Assignment for Module #3: ActionPack

The overall goal of this assignment is to assess your ability to implement and customize Rails scaffold.

This includes:

- Creating a Rails scaffold for a Model, View, and Controller (MVC)
- Modifying the runtime application flow
- Re-using and customizing partial views
- Sharing state through the controller attributes with the view
- Implementing an end-to-end display of a custom query result

The functional goal of this assignment is to implement a web application to manage TodoItems.

Functional Requirements

1. Create the scaffold for the following model type:
   - TodoItem
2. Create the Database (DB) schema for `TodoItem`
3. Change the default scaffolding to route to the `index` page versus the `show` page after a TodoItem is created
4. Remove the `Edit` link from the `index` view.
5. Modify the TodoItem view partial to display the `Completed` property when a TodoItem is being edited and not during creation.
6. Display the number of completed TodoItems.

Getting Started

1. Create a new Rails application called `todolists`
2. Add the following specification to your Gemfile

```
group :test do
  gem 'rspec-rails', '~> 3.0'
  gem 'capybara'
end
```

3. Run the `bundle` command to resolve new gems
4. From the `todolists` application root directory, initialize the rspec tests using the `rails generate rspec:install` command

```
[todolists]$ rails generate rspec:install
    create  .rspec
    create  spec
    create  spec/spec_helper.rb
    create  spec/rails_helper.rb
```

   Add the following line to `.rspec` to add verbose output to test results.

```
--format documentation
```

5. Download and extract the starter set of bootstrap files.

```
|-- Gemfile
|-- db
|    `-- seed.rb
`-- spec
    `-- features
        `-- module3_action_pack_spec.rb
```

- overwrite your existing Gemfile with the Gemfile from the bootstrap fileset. They should be nearly identical, but this is done to make sure the gems and versions you use in your solution can be processed by the automated Grader when you submit. Any submission should be tested with this version of the file.
- overwrite your existing `db/seed.rb` file using the `seeds.rb` provided with the bootstrap fileset. The bootstrap `seeds.rb` file contains some test data that will be useful during development and unit tests.
- add the `spec/features/module3_action_pack_spec.rb` file provided with the bootstrap fileset to your `todolists` application. Within your application root directory, you will first need to create a corresponding `spec/features` sub-directory to place the `module3_action_pack_spec.rb` file. This file contains tests that will help determine whether you have completed the assignment.

6. Run the rspec test(s) to receive feedback. `rspec` must be run from the root directory of your application. All tests will (obviously) fail until you complete the specified solution.

```
$ rspec
...
19 examples, 1 failure, 17 pending
```

To focus test feedback on a specific step of the requirements, add "-e rq##" to the rspec command line to only evaluate that requirement. Pad all step numbers to two digits.

```
$ rspec -e rq01
Run options: include {:full_description=>/rq01/}

Module #3
  rq01
    Generate Rails application
      must have top level structure of a rails application

Finished in 0.00465 seconds (files took 1.56 seconds to load)
1 example, 0 failures
```

7. Implement your solution to the technical requirements and use the rspec tests to help verify your completed solution.
8. Submit your Rails app solution for grading.

Technical Requirements

1. Create a new Rails app called `todolists`. Use the Gemfile provided in the bootstrap files (as stated in Step 5 within the Getting Started section). Do not change this 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).

```
$ rails new ...
$ rspec -e rq01
```

2. Using the `rails generate scaffold` command, create a Rails MVC artifact for a TodoItem that has the following business-related fields:

- TodoItem
  - due_date - date when the specific task is to be complete

- title - a string with short name for specific task
- description - a string with narrative text for specific task
- completed - a boolean value (default=false), indicating whether item is complete

It is assumed that this type will also contain the `id`, `created_at`, and `updated_at` fields. Migrate the database as a part of this requirement to populate the database with the TodoItem schema.

```
$ rails g scaffold ...
$ rake db:migrate
$ rspec -e rq02
```

Note that the above `rake db:migrate` command will execute against the `db/development.sqlite3` database instance. The capybara rspec tests will use the `db/test.sqlite3` database instance and automatically run `db:migrate` and `db:seed` on its own. The default database for all commands is the development database.

- Use `rake db:seed RAILS_ENV=test` to execute the db/seed.rb against the test database.
- Use `rails db -e test` to access the test database.
- Use `rails c test` to use the Rails console to interact with the test database.

Since the grader uses a separate `test` database instance, you can modify the state of the `development` database as you wish during your development.

We will have specific interest in the following artifacts.

```
db/migrate/..._create_todo_items.rb
app/models/todo_item.rb
app/controllers/todo_items_controller.rb
app/views/todo_items/index.html.erb
app/views/todo_items/edit.html.erb
app/views/todo_items/show.html.erb
app/views/todo_items/new.html.erb
app/views/todo_items/_form.html.erb
```

3. Seed the database with the `db/seeds.rb` file supplied in the student-start bootstrap files. Do not modify this file. The grader expects test results to be based on this input.

```
$ rake db:seed
$ rspec -e rq03
```

4. Start the Rails server and navigate to the todo_items index page.

```
$ rails s
```

http://localhost:3000/todo_items

```
$ rspec -e rq04
```

5. Using the `New Todo item` link, create a new TodoItem (with any data). After a new Todo Item has been successfully created, notice the page that it navigated you to. This is the show page.

```
$ rspec -e rq05
```

