THE MVC FRAMEWORK-CODEIGNITER

[1]Shraddha R. Mandaviya, [2]Riddhi K. Raval, [3]Arpit B. Parekh
[1]Assistant Professor, [2] Assistant Professor,[3]Assistant Professor
[1]Computer Science,
[1]Shree Swaminarayan College of Computer Science, Bhavnagar, India

Abstract— The objective of this paper is to describes how to work on codeigniter framework and installation of codeigniter. With the use of codeigniter less utilizes memory and fit to every hosting. The model, view, controller architecture is all the coding framework are MVC nowadays some benefits Little too server requirements, Easy to understand and extend, All The tools you need in one little package, available security tools. Codeigniter is an open source rapid application development framework that is used to develop dynamic websites with PHP. MVC framework can help to speed up the development of any web application.

Keywords: Codeigniter, PHP Framework, Installation, File Structure, MVC

I. INTRODUCTION

Frameworks are of key importance for developing large-scale object-oriented software system. They promise higher productivity and shorter time-to-market through design and code reuse. Codeigniter is a php framework which is developed by EllisLab.s it needs no extra configuration. You do not need to use the command line. It is extremely light. It provides a rich set of libraries for commonly needed tasks, as well as a simple interface and logical structure to access these libraries. Codeigniter face challenges you to find a framework that has better performance than codeigniter. Codeigniter supports M-V-C uses the model-view-controller approach. Codeigniter thoroughly documented. Codeigniter has a friendly community of users. PHP5 is a doozy of a scripting language for building state-of-the-art, dynamic web-based applications; its rapport with HTML is top-notch; plus it's a great language for honing your OOP skills. It's tight, secure and intuitive. For many of us, it's the only server-side language for our needs. It makes coding in PHP simple, quick and user-friendly. Underpins the Model/View/Controller (MVC) approach to web development a best practice philosophy all developers should adhere to. It's built on a linear, easy-to-use folder structure. Its open source and simple to configure and customize for your own needs. You can construct your own cleaner URI lines in codeIgniter. For this we will create a Controller, a Model and a View, as well as the database to hold the information.

II. FEATURES OF CODEIGITER

1. FREE TO USE

It is licensed under MIT license, so it is free to use.

2. FOLLOWS MVC PATTERN

It uses Model-View-Controller which basically separates logic and presentation parts. Request comes to controller, database action is performed through model and output is displayed through views. But in normal PHP scripting, every page represents MVC which increases complexity.

3. LIGHT WEIGHT

It is extremely light-weighted. CodeIgniter core system requires very small library, other libraries may be added upon dynamic request based upon your needs. That is why it is quite fast and light weighted.

4. GENERATE SEO FRIENDLY URLS

URLs generated by CodeIgniter are search-engine friendly and clean. It uses a segment based approach rather than standard query based approach.

5. BUILT-IN LIBRARIES

It comes with full packet libraries that enable all the web needed tasks like database, form validation, sending email, manipulating images, sending emails,etc.

III. CODEIGNITER INSTALLATION

Download CodeIgniter from its official website**.** Download current version of CodeIgniter.Unzip Codeigniter package.Downloaded CodeIgniter will be in zip format. Copy it and place it in your htdocs folder. See in fig.1. Unzip and anew it. We are naming it as Codeigniter**.**

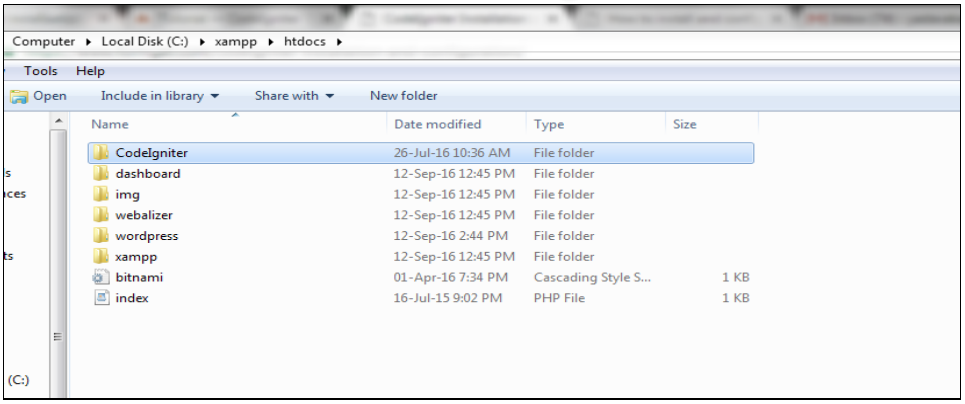

Fig. 1.Unzipped Codeigniter

Now CodeIniter user guide on browser type localhost/CodeIgniter/ (after localhost type name of your unzipped folder). Then codeigniter is successfully installed see fig.2.

