Working with Server Side Scripting: PHP Framework (CodeIgniter)

Lesson Objectives

- Understanding about How to Working with Server Side Scripting using PHP Framework (CodeIgniter)

Outline

- Introduction to CodeIgniter (CI)
- The Structures of CI Website
- Using CI to Simplify Databases

Introduction to CodeIgniter (CI)

What is CI?

- CI is a powerful PHP framework with a very small footprint, built for PHP coders who need a simple and elegant toolkit to create full-featured web applications

CI Features

- Model-View-Controller Based System
- PHP 4 Compatible
- Extremely Light Weight
- Full Featured database classes with support for several platforms.
- Active Record Database Support
- Form and Data Validation
- Security and XSS Filtering
- Session Management

CI Features

- 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

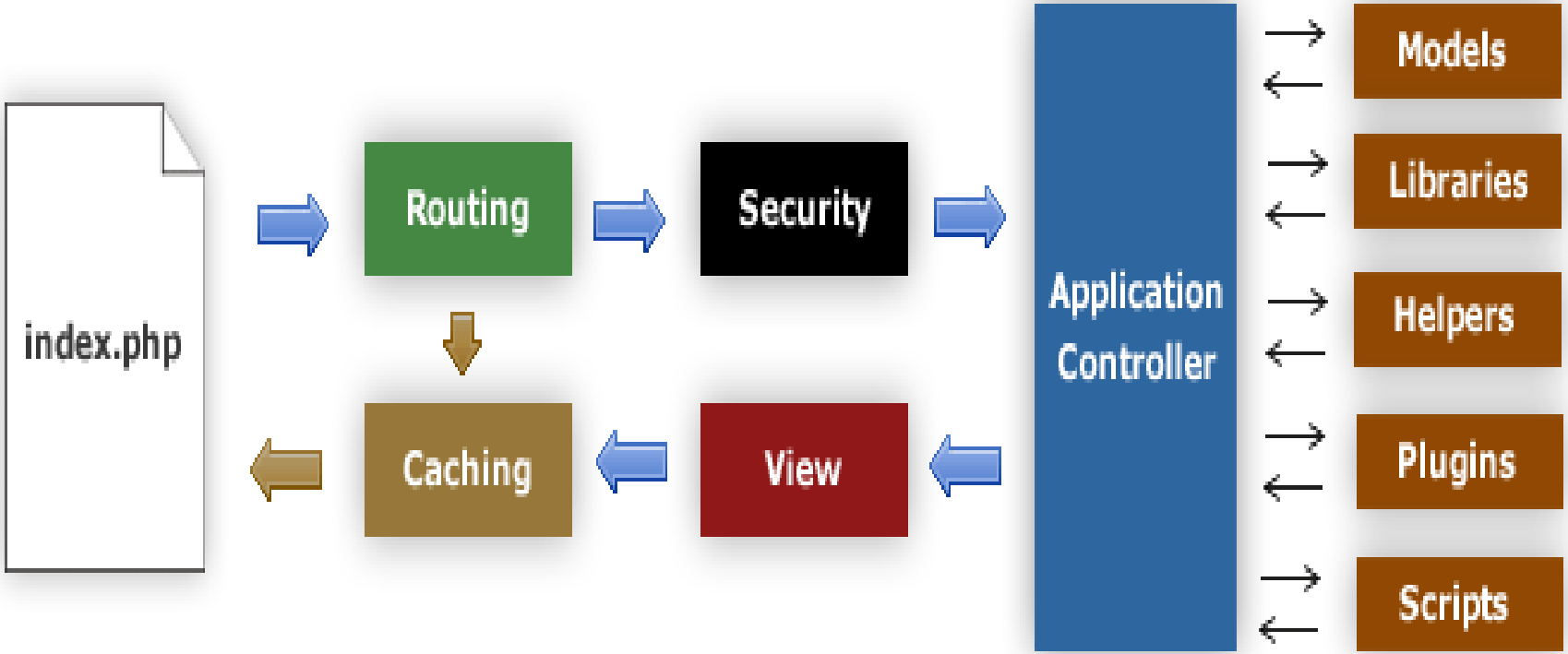
CI Features

- Data Encryption
- Benchmarking
- Full Page Caching
- Error Logging
- Application Profiling
- Scaffolding
- Calendaring Class
- User Agent Class

