**Code igniter**

**Learning Code igniter**

1. Download code igniter.

2. Create one separate database.

3. Rewrite the autoload and database files in the config folder in the code igniter.

4. Then we have to save the files in controller, model and views.

**Controller**

**Path - application->controller**

**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.

**Syntax for controller**

**Creating a controller**

<?php

class Test extends CI\_Controller {

public function index() {

echo "Hello World!";

}

}

?>

**Calling Constructor**

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

### Creating Constructor Method

<?php

class Test extends CI\_Controller {

public function index() {

echo "This is default function.";

}

public function hello() {

echo "This is hello function.";

}

}

?>

**Calling Constructor Method**

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

**View**

**Path – application->view**

**Points to Remember**

1. The view page are used to display what we are creating.

2. It contain like head, body, title, form and script etc..

3. Simply it is a design page.

**Syntax for view**

<!DOCTYPE html>

<html lang = "en">

<head>

<meta charset = "utf-8">

<title>CodeIgniter View Example</title>

</head>

<body>

CodeIgniter View Example

</body>

</html>

**when represent in the constructor of view file**

<?php

class Test extends CI\_Controller {

public function index() {

$this->load->view('test');

}

}

?>

**Represent the view in constructor**

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

If the view file is in folder then we represent,

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

name->view file name

**Model**

**Path - application->model**

**Points to Remember**

1. The name of the model class must start with an uppercase letter.
2. It contain the database function.

**Syntax for model**

### Creating Model Class

<?php

Class Model\_name extends CI\_Model {

Public function \_\_construct() {

parent::\_\_construct();

}

}

?>

**Creating Model Class with function**

<?php

Class Model\_name extends CI\_Model {

Public function \_\_construct() {

parent::\_\_construct();

}

function function\_name() {

Any db function

/\* $this->db->select('invite\_user\_id');

$this->db->from(TABLE\_PRE . 'invite');

$this->db->where($where\_aray);

$query = $this->db->get();

if ($query->num\_rows > 0) {

return true;

}

return false;

} \*/

}

?>

**Represent the model in constructor**

$this->load->model('model\_name');

After initiate the model we can simply call the function as

$this->model\_name->method();

**Helpers**

**Path – application->controller->config**

**Points to Remember**

1. It will help our build your system.
2. The helper name and the URL are loaded in the controller folder and the helper file.

**When load the helpers in controller file and also in the config file**

$this->load->helper('name');

If we want to load the URL

$this->load->helper('url');

## Configuring Base URL

**Syntax**

$config['base\_url'] = 'http://your-domain.com';

## Database Configuration

**Syntax**

$db['default'] = array(

'dsn' => '',

'hostname' => 'localhost',

'username' => 'root',

'password' => '',

'database' => 'database\_name',

'dbdriver' => 'mysqli',

'dbprefix' => '',

'pconnect' => TRUE,

'db\_debug' => TRUE,

'cache\_on' => FALSE,

'cachedir' => '',

'char\_set' => 'utf8',

'dbcollat' => 'utf8\_general\_ci',

'swap\_pre' => '',

'encrypt' => FALSE,

'compress' => FALSE,

'stricton' => FALSE,

'failover' => array()

);

## Autoload Configuration

$autoload['libraries'] = array('database', 'email', 'session');

$autoload['drivers'] = array('cache');

$autoload['helper'] = array('url', 'file');

$autoload['language'] = array('lang1', 'lang2')

$autoload['model'] = array('first\_model', 'second\_model');

## Connecting to a Database

**Automatic Connecting**

$autoload['libraries'] = array(‘library’);

**Manual Connecting**

$this->load->database();

## Inserting a Record

**Syntax**

insert([*$table = ''*[, *$set = NULL*[, *$escape = NULL*]]])

**Parameters**

1. **$table** (*string*) − Table name
2. **$set** (*array*) − An associative array of field/value pairs
3. **$escape** (*bool*) − Whether to escape values and identifiers

**Initiate in the controller**

$data = array(

'roll\_no' => ‘1’,

'name' => ‘Virat’

);

$this->db->insert("stud", $data);

## Updating a Record

**Syntax**

set(*$key*[, *$value = ''*[, *$escape = NULL*]])

**Parameters**

1. **$key** (*mixed*) − Field name, or an array of field/value pairs
2. **$value** (*string*) − Field value, if $key is a single field
3. **$escape** (*bool*) − Whether to escape values and identifiers

**Initiate in the controller**

$data = array(

'roll\_no' => ‘1’,

'name' => ‘Virat’

);

$this->db->set($data);

$this->db->where("roll\_no", ‘1’);

$this->db->update("stud", $data);

## Deleting a Record

**Syntax**

delete([*$table = ''*[, *$where = ''*[, *$limit = NULL[, $reset\_data = TRUE*]]]])

**Parameters**

1. **$table** (*mixed*) − The table(s) to delete from; string or array
2. **$where** (*string*) − The WHERE clause
3. **$limit** (*int*) − The LIMIT clause
4. **$reset\_data** (*bool*) − TRUE to reset the query “write” clause

**Initiate in the controller**

$this->db->delete("stud", "roll\_no = 1");

## Selecting a Record

**Syntax**

get([*$table = ''*[, *$limit = NULL*[, *$offset = NULL*]]])

**Parameters**

1. **$table** (*string*) − The table to query array
2. **$limit** (*int*) − The LIMIT clause
3. **$offset** (*int*) − The OFFSET clause

**Initiate in the controller**

$query = $this->db->get("stud");

$data['records'] = $query->result();

# Form Validation

**Syntax**

$this->load->library('form\_validation');

$this->form\_validation->set\_rules('form field name', 'Name', 'required');

**Segment**

**Syntax**

$this->uri->segment(3)

Where,  
 segment – represent how many folders contain next next step,

segment(3) – It contain three like [fans/my profile/new].

How To Get Last segment From URL InCodeigniter?

$total = $this->uri->total\_segments();

$last = $this->uri->segment($total);