

Working With Addins

tutorial #ci-normal05

James L. Parry
B.C. Institute of Technology

Tutorial Goals

This tutorial is meant to give you some practice with extending a CodeIgniter webapp in some standard ways.

I have prepared a [starter webapp](#) – based on the "quotes" webapp you saw and fixed earlier. This webapp is complete, showing the most recent quote on the homepage, and then a gallery when you drill down, but it is painfully boring. We will add some ratings to the individual quote presentations, and demonstrate one use of a "hook".

The End Result

This is what the end result should look like, **after** fixing!



Database Setup

You will need to setup your webapp's database, using the supplied script, `quotes-setup.sql`. Please name the database "quotes", so that your webapp will work without additional setup when I run it on my system.

As part of the tutorial, you will add a few columns to the table, for ratings.

If you need different database configuration settings for your local system, remember to put those inside `application/config/development/database.php`!

The Database

The database provided contains a single table, with a single field as the primary key. The model provided, `Quotes`, extends `MY_Model`, inheriting all of the CRUD methods we need.

The quotes table has four columns:

- `id` is a unique numeric identifier for a quote; it must be positive
- `who` is the name of the person quoted; it cannot be blank, and has a maximum length of 64
- `mug` is the filename of the author's picture; images are stored in the `data` folder off of the document root; filenames have a maximum length of 64
- `text` is the actual quoted text, of arbitrary length

What Needs Doing?

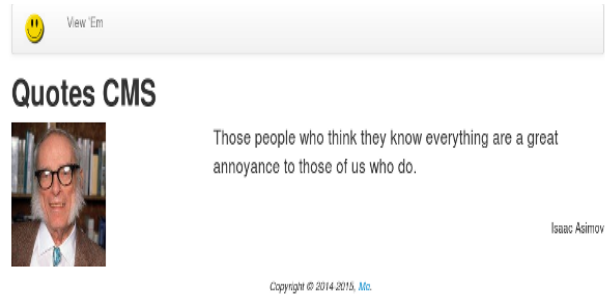
1. [Enable the supplied bolding hook](#)
2. [Dive into the Caboose library](#)
3. [Incorporate a rating widget](#)

The Starting Point

The webapp is functional when you start, with the homepage shown to the right.

It has a simple menubar on the top, with the smiley face linked back to the homepage, and the "View 'Em" menu link taking you to the "viewer" page.

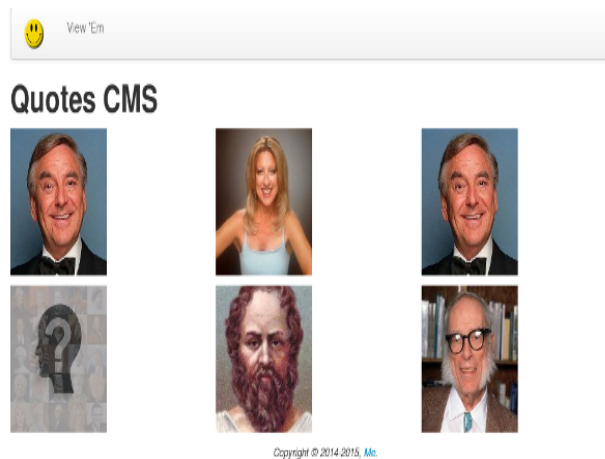
The quote shown is the most recently added, i.e. the one with the highest ID.



And Beyond the Homepage?

The "viewer" presents a table of the quotes on file, showing the author's mugshots. This is ordered from left-to-right and top-to-bottom by ID#.

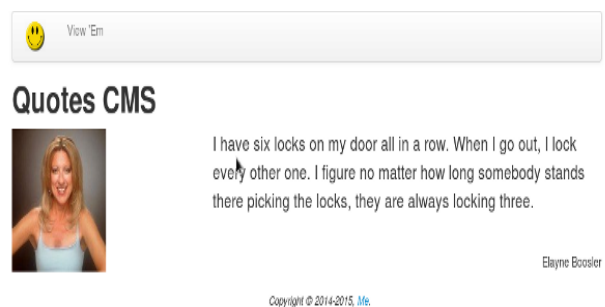
Each image is itself a link to the page that presents just that one quote, implemented by the quote method inside the Viewer controller, and using the justone view that you saw in lab 3.



