CodeIgniter®

Modern Programming paradigms in Web Engineering

Lecture 01

Introduction to CodeIgniter

By: Ashuqullah Alizai
Herat University
Computer Science Faculty
0795642400
Alizai.csf@hotmail.com

Motivation

- If you need to produce results, have better and more maintainable code, and you enjoy programming, then you should try using CodeIgniter.
- CI is
  - free,
  - lightweight,
  - and simple to install,
  - and it really makes your life much easier

What can CodeIgniter do for you?

- If you are already writing code in PHP, CI will help you to do it in a better and easier way.
- It will cut down the amount of code you actually type.
- Your scripts will be easier to read and update
- It improving team work and maintainability.
- It will help you to give large websites a coherent structure.
- It will discipline your code and make it more robust, in some cases even without your knowing it.

NOTE: That's quite a big claim. You have already spent some time learning PHP, HTML, CSS, a database, and so on. You need basic, not necessarily expert knowledge of PHP to benefit from CI.

CI Is Not Convenient

- CI is not For those who
  - don't have a minimum knowledge of PHP and HTML.
  - like to write all of their code. There are people who prefer to write their code instead of using already built solutions.
  - don't like PHP;
  - And definitely CI is not for those whom don't like to finish your projects on time, in a well-structured fashion, and without having to redo the same things again and again.

Benefits of CI: Save time

- CI doesn't take long to learn, and it quickly pays for your effort in the time saved later.
- The less you type, the fewer mistakes you make, and the less time you spend debugging your code.

Let's take two examples

```
$connection = mysql_connect("localhost","fred","12345");
mysql_select_db("websites", $connection);
$result = mysql_query ("SELECT * FROM sites", $connection);
while ($row = mysql_fetch_array($result, MYSQL_NUM)){
foreach ($row as $attribute)
print "{$attribute[1]}"; }
```

Now see how a CI function would handle a similar query:

```
$this->load->database('websites');
$query = $this->db->get('sites');
foreach ($query->result() as $row){
print $row->url;  }
```

Compare the character count—244 for the traditional syntax and 112 for CI. Another thing that you have to take into account when using Active Record is that you can change your database from MySQL to Postgres (or any other that is supported by CI) and you won't need to change your queries—a very helpful thing.

Benefits of CI: Save time

- **Second example: writing a data entry form in HTML,**
- **you want a drop-down query box. Let's say this drop-down query box shows three options and allows the user to select one of them.**

```
<select name="url">
<option value="this">www.this.com</option>
<option value="that">www.that.com</option>
<option value="theother" selected>www.theother.com</option>
</select>
```

- **CI's version is both shorter and, because it works from an array, more adapted to PHP processing:**

```
$urlarray = array(
'this' => 'www.this.com',
'that' => 'www.that.com',
'theother' => 'www.theother.com',
);
$variable = form_dropdown('url', $urlarray, 'this');
```

In HTML, you need to type 154 characters, and 128 in CI. Note how easily we can define the "selected" element of the drop-down menu, putting it in the third parameter. This thing alone will save us a lot of time.

Make your site more robust

- Although you don't need to write much code, CI provides a lot of the standard functionality and better security, and it remembers all those excitement.
- It keeps track of things you may have forgotten about (those little touches that distinguish amateur sites from professional ones).

Keep your links up-to-date automatically

- Suppose you've just written a menu page, with lot of hyperlinks to other pages in your site.
- They are all in the traditional HTML format as shown:

```
<a href="http://www.mysite.com/index.php/start/hello/fred ">say hello to Fred</a>
```

- Then, you decide to move the site to another URL.
- CI gives you a simple function to write hyperlinks like this:

```
echo anchor('start/hello/fred', 'Say hello to Fred');
```

- CI also encourages you to put the URL of your site in a configuration file that the rest of your site can access
- CI's anchor function that we've used here, automatically refers to that configuration file. So, when you come to move your site, you only need to change that one entry in the configuration file, and all your hyperlinks are updated automatically.