CI Features

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

CI Application Flowchart



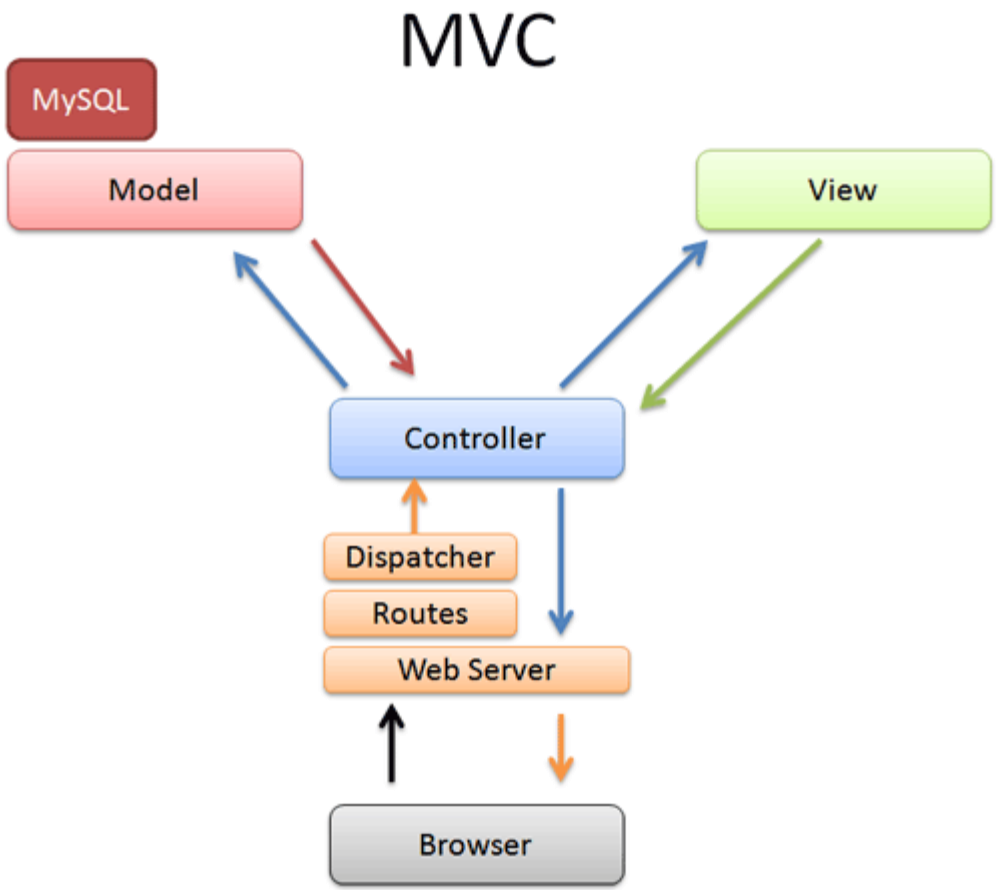
CI Application Flowchart

- The Flow
  1. The index.php serves as the front controller, initializing the base resources needed to run CodeIgniter.
  2. The Router examines 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.
  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.

MVC Concept in CI

- CI is based on the Model-View-Controller development pattern
  - Model
    - represents data structures, typically will contain functions to retrieve, insert, and update information in a database.
  - View
    - the information that is being presented to a user, normally a web page, but can also be a page fragment like a header or footer, RSS page, or any other type of "page"
  - 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

MVC Concept in CI



CI Design and Architectural Goals

- CI goal is maximum performance, capability, and flexibility in the smallest, lightest possible package
  - Dynamic Instantiation
    - components are loaded and routines executed only when requested, rather than globally
  - Loose Coupling
    - the less components depend on each other the more reusable and flexible the system becomes
  - Component Singularity
    - each class and its functions are highly autonomous in order to allow maximum usefulness

The Structures of CI Website

Controller

- Controller is simply a class file that is named in a way that can be associated with a URI
  - localhost/index.php/hello/

Controller

- Function
  - function can be add to the controller and called by the URI segment
    - localhost/index.php/hello/index

View

- View is simply a web page, or a page fragment, like a header, footer, sidebar, etc.
  - views can flexibly be embedded within other views (within other views, etc., etc.)
  - views are never called directly, they must be loaded by a controller

View

- Loading a View
  - $this->load->view('name');
    - name is the name of your view file
      - the .php file extension does not need to be specified unless you use something other than .php.

