Programming IV: SCS2108

Handout 5: CodeIgnitor

Prasad Wimalaratne, PhD, SMIEEE
spw@ucsc.cmb.ac.lk

1

Working with Server Side Scripting: PHP Frameworks (e.g CodeIgniter)

Code Igniter

2

Overview

- Understand the MVC software Architectural Design Pattern
- Understanding about How to Working with Server Side Scripting using PHP Framework (CodeIgniter)
- CI is a PHP framework for PHP coders to create full-featured web applications
  - Introduction to CodeIgniter (CI)
  - The Structure of CI Websites
  - Using Database connectivity in CI

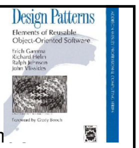
3

What is a Design Pattern?

- A general and reusable solution to a commonly occurring problem in the design of software
- A template for how to solve a problem that has been used in many different situations
  - NOT a finished design
  - the pattern must be adapted to the application
  - cannot simply translate into code

4

Origin of Design Patterns



- Architectural concept by Christopher Alexan (1977/79)
- Adapted to OO Programming by Beck and Cunningham (1987)
- Popularity in CS after the book: "Design Patterns: Elements of Re-useable Object-oriented software", 1994. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
- Now widely-used in software engineering

5

Model-view-controller (MVC)

- Model-view-controller (MVC) is a software architectural pattern for implementing user interfaces.
- It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

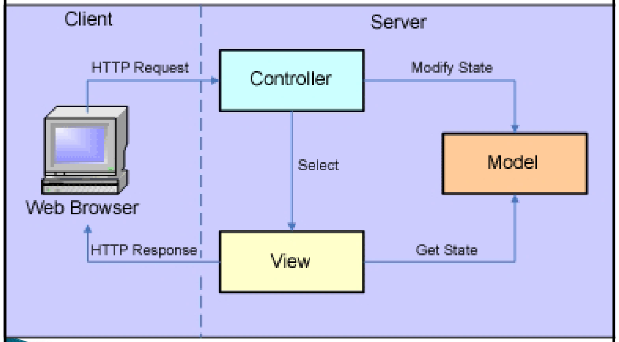
6

The MVC Design Pattern

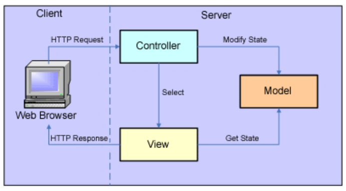
- Although originally developed for desktop computing, Model View Controller has been widely adopted as an architecture for Web applications in all major programming languages.
- Good design of architecture is crucial
- You can leverage on patterns and frameworks
  - Both have advantages and disadvantages
- MVC is the most commonly used pattern

The MVC Pattern (in practice)

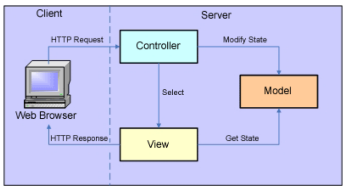
- Model
  - Contains domain-specific knowledge
  - Records the state of the application
    - E.g., what items are in shopping cart
  - Often linked to a database
  - Independent of view
  - One model can link to different views
- View
  - Presents data to the user
  - Allows user interaction
  - Does no processing
- Controller
  - defines how user interface reacts to user input (events)
  - receives messages from view (where events come from)
  - sends messages to model (tells what data to display)



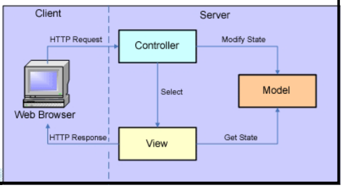
- Model
- The Model represents the state of the application and defines the business actions that modify it (persistent data and business logic). It can be inquired about its state (usually by the View) and be asked to change it (usually by the Controller). It knows nothing about the View or Controller.

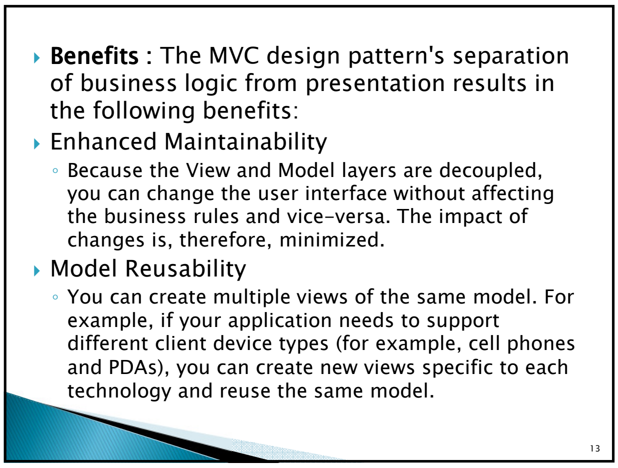


- **View**
- The View provides the presentation of the Model. It represents the look of the application, that is, its user interface. It is responsible for presenting and collecting data to and from the user. The View can get the Model's state but cannot modify it

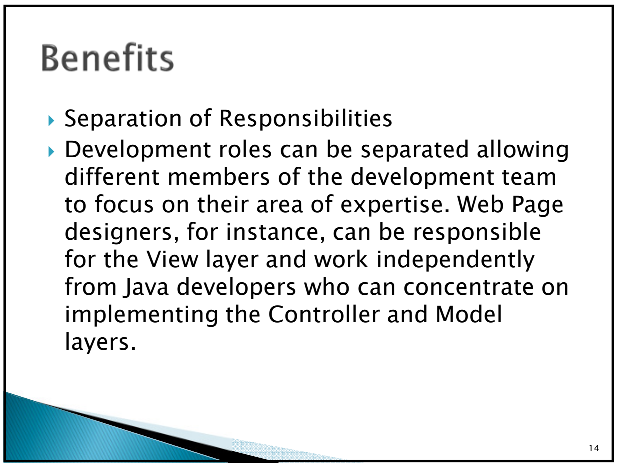


- **Controller**
- The Controller reacts to user input and informs the Model to change its state accordingly. Specifically, it processes incoming user requests by dispatching them to the appropriate business logic functions (in the Model) and selecting the response to the user (the View) based on the outcome.



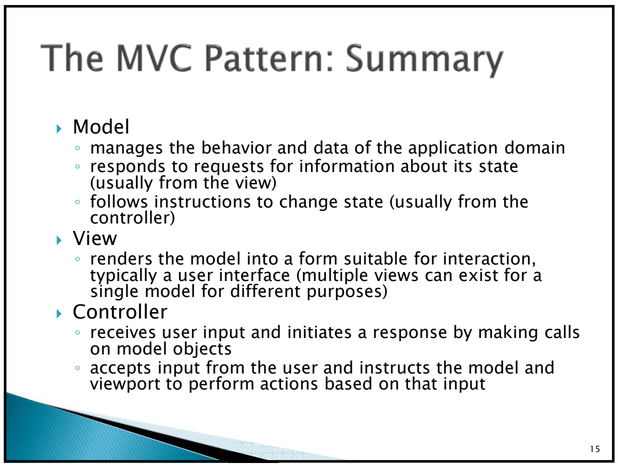


- **Benefits** : The MVC design pattern's separation of business logic from presentation results in the following benefits:
- Enhanced Maintainability
  - Because the View and Model layers are decoupled, you can change the user interface without affecting the business rules and vice-versa. The impact of changes is, therefore, minimized.
- Model Reusability
  - You can create multiple views of the same model. For example, if your application needs to support different client device types (for example, cell phones and PDAs), you can create new views specific to each technology and reuse the same model.



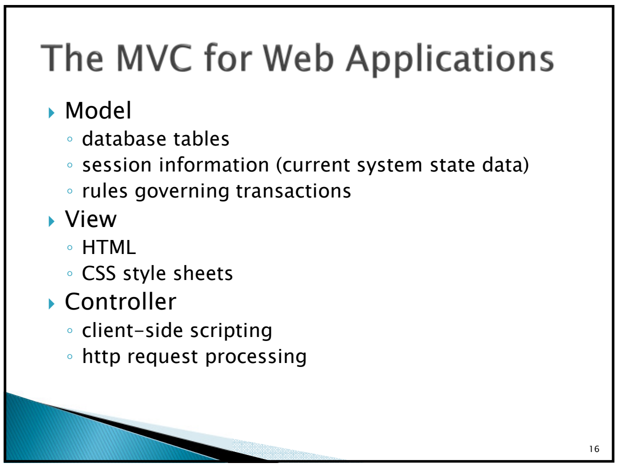
Benefits

- Separation of Responsibilities
- Development roles can be separated allowing different members of the development team to focus on their area of expertise. Web Page designers, for instance, can be responsible for the View layer and work independently from Java developers who can concentrate on implementing the Controller and Model layers.



