Assignment for Module #3: Recipe Finder

The overall goal of this assignment is to implement a Rails application using model, view, and controller classes.

- the model class will access information
- the view class will display information and accept commands from the user
- the controller class will implement actions through internal service logic and the delegation to model and view classes.

The functional goal is to provide web page access to recipe information served by http://food2fork.com/api through JSON and images. Documentation for the API can be found at http://food2fork.com/about/api.

Functional Requirements

You are tasked with creating a Rails app that will display a recipe index page based on a search keyword entered.

- the user will supply a keyword to search for
- the Rails app will pass that query to http://food2fork.com/api and accept the results
- the Rails app will build a web page display of the results and accept the next keyword search
- the web page displayed will provide HTML links to more detailed recipe information from other web sites.

You should already have the Recipe class from an earlier assignment. (Remember, that unlike in that assignment - you will not need to require HTTParty gem in your code, since loading HTTParty gem should be the Bundler's job.)

You are also tasked with deploying your solution to Heroku - to be accessed by friends, family, other students, co-workers, and prospective employers.

Getting Started

1. Create a new Rails application using the rails command called recipefinder.
2. Download and extract the starter set of boostrap files into the recipefinder directory.
    - replace the generated Gemfile with the Gemfile from the bootstrap fileset
    - run the bundle command to resolve new gems

```
|-- Gemfile
|-- README.md
|-- .rspec (important hidden file)
`-- spec
   |-- recipes_app_spec.rb
   `-- spec_helper.rb
```

3. Install the following gems used by the rspec unit tests. You may have some of these already installed. The last gem is used for headless web page testing.

```
$ gem install rspec
$ gem install rspec-its
$ gem install capybara
$ gem install poltergeist
```

4. Make sure phantomJS is installed and in your bin PATH on your system ($ phantomjs --version). This binary is used by the poltergeist gem to implement a headless unit test for the Web interface. You can interact with your Rails app directly using a browser without this library. It is only needed by the rspec tests to provide you feedback for example criteria the grader will be looking for later when submitted. PhantomJS installation was covered in Module 1. In case you need more information, the download URLs are below. Linux users will need to use version 1.9.8 or build from source. All other platforms can easily use 2.0.0.
    - phantomjs downloads: http://phantomjs.org/download.html
    - bitbucket: https://bitbucket.org/ariya/phantomjs/downloads
5. Run the rspec test(s) to receive feedback. They must be run from their location at the root of your rails application. All tests will (obviously) fail until you complete the specified solution.

```
Finished in 1.69 seconds (files took 0.41211 seconds to load)
8 examples, 8 failures

Failed examples:

rspec ./spec/recipes_app_spec.rb:6 # Recipes App displays
  'Kahlúa-Spiked' when request parameter 'search' is mocha
rspec ./spec/recipes_app_spec.rb:11 # Recipes App utilizes
  the FOOD2FORK_SERVER_AND_PORT environment variable
rspec ./spec/recipes_app_spec.rb:16 # Recipes App utilizes
  the FOOD2FORK_KEY environment variable
rspec ./spec/recipes_app_spec.rb:24 # Recipes App visit root
  displays chocolate (default)
rspec ./spec/recipes_app_spec.rb:28 # Recipes App visit root
  displays 'Powered By Food2Fork.com'
rspec ./spec/recipes_app_spec.rb:32 # Recipes App visit root
  displays table element that has a row with 3 columns
rspec ./spec/recipes_app_spec.rb:36 # Recipes App visit root
  column 1 should have the thumbnail inside img tag inside a link tag
rspec ./spec/recipes_app_spec.rb:40 # Recipes App visit root
  title should be inside a second column inside a link tagink tag
```

6. Implement your Rails app solution and use the rspec tests to help verify your completed Rails app solution.
7. (Optional) Post your Rails app solution to Heroku.
8. Submit your Rails app solution for grading.

Technical Requirements

1. Create a new Rails app called recipefinder. Use the Gemfile provided in the boostrap files. Do not change the Gemfile from what is provided or your submitted solution may not be able to be processed by the grader (i.e., do not add any additional gems or change gem versions).
2. Generate RecipesController (recipes_controller.rb) that will have an index action
3. The RecipesController index action should
    - check if a request parameter search was passed in.
    - use the search term as the keyword if supplied, and use a default value of chocolate if not supplied
4. Create a model, Recipe (recipe.rb) that will contain a for class method.
5. The Recipe for class method should
    - take a keyword to query
    - query the Food2Fork API for a result.
    - add the HTTP query parameter key (your developer key) to each outgoing URL request to http://food2fork.com/api using HTTParty default_params.
    - obtain the key value from an environment variable FOOD2FORK_KEY
    - obtain the url (and/or port) value from an environment variable FOOD2FORK_SERVER_AND_PORT

