Trivial CI to Basic CI

tutorial #ci-basic02

James L. Parry
B.C. Institute of Technology

Tutorial Goals

This tutorial will walk you through converting a trivial webapp (the result of completing tutorial "basic01" into a basic CodeIgniter webapp.

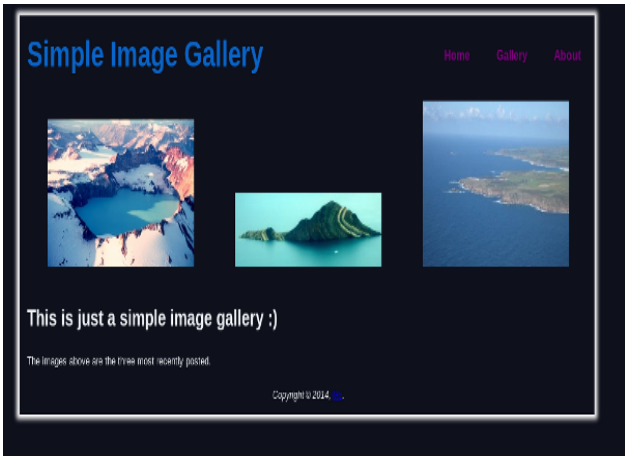
The approach taken is not the only one possible, or even the best one, but it suits "baby steps".

Suggestion: you may want to skim the slideshow first, before working your way through it.

What Do You Have?

The trivial website looks the same as the original, but it has been partially transformed.

In this tutorial, we will take the website to the next level.



Build a Master View Template

Eliminate duplicate code in our views by extracting the common opening and closing HTML into views/template.php.

Let's do that, and add a placeholder, "{content}", which will get substituted appropriately in our next tutorial step. Add another placeholder, "{pagetitle}", for the page title.

I would do this by copying the about view, renaming it to "template.php", and then zapping its middle.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>{pagetitle}</title>
        <meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8"/>
        <link rel="stylesheet" type="text/css" media="all" href="css/reset.css" />
        <link rel="stylesheet" type="text/css" media="all" href="css/text.css" />
        <link rel="stylesheet" type="text/css" media="all" href="css/style.css" />
        <link rel="stylesheet" type="text/css" media="all" href="css/lightbox.css" />
    </head>
    <body>
        <div id="wrapper">
            <div id="header">
                <span class="myhead">Simple Image Gallery</span>
                <span class="mynav">
                    <ul>
                        <li><a href="/">Home</a></li>
                        <li><a href="/gallery">Gallery</a></li>
                        <li><a href="/about">About</a></li>
                    </ul>
                </span>
            </div>
            <div class="alone"></div>
            <div id="content">
                {content}
            </div>
            <div id="footer" class="span12">
                Copyright &copy; 2014,  <a href="mailto:someone@somewhere.com">Me</a>.
            </div>
        </div>
        <script type="text/javascript" src="js/jquery-1.11.0.min.js"></script>
        <script type="text/javascript" src="js/lightbox.min.js"></script>
    </body>
</html>
```

Simplify Our Views

We can now eliminate the common opening and closing HTML from our three original views, leaving just the "meat".

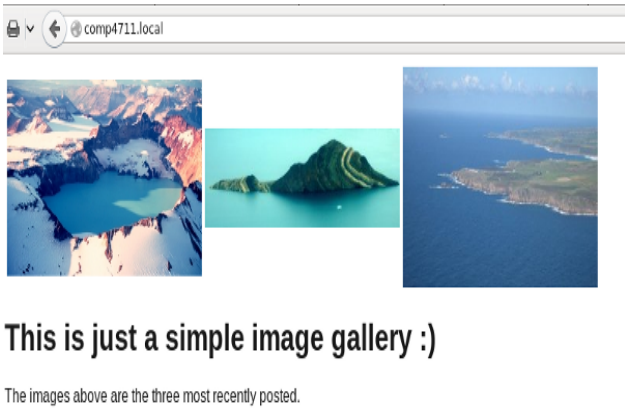
Our revised views/welcome.php is shown to the right.