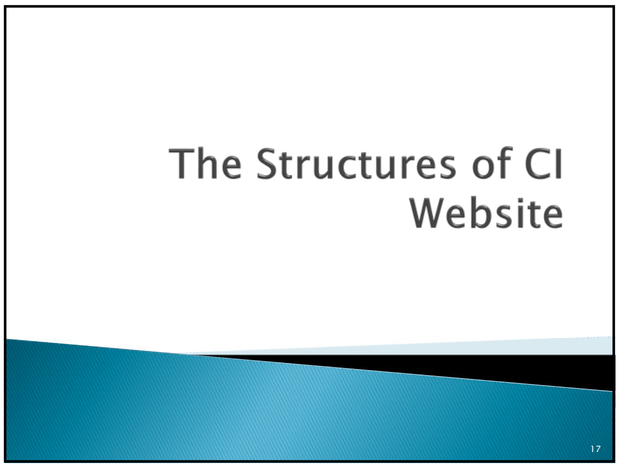
The MVC Pattern: Summary

- Model
  - manages the behavior and data of the application domain
  - responds to requests for information about its state (usually from the view)
  - follows instructions to change state (usually from the controller)
- View
  - renders the model into a form suitable for interaction, typically a user interface (multiple views can exist for a single model for different purposes)
- Controller
  - receives user input and initiates a response by making calls on model objects
  - accepts input from the user and instructs the model and viewport to perform actions based on that input

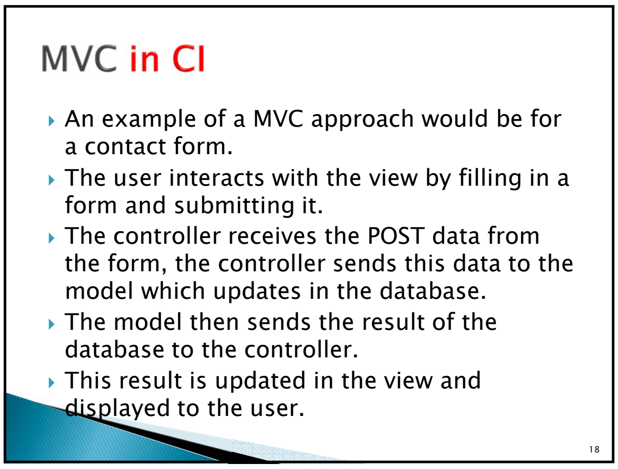


The MVC for Web Applications

- Model
  - database tables
  - session information (current system state data)
  - rules governing transactions
- View
  - HTML
  - CSS style sheets
- Controller
  - client-side scripting
  - http request processing



The Structures of CI Website



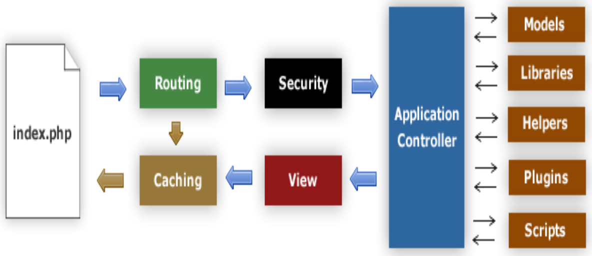
MVC in CI

- An example of a MVC approach would be for a contact form.
- The user interacts with the view by filling in a form and submitting it.
- The controller receives the POST data from the form, the controller sends this data to the model which updates in the database.
- The model then sends the result of the database to the controller.
- This result is updated in the view and displayed to the user.

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

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.

CI Design and Architectural Goals

- CI goal is to optimize 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

CI Features

- Model-View-Controller Based System
- Light Weight
- Full Featured database classes with support for several platforms.
- Form and Data Validation
- Security features
- Session Management
- Data Encryption
- Page Caching
- Error Logging
- Calendaring Class
- ...etc

CI Features

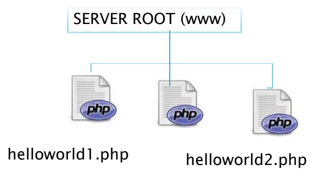
- Email Sending Class. Supports Attachments, HTML/Text email, multiple protocols (sendmail, SMTP, and Mail) and more.
- Image Manipulation Library (cropping, resizing, rotating, etc.). Supports GD, ImageMagick, and NetPBM
- File Uploading Class
- FTP Class
- Localization

CI Features

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

25

Server Access(Regular PHP)

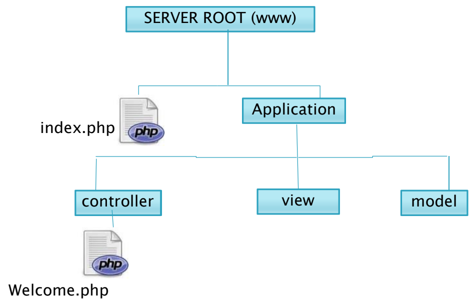


http://localhost/helloworld1

http://localhost/helloworld2

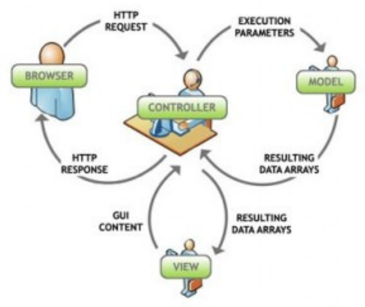
26

CI Routing



http://localhost/index.php/Welcome

27



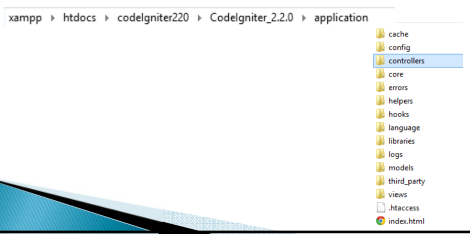
28

Controller in CI

29

http://localhost/index.php/Welcome

- Not Directly Access file Application/Controller/Welcome.php
- But Index.php route to Welcome.php in Controller



xampp ▸ htdocs ▸ codeIgniter220 ▸ CodeIgniter_2.2.0 ▸ application

cache
config
controllers
core
errors
helpers
hooks
language
libraries
logs
models
third_party
views
.htaccess
index.html

30

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**: The controller acts as the in between of view and model and, as the name suggests, it controls what is sent to the view from the model. In CI, the controller is also the place to load libraries and helpers.
- Note: Class names must start with an uppercase letter.
- always make sure your controller **extends** the parent controller class so that it can inherit all its functions.

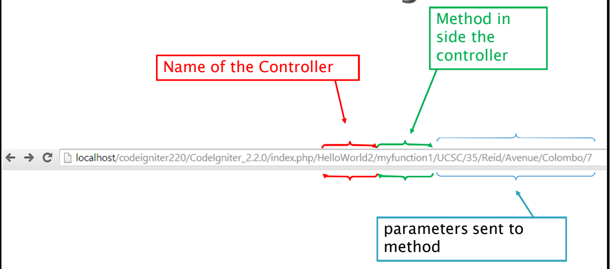
31

Controller

- Function
  - function can be add to the controller and called by the URI segment
    - localhost/index.php/hello/index2
- CodeIgniter can be told to load a default controller when a URI is not present, as will be the case when only your site root URL is requested. To specify a default controller, open your application/config/routes.php file and set this variable:
- $route['default_controller'] = “HellowWorld2';
- Where HellowWorld2 is the name of the controller class you want used. If you now load your main index.php file without specifying any URI segments you'll see your Hello World … message by default
- localhost/index.php/

32

URL format for routing



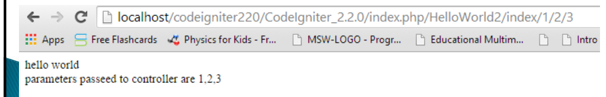
localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld2/myfunction1/UCSC/35/Reid/Avenue/Colombo/7

33

Method inside Controller and it parameters

HelloWorld2.php inside Controller Directory

```
<?php
    class HelloWorld2 extends CI_controller{
        public function index($param1,$param2,$param3)
        {
            echo "hello world <br>" ;
            echo "parameters passeed to controller are $param1,
$param2,$param3";
        }
    }
?>
```



localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld2/index/1/2/3

```
hello world
parameters passeed to controller are 1,2,3
```

