

The **Test** class extends an in-built class called **CI_Controller**. This class must be extended whenever you want to make your own Controller class.

Calling a Controller

The above controller can be called by URI as follows:

```
http://www.your-domain.com/index.php/test
```

Notice the word "**test**" in the above URI after index.php. This indicates the class name of controller. As we have given the name of the controller "**Test**", we are writing "**test**" after the index.php. The class name must start with **uppercase letter** but we need to write **lowercase letter** when we call that controller by URI. The general syntax for calling the controller is as follows:

```
http://www.your-domain.com/index.php/controller/method-name
```

Creating & Calling Constructor Method

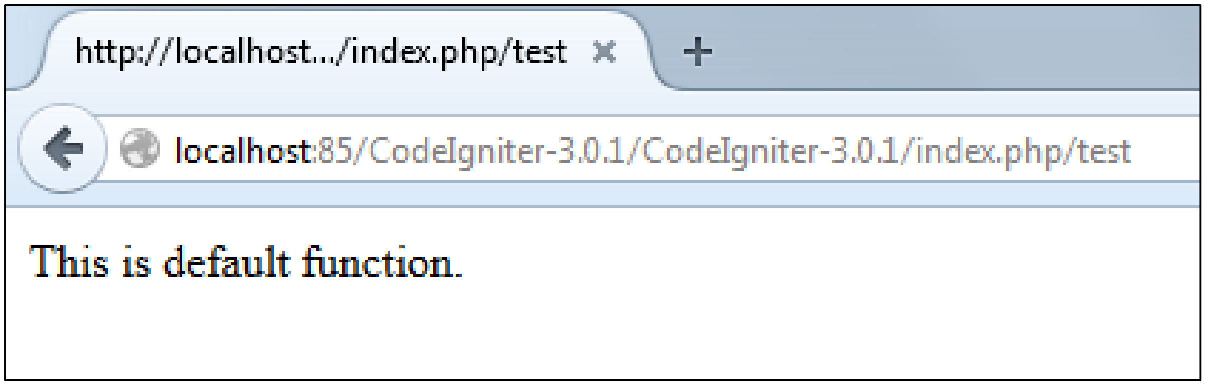
Let us modify the above class and create another method named "hello".

```
<?php
class Test extends CI_Controller {

     public function index()
     {
             echo "This is default function.";
     }

     public function hello()
     {
             echo "This is hello function.";
     }
}
?>
```

We can execute the above controller in the following three ways:

1. http://www.your-domain.com/index.php/test
2. http://www.your-domain.com/index.php/test/index
3. http://www.your-domain.com/index.php/test/hello

After visiting the first URI in the browser, we get the output as shown in the picture given below. As you can see, we got the output of the method "**index**", even though we did not pass the name of the method the URI. We have used only controller name in the URI. In such situations, the CodeIgniter calls the default method "**index**".

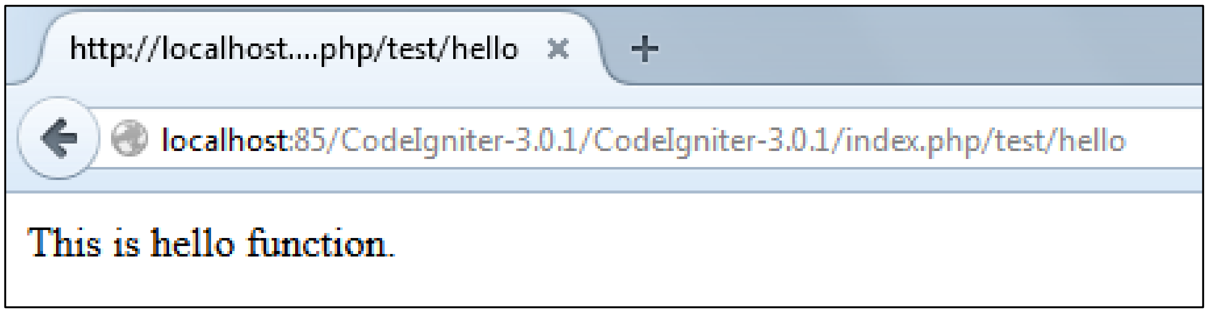


localhost:85/CodeIgniter-3.0.1/CodeIgniter-3.0.1/index.php/test

```
This is default function.
```

Visiting the second URI in the browser, we get the same output as shown in the above picture. Here, we have passed method's name after controller's name in the URI. As the name of the method is "**index**", we are getting the same output.

Visiting the third URI in the browser, we get the output as shown in picture given below. As you can see, we are getting the output of the method "**hello**" because we have passed "**hello**" as the method name, after the name of the controller "**test**" in the URI.


http://localhost....php/test/hello
localhost:85/CodeIgniter-3.0.1/CodeIgniter-3.0.1/index.php/test/hello

```
This is hello function.
```

Points to Remember:

- The name of the controller class must start with an uppercase letter.
- The controller must be called with lowercase letter.
- Do not use the same name of the method as your parent class, as it will override parent class's functionality.