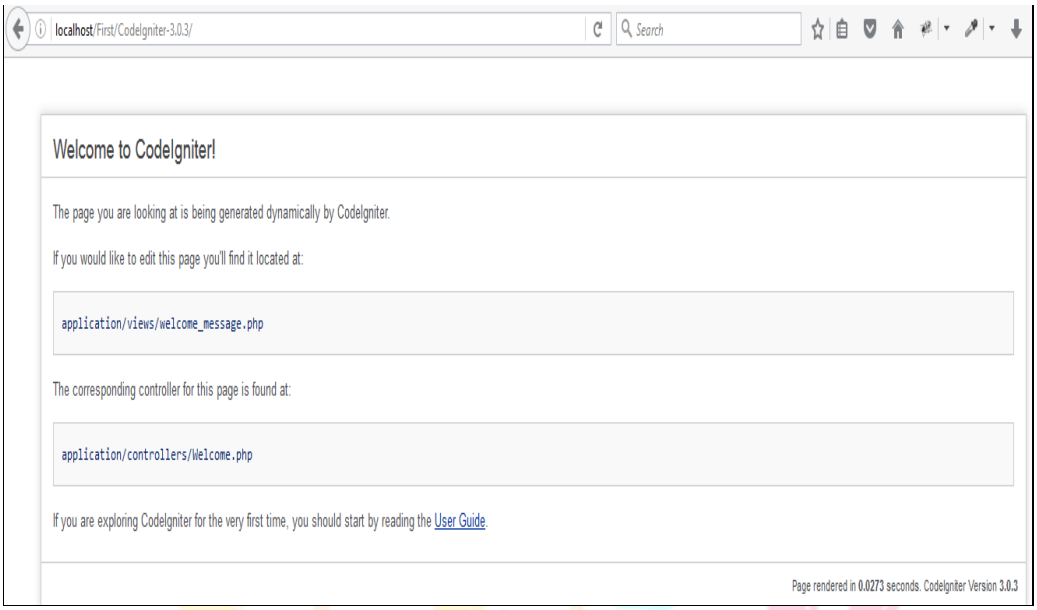

Fig. 2.Installed Codeigniter

Then set the base URL in application/config/config.php file with any text editor see in Fig.3. As under.

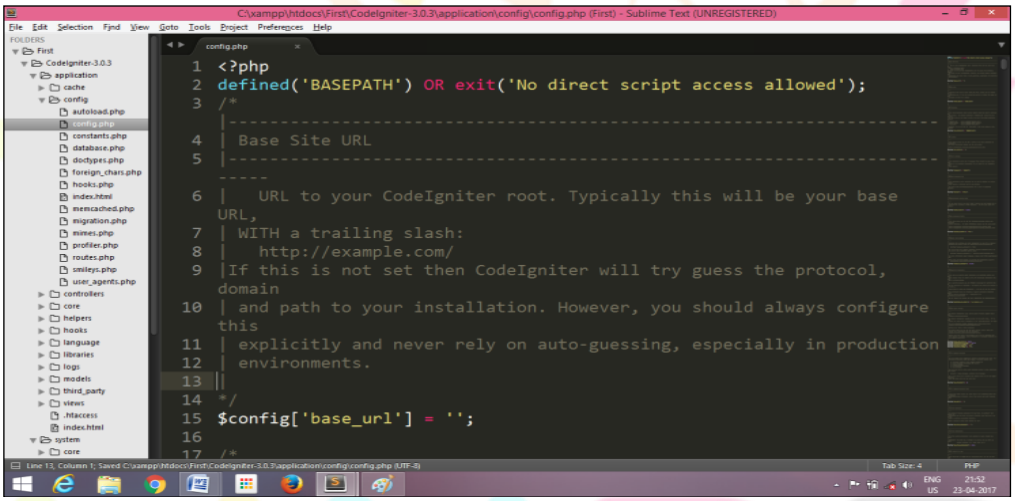

Fig. 3.Set URL

You need to establish the connectivity to your database. Go to the path application/config/ database.php file.