34

- In some cases you may want certain functions hidden from public access. To make a function private, simply add an underscore as the name prefix and it will not be served via a URL request. For example, if you were to have a function like this

```
private function _utility()
{
  // some code
}
```

- Trying to access it via the URL, like this, will not work:: example.com/index.php/**blog**/**_utility**/

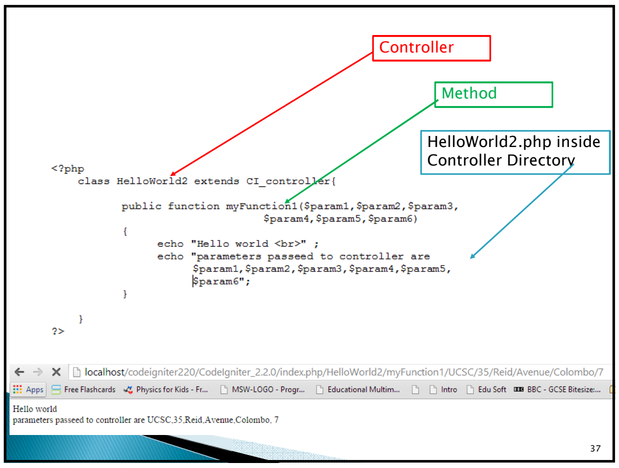
35

Passing URI Segments to your Functions

- If your URI contains more then two segments they will be passed to your function as parameters.
- For example, lets say you have a URI like this:
- example.com/index.php/products/shoes/sandals/123
- Your function will be passed URI segments 3 and 4 ("sandals" and "123"):

```
<?php
class Products extends CI_Controller {

   public function shoes($sandals, $id)
   {
      echo $sandals;
      echo $id;
   }
}
?>
```

36

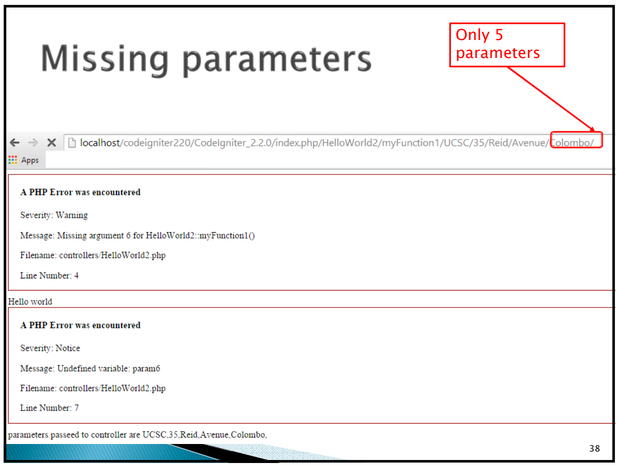


```
<?php
    class HelloWorld2 extends CI_controller{

            public function myFunction1($param1,$param2,$param3,
                                $param4,$param5,$param6)
            {
                echo "Hello world <br>" ;
                echo "parameters passeed to controller are
                    $param1,$param2,$param3,$param4,$param5,
                    $param6";
            }

    }
?>
```

localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld2/myFunction1/UCSC/35/Reid/Avenue/Colombo/7

```
Hello world
parameters passeed to controller are UCSC,35,Reid,Avenue,Colombo, 7
```

37

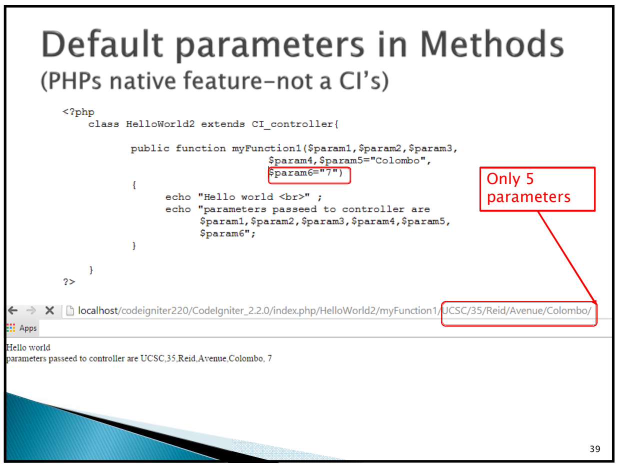


Missing parameters

localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld2/myFunction1/UCSC/35/Reid/Avenue/Colombo/

```
A PHP Error was encountered

Severity: Warning

Message: Missing argument 6 for HelloWorld2::myFunction1()

Filename: controllers/HelloWorld2.php

Line Number: 4

Hello world

A PHP Error was encountered

Severity: Notice

Message: Undefined variable: param6

Filename: controllers/HelloWorld2.php

Line Number: 7

parameters passeed to controller are UCSC,35,Reid,Avenue,Colombo,
```

38



Default parameters in Methods

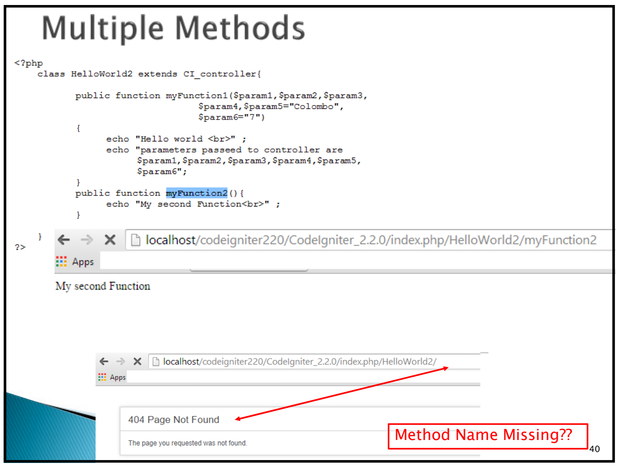
(PHPs native feature-not a CI's)

```
<?php
    class HelloWorld2 extends CI_controller{

            public function myFunction1($param1,$param2,$param3,
                                $param4,$param5="Colombo",
                                $param6="7")
            {
                echo "Hello world <br>" ;
                echo "parameters passeed to controller are
                    $param1,$param2,$param3,$param4,$param5,
                    $param6";
            }

    }
?>
```

localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld2/myFunction1/UCSC/35/Reid/Avenue/Colombo/

```
Hello world
parameters passeed to controller are UCSC,35,Reid,Avenue,Colombo, 7
```

39



Multiple Methods

```
<?php
    class HelloWorld2 extends CI_controller{

            public function myFunction1($param1,$param2,$param3,
                                $param4,$param5="Colombo",
                                $param6="7")
            {
                echo "Hello world <br>" ;
                echo "parameters passeed to controller are
                    $param1,$param2,$param3,$param4,$param5,
                    $param6";
            }
            public function myFunction2(){
                echo "My second Function<br>" ;
            }

    }
?>
```

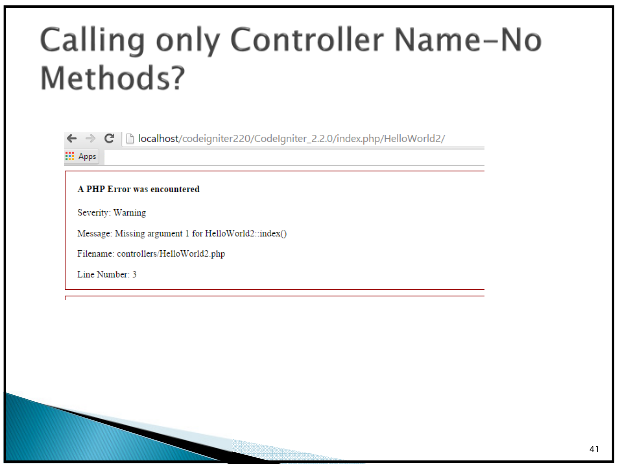
localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld2/myFunction2

```
My second Function
```

localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld2/

```
404 Page Not Found

The page you requested was not found.
```

40

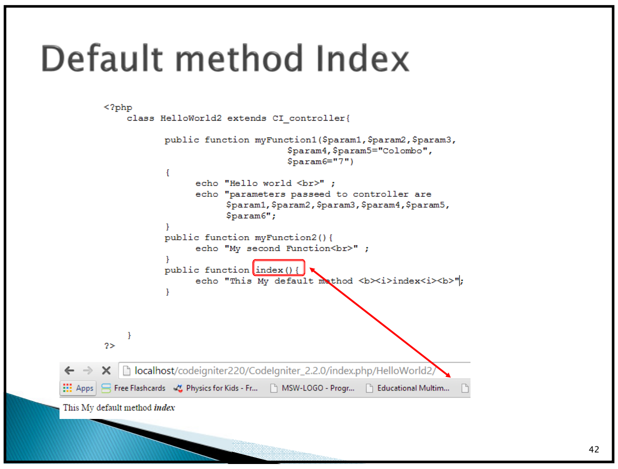


Calling only Controller Name-No Methods?

localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld2/

```
A PHP Error was encountered

Severity: Warning

Message: Missing argument 1 for HelloWorld2::index()

Filename: controllers/HelloWorld2.php

Line Number: 3
```

41



Default method Index

```
<?php
    class HelloWorld2 extends CI_controller{

            public function myFunction1($param1,$param2,$param3,
                                $param4,$param5="Colombo",
                                $param6="7")
            {
                echo "Hello world <br>" ;
                echo "parameters passeed to controller are
                    $param1,$param2,$param3,$param4,$param5,
                    $param6";
            }
            public function myFunction2(){
                echo "My second Function<br>" ;
            }
            public function index(){
                echo "This My default method <b><i>index<i><b>";
            }


    }
?>
```

localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld2/

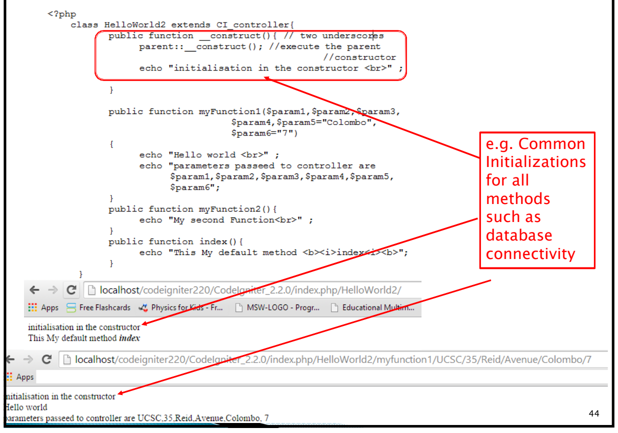
This My default method ***index***

42

Default method Index

- Function name is **index()**. The "index" function is always loaded by default if the **second segment** of the URI is empty.
- **The second segment of the URI determines which function in the controller gets called.**

Constructor – Initializations



```
<?php
    class HelloWorld2 extends CI_controller{
        public function __construct(){ // two underscores
            parent::__construct(); //execute the parent
                                    //constructor
            echo "initialisation in the constructor <br>" ;

        }

        public function myFunction1($param1,$param2,$param3,
                                    $param4,$param5="Colombo",
                                    $param6="7")
        {
            echo "Hello world <br>" ;
            echo "parameters passeed to controller are
                  $param1,$param2,$param3,$param4,$param5,
                  $param6";
        }
        public function myFunction2(){
            echo "My second Function<br>" ;
        }
        public function index(){
            echo "This My default method <b><i>index<i><b>";
        }
    }
```

e.g. Common Initializations for all methods such as database connectivity

localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld2/

```
initialisation in the constructor
This My default method index
```

localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld2/myfunction1/UCSC/35/Reid/Avenue/Colombo/7

```
nitialisation in the constructor
Hello world
parameters passeed to controller are UCSC,35,Reid,Avenue,Colombo, 7
```

View in CI

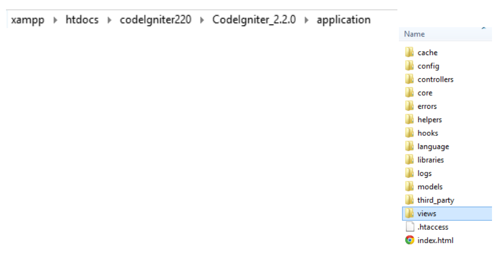
View

- 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.)
  - views are never called directly, they must be loaded by a controller
- **View**: The view deals with displaying the data and interface controls to the user with.
  In CI the view could be a web page, rss feed, ajax data or any other "page".

View

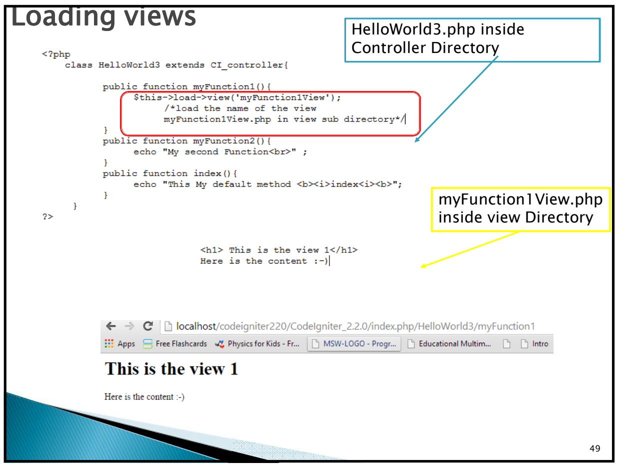
- Loading a View
  - $this->load->view('name');
    - name is the name of your view file
      - the .php file extension does not need to be specified unless you use something other than .php.

View



xampp ▸ htdocs ▸ codeIgniter220 ▸ CodeIgniter_2.2.0 ▸ application

```
Name
cache
config
controllers
core
errors
helpers
hooks
language
libraries
logs
models
third_party
views
.htaccess
index.html
```



Loading views

HelloWorld3.php inside Controller Directory

```
<?php
    class HelloWorld3 extends CI_controller{

        public function myFunction1(){
            $this->load->view('myFunction1View');
                /*load the name of the view
                myFunction1View.php in view sub directory*/
        }
        public function myFunction2(){
            echo "My second Function<br>" ;
        }
        public function index(){
            echo "This My default method <b><i>index<i><b>";
        }
    }
?>
```

myFunction1View.php inside view Directory

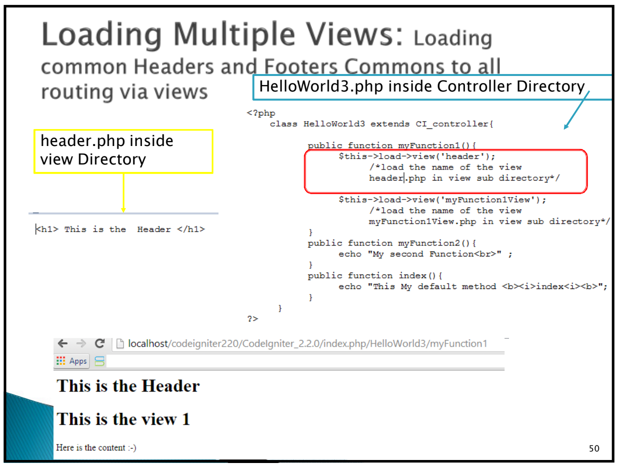
```
<h1> This is the view 1</h1>
Here is the content :-)
```

localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld3/myFunction1

This is the view 1

Here is the content :-)

49



Loading Multiple Views: Loading common Headers and Footers Commons to all routing via views

HelloWorld3.php inside Controller Directory

```
<?php
    class HelloWorld3 extends CI_controller{

        public function myFunction1(){
            $this->load->view('header');
                /*load the name of the view
                header.php in view sub directory*/

            $this->load->view('myFunction1View');
                /*load the name of the view
                myFunction1View.php in view sub directory*/
        }
        public function myFunction2(){
            echo "My second Function<br>" ;
        }
        public function index(){
            echo "This My default method <b><i>index<i><b>";
        }
    }
?>
```

header.php inside view Directory

```
<h1> This is the  Header </h1>
```

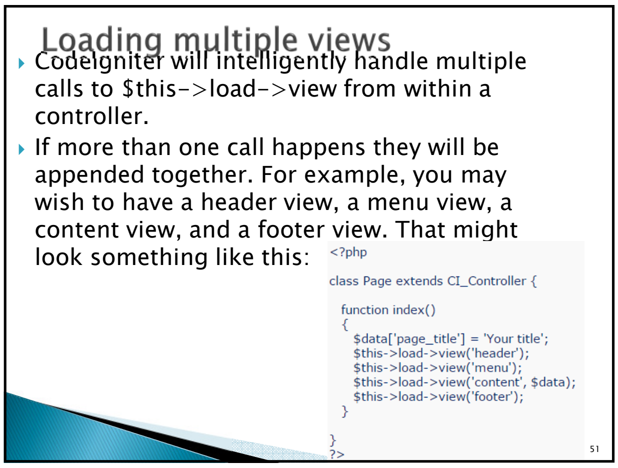
localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld3/myFunction1

