

Working With Libraries and Helpers

tutorial #tutorial05

James L. Parry
B.C. Institute of Technology

Tutorial Goals

This tutorial is meant to give you some practice using and extending a library and a helper, as described in lesson 6.

I have prepared a starter webapp – based on the "quotes" webapp you saw and fixed earlier, in tutorial 3. The 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, as well as a simple maintenance component for the quotes.

Revisions & Notes

- The screenshot on original page 3, "The End Result", was incorrect - it showed the end result of the previous tutorial.
- One of the rating widget parameters shown on original page 24, "Initializing a Rating Widger", is incorrect. The highest rating should use the parameter `rateMax` and not `maxRate`

Preparation

I have prepared a [starter webapp](#). The lab 5 starter, Quotes, is meant to handle ordering in a small restaurant.

The webapp is fully functional as delivered, but painfully boring. You are going to spice it up a bit :)

Fork the github project, and clone it locally to work with, the same as you have done with the previous tutorial.

The End Result

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



Order # 12 (0)

Click on a menu item below to add it to your order, or [Checkout](#)



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.

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
- `what` is the actual quoted text, of arbitrary length

What Needs Doing?

1. Enable the supplied bolding hook
2. Add a viewer rating widget
3. Randomize the quote presented on the homepage
4. Add a maintenance controller, presenting an ordered list of the quotes (#5b)
5. Add the ability to add a new quote (#5b)
6. Add the ability to edit a quote (#5c)
7. Add the ability to delete a quote (#5c)
8. Add the ability to fake admin rights (#5c)
9. Add an on-page editing for an admin (#5c)
10. Return to the page we were on after editing a quote (#5c)

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.

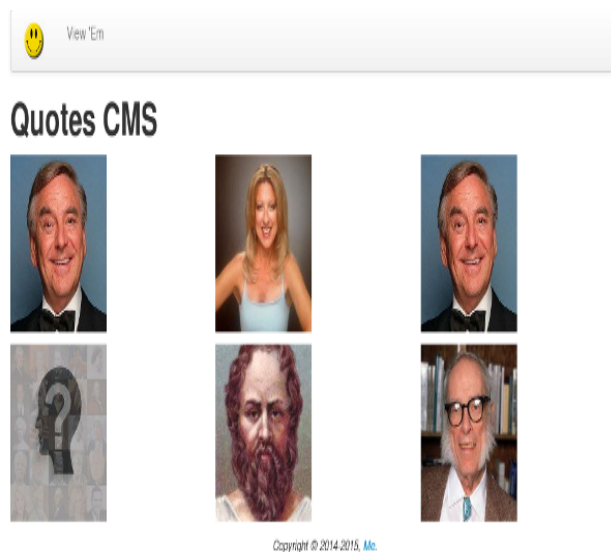
The quote shown is the most recently added, i.e. the one with the highest ID.



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#.

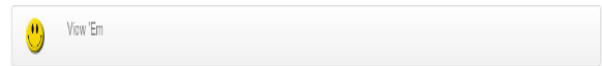
Each image is itself a link to the page that presents just that one quote, implemented by the quote method inside the Viewer controller, and using the justone view that you saw in lab 3.



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.



Quotes CMS



I have six locks on my door all in a row. When I go out, I lock every other one. I figure no matter how long somebody stands there picking the locks, they are always locking three.

Elayne Bousler

Copyright © 2014-2015, Me.

1. ENABLE THE SUPPLIED BOLDING HOOK

The challenge part of lab 3 was to add a "hook" to your webapp, to bold capitalized words in the quoted text. A couple of students asked for a solution for that, so I have included one of the simpler ones submitted.

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.

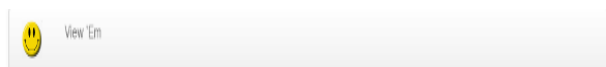
So, why isn't it working already?

