Week 3 Section

Demo MVC, RESTful Routes and CRUD w/ Sinatra

Quick setup:

```
git clone https://github.com/jeremywrnr/sinatra-intro
cd sinatra-intro
bundle install
ruby template.rb # OR: bundle exec ruby template.rb
```

Then open this webpage:

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

Also try with `curl`:

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

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're done, 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 together so pair up and get the starter code at:

https://github.com/jeremywrnr/sinatra-intro

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?

Model includes Users and Todos.

- (a) index: list all todos for user
- (b) create: CREATE user, CREATE todo
- (c) read: GET specific todo
- (d) update: UPDATE todo, mark completed
- (e) destroy: DELETE todo

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:443/search?q=test#copy`

From [reference link]:

scheme://host:port/path?query#fragment

A URL for HTTP (or HTTPS) is normally made up of three or four components:

A scheme. The scheme identifies the protocol to be used to access the resource on the Internet. It can be HTTP (without SSL) or HTTPS (with SSL).

A host. The host name identifies the host that holds the resource. For example, www.example.com. A server provides services in the name of the host, but there is not a one-to-one mapping between hosts and servers. Host names explains more about host names.

Host names can also be followed by a port number. Port numbers explains more about these. Well-known port numbers for a service are normally omitted from the URL. Most servers use the well-known port numbers for HTTP and HTTPS , so most HTTP URLs omit the port number.

A path. The path identifies the specific resource within the host that the Web client wants to access. For example, /software/htp/cics/index.html.

A query string. If a query string is used, it follows the path component, and provides a string of information that the resource can use for some purpose (for example, as parameters for a search or as data to be processed). The query string is usually a string of name and value pairs, for example, term=bluebird. Name and value pairs are separated from each other by an ampersand (&), for example, term=bluebird&source=browser-search.

Break down the URL into its component parts:

- https:// : scheme
- www.etsy.com : host
- 443 : port
- /search : path
- q=test : params
- copy : anchor (This is not specified in the reference, but you can use this to specify a specific part of the page you would like the browser to scroll to, based on that elements ids. For example; https://en.wikipedia.org/wiki/Battus_polydamas#Habitat will load the page and go directly to the element that has the *Habitat* id.)

In Sinatra the routing and controller are coupled. It's easy 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

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.

Reference: http://sinatrarb.com/intro.html