Module #3, Assignment 1: Recipe Puppy HTTP API Interface

The purpose of this assignment is to implement a source of data – for use in an MVC Model class implementation.

The functional goal is to implement a restful-flavored API client for `http://www.recipepuppy.com/about/api/` to return JSON documents containing recipe information.

Functional Requirements

1. Implement a Ruby class that will

   - accept a keyword search term
   - forward the keyword search term according to the `www.recipepuppy.com` interface definition using the HTTParty API
   - return the JSON document containing recipe information provided by `www.recipepuppy.com`

Getting Started

1. Download and extract the starter set of boostrap files from (URL).

   ```
   --- student-start
     |-- .rspec (important hidden file)
     |-- chocolate_recipes.json
     |-- module3_1_assignment.rb
     |-- solution.rb
     `-- spec
         |-- recipe_spec.rb
         `-- spec_helper.rb
   ```

   - .rspec - configuration file for unit tests. If you move your files you must take care to also copy this file.
   - module3_lesson1_assignment.rb - contains the starting example. Your solution must be placed within this file.
   - spec - this directory contains tests to verify your solution. You should not modify anything in this directory
   - chocolate_recipes.json - used for off-line unit testing by rspec tests

2. Install the following gems used by the rspec unit tests. You may have some of these already installed. The last gem is used for testing HTTP calls without using the live `www.recipepuppy.com` site.

   ```
   $ gem install rspec
   $ gem install rspec-its
   $ gem install webmock
   ```

3. Read thru the recipepuppy and HTTParty documentation.

   - HTTParty API document is located at `https://github.com/jnunemaker/httparty`
   - recipepuppy interface definition is located at `http://www.recipepuppy.com/about/api/`

4. Implement the Ruby class in a file called `module3_1_assignment.rb`.

5. Run the rspec test(s) to receive feedback. If you copy/move them, be sure to include the important .rspec hidden file. All tests will (obviously) fail until you complete the specified solution.

   ```
   $ rspec

   Recipe
     should respond to #for (FAILED - 1)
     default_params
   ```

```
    example at ./spec/recipe_spec.rb:10 (FAILED - 2)
  base_uri
    example at ./spec/recipe_spec.rb:11 (FAILED - 3)
  Chocolate Search
    example at ./spec/recipe_spec.rb:22 (FAILED - 4)
    size
      example at ./spec/recipe_spec.rb:23 (FAILED - 5)

Failures:

  1) Recipe should respond to #for
  2) Recipe default_params
  3) Recipe base_uri
  4) Recipe Chocolate Search
  5) Recipe Chocolate Search size

Finished in 0.06497 seconds (files took 1.51 seconds to load)
5 examples, 5 failures

Failed examples:

rspec ./spec/recipe_spec.rb:9 # Recipe should respond to #for
rspec ./spec/recipe_spec.rb:10 # Recipe default_params
rspec ./spec/recipe_spec.rb:11 # Recipe base_uri
rspec ./spec/recipe_spec.rb:22 # Recipe Chocolate Search
rspec ./spec/recipe_spec.rb:23 # Recipe Chocolate Search size
```

6. Run the `solution.rb` Ruby script to execute a sample call.

```ruby
require_relative "module3_1_assignment"

puts Recipe.for("chocolate")
```

Technical Requirements

1. Implement a `Recipe` class that will implement the HTTP API to `http://www.recipepuppy.com/about/api/`. The unit tests will expect a class by that exact name.
2. The `Recipe` class should
   - be implemented in a file called `module3_1_assignment.rb`. The unit tests will expect a file by that name.
   - import the HTTParty mixin
   - define a base_uri to use `http://www.recipepuppy.com/api`
   - define a default query param of `onlyImages=1` for all HTTP GET requests in order to only return results that contain associated thumbnails.
   - specify the desired format as `json`
   - specify all the above using legal Ruby syntax and
3. The `Recipe` class must have a `for` class method that
   - accepts a keyword for a search term
   - issues an HTTP GET request using the HTTParty gem
   - the HTTP GET request must have the "q=keyword" query argument
   - returns the JSON payload document supplied in the `results` element of the hash returned by HTTParty

Self Grading/Feedback

Some unit tests have been provided in the bootstrap files that can be used to evaluate your solution. They must be run from the same directory as your solution.

```
$ rspec

Recipe
  should respond to #for
  default_params
    should include {:onlyImages => 1}
  base_uri
    should include "http://www.recipepuppy.com/api"
  Chocolate Search
    should be a kind of Array
    size
      should eq 10

Finished in 0.03678 seconds (files took 0.81808 seconds to load)
5 examples, 0 failures
```

A client script (`solution.rb`) is also provided in the bootstrap and can be used to issue a sample client request.

```
$ ruby solution.rb
{"title"=>"Tim and Tracy's Chocolate Cake (Boiled)", "href"=>"http://www.recipezaar.com/Tim-and-Tracys-Choco
 ...
{"title"=>"Double Chocolate Cookies &amp;amp; Mint Chocolate Variation", "href"=>"http://www.recipezaar.com/Doub
```

Submission

There is no submission required for this assignment but the implementation will be part of a follow-on assignment so please complete this to the requirements of the unit test.

Your final directory contents should look as follows:

```
|-- module3_1_assignment.rb
|-- chocolate_recipes.json
|-- solution.rb
|-- .rspec (important hidden file)
`-- spec
    |-- recipe_spec.rb
    `-- spec_helper.rb
```

Updated: 2015-09-19