Preventing database SQL injection attacks and form prepping

- There are certain limitations of HTML and databases
- Data containing symbols such as apostrophes and quotation marks may not be saved correctly,
- your database may be open to malicious attacks.
- For example:
- SELECT id, name FROM users WHERE user = '{$user}' AND password ='{$password}';
- **Consider that the variables have the following values:**
- $user = "Fred"; $password = "1234";
- **Now our query would translate to:**
- SELECT id, name FROM users WHERE user = 'Fred' AND password = '1234';
- **This query will return a good result, but, what if our variables were:**
- $user = "Fred"; $password = "1234' OR '1' = '1";
- **Now our query would produce:**
- SELECT id, name FROM users WHERE user = 'Fred' AND password = '1234' OR '1' = '1';
- **This time the variable's data contains a new "where" clause with a condition that is always true.**

Preventing database SQL injection attacks and form prepping

- Use of some character such as " ' " cause problems and make our query behave in a way we don't want, and give bad results.
- Example: $user = "Fred"; $password = "12xWgBq'wS";
- Our password variable contains a password that looks quite secure, but will produce a problem in our query:
- SELECT id, name FROM users WHERE user = 'Fred' AND password ='12xWgBq'wS';
- The data will cut the query, producing some errors when executed. What can we do to prevent these problems? Well the answer to this is to prepare or "prep" our data in our data entry form, before it is submitted to the database.
- CI's form helper does this, automatically. So, when you create an input box by typing:
- echo form_input('user', 'Fred');

Protect your site from XSS attacks

- As stated on Wikipedia (http://en.wikipedia.org/wiki/Cross-site_scripting), XSS (cross site scripting) is a kind of vulnerability that allows some unwanted code to be executed in our application, phising attacks, data theft, and more.
- In order to avoid this you should validate your data.
- CodeIgniter helps you to do so, in all your applications if you set global XSS filter to true in your configuration file, or whenever you need it:
- $data = $this->input->xss_clean($data);
- You can even use it to check potential XSS attacks within image files:
- $this->input->xss_clean($file, TRUE);
- The second parameter tells CI that it is an image that needs validation.

Make your code bolder

- CI also makes it easy to do things you might not have tried before.
- Of course, PHP users can always integrate libraries from **PHP Extension and Application Repository (PEAR) and other sources.**
- **They aren't always easy to integrate, or use,** and their syntax and standards differ greatly.
- CI has a common set of standards, and once you've mastered its syntax, all its parts work together without complication.
- All its code is well-written and reliable, and is tested by its user community.
- It puts much more sophistication in your hands.

Send email attachments without hassles

- Sending emails is a complex business. CI's code for doing it looks easy to follow:

```
$this->load->library('email');
$this->email->from('your@your-site.com', 'Your Name');
$this->email->subject('Email Test');
$this->email->message('Testing the email class.');
$this->email->send();
```

- There are a number of issues involved in sending emails—setting word wrapping and escaping it (so that long URLs don't get wrapped and broken up).
- For example, when sending attachments, the standard PHP functions can get quite complex. As a result many code writers are tempted to avoid using these functions if possible.
- CI's email class makes it simple to send an attachment. You write:
- $this->email->attach('/path/to/photo1.jpg');
- CI does the rest, working behind the scenes. There is a function that sorts out MIME types for nearly a hundred type of attachments. So it knows that your photo, photo1.jpg, is an image/jpeg MIME type. It remembers to generate boundary delimiters in the right places around your attachments. It takes care of wrapping your text, and it allows you to easily mark chunks of text you don't want wrapped.

