CodeIgniter and MVC

Enterprise level web application development

prepared by : chirag jagani

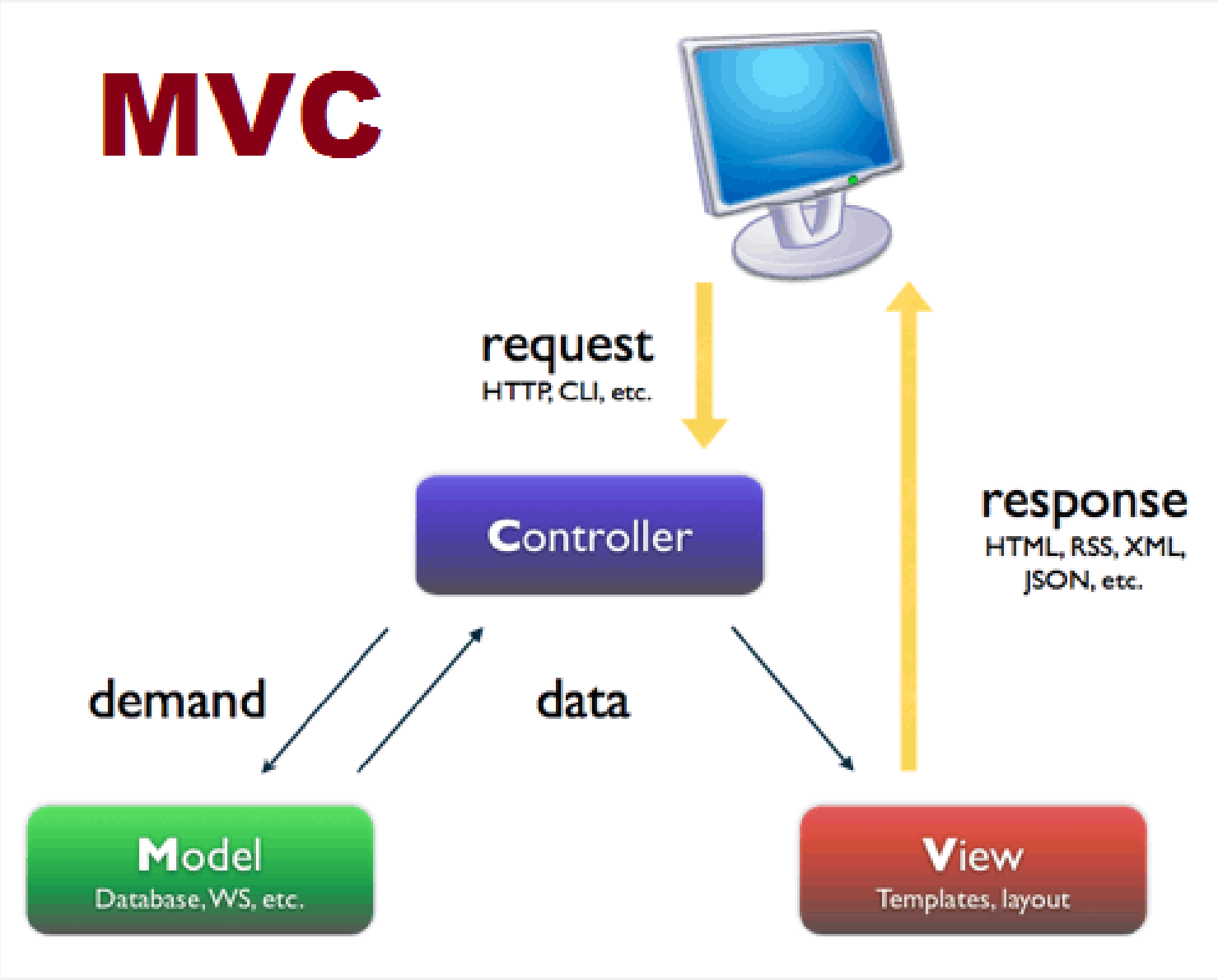
Motivation

- You have worked with PHP, for small sites this works very well. HTML files can be easily extended with dynamic content from the database, form processing, etc.
- • When sites grow, you might have realized that across multiple pages lots of code repetition occurs. This is a problem when you need to **change certain parts of a page**, that affects many or all pages.
- Furthermore, its **hard to introduce new developers** to code someone else has written. It takes a long time to get familar with the code

Model-View-Controller

- “Separation of concerns” of **Logic** and **Presentation**
- Controller: Handles all incoming HTTP requests, passes data to the views
- View: Renders the HTML output
- Models: Encapsulate Business Logic, such as interaction with the database
- For PHP we use CodeIgniter

MVC


request
HTTP, CLI, etc.
Controller
response
HTML, RSS, XML,
JSON, etc.
demand
data
Model
Database, WS, etc.
View
Templates, layout