This is the Header

This is the view 1

Here is the content :-)

50

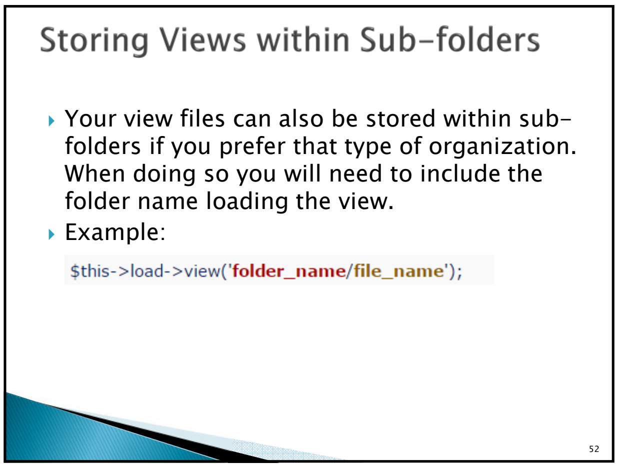


Loading multiple views

- CodeIgniter will intelligently handle multiple calls to $this->load->view from within a controller.
- If more than one call happens they will be appended together. For example, you may wish to have a header view, a menu view, a content view, and a footer view. That might look something like this:

```
<?php

class Page extends CI_Controller {

  function index()
  {
    $data['page_title'] = 'Your title';
    $this->load->view('header');
    $this->load->view('menu');
    $this->load->view('content', $data);
    $this->load->view('footer');
  }

}
?>
```

51

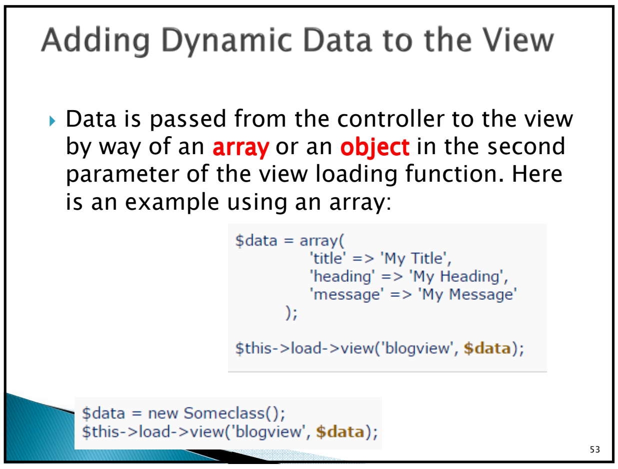


Storing Views within Sub-folders

- Your view files can also be stored within sub-folders if you prefer that type of organization. When doing so you will need to include the folder name loading the view.
- Example:

```
$this->load->view('folder_name/file_name');
```

52



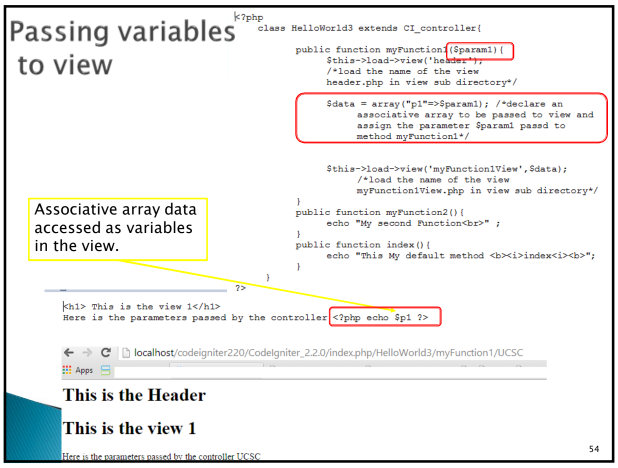
Adding Dynamic Data to the View

- 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. Here is an example using an array:

```
$data = array(
          'title' => 'My Title',
          'heading' => 'My Heading',
          'message' => 'My Message'
     );

$this->load->view('blogview', $data);
```

```
$data = new Someclass();
$this->load->view('blogview', $data);
```

53



Passing variables to view

```
<?php
    class HelloWorld3 extends CI_controller{

        public function myFunction1($param1){
            $this->load->view('header');
            /*load the name of the view
            header.php in view sub directory*/

            $data = array("p1"=>$param1); /*declare an
                associative array to be passed to view and
                assign the parameter $param1 passd to
                method myFunction1*/


            $this->load->view('myFunction1View',$data);
                /*load the name of the view
                myFunction1View.php in view sub directory*/
        }
        public function myFunction2(){
            echo "My second Function<br>" ;
        }
        public function index(){
            echo "This My default method <b><i>index<i><b>";
        }
    }
?>
```

Associative array data accessed as variables in the view.

```
<h1> This is the view 1</h1>
Here is the parameters passed by the controller <?php echo $p1 ?>
```

localhost/codeigniter220/CodeIgniter_2.2.0/index.php/HelloWorld3/myFunction1/UCSC

This is the Header

This is the view 1

Here is the parameters passed by the controller UCSC

54

```
<?php
class Blog extends CI_Controller {

	function index()
	{
		$data['title'] = "My Real Title";
		$data['heading'] = "My Real Heading";

		$this->load->view('blogview', $data);
	}
}
?>
```

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

Creating Loops

- The data array you pass to your view files is not limited to simple variables. You can pass multi dimensional arrays, which can be looped to generate multiple rows. For example, if you pull data from your database it will typically be in the form of a multi-dimensional array.

- Here's a simple example. Add this to your controller:

```
<?php
class Blog extends CI_Controller {

	function index()
	{
		$data['todo_list'] = array('Clean House', 'Call Mom', 'Run Errands');

		$data['title'] = "My Real Title";
		$data['heading'] = "My Real Heading";

		$this->load->view('blogview', $data);
	}
}
?>
```

```
<?php
class Blog extends CI_Controller {

	function index()
	{
		$data['todo_list'] = array('Clean House', 'Call Mom', 'Run Errands');

		$data['title'] = "My Real Title";
		$data['heading'] = "My Real Heading";

		$this->load->view('blogview', $data);
	}
}
?>
```

- Now open your view file and create a loop:

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

<h3>My Todo List</h3>

<ul>
<?php foreach ($todo_list as $item):?>

<li><?php echo $item;?></li>

<?php endforeach;?>
</ul>

</body>
</html>
```

Alternate PHP Syntax for View Files

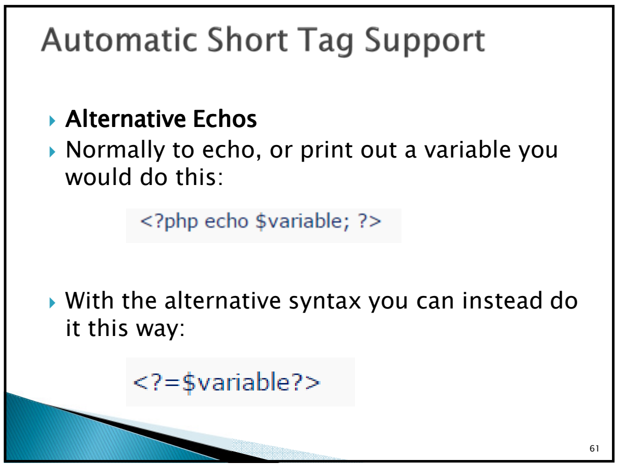
- https://ellislab.com/codeigniter/user-guide/general/alternative_php.ht
- To minimize the PHP code in these files, and to make it easier to identify the code blocks it is recommended that you use PHPs alternative syntax for control structures and short tag echo statements. If you are not familiar with this syntax, it allows you to eliminate the braces from your code, and eliminate "echo" statements

http://php.net/manual/en/control-structures.alternative-syntax.php

Alternate PHP Syntax

- php supports an alternative syntax for control structures allowing you to jump in and out of php and html. Any code (html, php, anything) nested within a conditional will only parse if the defined conditions are matched. The end result is clean, organized, efficient code.

- The best time to use this alternative style is when you need to mix PHP codes with HTML. In CI, this normally happens in views.



Automatic Short Tag Support

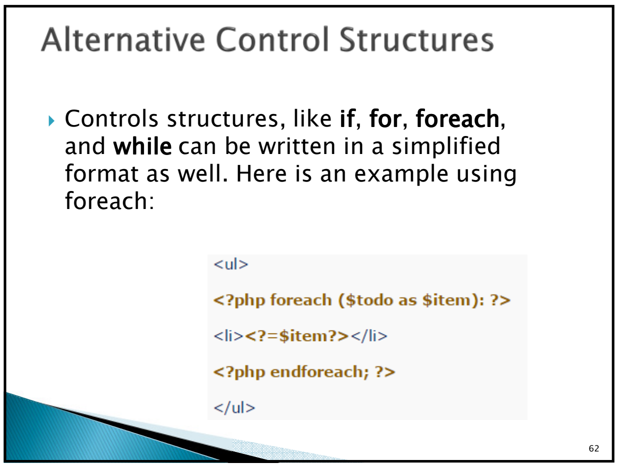
- **Alternative Echos**
- Normally to echo, or print out a variable you would do this:

```
<?php echo $variable; ?>
```

- With the alternative syntax you can instead do it this way:

```
<?=$variable?>
```

61

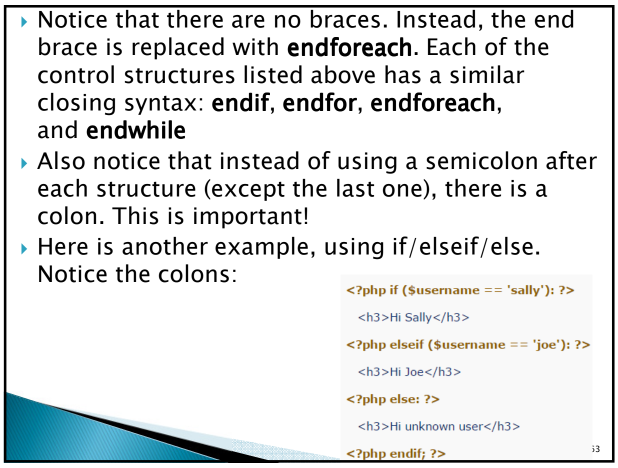


Alternative Control Structures

- Controls structures, like **if**, **for**, **foreach**, and **while** can be written in a simplified format as well. Here is an example using foreach:

```
<ul>
<?php foreach ($todo as $item): ?>
<li><?=$item?></li>
<?php endforeach; ?>
</ul>
```

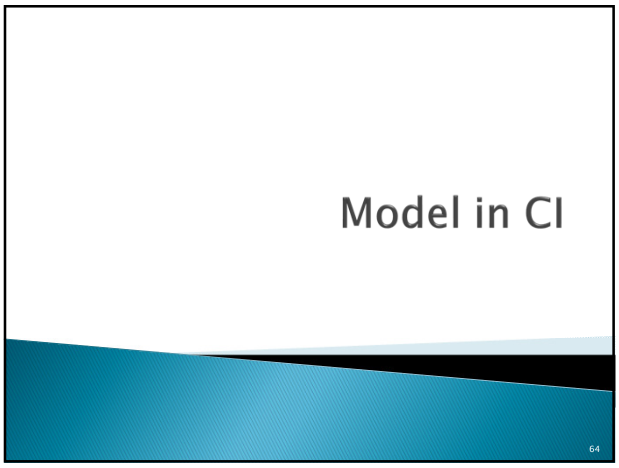
62



- Notice that there are no braces. Instead, the end brace is replaced with **endforeach**. Each of the control structures listed above has a similar closing syntax: **endif**, **endfor**, **endforeach**, and **endwhile**
- Also notice that instead of using a semicolon after each structure (except the last one), there is a colon. This is important!
- Here is another example, using if/elseif/else. Notice the colons:

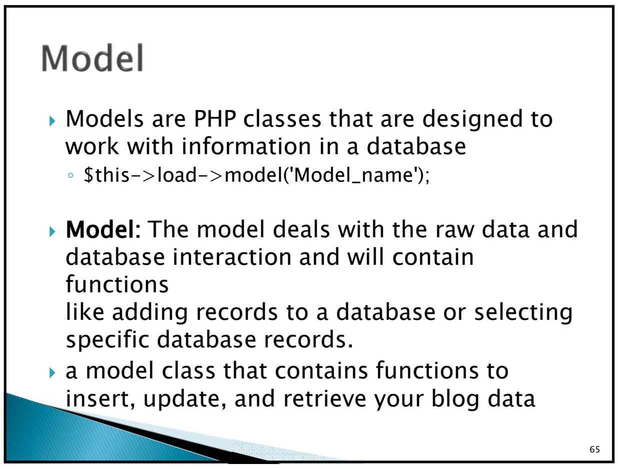
```
<?php if ($username == 'sally'): ?>
  <h3>Hi Sally</h3>
<?php elseif ($username == 'joe'): ?>
  <h3>Hi Joe</h3>
<?php else: ?>
  <h3>Hi unknown user</h3>
<?php endif; ?>
```

63



Model in CI

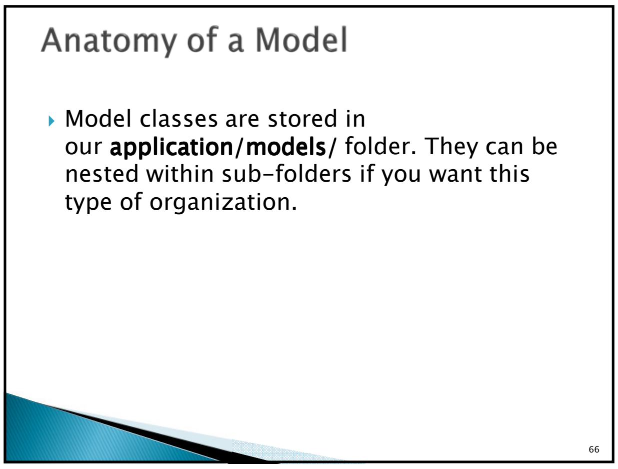
64



Model

- Models are PHP classes that are designed to work with information in a database
  - $this->load->model('Model_name');
- **Model:** The model deals with the raw data and database interaction and will contain functions
  like adding records to a database or selecting specific database records.
- a model class that contains functions to insert, update, and retrieve your blog data

65



Anatomy of a Model

- Model classes are stored in our **application/models/** folder. They can be nested within sub-folders if you want this type of organization.

66

- Where **Model_name** is the name of your class. Class names **must** have the first letter capitalized with the rest of the name lowercase. Make sure your class extends the base Model class.
- The file name will be a lower case version of your class name.
- Your file will be this:

```
application/models/user_model.php
```

Model

- Your models will typically be loaded and called from within your controller functions. To load a model you will use the following function

```
$this->load->model('blog/queries');
```

- If your model is located in a sub-folder, include the relative path from your models folder. For example, if you have a model located at **application/models/blog/queries.php** you'll load it using:

```
$this->load->model('blog/queries');
```

Model

- Once loaded, you will access your model functions using an object with the same name as your class:

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

$this->Model_name->function();
```

- If you would like your model assigned to a different object name you can specify it via the second parameter of the loading function:

```
$this->load->model('Model_name', 'fubar');

$this->fubar->function();
```

Model

- Here is an example of a controller, that loads a model, then serves a view:

```
class Blog_controller extends CI_Controller {

   function blog()
   {
      $this->load->model('Blog');

      $data['query'] = $this->Blog->get_last_ten_entries();

      $this->load->view('blog', $data);
   }
}
```

Helper Functions

- Helpers, as the name suggests, help you with tasks.
- Each helper file is simply <u>a collection of functions in a particular category</u>.
- There are **URL** Helpers, that assist in creating links, there are **Form** Helpers that help you create form elements, **Text** Helpers perform various text formatting routines, Cookie Helpers set and read cookies, File Helpers help you deal with files, etc.

Helper Functions

- Helpers are typically stored in your system/helpers, or application/helpers directory.
-
- 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'));

Using a Helper

https://ellislab.com/codeigniter/user-guide/general/helpers.html

- Once you've loaded the Helper File containing the function you intend to use, you'll call it the way you would a standard PHP function.
- For example, to create a link using the **anchor()** function in one of your view files you would do this:

```
<?php echo anchor('blog/comments', 'Click Here');?>
```

- Where "Click Here" is the name of the link, and "blog/comments" is the URI to the controller/function you wish to link to.

Plugins

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

Plugins

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

Example: Model

- Connecting Model to controller:
  - E.g pass *User Name* from **Controller** to **Model** and **Model** returns the *User Profile*. Then the **Controller** passes *User Profile* is to view for display. (Note: that this step hard cords *User Name* and *User Profile* for simplicity. Subsequently learn how to connect to database within the **Model** and access *User Profile* from a database.