And Then There Be Quotes

The single page presentation of a quote shows all of the quote's data fields, arranged somewhat pleasingly though neutral.

The justone view uses the template parser to substitute actual data values.



1. ENABLE THE SUPPLIED BOLDING HOOK

We shall start gently, with a "hook" for your webapp, to bold capitalized words in the quoted text.

The "hook" itself is found in `application/hooks/Hooks.php`. Check it out ... I'll wait.

The hook's configuration is in `application/config/hooks.php`. Go ahead - check that one out too.

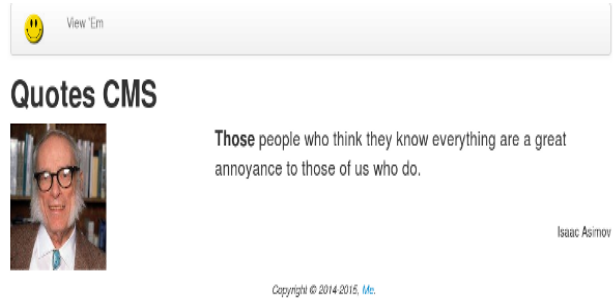
So, why isn't it working already?

One Teeny Step Needed

We need to enable hooks if ours is to do anything.

In `application/config/config.php` change the value of the 'enable_hooks' setting, and you should see a change in the homepage, shown right - the capitalized words are bolded as expected.

```
$config['enable_hooks'] = TRUE;
```



2. DIVE INTO THE CABOOSE LIBRARY

The **Caboose** library that I mentioned in class is an addin for CodeIgniter, to manage CSS and Javascript dependencies, including those for widgets for such frameworks.

Caboose is not the only way to do this, but it presents a good example of an addin library.

I plan to spin it off into a separate package, but after the school term, when I have a bit more time :)

The Caboose pieces are already in the starter project, with provision for a couple of widgets:

```
/libraries/Caboose.php
/views/_components
    confirm.php, lightbox.php
/assets
    /css/
        jquery-lightbox-0.5.css
    /js/
        jquery.lightbox-0.5.min.js
```

Enable Caboose in our webapp

We need to load the Caboose library in order to use it. Let's autoload it, in `application/config/autoload.php`.

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

Nothing looks different, but nothing broke :)



Extract Caboose output

We want to set some view parameters from our Caboose object, inside `core/MY_Controller`, as discussed in the lesson.

```
/**
 * Render this page
 */
function render() {
    $this->data['menubar'] = $this->parser->parse('_menubar', $this->config);
    $this->data['content'] = $this->parser->parse($this->data['pagebody'], !

    // convert Caboose output into view parameters
    $this->data['caboose_styles'] = $this->caboose->styles();
    $this->data['caboose_scripts'] = $this->caboose->scripts();
    $this->data['caboose_trailings'] = $this->caboose->trailings();
}
```

Incorporate Caboose in our view template

And we want to include the Caboose output at the appropriate points in our `views/_template.php`, using substitution parameters, again as discussed in the lesson.