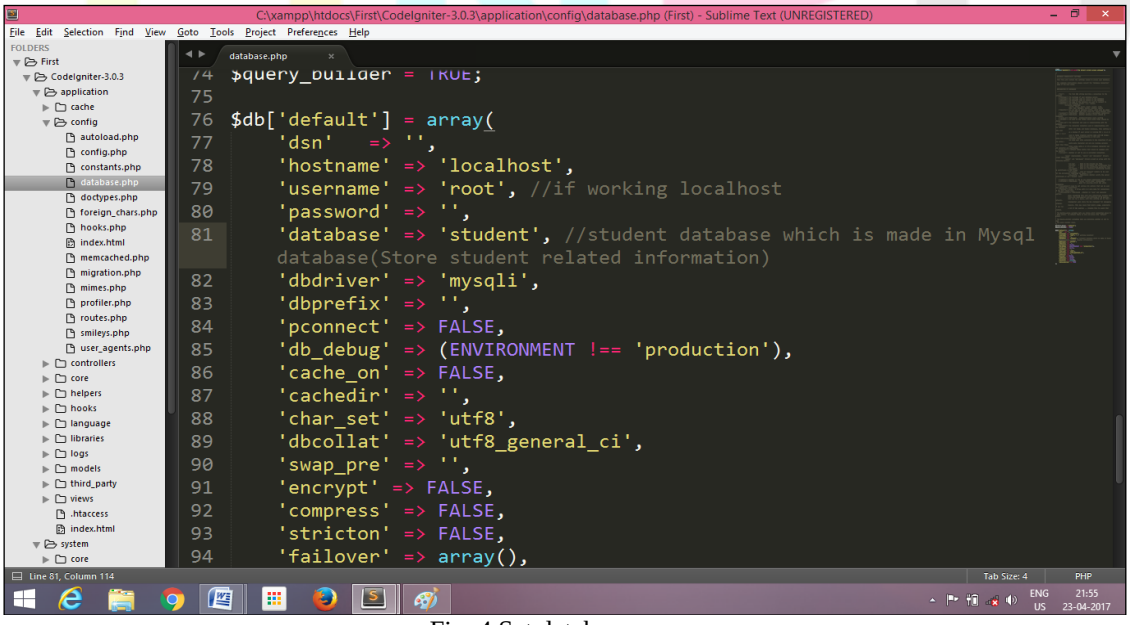

Fig. 4.Set database name

IV. FILE STRUCTURE

Codeigniter file structure is mainly divided into three parts: Application, System, and User_guide.

1. APPLICATION:

Application folder is the main development folder for you where you will develop your project. It contains your models, views, controllers, configuration and many other files. It contains all the code of the project you are working on Cache: Cache stores the processed data so that this data can be easily loaded within no time for the future use. It increases the speed of your page access. The config folder contains configuration files as shown Fig.5. These files allow configuring CodeIgniter application.

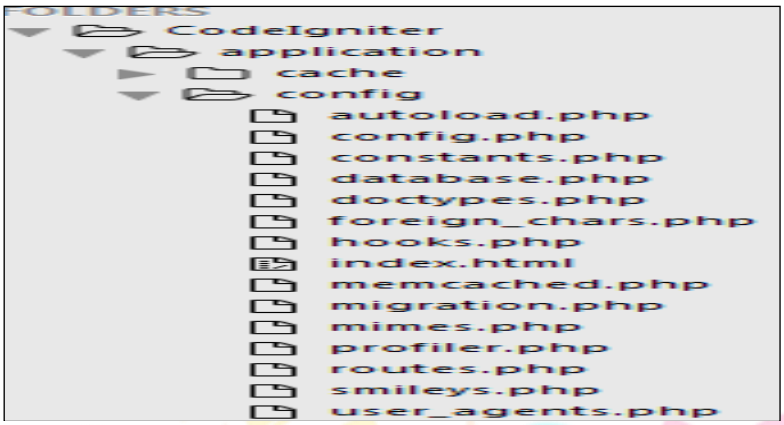

Fig. 5.Application

2. SYSTEM

It is used to performed action. It contains files which makes the coding easy. It contains Codeigniter core class. Do not make any changes in this folder. It contains database drivers and other utilities. It contains font related information. It contains default helpers such as URL, date and cookie. CodeIgniter supports multilingual web applications. It contains default language file. It contains libraries like calendars, file upload, email, etc. libraries created by you will be saved in "application/libraries". Here, only standard libraries will be stored.

