Homework 2: Building a Blog in Rails

Due **11:59PM, Wednesday, Oct.23rd**
The deadline is closer than you think. Please plan accordingly.
You can use your late days on this homework.

Introduction

One of the contributors to Rails' explosive growth was a talk given by the creator, David Heinemeier Hansson, who shows how to create a blog in less than 15 minutes (viewing is optional). This was completely novel at the time, and shows how powerful Rails is and how quickly you can get an application up and running.

With just the few lectures we have had together, we will build a blog of our own from scratch- from going from the starting rails project all the way to having
our blog on the world wide web!

In this homework, you will

1. Create a blog that:
    - has a home page, with a gateway into a page with all your blog posts.
    - can create, edit, and delete blog posts (by you).
2. Write HTML and CSS to make it look good.
3. Deploy the blog on the open web.
    - ideally as your actual personal website/blog!

Here is a quick demo of the basic functionality of the blog.
But aesthetic-wise, we hope your project can look much more better than this one. Perhaps something like this is what you should aim for.

Please start early- some parts of this project (namely deployment) can be very
time intensive to debug.

Note that in we are not providing any starter code for this assignment. You will build everything using Rails from the ground up!

Starting up and the Home Page

Start off your project by creating a new Rails application named `hw3`. You can run `rails s` on this blank project, and you will be greeted with the Rails welcome page. Well done!

...but this isn't a blog/website.

First things first, let's create a home page. Our home page should contain the title, and the link to your blog posts.

1.1 - Our Home Page

To do that, create a `controller` called `WelcomeController` with a single action `index` and its corresponding view. In the view, put in the **title** of your blog. Don't put the link to your blog posts yet! We'll do that later once we have that all figured out.

Then, in `routes.rb`, define the root as the index action of your welcome controller.

Upon reload, your app should now have the title of your blog on top of the page. Great! We're on our way.

The Posts

With this, let's continue with our application. We need to have some way to create blog posts. We'll represent our blog posts with the model `BlogPost`.

Our BlogPost will have the following properties:

- `title`, of type `string`, will be the title of each blog post.
- `content`, of type `text`, will contain the content (the text) of each post.

(`string` and `text` both store strings of characters, but `string` is more appropriate for things that will be shorter).

1.2 - The Model

Make a model with those properties. Remember to migrate!

Cool, so we have our model... but what do we do with it? This is where the View and the Controller kick in! If you'd recall, the person interacting with your blog page will interact with the view. The view will then interact with the controller to figure out what to do.

1.3 - The Controller

1. Make `controllers/blog_posts_controller.rb` and add its appropriate methods! We'd want to show all the posts, show an individual post on a page, make a post, edit a post, and delete a post. Can you figure out which methods we need?

   Hint: Take a look at this link for the list of controller actions.

2. Hook up `routes.rb` to use that controller!

With these done, we can now hypothetically go to your `https://localhost:3000/blog_posts`, but we don't have anything to show the users! Also, how do we know that our controller methods actually work? Wouldn't it be nice to have visual confirmation for ourselves that our application works as intended?

Let's go onto the views!

Let's recall what our application can do: we can show all posts, show only a single post, make a post, edit a post, and delete a post. This means we'll need four view files in our `views/blog_posts` directory.

- `index.html.erb` : This will show all the posts
- `show.html.erb` : This will show only a single blog post.
- `new.html.erb` : This will have a form to make a post.
- `edit.html.erb` : This will have a form **with an already existing post's information** filled in, and you can edit it.

(Notice that we don't have a page for deleting a post. Why?)

1.4 - The Views

1. Make `index.html.erb` : List all the posts on this page. For each post, have a link to `show` , `edit` , and `destroy` . You should also have a link here that will take you to the `new` page as well.

   Going back to the homepage a little (the thing we made all the way in 1.1), you can also finally add a link from your homepage to this `index` page.

   You can decide how much you want to show on this page. I personally show both the title and the content as well, but only showing the title is fine. Your app, your design!

2. Make `show.html.erb` , which will show both the title and the content of the selected blog post. You should also add a button to go back to the index here.

3. Make `new.html.erb` , which will have a form to create a new blog post! You should also add a button to go back to the index as well.

4. Make `edit.html.erb` , which will have a form with a blog post's information filled out, and that you can edit.

This is also when you can have visual confirmation that your controller and model work!

Making the blog look good

Great, our application should be working now! You should be able to read your posts, create new posts, edit them, and delete! With our functionality done, let's go on to the fun part- making our application look good!

1.5 - HTML and CSS

Utilize HTML and CSS to make your app look good! You can utilize Bootstrap here,

and a link to installing Bootstrap can be seen here.

There aren't strict guidelines for how your blog should look, but it should maintain the general structure and functionality as above. It should also be more than just a few lines of CSS- we'll be grading you on how much effort was put into styling!

For additional resources outside of the CIS 19X lectures, we suggest looking at w3schools for examples and syntax, and Scrimba for a more active learning experience.

Deployment

We're not quite done yet- we want to share our thoughts with the world! We must leave our familiar `localhost:3000` behind and move into deployment!

We will use Heroku in this class for deployment. You can use any deployment service you prefer, but the 196 team strongly recommends Heroku and can only provide technical support for this.

To deploy using Heroku, follow the guide here.

Submission

To submit:

1. Commit your changes and push your changes to Gitlab.
2. Make a zip for this folder, and upload it on Canvas.
3. In your canvas submission, make sure you put the link of your deployed blog in submission comment.

Final Remarks

Even with the few tools we've learned, we've made a very powerful application and showcased some of the things Rails can offer. Of course, there's many things we can improve on this blog- having users sign in and comment, making sure that only you can make blog posts (as it stands, anyone can make a blog post), maybe even interfacing with social media! We'll learn all this and much more in the following weeks.