We don't need the original link and script elements in the template, as those now come from Caboose.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>{title}</title>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8"/>
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
{caboose_styles}
<link rel="stylesheet" type="text/css" href="/assets/css/style.css"/>
</head>
<body>
<div class="container">
<div class="navbar">
<div class="navbar-inner">
<a class="brand" href="/"></a>
{menubar} </div>
</div>
<div id="content">
<h1>{title}</h1>
{content}
</div>
<div id="footer" class="span12">
Copyright &copy; 2014-2015, <a href="mailto:someone@somewhere.com">Me</a>.
</div>
</div>
{caboose_scripts}
{caboose_trailings}
</body>
</html>
```

We Have Caboose

At this point, we have enabled the Caboose, without making any requests of it.

We should also not have broken anything in doing so. Our webapp should look the same as it did a short while ago...



Quotes CMS



Those people who think they know everything are a great annoyance to those of us who do.

Isaac Asimov

Copyright © 2014-2015, Me.

3. ADD A VIEWER RATING WIDGET

This is a fun little idea, and an excuse to explore the **Caboose** widget manager mentioned in class. The idea is to add a rating star widget somewhere in the view fragment used to present a quote, and to track the votes as part of our webapp.

A rating widget looks something like:



1 star for normal, 5 for outrageous.

Database Mods for Voting

We need to add two fields to our quotes table:

- **vote_total** - an integer to hold the total of all votes
- **vote_count** - an integer to hold the number of votes cast

I suggest using phpMyAdmin to do this, resulting in the table structure shown right.

Server: localhost » Database: quotes » Table: quotes

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	id	int(11)			No	None	AUTO_INCREMENT
2	who	varchar(64)	utf8_general_ci		No	None	
3	mug	varchar(64)	utf8_general_ci		No	None	
4	what	text	utf8_general_ci		No	None	
5	vote_total	int(11)			No	None	
6	vote_count	int(11)			No	None	

Get a Rating Widget

A rating widget that I have used is [jRating](#). Its original author has abandoned it, so it is now an orphan project and you might be advised to find an alternative to use in your webapp.

The jRating link above leads to the documentation for the widget, which is pretty easy to follow for our purposes. The Download link lets you download a zip of the project. Do so!

jQuery Script.net (jQuery Slider) Search

Home Latest jQuery Plugins Most Popular Plugins Recommended Plugins Blog Submit Plugins & News

JQUERY PLUGINS » JQUERY OTHER PLUGINS » JRATING

Flexible Star Rating System with jQuery - jRating

[Demo](#) [Download](#)

Screen / Back screen when open:

File Size: 46.6 KB
Views Total: 01/29/2013 07:39:42 UTC
Last Update: 01/29/2013 07:39:33 UTC
Publish Date: 01/29/2013 07:39:33 UTC
Official Website: Go to website
License: MIT

More in this category...
View our Recommended Plugins
TOP 100 jQuery Plugins 2015

Categories:

- All jQuery Plugins
- Animations (442)
- CSS3 & HTML5 (28)
- Gallery (193)
- Lightbox (146)
- Menus & Nav (442)
- Others (988)
- Slider (420)
- Social Media (115)
- Text (208)
- Tooltip (74)
- Accordion (82)
- Chart & Graph (79)
- Form (518)
- Layout (220)
- Loading (122)
- Mobile (77)
- Rotator (48)
- Slideshow (219)
- Table (112)
- Time & Clock (200)
- Zoom (91)

You Might Like:

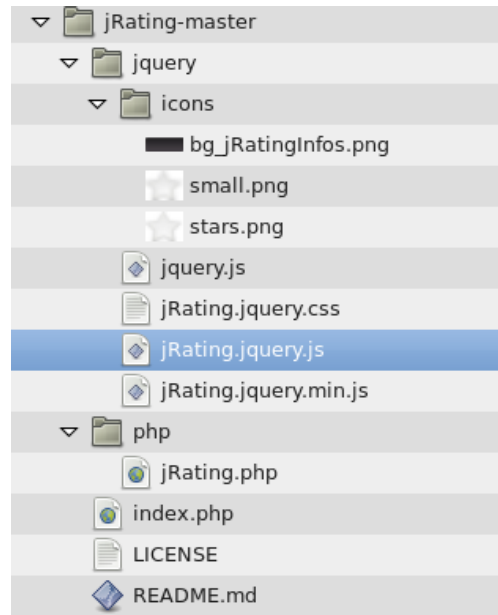
- Back to top
- Bootstrap
- Carousel
- Countdown
- Dropdown Menu
- Form Validation
- Grid Layout
- Validator
- Mixit
- MouseHover
- News Ticker
- Notification
- Parallax
- Prism Layout

What's In the Widget?

Extract the jRating once downloaded. When you checkout its contents (shown right), you can see several files or folders of interest:

- **icons/** - images used by the widget
- **jRating.jquery.css** - CSS styling for the widget
- **jRating.jquery.js** - the widget's Javascript source

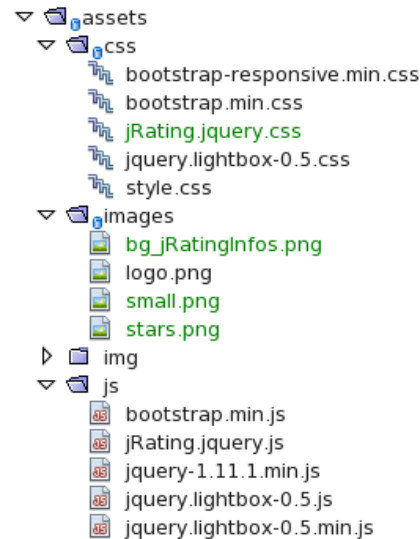
We won't need the others. **The supplied php/jRating.php does not belong anywhere inside our project!**



Copy the Widget Into Our Project

Let's copy the widget pieces that we need into our project's assets folder, and we can then tailor them.

- **icons/*** - the .png images go into our images folder
- **jRating.jquery.css** - goes inside css
- **jRating.jquery.js** - goes inside js



Fix the CSS Image References

We copied the images into our assets/images folder, and need to reflect that in jRating.jquery.css, as shown below.