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;
```



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.

2. ADD A VIEWER RATING WIDGET

This is a fun little idea, and an excuse to explore the **Caboose** widget manager mentioned in lesson 6. 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							
Browse Structure SQL Search Insert Export							
#	Name	Type	Collation	Attributes	Null	Default	Extra
<input type="checkbox"/> 1	id	int(11)			No	None	AUTO_INCREMENT
<input type="checkbox"/> 2	who	varchar(64)	utf8_general_ci		No	None	
<input type="checkbox"/> 3	mug	varchar(64)	utf8_general_ci		No	None	
<input type="checkbox"/> 4	what	text	utf8_general_ci		No	None	
<input type="checkbox"/> 5	vote_total	int(11)			No	None	
<input type="checkbox"/> 6	vote_count	int(11)			No	None	

Get a Rating Widget

A rating widget that I have used is [jRating](#), from [Alpixel](#).

The jRating link above leads to the documentation for the widget, which is pretty easy to follow for our purposes. The Download link leads to their Github page, from which you can [download](#) a zip of the project. Do so!

jRating is a very flexible jQuery plugin for quickly creating an Ajaxed star rating system. It is possible to configure every details from "the number of the stars" to "if the stars can represent decimals or not".

There is also an option to display small or big stars and images can be changed with any other file easily.

Although jRating can be used with any scripting language, a PHP file that handles the requests is already included in the download package on github.

[DOWNLOAD](#)
[DEMONSTRATION](#)
[PLUGIN HOMEPAGE](#)

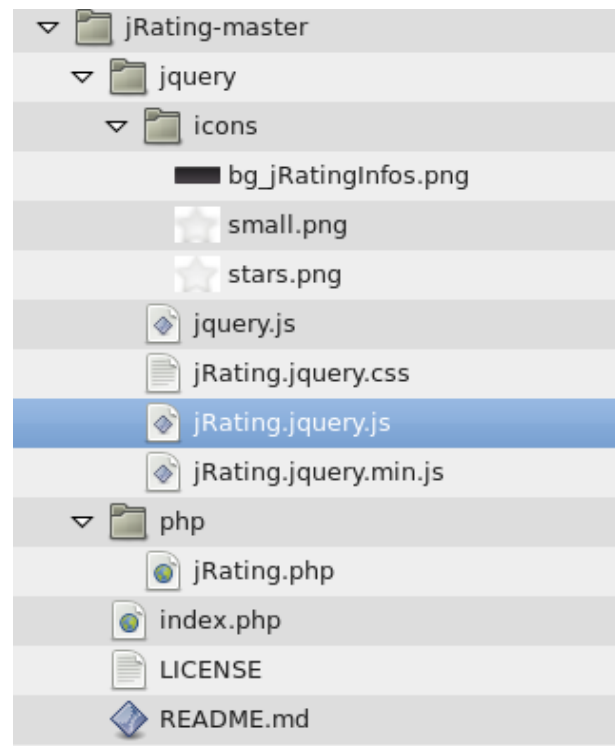
[IMPLEMENTATION](#)
[OPTIONS](#)
[METHODS](#)
[SCREENSHOTS](#)

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

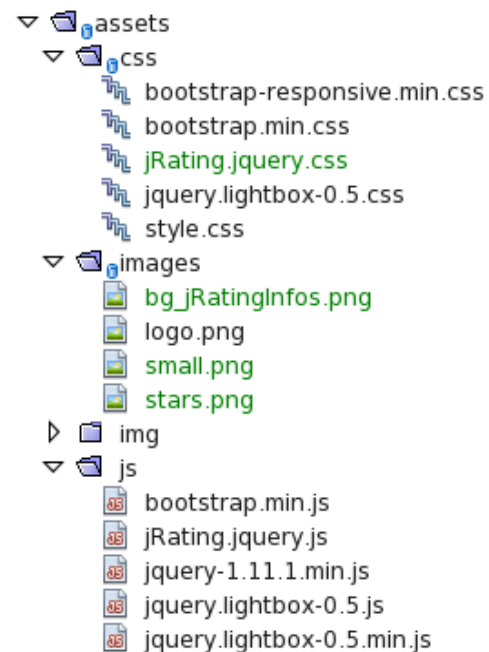
We won't need the others.



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'  
        }  
    }  
})(jQuery);
```

Enable Caboose in our webapp (1/3)

We want to autoload the Caboose library, as it will be used in our master template.