```
<table cols="3" class="gallery">
    <tr>
        <td class="oneimage">
            <a href="./data/Katmai_Crater_1980.jpg" data-lightbox="gallery"
        </td>
        <td class="oneimage">
            <a href="./data/Caledonian_orogeny_fold_in_King_Oscar_Fjord.jpg"
        </td>
        <td class="oneimage">
            <a href="./data/Lands_End_Cape_Cornwall.jpg" data-lightbox="gall
        </td>
    </tr>
</table>
<h1>This is just a simple image gallery :)</h1>
<p>The images above are the three most recently posted.</p>
```

Is This Better?

The site still "works", but looks funny without the proper styling :-/

We will fix that next!



Build a base controller

We will make a base controller to pull the view pieces together consistently, using our template.

This will be similar to the one referenced in lesson 2, from the contacts sample webapp. Copy yours from the contacts project to core/MY_Controller.php inside your lab 2 project.

Fix the menu to suit our app, and change the ['title'] reference to ['pagetitle'], to match what we already setup in our template.

Note: Your webapp will not run yet - more to fix!

```php
<?php

/**
 * core/MY_Controller.php
 *
 * Default application controller
 */
class Application extends CI_Controller {

    protected $data = array();      // parameters for view components
    protected $id;    // identifier for our content
    protected $choices = array(// our menu navbar
        'Home' => '/', 'Gallery' => '/gallery', 'About' => '/ about'
    );

    /**
     * Constructor.
     * Establish view parameters & load common helpers
     */
    function __construct() {
        parent::__construct();
        $this->data = array();
        $this->data['pagetitle'] = 'Sample Image Gallery';
    }

    /**
     * Render this page
     */
    function render() {
        $this->data['menubar'] = build_menu_bar($this->choices);
        $this->data['content'] = $this->parser->parse($this->data['pagebody'], $this->
        $this->data['data'] = &$this->data;
        $this->parser->parse('_template', $this->data);
    }
}

/* End of file MY_Controller.php */
/* Location: application/core/MY_Controller.php */
```

Apply our base controller

Modify our three controllers to extend our base controller, Application, instead of CI_Controller.

In these controllers, instead of $this->load->view('x') we should set a view parameter instead, and invoke the render method of the base controller.

$this->data['pagebody'] = 'x'; $this->render();

```
class Welcome extends Application {

    /**
     * Index Page for this controller.
     */
    public function index()
    {
//      $this->load->view('welcome');
        $this->data['pagebody'] = 'welcome';
        $this->render();
    }

}
```

What Do You Think Now?

We also need to copy the common_helper from the contacts starter, so we can use its build_menu function. Copy application/helpers /common_helper.php from the example-contacts folder (last week) into the same place inside your current project.

We should autoload the common and url helpers, and the parser library, since they are used in Application.

Looking better, although the image references are still hard-coded. We'll fix that next.

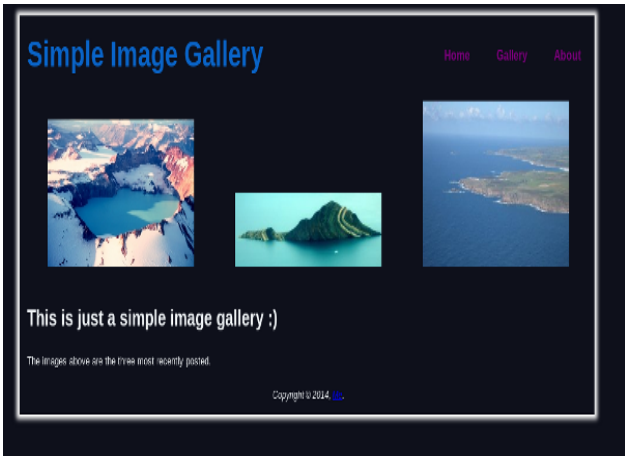

```
$autoload['helper'] = array('common', 'url');
```

Are We There Yet?

The site is looking the same as when we started, but it is constructed very differently internally.

The next tutorial will add a database-driven model to make the content dynamic.

This would be an appropriate point to push your local changes to your github repo.



Congratulations!

You have completed tutorial #ci-basic02: Trivial CI to Basic CI

If you would take a minute to provide some feedback, we would appreciate it!

The next activity in sequence is: ci-basic03 Basic CI to Good CI

You can use your browser's back button to return to the page you were on before starting this activity, or you can jump directly to the course homepage, organizer, or reference page.