    You will use the http://food2fork.com/api host and port# (default=:80) during development and Heroku deployment. However, your assignment will be graded off-line and should get its host and port# from the FOOD2FORK_SERVER_AND_PORT environment variable. Your assignment must use the defined value if present and default to the real value otherwise.

```
class Recipe
  ...
  key_value = ENV['FOOD2FORK_KEY']
  hostport = ENV['FOOD2FORK_SERVER_AND_PORT'] || 'www.food2fork.com'
  base_uri "http://#{hostport}/api"
  ...
```

6. Foods2Fork requires attribution when using their API. Place the following somewhere in your application layout file (application.html.erb) to be displayed alongside the recipes.

```
<p>Powered By Food2Fork.com</p>
```

7. Create your view that should

    - list each recipe as a row in an HTML table (<table>)
    - Each row (<tr>) should have 3 columns (<td>) where

        - column 1 should contain the thumbnail of the recipe,
        - column 2 should contain the title and
        - column 3 should contain the social rank of the recipe.

    You are not required to create an HTML form for the search term. You may specify the search keyword using just the URL with the following syntax in the browser.

```
http://localhost:3000/recipes/index?search=swiss
```

8. Add href tags to your image and title. You should be able to click on either the title or the thumbnail and go straight to the actual recipe (out there on the web). Look at image_tag Rails helper for help with defining an img tag (http://api.rubyonrails.org/classes/ActionView/Helpers/AssetTagHelper.html#method-i-image_tag) and use this helper as the first argument to link_to helper.

9. Inside the image_tag specify width and height of 100 for your images.

10. Sanitize recipe titles displayed. Rails automatically escapes HTML in your strings (to avoid XSS attacks http://en.wikipedia.org/wiki/Cross-site_scripting). Because of this, some of your titles will look wrong. For example, try searching for mocha and look at your titles. To get around this issue, Rails has a sanitize (or raw) helper (http://api.rubyonrails.org/classes/ActionView/Helpers/SanitizeHelper.html#method-i-sanitize) that will help you display HTML characters properly

11. Make the RecipesController index action the default (root) page for your application. Instead of having to go to http://localhost:3000/recipes/index to get to your recipes, you want this page to be the default (root). You should therefore be able to go to http://localhost:3000/?search=apple%20pie for example and see your results.

12. (optional -- ungraded) Deploy your app to Heroku at recipefinderX.herokuapp.com where X is any available number from 1 to 10000000. In order to do this you will have to define the FOOD2FORK_KEY with your key to use the food2fork api on Heroku. Instructions for doing that can be found at the following link:

    https://devcenter.heroku.com/articles/config-vars#example.

Self Grading/Feedback

Some unit tests have been provided in the bootstrap files and provide examples of some tests the grader will be evaluating for when you submit your solution. They can be run from any location but be sure to copy the hidden .rspec file if you move them.

```
$ rspec
...
Recipes App
 displays 'Kahlúa-Spiked' when request parameter 'search' is mocha
 utilizes the FOOD2FORK_SERVER_AND_PORT environment variable
 utilizes the FOOD2FORK_KEY environment variable
 visit root
   displays chocolate (default)
   displays 'Powered By Food2Fork.com'
   displays table element that has a row with 3 columns
   column 1 should have the thumbnail inside img tag inside a link tag
   title should be inside a second column inside a link tag

Finished in 2.73 seconds (files took 0.54954 seconds to load)
8 examples, 0 failures
```

The tests assume your server is running on localhost:3000. Please adjust the source code in recipes_app_spec.rb if that is not the case with your development environment.

```
Capybara.app_host = "http://localhost:3000"
```

Submission

Submit an .zip archive (other archive forms not currently supported) with your solution root directory as the top-level (e.g., your Gemfile and sibling files must be in the root of the archive and *not* in a sub-folder. The grader will replace the spec files with fresh copies and will perform a test with different query terms.

```
|-- app
|   |-- assets
|   |-- controllers
|   |-- helpers
|   |-- mailers
|   |-- models
|   `-- views
|-- bin
|-- config
|-- config.ru
|-- db
|-- Gemfile
|-- Gemfile.lock
|-- lib
|-- log
|-- public
|-- Rakefile
|-- README.rdoc
|-- test
`-- vendor
```

Last Updated: 2015-09-22