Save bandwidth by zipping files that users need to download

- To save bandwidth, it's a fairly common practice to compress or zip files before you download them.
- That's something you might have never done, and you wouldn't know how to go about it.
- CI has a nice facility that allows you to produce ZIP files with four lines of code:

```
$name = 'mydata1.txt';
$data = 'the contents of my file……';
$this->zip->add_data($name, $data);
$this->zip->archive('c:/my_backup.zip');
```

What CI doesn't do

- There are some things that CI doesn't do.
- CI was intended to be a small and **lightweight framework**. whereas the Zend framework is 10 MB. It won't answer all the problems you will have.
- But it does:
- Make it easy and quick to program in PHP
- Structure your site and help you through the architectural decisions
- As a result of being lightweight, it does not have as many features as some of its rivals. Other frameworks such as Rails, CakePHP, or Symfony have **scaffolding and generators**.
- **These tools automatically write certain basic scripts for you.**
- Once you have set up a database, Rails creates out-of-the-box web pages to do basic **Create, Read, Update, and Delete (CRUD) operations on the database tables. In** addition, Rails allows you to write generators—pieces of code that automatically write other basic scripts. The Rails community has created quite a lot of these, so you can automatically generate scripts that do all sorts of clever things.

what is Framework

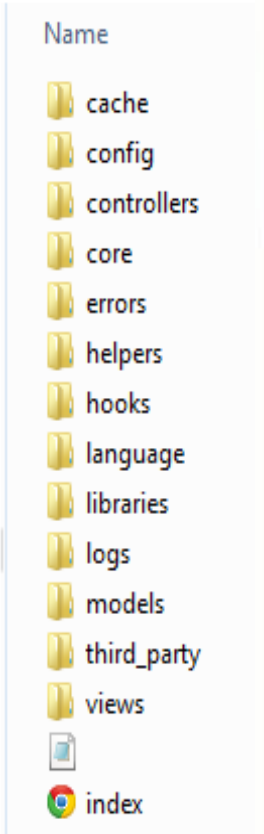
- Shortly after programming was invented, it was noticed that it involved many repetitive operations. So, you only had to write certain chunks once, and could then reuse them. PHP programmers are used to writing separate chunks of code in functions, and then storing those functions in include files.
- At one level, a framework is just that—lot of chunks of code stored in separate files, which simplify the coding of repetitive operations.

What is CodeIgniter?

- Codeigniter is a powerful HP framework with extensive list of libraries and helpers.

CodeIgniter Vs FlatPHP

- **Ready to Go & Almost Zero Configuration**
- If you are going to start a new project with custom PHP then you obviously need to set and collect couple of files to get this start. You must need to build database & other configuration files.
- **Custom PHP**
- connection.php
  - config.php
  - Database.Class.php
  - login.php
  - logout.php
- **Codeigniter**
- In Codeigniter we dont need to worry about such classes and configurations. We will only need to download latest version from there website and after unzipping and setting up few things you are ready to go. The whole zip comes with 3 folders usually,
- - application
  - system
  - user_guide



CodeIgniter Vs FlatPHP

- **MVC Structure**
  - As we know, Codeigniter is a model/View/Controller (MVC) based framework so this is best structure to build any application.
  - Each request comes to controller and performing any database interaction though Model, all out put goes to View for the end result.
  - Where as, in custom PHP each single page serves as MVC where we can see so much complexity.

CodeIgniter Vs FlatPHP

- **No PHP Version Conflicts**
- We often see that whenever new PHP version releases, many new and useful functions are offered with it. Also few of them take places of old functions with new set of features.
- In custom PHP development, everything is open and directly used. So its hard to go into hundred of files and change all those old functions which are now depreciated.
- But in using a framework we dont need to worry about version conflicts. Because every functions comes in wrapper and a developer should not be worry about fixing such issue but a newer version of framework will be one step solution for this.

CodeIgniter Vs FlatPHP

- **Less Chances to Mess The Code**
  - If you are working in a team on same project there is less chances that anyone of your team would mess up the code as you all are following a predefined way to perform the tasks.
  - The separation of code has been very easy in Codeigniter.
  - You can break down a module into a number of controllers, models and views which will make it easy to work on same project by a team simultaneously.

