Basic CI to Good CI

tutorial #ci-basic03

James L. Parry
B.C. Institute of Technology

Tutorial Goals

This tutorial will walk you through converting a basic CodeIgniter into a "good" one.

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.

Add a Model?

We already have a SQL script in the data/setup folder of our starter. How convenient! It's like someone planned to use a database eventually :-/

The first step is to create a database, and import the data/setup/images.sql script to create a table we can then use. This can be done using phpMyAdmin, or by using the database browser inside NetBeans.

In config/autoload.php, we should add the database library, and in config/database.php you should adjust your settings for your system.

```php
$active_group = 'default';
$query_builder = TRUE;

$db['default'] = array(
        'dsn'   => '',
        'hostname' => 'localhost',
        'username' => 'root',
        'password' => '',
        'database' => 'picassos',
        'dbdriver' => 'mysqli',
        'dbprefix' => '',
        'pconnect' => TRUE,
        'db_debug' => TRUE,
        'cache_on' => FALSE,
        'cachedir' => '',
        'char_set' => 'utf8',
        'dbcollat' => 'utf8_general_ci',
        'swap_pre' => '',
        'autoinit' => TRUE,
        'encrypt' => FALSE,
        'compress' => FALSE,
        'stricton' => FALSE,
        'failover' => array(),
        'save_queries' => TRUE
);

$autoload['libraries'] = array('database', 'parser');
```

Now We Can Add a Model

The contacts example had a base model, but that feels like overkill for this simple app. We only need two methods in our model: get all images, ordered properly, and get the three newest images.

Of course, if you want a complete MY_Model in your starter app, for convenience, feel free to use that here.

We just need a simple model, which I suggest calling "Images". It is used on most of our pages, hence is safe to autoload in our configuration.

```php
<?php

class Images extends CI_Model {

    // constructor (a good practice)
    function __construct()
    {
        parent::__construct();
    }

}
```

```php
$autoload['model'] = array('images');
```

Revisit the gallery page

Instead of hard-coding image names like the original static site, we can generate the HTML for those programmatically, in our Gallery controller.

We will need to add a suitable method to our Images model to provide all of the image data. Let's call it *all()*, and have it return an array of images, in reverse posting order (newest ones first).

```php
<?php

class Images extends CI_Model {

    // constructor (a good practice)
    function __construct()
    {
        parent::__construct();
    }

    // return all images, descending order by post date
    function all()
    {
        $this->db->order_by("id", "desc");
        $query = $this->db->get('images');
        return $query->result_array();
    }

}
```

Let's Handle a Single Image

We can make a view fragment to handle the HTML for a single image.

Copy the code for one image table cell, and tailor it to use the appropriate field names from our database as substitution fields.

Save this as views/_cell.php. The underscore at the front of the name reinforces that this is for internal use.

Notice that there is no PHP in the view fragment.

```php
<?php
/*
The original HTML:
    <a href="./data/Katmai_Crater_1980.jpg" data-lightbox="gallery"
        data-title="Katmai Crater - Mount Katmai, Alaska ... Posted 2014.05.05 by donald, in landscape">
            <img src="./data/thumb/Katmai_Crater_1980.jpg"></a>

Converted to be a view fragment template:
*/
?>
<a href="./data/{filename}" data-lightbox="gallery"
   data-title="{title} ... Posted {uploaded} by {uploader}, in {category}">
    <img src="./data/thumb/{filename}"/>
</a>
```