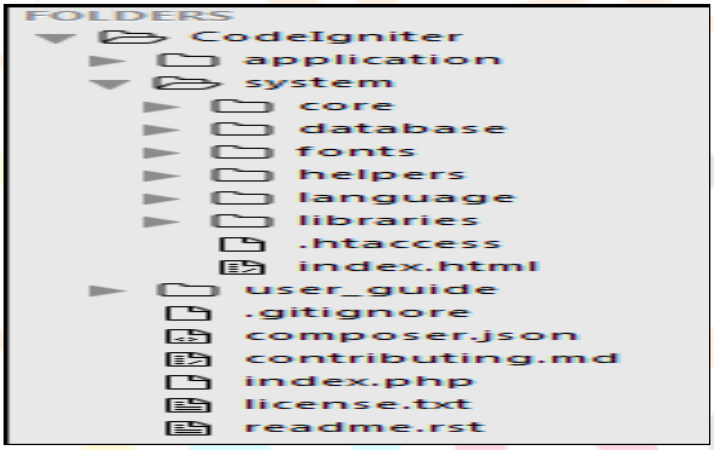

Fig.6. System

3. USER_GUIDE

It is the offline CodeIgniter guide. It comes with every CodeIgniter downloaded version. In case of any query, you can read its user guide. You can learn here all the functions, libraries, helpers of CodeIgniter. Before starting use of CodeIgniter go through this guide

V. MVC COMPONENTS

MVC stands for Model, View, and Controller. It is a programming pattern used in developing Web applications. This pattern separates the user interface and backend. Model–View–Controller (MVC) is a software architectural pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts in order to separate internal.

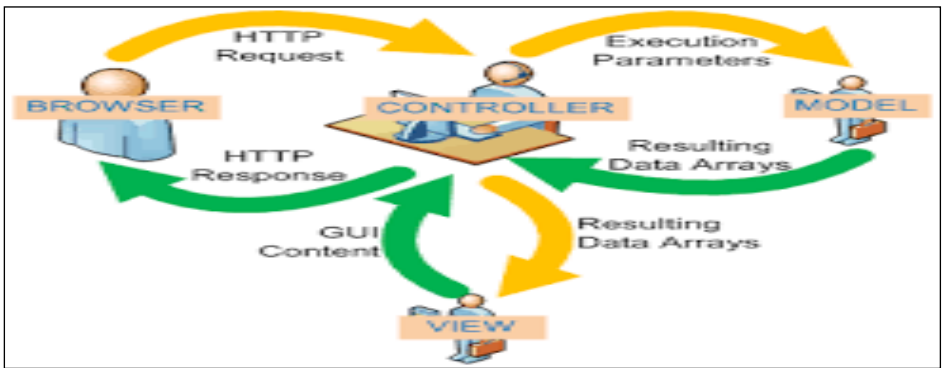

Fig.7. Model-View-Controller

1. MODEL:

Models are directed by the Controller. It represents your data structure. Model classes contain functions through which you can insert, retrieve or update for send and receive data into your database model is implemented as a series of business entity classes each of which contains all the properties and methods required by a single business entity. It is actually a subclass to an abstract super class which contains properties and methods which are common to all database tables. The representations of information from the ways information is presented to and accepted from the user. MVC or Model-View-Controller is a software architecture, or design pattern, that is used in software engineering, whose fundamental principle is based on the idea that the logic of an application should be separated from its presentation. Put simply, The MVC principle is to separate the application into 3 main parts, known as the Model, the View, and the Controller. Apparent from the diagram are the direct associations (solid lines) and the inferred associations (dashed lines). The inferred associations are associations that might seem apparent from the point of view of the user, and not from the actual software design.