View

- Adding Dynamic Data to the View
  - data is passed from the controller to the view by way of an array or an object in the second parameter of the view loading function

```
<html>
<head>
<title><?php echo $title;?></title>
</head>
<body>
    <h1><?php echo $heading;?></h1>
    <p><?php echo $message;?></p>
</body>
</html>
```

Model

- Models are PHP classes that are designed to work with information in a database
  - $this->load->model('Model_name');

Helper Functions

- Helpers help us with tasks
  - each helper file is simply a collection of functions in a particular category
    - URL Helpers, that assist in creating links,
    - Form Helpers that help to create form elements,
    - Text Helpers perform various text formatting routines,
    - Cookie Helpers set and read cookies,
    - File Helpers help to deal with files
    - etc

Helper Functions

- Loading a Helper
  - $this->load->helper('name');
  - $this->load->helper(array('helper1', 'helper2', 'helper3'));

Plugins

- Plugins work almost identically to Helpers
  - the main difference is that a plugin usually provides a single function, whereas a Helper is usually a collection of functions
  - Helpers are also considered a part of the core system; plugins are intended to be created and shared by the community

Plugins

- Loading Plugins
  - $this->load->plugin('name');
  - $this->load->plugin(array('plugin1', 'plugin2', 'plugin3'));

Using CI to Simplify Databases

Database Configuration

- CodeIgniter has a config file that lets to store database connection values (username, password, database name, etc.)
  - application/config/database.php

Database Configuration

- Active Record
  - the Active Record Class is globally enabled or disabled by setting the $active_record variable in the database configuration file to TRUE/FALSE (boolean)
    - if the active record class not used, setting it to FALSE will utilize fewer resources when the database classes are initialized

Connecting to Database

- There are two ways to connect to a database
  - Automatically Connecting
    - the "auto connect" feature will load and instantiate the database class with every page load
    - to enable "auto connecting", add the word database to the library array, as indicated in the following file **application/config/autoload.php**

Connecting to Database

- There are two ways to connect to a database
  - Manually Connecting
    - if only some of the pages require database connectivity it can be manually connect to the database by adding this line of code in any function where it is needed, or in the class constructor to make the database available globally in that class
      - $this->load->database();

Running Queries

- To submit a query, use the following function
  - $this->db->query('YOUR QUERY HERE');
    - the query() function returns a database result object when "read" type queries are run
  - $this->db->simple_query();
    - a simplified version of the $this->db->query() function
    - it ONLY returns TRUE/FALSE on success or failure

Running Queries

- Query Bindings
  - bindings enable to simplify query syntax by letting the system put the queries together

Generating Query Results

- result()
  - this function returns the query result as an array of objects, or an empty array on failure

Generating Query Results

- result_array()
  - this function returns the query result as a pure array, or an empty array when no result is produced

Generating Query Results

- row()
  - this function returns a single result row
    - if the query has more than one row, it returns only the first row (the result is returned as an object)

Generating Query Results

- row_array()
  - identical to the above row() function, except it returns an array

Active Record Class

- CodeIgniter uses a modified version of the Active Record Database Pattern
  - this pattern allows information to be retrieved, inserted, and updated in your database with minimal scripting
  - CodeIgniter does not require that each database table be its own class file

Active Record Class

- Selecting Data
  - $this->db->get();
    - runs the selection query and returns the result.

Active Record Class

- Selecting Data
  - $this->db->get_where();
    - identical to the get() function except that it permits you to add a "where" clause in the second parameter, instead of using the db->where() function

Active Record Class

- Selecting Data
  - $this->db->select();
    - permits to write the SELECT portion of query

Active Record Class

- Inserting Data
  - $this->db->insert();
    - generates an insert string based on the data you supply, and runs the query

Active Record Class

- Inserting Data
  - $this->db->set();
    - this function enables you to set values for inserts or updates

Active Record Class

- Updating Data
  - $this->db->update();
    - Generates an update string and runs the query based on the data you supply

Active Record Class

- Deleting Data
  - $this->db->delete();
    - generates a delete SQL string and runs the query

Active Record Class

- Deleting Data
  - $this->db->empty_table();
    - generates a delete SQL string and runs the query

Active Record Class

- Method Chaining
  - allows you to simplify your syntax by connecting multiple functions (PHP5)

References

- CodeIgniter User Guide

It's Discussion Time!