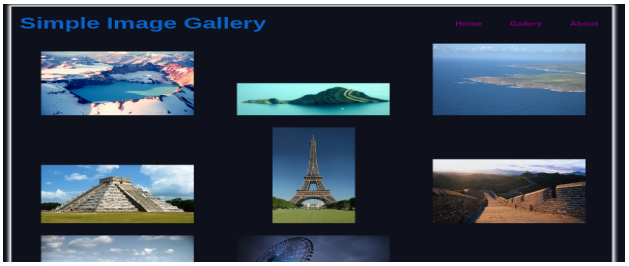
Let's Generate Our Gallery

We can use the HTML table class to generate the HTML for our table now. This logic would go inside our Gallery controller.

```
class Gallery extends Application {

    /**
     * Index Page for this controller.
     */
    public function index()
    {
        // get all the images from our model
        $pix = $this->images->all();

        // build an array of formatted cells for them
        foreach ($pix as $picture)
            $cells[] = $this->parser->parse('_cell', (array) $picture, true);

        // prime the table class
        $this->load->library('table');
        $parms = array(
            'table_open' => '<table class="gallery">',
            'cell_start' => '<td class="oneimage">',
            'cell_alt_start' => '<td class="oneimage">'
        );
        $this->table->set_template($parms);

        // finally! generate the table
        $rows = $this->table->make_columns($cells, 3);
        $this->data['thetable'] = $this->table->generate($rows);

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

}
```

What Does It Look Like?

Can you see a difference?



Revisit the Homepage

The homepage shows the 3 most recent images posted. This is very similar to the gallery, and we can leverage what we did for it.

Add a newest() method to our model, which just returns the 3 newest images.

```
class Images extends CI_Model {

    // constructor (a good practice)
    function __construct()
    {
        parent::__construct();
    }

    // return all images, descending order by post date
    function all()
    {
        $this->db->order_by("id", "desc");
        $query = $this->db->get('images');
        return $query->result_array();
    }

    // return just the 3 newest images.
    function newest()
    {
        $this->db->order_by("id", "desc");
        $this->db->limit(3);
        $query = $this->db->get('images');
        return $query->result_array();
    }

}
```

Fix the Homepage View

Adjust the "welcome" view, the same way we changed the "gallery" view.

```
{thetable}
<h1>This is just a simple image gallery :)</h1>
<p>The images above are the three most recently posted.</p>
```

Fix the Homepage Controller

Finally, update the "Welcome" controller, similarly to the Gallery one.

```
class Welcome extends Application {

    /**
     * Index Page for this controller.
     */
    public function index()
    {
        // get the newest images from our model
        $pix = $this->images->newest();

        // build an array of formatted cells for them
        foreach ($pix as $picture)
            $cells[] = $this->parser->parse('_cell', (array) $picture, true);

        // prime the table class
        $this->load->library('table');
        $parms = array(
            'table_open' => '<table class="gallery">',
            'cell_start' => '<td class="oneimage">',
            'cell_alt_start' => '<td class="oneimage">'
        );
        $this->table->set_template($parms);

        // finally! generate the table
        $rows = $this->table->make_columns($cells, 3);
        $this->data['thetable'] = $this->table->generate($rows);

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

}
```

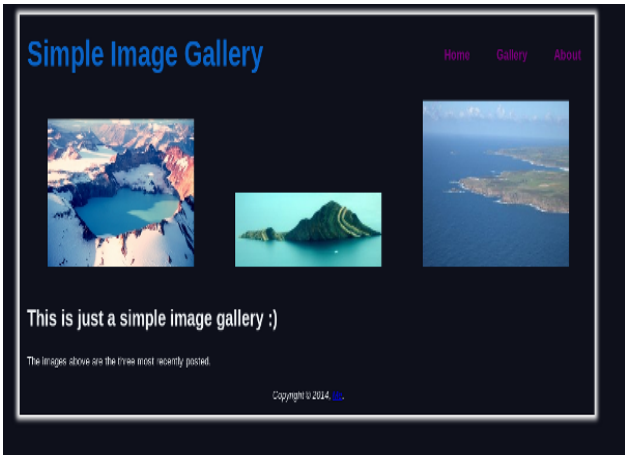
Are We There Yet?

We are indeed there:)

We have successfully converted a static website into a CodeIgniter webapp.

There are many efficiencies that could be achieved with re-factoring, but the idea and the process should be clear.

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



Congratulations!

You have completed tutorial #ci-basic03: Basic CI to Good CI

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

The next activity in sequence is: There is nothing further in this course

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.