A simple way to think of this would be to consider the following:

- **A.** A user interacts with the view - by clicking on a link or submitting a form.
- **B.** The Controller handles the user input, and transfers the information to the model.
- **C.** The Model receives the information and updates its state.
- **D.** The View checks the state of the Model and responds accordingly.
- **E.** The View waits for another interaction from the user.

Model is responsible for all data validation, business rules and task-specific conduct, while the actual generation of DML (Data Manipulation Language) statements is performed in a separate DML class. A model is an object representing data or even activity, e.g. a database table or even some plant-floor production-machine process. The model manages the behavior and data of the application domain, responds to requests for information about its state and responds to instructions to change state. The model represents enterprise data and the business rules that govern access to and updates of this data. Often the model serves as a software approximation to a real-world process, so simple real-world modeling techniques apply when defining the model. The model is the piece that represents the state and low-level behavior of the component. It manages the state and conducts all transformations on that state. The model has no specific knowledge of either its controllers or its views. The view is the piece that manages the visual display of the state represented by the model. A model can have more than one view.

2. VIEW:

The View is the 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. The view deals with displaying the data and interface controls to the user with. The View is implemented as a series of screen structure scripts which are combined with the output from each database table class to produce an XML document. For each database table there is typically a list view. A view is some form of visualization of the state of the model. The view manages the graphical and/or textual output to the portion of the bitmapped display that is allocated to its application. A view typically has a one to one correspondence with a display surface and knows how to render to it. A view attaches to a model and renders its contents to the display surface.

Example: View

```
<!DOCTYPE html>
<html>
    <head>
        <title>Hello World</title>
    </head>
<body>

<p>Hello World!!</p>
</body>
</html>
```

3. CONTROLLER:

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. Then navigate to the application/controllers folder. It act as a single point of control from where you can manage both model and view. A controller offers facilities to change the state of the model. A controller is the means by which the user interacts with the application. A controller accepts input from the user and instructs the model and view to perform actions based on that input. In effect, the controller is responsible for mapping end-user action to application response. The controller translates interactions with the view into actions to be performed by the model. In a stand-alone GUI client, user interactions could be button clicks or menu selections, whereas in a Web application they appear as HTTP GET and POST requests. The actions performed by the model include activating business processes or changing the state of the model. Based on the user interactions and the outcome of the model actions, the controller responds by selecting an appropriate view. The Controller is implemented as a series of *component scripts* which link to one of a series of transaction controller scripts. Unlike some implementations which require a separate controller for each transaction, each of my controllers is for a class (or type) of transaction, so the same controller script can be shared by all transactions of the same class.

Each of these scripts deals with the following:

- **A.** It handles the HTTP GET and POST requests.
- **B.** It calls methods on those objects as appropriate.

Most will perform some sort of communication with the database although some may not. Data is passed into and out of these objects as standard PHP arrays. In this way an object can deal with any number of database occurrences both as input or output.

Example:Controller

```
<?php
      defined('BASEPATH') OR exit('No
direct script access allowed');

  class Hello extends CI_Controller
{

                PUBLIC FUNCTION INDEX()

              {
               $this->load->view('hello_world');
              }
 }
?>
```

VI. CONCLUSION:

The main intend of this research paper is to introduce the fundamental concepts of codeigniter framework and installation of codeigniter. Codeigniter would be the preferred selection when developing bigger projects. Codeigniter study paper you should now have a superior understanding of how Codeigniter mechanism. You should identify how to set-up Codeigniter including the whole design, how to create models, views and controllers along with libraries and identify how to consignment and utilize them productively.

REFERENCES

[1] Thomas Myer, "Professional Codeigniter", publisher.Wrox (25 July 2008).
[2] Nitin Reddy Katkam,"Codeigniter v2 Guide, A quick and easy guide to using Codeigniter for PHP developers", 2013.
[3] https://www.codeigniter.com/userguide3/installation/index.html
[4] Eli Orr, Yehuda Zadik, " Programming with Codeigniter MVC, Build feature-rich web applications using the codeigniter MVC framework",, Publisher:Packt Publishing ,23th September 2013.
[5] Packagist. "PHP's package manager" Packagist.org.
[6] PHP documentation. "History of PHP" php.net. http://php.net/manual/en/history.php.php
[7] https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm
[8] https://en.wikipedia.org/wiki/CodeIgniter
[9] W3techs. "W3Techs - World Wide Web Technology Surveys", w3techs.com.