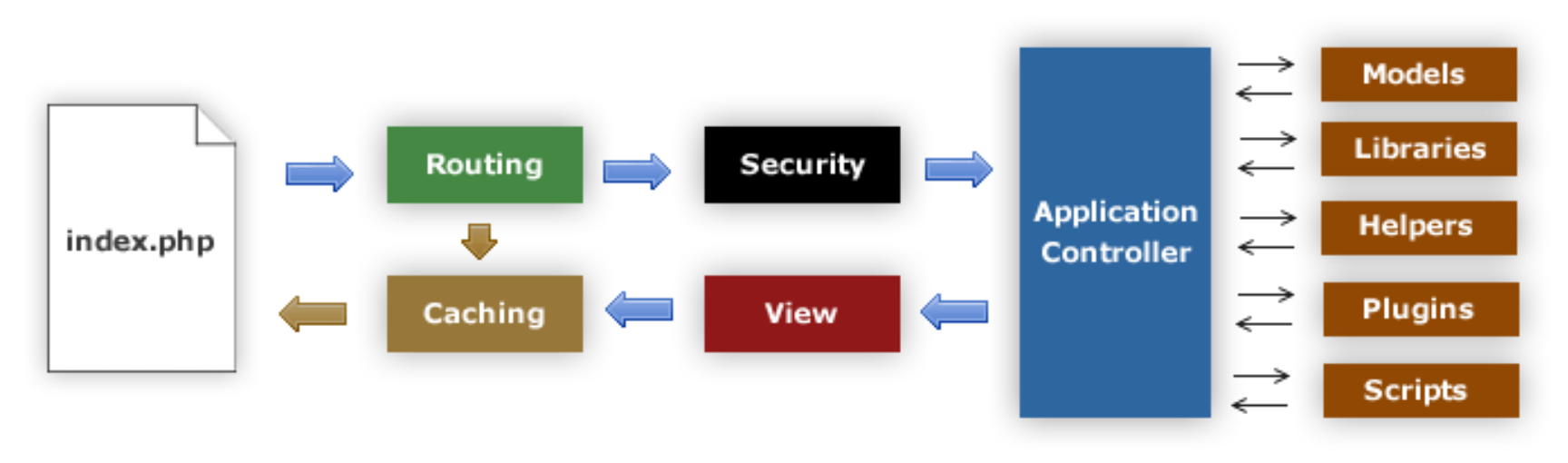
CodeIgniter

- There are countless PHP MVC frameworks, the most popular ones being **CodeIgniter** and **laravel**
- CodeIgniter is very light weight. It doesn‘t force any convention but provides many commonly required features through a set of build in libraries.
- Codeigniter is one of the best documented PHP web frameworks

CodeIgniter: Features

- Model-View-Controller Based System
- Extremely Light Weight, does not force any convention
- Full Featured database classes with support for several platforms, Active Record Database Support
- Custom Routing
- Form and Data Validation
- Security and XSS Filtering

Application Flow Chart



1. The index.php serves as the front controller, initializing the base resources needed to run CodeIgniter.
2. The Router check the HTTP request to determine what should be done with it
3. If a cache file exists, it is sent directly to the browser, bypassing the normal system execution.
4. Security. Before the application controller is loaded, the HTTP request and any user submitted data is filtered for security

Application Flow Chart(continue..)

5. The Controller loads the model, core libraries, plugins, helpers, and any other resources needed to process the specific request.
6. The finalized View is rendered then sent to the web browser to be seen. If caching is enabled, the view is cached first so that on subsequent requests it can be served