Views

This can be a simple or complex webpage, which can be called by the controller. The webpage may contain header, footer, sidebar etc. View cannot be called directly. Let us create a simple view. Create a new file under **application/views** with name "**test.php**" and copy the below given code in that file.

```
<!DOCTYPE html>
<html lang="en">
<head>
     <meta charset="utf-8">
     <title>CodeIgniter View Example</title>
</head>
<body>
     CodeIgniter View Example
</body>
</html>
```

Change the code of **application/controllers/test.php** file as shown in the below.

Loading the View

The view can be loaded by the following syntax:

```
$this->load->view('name');
```

Where name is the view file, which is being rendered. If you have planned to store the view file in some directory then you can use the following syntax:

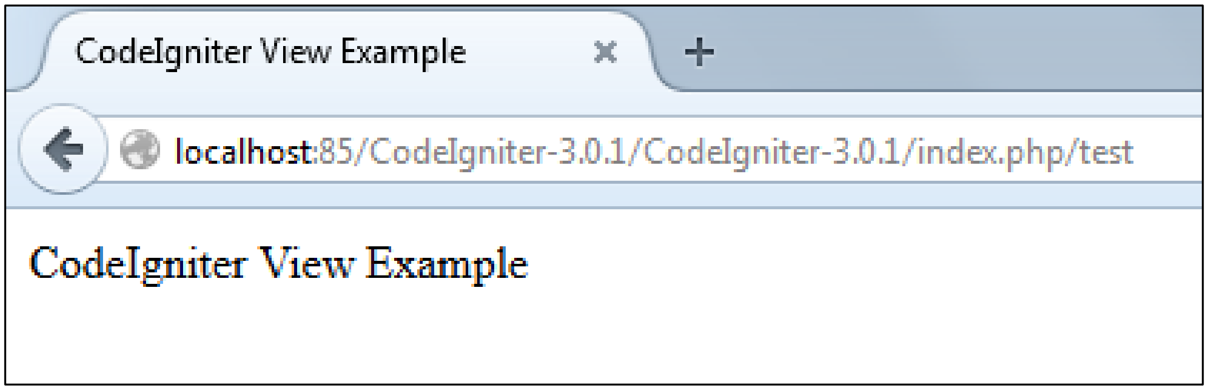
```
$this->load->view('directory-name/name');
```

It is not necessary to specify the extension as php, unless something other than .php is used.

The index() method is calling the view method and passing the "test" as argument to view() method because we have stored the html coding in "**test.php**" file under **application/views/test.php**.

```
<?php
class Test extends CI_Controller {


      public function index()
      {
             $this->load->view('test');
      }
}
?>
```

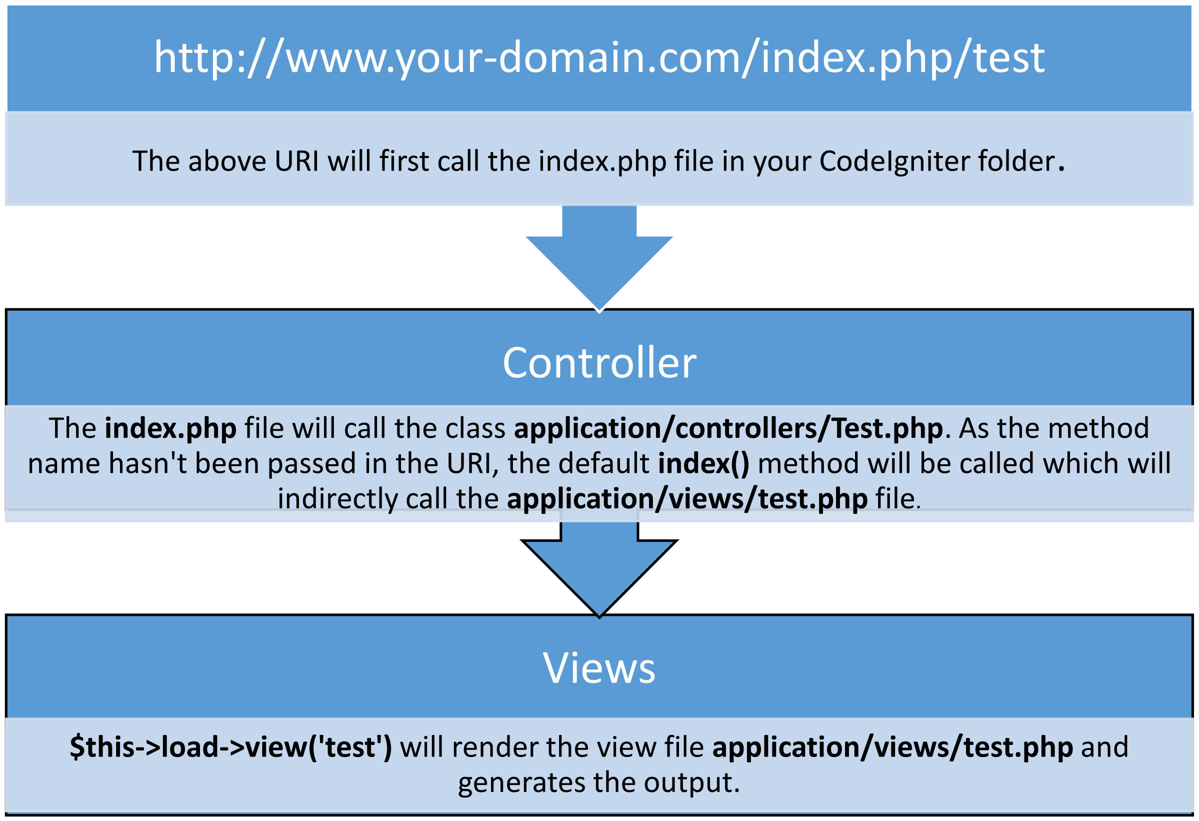
Here is the output of the above code:



localhost:85/CodeIgniter-3.0.1/CodeIgniter-3.0.1/index.php/test

CodeIgniter View Example

The following flowchart illustrates of how everything works:


http://www.your-domain.com/index.php/test

The above URI will first call the index.php file in your CodeIgniter folder.

Controller

The **index.php** file will call the class **application/controllers/Test.php**. As the method name hasn't been passed in the URI, the default **index()** method will be called which will indirectly call the **application/views/test.php** file.

Views

$this->load->view('test') will render the view file **application/views/test.php** and generates the output.

Models

Models classes are designed to work with information in the database. As an example, if you are using CodeIgniter to manage users in your application then you must have model class, which contains functions to insert, delete, update and retrieve your users' data.

Creating Model Class

Model classes are stored in **application/models** directory. Following code shows how to create model class in CodeIgniter.

```
<?php
Class Model_name extends CI_Model{


     Public function __construct()
     {
            parent::__construct();
     }
}
?>
```

Where Model_name is the name of the model class that you want to give. Each model class must inherit the CodeIgniter's CI_Model class. The first letter of the model class must be in capital letter. Following is the code for users' model class.

```
<?php
Class zzzextends CI_Model{


Public function __construct()
{
     parent::__construct();
}


}
?>
```

The above model class must be saved as User_model.php. The class name and file name must be same.

Loading Model

Model can be called in controller. Following code can be used to load any model.

```
$this->load->model('model_name');
```

Where model_name is the name of the model to be loaded. After loading the model you can simply call its method as shown below.

```
$this->model_name->method();
```

Auto-loading Models

There may be situations where you want some model class throughout your application. In such situations, it is better if we autoload it.

```
/*
| -------------------------------------------------------------------
|  Auto-load Models
| -------------------------------------------------------------------
| Prototype:
|
|   $autoload['model'] = array('first_model', 'second_model');
|
| You can also supply an alternative model name to be assigned
| in the controller:
|
|   $autoload['model'] = array('first_model' => 'first');
*/
$autoload['model'] = array();
```

As shown in the above figure, pass the name of the model in the array that you want to autoload and it will be autoloaded, while system is in initialization state and is accessible throughout the application.

Helpers

As the name suggests, it will help you build your system. It is divided into small functions to serve different functionality. A number of helpers are available in CodeIgniter, which are listed in the table below. We can build our own helpers too.

Helpers are typically stored in your **system/helpers**, or **application/helpers directory**. Custom helpers are stored in **application/helpers** directory and systems' helpers are stored in **system/helpers** directory. CodeIgniter will look first in your **application/helpers directory**. If the directory does not exist or the specified helper is not located, CodeIgniter will instead, look in your global ***system/helpers/*** **directory**. Each helper, whether it is custom or system helper, must be loaded before using it.

Helper Name	Description

Array Helper	The Array Helper file contains functions that assist in working with arrays.
CAPTCHA Helper	The CAPTCHA Helper file contains functions that assist in creating CAPTCHA images.
Cookie Helper	The Cookie Helper file contains functions that assist in working with cookies.
Date Helper	The Date Helper file contains functions that help you work with dates.
Directory Helper	The Directory Helper file contains functions that assist in working with directories.
Download Helper	The Download Helper lets you download data to your desktop.
Email Helper	The Email Helper provides some assistive functions for working with Email. For a more robust email solution, see CodeIgniter's Email Class.
File Helper	The File Helper file contains functions that assist in working with files.
Form Helper	The Form Helper file contains functions that assist in working with forms.
HTML Helper	The HTML Helper file contains functions that assist in working with HTML.
Inflector Helper	The Inflector Helper file contains functions that permits you to change words to plural, singular, camel case, etc.
Language Helper	The Language Helper file contains functions that assist in working with language files.
Number Helper	The Number Helper file contains functions that help you work with numeric data.
Path Helper	The Path Helper file contains functions that permits you to work with file paths on the server.
Security Helper	The Security Helper file contains security related functions.