CS 312 - Assignment 5
Rails

Due 6pm on Friday, March 23

For this assignment you will be walking through a tutorial for building a simple blog application[1].

Partners
You can (and are encouraged to) work with a partner on this project. If you do, you must both be there whenever you are working on the project.

Git
For this assignment, you should create a new Git repository and commit the changes you make regularly. This will be useful for both practicing GIt as well as for if you make a mistake and need to go back. As before, you will turn in the commit log with your assignment.

[1] It turned out to be too steep a learning curve to try and do the Animal game in ruby, but I encourage you to try it for fun after you finish this.

Questions

Throughout this handout, I have interspersed questions regarding the tutorial. I've tried to make them very obvious what the questions are and what sections in the tutorial they reference. As you work through the tutorial, keep this handout open and answer the questions as you go. Keep your answers in a file and submit this when you submit your assignment. Feel free to search the web or use the rails book to figure out the answers if you can't figure them out from the context of the tutorial.

Take your time

The goal of this assignment is not to see how fast you can finish, but to learn more about ruby on rails. As you work through the tutorial, read all of the text and look at all of the code/html that you're editing and try and figure out exactly what is going on. Rails relies heavily on conventions and it can be confusing at first, but if you take your time, you should be able to figure things out.

READ THE FOLLOWING BEFORE PROCEEDING:

Before rails will work properly on any of the computers in MBH 632 you *must* run the following command:

```
rvm --default use 1.9.3-p0
```

This tells rails what version of ruby to use and initializes a number of variables that are important.

- You only need to do this once *per computer*.
- So if you move to a new computer, you will have to run this again on that computer before it will work.
- If you want to be extra careful feel free to put this command in your `.bashrc` file (though you might get errors if you login to machines that don't have ruby installed, for example basin).

The tutorial we will be using is:
`http://guides.rubyonrails.org/getting_started.html`

Work through this tutorial, generating the blog application as you go.

1 Guide Assumptions

Read or skip this section

2 What is Rails?

Read this section

3 Creating a New Rails Project

Only do section 3.2

4 Hello, Rails!

After you have your "Hello, Rails" project up and running create a second rails project (e.g. `rails new testproject`). Stop the server on your blog project (`Ctrl+C`) then `cd` into your second project and start the `rails server`. If you now go to `http://localhost:3000/` you should see the generic start page again (if you'd like, feel free to edit the `public/index.html` file to show something else.

You may be developing multiple rails projects, but you may only have a single web server running on any given port. Go back to your `blog` project and start the `rails server` there.

5 Getting Up and Running Quickly with Scaffolding

Read this section

6 Creating a Resource

Question 1: In the statement:

```
$ rails generate scaffold Post name:string title:string content:text
```

what are two other types you could have put after the ':'?

6.1 Running a Migration

After you have migrated the database, go into the `db` directory and run:

```
$ sqlite3 development.sqlite3
```

You should now see your `posts` table in the database. Sometimes it can be nice to look at the files in the database when you're working on a project to see exactly what is in there.

If you ever want to under a database migration, you can type:

```
$ rake db:rollback
```

which will undo the most recent migration (you can also undo particular migrations by specifying more parameters). Give this a try and then look in the database. Notice that the table is now gone. When you're done, make sure to apply the db migration again:

```
$ rake db:migrate
```

6.2 Adding a Link

Question 2: If instead of generating a scaffold called `Post` you called it `Entry` what would you write in `app/views/home/index.html.erb` to link to this content (i.e. how would you change the first box in 6.2)? (Don't actually make this change).

6.5 Adding Some Validation

Question 3: When validating the posts, what you you need to enter into the `post.rb` file to check that the post content was non-empty AND was at most 500 characters?

(the link below may be useful:
http://guides.rubyonrails.org/active_record_validations_callbacks.html#validations-overview

---

If you want you can add this additional check and then check that it works in the rails console.

6.7 Listing All Posts

Question 4: Why isn't the line `<% @posts.each do |post| %>` in `app/views/posts/index.html.erb` `<%= @posts.each do |post| %>`?

6.8 Customizing the Layout

Question 5: What does the "yield" statement indicate in `app/views/layouts/application.html.erb`

If you're not sure, try moving it around (or putting it in multiple times and see what happens to the display of your application.

6.9 Creating New Posts

Question 6: What happens when we remove:

```
<div class="field">
    <%= f.label :name %><br />
    <%= f.text_field :name %>
</div>
```

from `views/posts/_form.html.erb`? Why?

6.10 Showing an Individual Post

Question 7: What would `:id` be in the method `show` if we visited the url `http://localhost:3000/posts/15`?

6.11 Editing Posts

Question 8: What does `Post.find(params[:id])` do?

6.12 Destroying a Post

Question 9: When are the following controller methods called inside the `app/controllers/posts_controller.rb`:

- `index`
- `show`
- `new`

- `edit`
- `create`
- `update`
- `destroy`

---

7 Adding a Second Model

7.1 Generating a Model

Question 10: What does the `post:references` indicate in the `rails generate` command?

Question 11: After applying the new database migration, what column name does the `post:references` correspond to in the `comments` table? Note, you will probably have to look in the database to see this.

7.2 Associating Models

Question 11: If we wanted to make sure that comments weren't empty and contained at least 5 characters:

- What file would we edit (give the complete path to it within the project)?
- What would we add to that file?

7.4 Generating a Controller

Question 12: Explain what each of the lines below (which you added to the `CommentController`) is doing:

```
@post = Post.find(params[:post_id])
@comment = @post.comments.create(params[:comment])
redirect_to post_path(@post)
```

---

8 Refactoring

8.1 Rendering Partial Collections

Question 13: What does the line `<%= render @post.comments %>` do in your new `app/views/posts/show.html.erb`?

---

9 Deleting Comments

Question 14: When you click the "Destroy Comment" link when viewing a post, what controller method is called (be specific about the controller class and the controller method)?

---

10 Security

Question 15: What does `:except => [:index, :show]` indicate? What

does `:only => :destroy` indicate?

---

11 Building a Multi-Modal Form

Can do if you'd like, but not required.

12 View Helpers

Can do if you'd like, but not required.

13 Testing, Testing, Testing

Add a few unit tests to check that your code is doing the right thing. You should have some standalone tests as well as at least one test fixture. You don't need to go crazy, but I want you to practice writing unit tests in Ruby. Chapter 7 from the Rails book has some useful information on how to do this.

14 Sprucing things up

Once you've completed the sections above, you should have a simple working blog application. Right now, though, it's pretty plain. Spruce up the overall application by editing the .css files associated with the application. Chapter 8 from the rails book has some useful information on how to do this.

What to submit

As always, create a folder with your name and assignment number on it and follow the normal directions for submitting (if there are two of you, put both of your names on the folder, but only one person needs to submit it). Within this folder, your submission should include the following:

- Your `blog` directory containing your final application
- A Git log file showing your commits as you worked through the tutorial. For example:

  ```
  $ git log > git_log.txt
  ```

- A filed called `answers.txt` which contains your answers to the questions above. Number your answers so its clear what question you're answering.