- Review how the `submit` action in the view invokes a `create` URI when the user presses the `Create Todo Item` button.

  ```
  app/views/todo_items/new.html.erb
  app/views/todo_items/_form.html.erb
  ```

```
<%= form_for(@todo_item) do |f| %>
  ...
  <div class="actions">
    <%= f.submit %>
  </div>
<%= end %>
```

- Notice that the `create` method in the controller handles the `create` URI call passed by the view and persists the new TodoItem. When the save operation is completed, the controller then redirects the flow to the `show` URI for the @todo_item.

```
# POST /todo_items
# POST /todo_items.json
def create
  @todo_item = TodoItem.new(todo_item_params)

  respond_to do |format|
    if @todo_item.save
      format.html { redirect_to @todo_item, notice: 'Todo item was successfully created.' }
      format.json { render :show, status: :created, location: @todo_item }
    ...
    end
  end
end
```

- Notice that the `show` method in the controller retrieves the persisted TodoItem by `:id`. By default the flow continues to the `show` page, where the view displays the details of the newly created @todo_item.

```
class TodoItemsController < ApplicationController
  before_action :set_todo_item, only: [:show, ...]

...

# GET /todo_items/1
# GET /todo_items/1.json
def show
end

...

private
    # Use callbacks to share common setup or constraints between actions.
    def set_todo_item
      @todo_item = TodoItem.find(params[:id])
    end

    ...
```

app/views/todo_items/show.html.erb

```
<p>
  <strong>Due date:</strong>
  <%= @todo_item.due_date %>
</p>

<p>
  <strong>Title:</strong>
  <%= @todo_item.title %>
</p>

<p>
```

```
  <strong>Description:</strong>
  <%= @todo_item.description %>
</p>

<p>
  <strong>Completed:</strong>
  <%= @todo_item.completed %>
</p>
```

- Note the mapping of helper_method_prefix, method/URI, and controller#method mappings shown by the `rake routes` command. This shows which action in the controller will be called when a method/URI is invoked.
  - The controller method is optional and the flow will continue to the view of the same name as the intended action when that occurs.
  - If a controller method does exist and matches the action's name, it has the ability to add member @variables with state for the views to use (e.g., set_todo_item called prior to show).
  - If the controller method does not change the route (e.g., show does not change the route), then the flow will continue to the view of the same name as the action.
  - If the controller method changes the route through a `redirect_to` (e.g., the create action re-directs the flow to the show URI), the flow will follow the newly defined path by sending the HTTP client a re-direct.
  - If the controller method changes the route using a `render` (e.g., the create action redenders a JSON document response when JSON is requested), the specified view is returned directly to the client.

```
$ rake routes
        Prefix Verb   URI Pattern                    Controller#Action
    todo_items GET    /todo_items(.:format)          todo_items#index
               POST   /todo_items(.:format)          todo_items#create
 new_todo_item GET    /todo_items/new(.:format)      todo_items#new
edit_todo_item GET    /todo_items/:id/edit(.:format) todo_items#edit
     todo_item GET    /todo_items/:id(.:format)      todo_items#show
               PATCH  /todo_items/:id(.:format)      todo_items#update
               PUT    /todo_items/:id(.:format)      todo_items#update
               DELETE /todo_items/:id(.:format)      todo_items#destroy
```

6. Modify the flow so that the user is directed back to the `index` page after creating a TodoItem instead of the `show` page. (Hint: you are changing the URI redirected by the controller's `create` method. Use `rake routes` to help determine the appropriate helper_method_prefix, URI, and controller#method mappings. Append `_url` to the helper method prefix when implementing this redirection.)

   ```
   $ rake routes
   $ rspec -e rq06
   ```

7. Remove the `Edit` link from the `index` page view. (You will still be able to access edit from the `show` page view).

   ```
   $ rspec -e rq07
   ```

8. Add conditional logic to the `_form.html.erb` partial so that it only displays the `completed` property when editing and not when creating. (A TodoItem cannot possibly be done before it is created;)

   ```
   $ rspec -e rq08
   ```

   Hint: You can obtain the model object's persisted state using `object.persisted?` or `object.new_record?` to help determine if it is new or being edited.

9. Display the number of completed TodoItems on the `index` page.

   1. Implement a class method in the `TodoItem` model that returns the count of completed TodoItems.

2. Update the `index` method in the controller class to assign the count of completed TodoItems in a member variable (.e.g, @number_of_completed_todos)
3. Display the count of completed TodoItems on the `index` page using a reference by the view – to the member variable defined in the controller class. The grader is looking for the result to be expressed as `Number of Completed Todos:  #` anywhere on the page – where `#` is the number of completed todos. There must be a single space between the `:` and number.

```
$ rspec -e rq09
```

Hint: This should be very similar to how the view class gets the list of TodoItems from the controller using the `@todo_items` member variable.

Self Grading/Feedback

Some unit tests have been provided in the bootstrap files and provide examples of tests the grader will be evaluating for when you submit your solution. They must be run from the project root directory.

```
$ rspec
...
19 examples, 0 failures
```

You can run as many specific tests you wish be adding `-e rq## -e rq##`

```
$ rspec -e rq01 -e rq02
```

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: 2015-10-25