CodeIgniter Vs FlatPHP

- **Less Duplication of Code**
- Based on the mode writing of function it is possible in flat php to have two or more function to do same task.
- But in Codeigniter, no one writes their function on to page. Each function goes into Models, which is accessible within site.
- Codeigniter develops a habit to write each code to its proper place. This thing reduces the chances of duplication.

CodeIgniter Vs FlatPHP

- **Clear & Thorough Documentation**
- **Most Active Online Community**
- Codeigniter has a very effective and active community where your questions has been replied within minutes.
- **Built-in Libraries and Helpers**
- The one of great benefit of using Codeigniter is its rich set of libraries and helpers which comes within download.

CodeIgniter Vs FlatPHP

- **Easy Error Handling**
  - Error handling has never been so easy and transparent. In custom development, its hard to keep track of bugs and find them.
- **Security and Encryption**
  - Every one wants a secure and hack free site for them. They need their site secure from CSRF/XSRF Attacks and SQL injection etc. In using framework the security of site can be achieved by using built-in classes and libraries.
- **Cache Class**
  - Mostly websites don't have cache feature in it. Specially if you are doing custom PHP development, most of the developers never mentions cache benefit or don't want to mention as this is a thing they don't wanna trouble with them.
- **Easy Template Solution of Codeigniter**
  - Most of you know about template system and how they can be beneficial for our site. Template systems separates the PHP and HTML so that you can easily change layouts with taking care of actual PHP code

Comparing CI to other open source solutions (CakePHP and Joomla!)

- Using CakePHP requires you to adhere to some strict conventions (those can be changed but the out-of-the-box software comes with them). Most of the time they are naming conventions, but there are database conventions also.
- After some time, because of curiosity, in spite of incredible lack of learning time (some nearby shouting boss also helped), continuing research of PHP frameworks found CodeIgniter.
- The most convincing thing was the documentation (CakePHP wasn't as well documented as it is today).
- CI has less conventions, and minimal configuration, also you can forget about most of them and work as you have always done. You don't want to use models, and you don't have to (though it is recommended). That said, one of the strengths of CI is just that—download it and start programming.

Comparing CI to other open source solutions (CakePHP and Joomla!)

- Joomla! is not exactly a framework, it helps you build websites faster. Also in the latest versions it has turned into the **MVC (Model-View-Controller) pattern. The** good part of Joomla! is that it is a CMS (Content Management System), but I think it is also its bad part, as sometimes you just don't need so much as Joomla! has to offer.
- Of course you can develop your own solution over Joomla!, but then what's the point of using Joomla!, and not using what it has to offer?
- CI, usually, is the most well-balanced of the three, but let's see how we can choose one of them.

What to choose

- With all those options out there, why should you go with CI?
- The answer—it's not that easy and involves, in most cases, personal preference. There will be projects that will be better suited to one of the other two options. Let's see some examples that will guide us.

When to choose Joomla!

- If we need a CMS (it may sound redundant, but...).
- If there is already a component, module, or some other functionality that we need, then there is something built that can help us carry out the project in no time.
- If the client asks you to use it. Joomla! is a well known software, and sometimes your client can just ask you to use it.

When to choose CakePHP

- If there are lot of relations between database tables. Cake's Active Record capabilities are slightly more powerful than CI's.
- If you need to build some admin zone quickly. Cake's bake script can read your database and build some CRUD (Create, Read, Update, Delete) pages for your tables.

When to choose CodeIgniter

- If the project doesn't have or doesn't need a very rigid structure.
- CI is good for working with legacy code.
- If you need to start programming right away, without having to learn a lot of conventions.
- If you need some software that helps you, with the confidence that it will help you in the way you really need, and learn the way something needs to be done.
- If you know PHP, you can use CI.
- The client needs some solution built specifically for him/her, and not an adapted solution (that would be in case of some Joomla! components/modules).
- Most of the time we will be using CI because it is very flexible. If you like programming CI will help you in doing it.
- If you are new to frameworks it is recommended that you get CI, maybe later you use another one (probably both of them), but CI will always have a place in your heart.

This Is The End For This Lecture