```
/** P containing the rate informations */
p.jRatingInfos {
    position: absolute;
    z-index: 9999;
    background: transparent url('/assets/images/bg_jRatingInfos.png') no-repeat;
    color: #FFF;
    display: none;
    width: 91px;
    height: 29px;
    font-size: 16px;
    text-align: center;
    padding-top: 5px;
}
```

Fix the JS Image References

We need to make a similar change in jRating.jquery.js, as shown below.

```
(function($) {
    $.fn.jRating = function(op) {
        var defaults = {
            /** String vars */
            bigStarsPath : '/assets/images/stars.png', // path of the icon stars.png
            smallStarsPath : '/assets/images/small.png', // path of the icon small.png
            phpPath : 'php/jRating.php', // path of the php file jRating.php
            type : 'big', // can be set to 'small' or 'big'
        }
    }
})
```

Add a Rating to a Quote

The jRating documentation shows some sample HTML to hold an instance of their widget...

```
<div class="basic" data-average="12" data-id="1"></div>
```

In our case, the data-id will be the id from a quote, and we will need to calculate the average from the values stored in our table...

```
<div class="hollywood" data-average="{average}" data-id="{id}"></div>
```

Add this div below the paragraph holding the author, in views/justone.php

```
<div class="row">
    <div class="span3"></div>
    <div class="span8 offset1">
        <p class="lead">{what}</p><br>
        <p class="text-right">{who}</p>
        <div class="hollywood" data-average="{average}" data-id="{id}"></div>
    </div>
</div>
```

Initializing a Rating Widget

Again referring to the jRating documentaiton, we need to initialize the jRating widget, binding it to a component on the webpage. This will be our views/_components/jrating.php.

We can set some options when it is initialized, including the planned target URL for the AJAX call that the widget will make to record a vote.

Ours should look like:

```
$('.{field}').jRating({
  rateMax : 5, // highest rating
  phpPath : '/viewer/rate' // target for the AJAX call
});
```

Configure Caboose for Ratings

We are finally ready to add a configuration entry for our ratings widget, in libraries/Caboose.php. We can add it to the \$components array after the existing two configuration entries...

```
'jrating' => array(
  'css' => 'jRating.jquery.css',
  'js' => 'jRating.jquery.js',
  'template' => 'jrating'
),
```

jRating Housekeeping

We have another bit of housekeeping to tend to - removing the "demo only" blocks in the jrating's javascript file, assets/js/jRating.jquery.js

For example, the original code is shown to the right, and after trimming below.

```
function(data) {
  if(!data.error)
  {
    /** Here you can display an alert box,
        or use the jNotify Plugin : http://www.myjqueryplugins.com/jNotify
        exemple : */
    if(opts.onSuccess) opts.onSuccess( element, rate );
  }
  else
  {
    /** Here you can display an alert box,
        or use the jNotify Plugin : http://www.myjqueryplugins.com/jNotify
        exemple : */
    if(opts.onError) opts.onError( element, rate );
  }
}
```

```
function(data) {
  if(!data.error)
  {
    /** ONLY FOR THE DEMO, YOU CAN REMOVE THIS CODE **/
    $('<div>serverResponse p</div>').html(data.server);
    /** END ONLY FOR THE DEMO **/