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

Enable Caboose in our webapp (2/3)

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();
}
```


Enable Caboose in our webapp (3/3)

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.

You did notice both places where material was added, didn't you? Check the green bars down the left side of the code editor window :)

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>{title}</title>
    <meta HTTP-EQUIV="Content-Type" CONTENT="text/html" />
    <meta name="viewport" content="width=device-width" />
    <link href="/assets/css/bootstrap.min.css" rel="stylesheet" />
    <link rel="stylesheet" type="text/css" href="/assets/css/{caboose_styles}" />
  </head>
  <body>
    <div class="container">
      <div class="navbar">
        <div class="navbar-inner">
          <a class="brand" href="/"></a>
        </div>
      </div>
      <div id="content">
        <h1>{title}</h1>
        {content}
      </div>
      <div id="footer" class="span12">
        Copyright &copy; 2014-2015, <a href="mailto:{email}">{email}</a>
      </div>
    </div>
    <script src="/assets/js/jquery-1.11.1.min.js"></script>
    <script src="/assets/js/bootstrap.min.js"></script>
    {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...


[View 'Em](#)

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.

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 documentation, 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
      example : */
    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
      example : */
    if(opts.onError) opts.onError( element, rate );
  }
}
```

```
function(data) {
  if(!data.error)
  {
    /** ONLY FOR THE DEMO, YOU CAN REMOVE THIS CODE **/
    $('#serverResponse p').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
      example : */
    if(opts.onSuccess) opts.onSuccess( element, rate );
  }
  else
  {
    /** ONLY FOR THE DEMO, YOU CAN REMOVE THIS CODE **/
    $('#serverResponse p').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
      example : */
    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...

 View 'Em

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.

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;
```

And the result is...

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



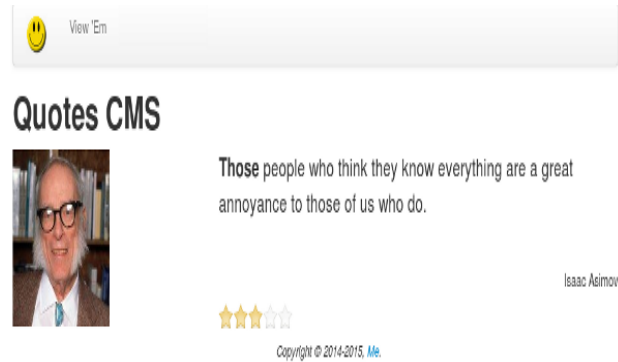
And We Need to Process the Rating

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);
}
```

And The Result Is...

The results after voting a few times...



3. RANDOMIZE THE HOMEPAGE QUOTE

Now that we are voting, we should mix things up a bit on the homepage. Let's choose a quote randomly instead. We'll need to make an adjustment in the `index()` method of our `Welcome` controller...

Instead of

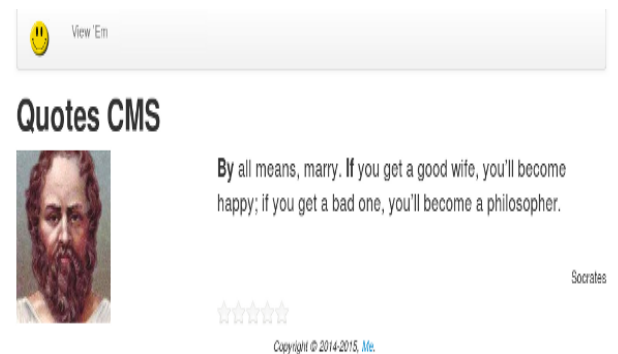
```
$this->data = array_merge($this->data, (array) $this->quotes->last());
```

We can choose a quote randomly instead...

```
$choice = rand(1,$this->quotes->size());  
$this->data = array_merge($this->data, (array) $this->quotes->get($choice));
```

Our Improved Homepage

It doesn't look any different, except that we don't get the same quotation every time we reload the page.



Congratulations!

You have completed tutorial #tutorial05: Working With Libraries and Helpers

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

The next activity in sequence is: [tutorial05b](#) Libraries and Helpers, Part (b)

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.