HelloWorld4.php Controller

```
<?php
    class HelloWorld4 extends CI_controller{

            public function myFunction1($param1){

                  $this->load->model('hello_model');
                  /*load the name of the model -
                        hello_model.php in model sub directory*/

            $profile =  $this->hello_model->getUserProfile("spw");
                  /*access the profile of the given user and assign
                  the profile in to a variable*/

                  print_r($profile);
                  // debug code to print the value in $profile
```

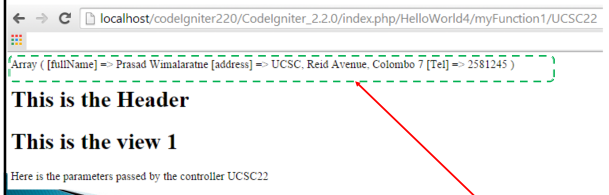
Hello_model.php Model

```
<?php
class Hello_model extends CI_Model {

    public function getUserProfile($userName)
    {
        return array("fullName"=>"Prasad Wimalaratne",
"address"=> "UCSC, Reid Avenue, Colombo 7", "Tel"=>2581245);
    }

}
?>
```

79

Output: Controller ->Model->Controller

Controller prints the profile data sent by the Model as a debug code.
http://localhost/codeIgniter220/CodeIgniter_2.2.0/index.php/HelloWorld4/myFunction1/UCSC22



localhost/codeIgniter220/CodeIgniter_2.2.0/index.php/HelloWorld4/myFunction1/UCSC22

```
Array ( [fullName] => Prasad Wimalaratne [address] => UCSC, Reid Avenue, Colombo 7 [Tel] => 2581245 )
```

This is the Header

This is the view 1

Here is the parameters passed by the controller UCSC22

```
print_r($profile);
// debug code to print the value in $profile
```

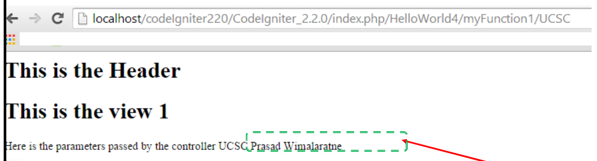
80

Output: Controller ->Model->Controller->view

```
1     $this->load->model('hello_model');
      /*load the name of the model
            hello_model.php in model sub directory*/
2  $profile =  $this->hello_model->getUserProfile("spw");
      /*access the profile of the given user and assign
      the profile in to a variable*/

3     $data['profile'] = $profile;
      /*define an index in the associative array called
'profile' and assign the values to data sent by controller*/
4     $this->load->view('myFunction1View',$data);
            /*load the name of the view
            myFunction1View.php in view sub directory*/
    }
```

81

View prints the profile data sent by the controller.
http://localhost/codeIgniter220/CodeIgniter_2.2.0/index.php/HelloWorld4/myFunction1/UCSC22



localhost/codeIgniter220/CodeIgniter_2.2.0/index.php/HelloWorld4/myFunction1/UCSC

This is the Header

This is the view 1

Here is the parameters passed by the controller UCSC Prasad Wimalaratne

- Output: Controller ->Model->Controller->view

82

Using CI to Simplify Databases

83

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**

84

Auto-loading Models

- If you find that you need a particular model globally throughout your application, you can tell CodeIgniter to auto-load it during system initialization. This is done by opening the application/config/autoload.php file and adding the model to the autoload array.

85

Connecting to your Database

- When a model is loaded it does NOT connect automatically to your database. The following options for connecting are available to you:
- You can connect using the standard database methods described here, either from within your Controller class or your Model class.
- You can tell the model loading function to auto-connect by passing TRUE (boolean) via the third parameter, and connectivity settings, as defined in your database config file will be used:

```
$this->load->model('Model_name', '', TRUE);
```

86

manually database connectivity

- You can manually pass database connectivity settings via the third parameter:

```
$config['hostname'] = "localhost";
$config['username'] = "myusername";
$config['password'] = "mypassword";
$config['database'] = "mydatabase";
$config['dbdriver'] = "mysql";
$config['dbprefix'] = "";
$config['pconnect'] = FALSE;
$config['db_debug'] = TRUE;

$this->load->model('Model_name', '', $config);
```

87

Connecting to Database

- **Manually Connecting**
  - If only some of your pages require database connectivity you can manually connect to your database by adding this line of code in any function where it is needed, or in your class constructor to make the database available globally in that class.
    - $this->load->database();

88

Database Configuration

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

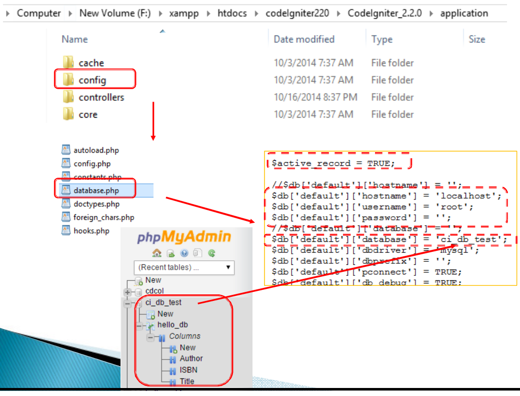
89

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

90

Configure 'database.php' in config directory



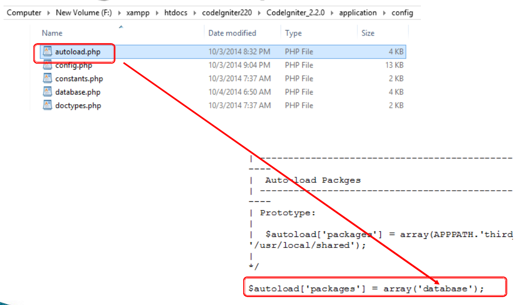
Computer ▸ New Volume (F:) ▸ xampp ▸ htdocs ▸ codeIgniter220 ▸ CodeIgniter_2.2.0 ▸ application

Name	Date modified	Type	Size
cache	10/3/2014 7:37 AM	File folder	
config	10/3/2014 7:37 AM	File folder	
controllers	10/16/2014 8:37 PM	File folder	
core	10/3/2014 7:37 AM	File folder	

autoload.php
config.php
constants.php
database.php
doctypes.php
foreign_chars.php
hooks.php

```
$active_record = TRUE;

//$db['default']['hostname'] = '';
$db['default']['hostname'] = 'localhost';
$db['default']['username'] = 'root';
$db['default']['password'] = '';
//$db['default']['database'] = '';
$db['default']['database'] = 'ci_db_test';
$db['default']['dbdriver'] = 'mysql';
$db['default']['dbprefix'] = '';
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
```

Configure 'autoload.php' in config directory



Computer ▸ New Volume (F:) ▸ xampp ▸ htdocs ▸ codeIgniter220 ▸ CodeIgniter_2.2.0 ▸ application ▸ config

Name	Date modified	Type	Size
autoload.php	10/3/2014 8:32 PM	PHP File	4 KB
config.php	10/3/2014 9:04 PM	PHP File	13 KB
constants.php	10/3/2014 7:37 AM	PHP File	2 KB
database.php	10/4/2014 6:50 AM	PHP File	4 KB
doctypes.php	10/3/2014 7:37 AM	PHP File	2 KB

```
| -------------------------------------------------------------------
----
|  Auto-load Packges
| -------------------------------------------------------------------
----
| Prototype:
|
|  $autoload['packages'] = array(APPPATH.'third
'/usr/local/shared');
|
*/

$autoload['packages'] = array('database');
```

CodeIgniter's Auto-load

- CodeIgniter comes with an "Auto-load" feature that permits libraries, helpers, and models to be initialized automatically every time the system runs.
- If you need certain resources globally throughout your application you should consider auto-loading them for convenience.

CodeIgniter's Auto-load

- The following items can be loaded automatically:
- Core classes found in the "libraries" folder
- Helper files found in the "helpers" folder
- Custom config files found in the "config" folder
- Language files found in the "system/language" folder
- Models found in the "models" folder

CodeIgniter's Auto-load

- To autoload resources, open the application/config/autoload.php file and add the item you want loaded to the autoload array. You'll find instructions in that file corresponding to each type of item.
- Note: Do not include the file extension (.php) when adding items to the autoload 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

CI database : 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.
- In some cases only one or two lines of code are necessary to perform a database action.

97

CI database : Active Record Class

- CodeIgniter provides a more simplified interface.
- Beyond simplicity, a major benefit to using the Active Record features is that it allows you to create database independent applications, since the query syntax is generated by each database adapter.