    /** Here you can display an alert box,
        or use the jNotify Plugin : http://www.myjqueryplugins.com/jNotify
        exemple : */
    if(opts.onSuccess) opts.onSuccess( element, rate );
  }
  else
  {
    /** ONLY FOR THE DEMO, YOU CAN REMOVE THIS CODE **/
    $('<div>serverResponse p</div>').html(data.server);
    /** END ONLY FOR THE DEMO **/

    /** Here you can display an alert box,
        or use the jNotify Plugin : http://www.myjqueryplugins.com/jNotify
        exemple : */
    if(opts.onError) opts.onError( element, rate );
  }
}
```


Let's Invoke the Caboose!

We show single quotations in two places: in `controllers/Welcome::index()` and in `controllers/Viewer::quote()`. In both cases, we need to invoke the rating widget before the `$this->render();`

```
$this->caboose->needed('jrating','hollywood');
```

And the result is...



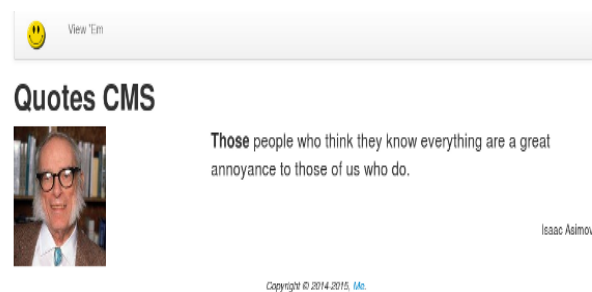
Oops - We Need the Average

The homepage didn't look any different. If you examine the page source, however, you will find...
`<div class="hollywood" data-average="{average}" data-id="6"></div>`

We need to add a bit of logic to calculate the rating average, and to pass it on as a view parameter.

```
$this->data['average'] =
($this->data['vote_count'] > 0) ?
($this->data['vote_total'] /
$this->data['vote_count']) : 0;
```

This calculation and parameter setting should go into both `Welcome::index()` and `Viewer::quote()`. And the result is...



Arghh ... the rating widget should be visible, but isn't!

And We Need to Process the Rating

We need to add the `rate()` method to our `Viewer`, to handle the AJAX request from the rating widget. Without going into too much detail about it, this will look like...

```
// handle a rating
function rate() {
    // detect non-AJAX entry
    if (!isset($_POST['action'])) redirect("/");
    // extract parameters
    $id = intval($_POST['idBox']);
    $rate = intval($_POST['rate']);
    // update the posting
    $record = $this->quotes->get($id);
    if ($record != null) {
        $record->vote_total += $rate;
        $record->vote_count++;
        $this->quotes->update($record);
    }
    $response = 'Thanks for voting!';
    echo json_encode($response);
}
```

Wait, What?

Some of you wanted the detail about the approach shown on the previous slide, so ...

The `rate()` method is an example of the handling of an AJAX request.

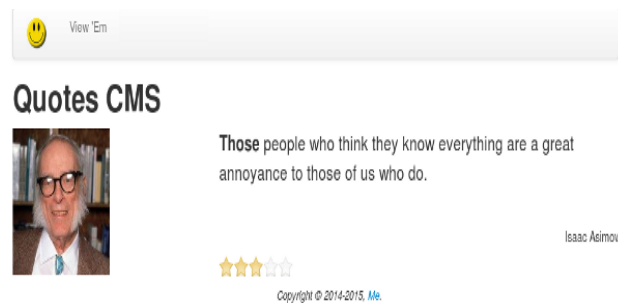
That would make this a "utility controller", i.e. one which is not meant to return HTML. Done properly, it would be a separate controller, perhaps `controllers/Rate`, and what you see here as `rate()` would then be `index()`.

Done properly, its endpoint (for an AJAX request) would be `mysite.com/rate`. Done this way, the endpoint is `mysite.com/viewer/rate`.

You asked for this!

And The Result Is...

The results after voting a few times...



Congratulations!

You have completed tutorial `#ci-normal05: Working With Addins`

If you would take a minute to [provide some feedback](#), we would appreciate it!

The next activity in sequence is: [lab06](#) Addins

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.