Module 4
MVC, RESTful Routes, CRUD, Sinatra

CS W169A: Software Engineering

John Yang | Summer 2020

1 Instructions

This section we will take a look at how to apply ideas of MVC, RESTful Routes, and CRUD in the context of the Sinatra framework to build a to-do list app. When you'redone, users should be able to go to your website, view their list of to-do items, create new list items, edit list items, and delete list items.

We will be building the codebase, with the starter code located at https://github.comjohn-b-yang/cs169-exercises. If you are able, find a classmate to pair program through these exercises! Here is the reference to Sinatra, which will be helpful!

2 Setup

Open a command line prompt and enter the following commands:

```
git clone https://github.com/john-b-yang/cs169-exercises
cd cs169-exercises/sinatra-intro/
bundle install
ruby template.rb # OR: bundle exec ruby template.rb
```

Then, enter the following link into a browser to view the webpage and check if it's working.

```
http://localhost:4567/todos
```

Also, try this following command using 'curl' to verify that the app is running locally and responsive. The command fires a GET request to retrieve the "to do" list, and you should receive a response displayed in the command line's standard output.

```
curl http://localhost:4567/todos
```

In the following exercises, we'll be adding more routes, and you can continue to use curl commands with different arguments to verify their behaviors' correctness.

Goal: Your task is to implement the parts of the file labeled "YOUR CODE HERE". The reference containing the solutions is in the final.rb file.

3 Task 1

The first thing we are going to do is create a model. Unlike Rails, Sinatra doesn't have MVC baked in so we're going to hack our own. We're going to use ActiveRecord on top of a SQLite database. In this application, what is our model going to be, and what CRUD operations are we going to apply to the model?

Based on the codebase, it looks like our model includes "user" and "todo" entities.

- index: Lists all "todo" items for user
- create: CREATE a new "user", CREATE a new "todo"

- read: GET specific "todo"
- update: UPDATE a "todo", mark it as completed
- destroy: DELETE a "todo"

4 Task 2

Next, let's create some routes so that users can interface with our app. Here is an example URL:

`https://www.etsy.com/search?q=test#copy`

First, specify what parts of the URL are which components based on our discussion of the anatomy of a URL from lecture. (If this doesn't ring a bell, review Module 4.4: Routes, Controllers, Views). Check out this post by IBM detailing the components of a URL.

The typical anatomy of a URL is as follows: "scheme://host:port/path?query"

- Scheme: Identifies protocol to be used to access the resource on the internet. Either HTTP or HTTPS (a.k.a. HTTP + SSL)
- Host: Host name identifies the host holding the resource (i.e. www.example.com). The host name can be thought of as an alias that pings the server which provides the desired service.
- Port Number (Optional): Follows a host name if present. Low port numbers for popular services (i.e. HTTP and HTTPS) are normally omitted from the URL.
- Path: Identifies specific resource within the host that the Web client wants to access (i.e. /user/repositories/repoName/directory.
- Query String: If query string used, it follows the path component. It is information the resource can use (i.e. as parameters for search, data to be processed). Query strings are usually a series of name-value pairs (i.e. term=bluebird) joined by ampersands (&).
- Anchor: Use this to specify a part of the page you'd like the browser to scroll to, based on that element's IDs. For instance, https://en.wikipedia.org/wiki/Software_engineeringSoftware_design loads the page, then goes directly to the element that has the "Software_design" ID.
- https:// : scheme
- etsy : host
- 443 : port
- /search : path
- q=test : parameters
- copy : anchor

In Sinatra the routing and controller are coupled, making it easier to declare paths. We're going to use declare some RESTful routes so that we can view a list of to-do items, create a to-do item, edit a to-do item, and delete a to-do item. What RESTful actions should we use for these?

CREATE, READ, UPDATE, DELETE

5 Task 3

Since HTTP is a RESTful protocol, every request must follow with a response, so we need to return a view or redirect to every request. We're going to use JSON for our responses, which is similar to what a lot of APIs do. Where should the response go?

The response should be sent at the end of the route handler.