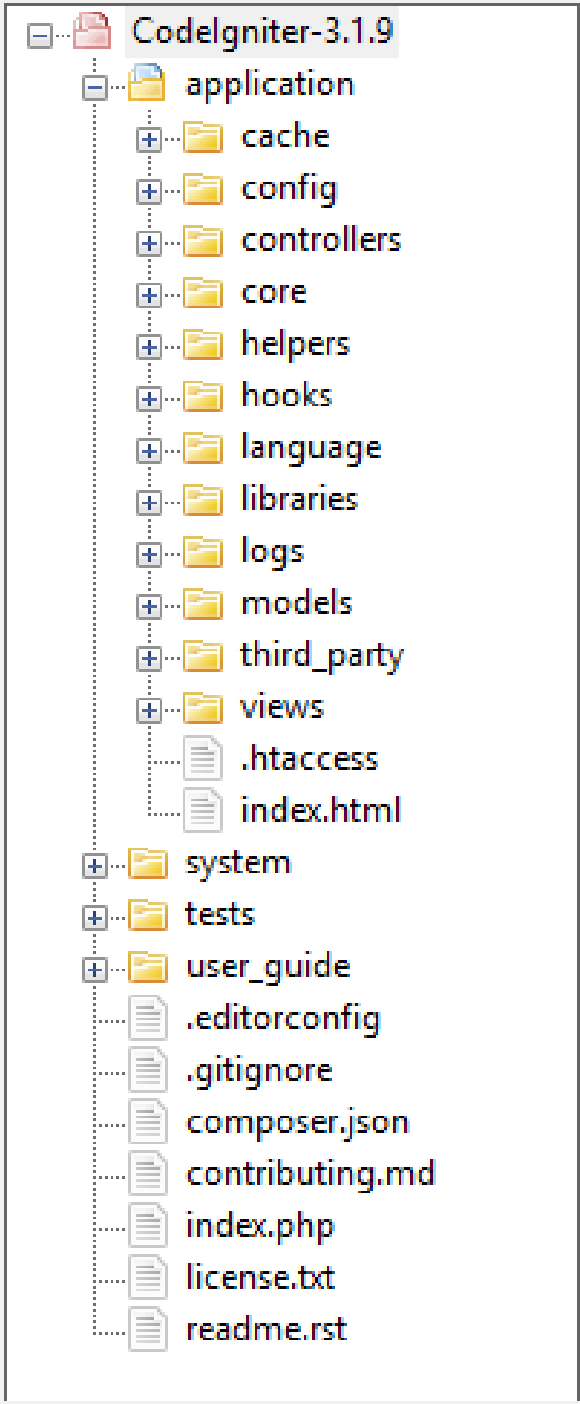
Getting Started

Directory Structure of CodeIgniter:

index.php - receives all requests and routes to the right controllers classes and actions, parameters are included in the URL

/system - contains all CodeIgniter classes and libraries provided by the framework

/application - this is where your application code is located, including the model, view and controller classes



What is Controllers?

- Take **incoming HTTP requests** and process them
- Must be the same **filename** as the capitalized **class name**
- Must **extend** the main **Controller class** of Codeigniter(**CI_Controller**)
- Each class function represents an **controller action**, which is redering a HTML page
- ***index*** is the default action

Controllers(continue..)

```php
<?php

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

class Auth extends CI_Controller {

    public function __construct() {
        parent::__construct();
        $this->load->database();
        $this->load->library(array('ion_auth', 'form_validation'));
        $this->load->helper(array('url', 'language'));

        $this->lang->load('auth');
    }

    // redirect if needed, otherwise display the user list
    public function index() {

        if (!$this->ion_auth->logged_in()) {
            // redirect them to the login page
            redirect('auth/login', 'refresh');
```

Routing Requests

- Per request default Codeigniter maps URL to controller actions:

  /index.php/controller/action

- The **default controller** is "*welcome*" and the **default action** is "*index*".
- Custom routing can be configured through:

  /application/config/routes.php

Routing Requests(continue..)

routes.php

```
46  | When you set this option to TRUE, it will replace ALL dashes in the
47  | controller and method URI segments.
48  |
49  | Examples: my-controller/index -> my_controller/index
50  |       my-controller/my-method -> my_controller/my_method
51  */
52  $route['default_controller'] = 'welcome';
53
54  $route['404_override'] = '';
55  $route['translate_uri_dashes'] = FALSE;
56
```

What is Views?

- Are HTML pages or page fragments
- Those are load and sent by the Controller to the Browser by the use of the following code.

  $this->load->view('blog_view');
- There is no application logic in the views, only display logic (at some level)
- **<?=** is short form for **<?php echo**

```
<html>
<head>
<title><?=$title?></title>
</head>
<body>
<h1><?=$heading?></h1>

<ol>

<?php foreach($todo as $item) ?

<li><?=$item?></li>

<?php endforeach; ?>
</ol>

</body>
</html>
```

What is a Model?

- In CodeIgniter**, Model** are the PHP classes where all database related manipulation is done e.g. fetching records, insert, update, and delete records.
- Within this, all data processing logic is done.
- All **model** files are managed in application/models directory and they are load and access by the controller
- Model classes are stored in your **application/models/** directory

```php
<?php
class Blog_model extends CI_Model {

        public $title;
        public $content;
        public $date;

        public function get_last_ten_entries()
        {
                $query = $this->db->get('entries', 10);
                return $query->result();
        }

        public function insert_entry()
        {
                $this->title    = $_POST['title']; // please read the below note
                $this->content  = $_POST['content'];
                $this->date     = time();

                $this->db->insert('entries', $this);
        }

        public function update_entry()
        {
                $this->title    = $_POST['title'];
                $this->content  = $_POST['content'];
                $this->date     = time();

                $this->db->update('entries', $this, array('id' => $_POST['id']));
        }

}
?>
```