98

Create Controller to get data from Model:

- Controller : HelloWorld4_DB.php

```
<?php
    class HelloWorld4_DB extends CI_controller{

        public function index(){
            //default method
        $this->load->model('helloworld_model_DB');
            /*load the name of the model -
            helloworld_model_DB.php in model sub directory*/

      $data['records'] =  $this->helloworld_model_DB->getData();
        $this->load->view('myDBView',$data);
                /*load the name of the view
                myDBView.php in view sub directory*/
```

99

Active Record Class

- Selecting Data
  - $this->db->get();
    - runs the selection query and returns the result.
- Selecting Data
- The following functions allow you to build SQL SELECT statements.
- $this->db->get();
  - Runs the selection query and returns the result. Can be used by itself to retrieve all records from a table:
- $query = $this->db->get('hello_db ');
- // Produces: SELECT * FROM hello_db

100

Create Model to get data from Database:

- Model: helloworld_model_DB.php

```
<?php
class Helloworld_model_DB extends CI_Model {
 /*
    function Helloworld_model_DB()
    {
      // Call the Model constructor
        parent::Model();
    }
     */
    function getData()
        {
          //Query the data table table 'hello_db' as assign
          //the result to variable $query
          $query = $this->db->get('hello_db');
              // Note need to set '$active_record = TRUE;'
              // in database.php

         return $query->result();
        }
     // function returns the result from query as an array

}
?>
```

// Produces: SELECT * FROM hello_db

101

Controller Loads the passes the data returned from Model to View:

- Controller : HelloWorld4_DB.php

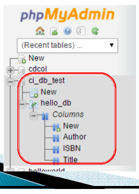
```
<?php
    class HelloWorld4_DB extends CI_controller{


        public function index(){
            //default method
        $this->load->model('helloworld_model_DB');
            /*load the name of the model -
                hello_model_DB.php in model sub directory*/

      $data['records'] =  $this->helloworld_model_DB->getData();

        $this->load->view('myDBView',$data);
                /*load the name of the view
                myDBView.php in view sub directory*/
```

102

View Displays the Records sent from Controller

- View : myDBView.php

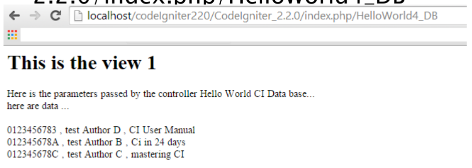
```
<h1> This is the view 1</h1>
Here is the parameters passed by the controller
     <?php
     echo "Hello World CI Data base... <br> here are data ...<br> <br>";
     foreach ( $records as $rec)
               echo  $rec->ISBN . " , " . $rec->Author . " , " . $rec->Title."<br>";

     ?>
```



103

Output: database access in CI

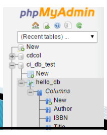
- http://localhost/codeIgniter220/CodeIgniter_2.2.0/index.php/HelloWorld4_DB

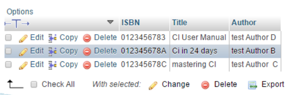


This is the view 1

Here is the parameters passed by the controller Hello World CI Data base...
here are data ...

```
0123456783 , test Author D , CI User Manual
012345678A , test Author B , Ci in 24 days
012345678C , test Author C , mastering CI
```





	ISBN	Title	Author
Edit Copy Delete	0123456783	CI User Manual	test Author D
Edit Copy Delete	012345678A	Ci in 24 days	test Author B
Edit Copy Delete	012345678C	mastering CI	test Author C

104

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

```
$where = "name='Joe' AND status='boss' OR status='active'";

$this->db->where($where);
```

105

Active Record Class

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

Writes a "SELECT MAX(field)" portion for your query

106

Active Record Class

Aggregate Functions : Writes a "SELECT MAX(field)" portion for your query. You can optionally include a second parameter to rename the resulting field

```
$this->db->select_max('age');
$query = $this->db->get('members');
// Produces: SELECT MAX(age) as age FROM members

$this->db->select_max('age', 'member_age');
$query = $this->db->get('members');
// Produces: SELECT MAX(age) as member_age FROM members
```

```
$this->db->select_avg('age');
$query = $this->db->get('members');
// Produces: SELECT AVG(age) as age FROM members
```

```
$this->db->select_min('age');
$query = $this->db->get('members');
// Produces: SELECT MIN(age) as age FROM members
```

107

Active Record Class

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

108

Active Record Class

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

```
$this->db->set('name', $name);
$this->db->insert('mytable');

// Produces: INSERT INTO mytable (name) VALUES ('{$name}')
```

```
$this->db->set('name', $name);
$this->db->set('title', $title);
$this->db->set('status', $status);
$this->db->insert('mytable');
```

109

Active Record Class

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

110

Active Record Class

- Deleting Data
  - $this->db->delete();
    - generates a delete SQL string and runs the query
    - The first parameter is the table name, the second is the where clause.

```
$this->db->delete('mytable', array('id' => $id));

// Produces:
// DELETE FROM mytable
// WHERE id = $id
```

111

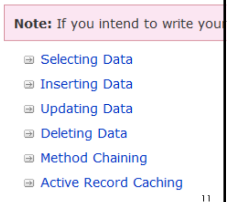
Active Record Class

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

112

CI database help: Active Record Class

- Refer to active record Class for database access
- https://ellislab.com/codeIgniter/user-guide/database/index.html
- http://goo.gl/h3ZMrV


Note: If you intend to write you

113

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

114

Generating Query Results

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

```
$query = $this->db->query("YOUR QUERY");

foreach ($query->result() as $row)
{
   echo $row->title;
   echo $row->name;
   echo $row->body;
}
```

```
$query = $this->db->query("YOUR QUERY");

if ($query->num_rows() > 0)
{
   foreach ($query->result() as $row)
   {
      echo $row->title;
      echo $row->name;
      echo $row->body;
   }
}
```

115

Generating Query Results

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

```
$query = $this->db->query("YOUR QUERY");

foreach ($query->result_array() as $row)
{
   echo $row['title'];
   echo $row['name'];
   echo $row['body'];
}
```

116

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)

```
$query = $this->db->query("YOUR QUERY");

if ($query->num_rows() > 0)
{
   $row = $query->row();

   echo $row->title;
   echo $row->name;
   echo $row->body;
}
```

117

Generating Query Results

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

```
$query = $this->db->query("YOUR QUERY");

if ($query->num_rows() > 0)
{
   $row = $query->row_array();

   echo $row['title'];
   echo $row['name'];
   echo $row['body'];
}
```

118

How to integrate Twitter Bootstrap 3 with PHP CodeIgniter Framework

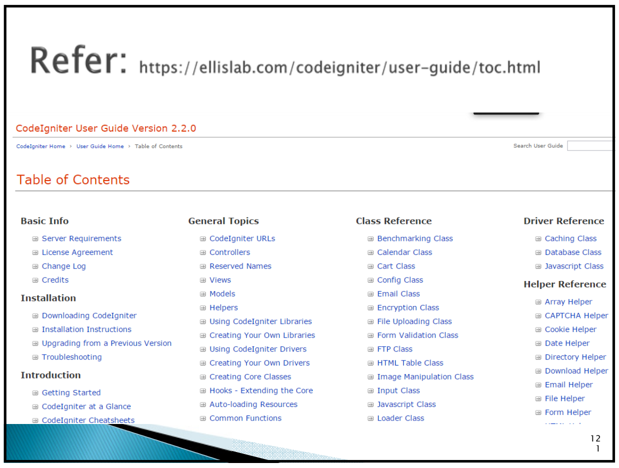
- http://www.kodingmadesimple.com/2014/06/integrate-twitter-bootstrap3-with-codeigniter-framework.html
- http://stackoverflow.com/questions/29794513/integrate-bootstrap-on-codeigniter
- http://stackoverflow.com/questions/20843265/adding-twitter-bootstrap-to-codeigniter

119

References

- CodeIgniter User Guide
  - https://ellislab.com/codeigniter/user-guide/
- Codeigniter – Getting Know Framework
  - http://goo.gl/Hsd9ta
- Everything You Need to Get Started With CodeIgniter
  - http://goo.gl/WJvCmt
- Codeigniter tutorial for beginners
  - http://goo.gl/ANwAFy
- PHP CodeIgniter Tutorial 2 – Models / Database http://goo.gl/alsiel

120



Refer: https://ellislab.com/codeigniter/user-guide/toc.html

CodeIgniter User Guide Version 2.2.0

CodeIgniter Home › User Guide Home › Table of Contents

Search User Guide

Table of Contents

12
1