Module 4
Rails Introduction

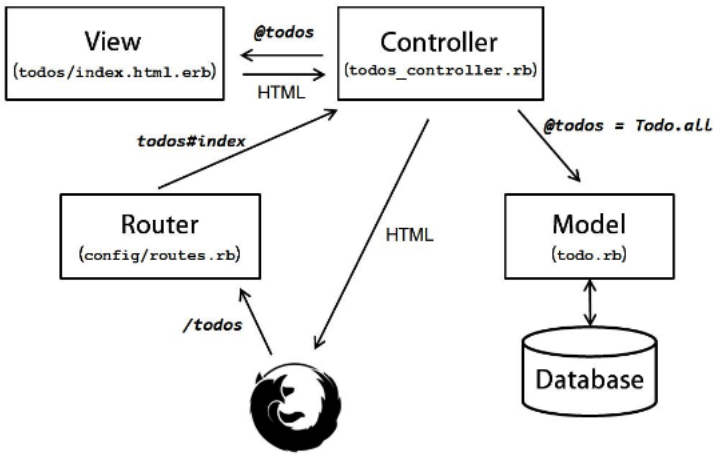
CS W169A: Software Engineering

1 Overview

In this worksheet, we're going to learn the basics of Rails by creating a Rails app from scratch! This exercise will use a lot of powerful Rails commands that abstract away many individual operations. The worksheet will detail the commands for different operations, but we encourage you to read the docs for a more in depth understanding of what this commands are doing under the hood. If possible, this would be a great exercise to be pair program.

To begin, you will not need to download any code at all. Simple open a command prompt. Make sure you have both Ruby and Rails installed. Throughout the exercise, you may run into some "package missing" errors, which is caused by the code calling a dependency that wasn't installed. You can resolve these issues with a simple "gem install". A reference of the completed project is linked here.

From a high level, this is a visualization of how the files and entities within our Rails application will relate to one another, framed in the context of MVC!.



2 Set Up

To create a Rails app, we will use the `rails new <app_name>` command (docs). Type the following into the command prompt and enter:

```
>> rails new todo_app
```

The command will have performed correctly if you see a list of `create <file name>` output, followed by another list of `Using <gem>`. The `rails new` command creates the skeleton of a simple application that runs. This output details the files and gems that were installed. If you run `ls`, you should see a new directory called "todo_app". If you `cd` into it and run `ls`, you will see the folders and files of your new application. Powerful stuff!

3 Our First Model

Now that we have our application framework, it's time to define the Models of our application. In other words, we are going to create the "To Do" class! The `rails generate` command is a powerful tool that helps us do just that.

Change directory into "todo_app", then type and enter the following in the command prompt:

```
>> rails generate scaffold todo description:string
```

What's happening here (docs)? If the command ran successfully, you should once again see a list of both `create <file name>` and `invoke <generator>` output.

Let's step through the output:

```
invoke  active_record
create    db/migrate/20200528014227_create_todos.rb
create    app/models/todo.rb
invoke    test_unit
create      test/models/todo_test.rb
create      test/fixtures/todos.yml
invoke  resource_route
 route    resources :todos
invoke  scaffold_controller
create    app/controllers/todos_controller.rb
invoke    erb
create      app/views/todos
create      app/views/todos/index.html.erb
create      app/views/todos/edit.html.erb
create      app/views/todos/show.html.erb
create      app/views/todos/new.html.erb
create      app/views/todos/_form.html.erb
invoke    test_unit
create      test/controllers/todos_controller_test.rb
create      test/system/todos_test.rb
invoke    helper
create      app/helpers/todos_helper.rb
invoke      test_unit
invoke    jbuilder
create      app/views/todos/index.json.jbuilder
create      app/views/todos/show.json.jbuilder
create      app/views/todos/_todo.json.jbuilder
invoke  assets
invoke    scss
create      app/assets/stylesheets/todos.scss
invoke  scss
create    app/assets/stylesheets/scaffolds.scss
```

Don't worry if yours doesn't look exactly like the above. The `invoke` commands' arguments are "generators". Generators can be thought of as scripts for generating useful items in Rails (docs, Tutorial). To create the Model of an entity, multiple generators are called upon. The third word, "scaffold", indicates that we'd like to create a resource called "todo" with a single attribute "description" of type "string". Running this command leads to creation of new migration, database, view (ERB, HTML, CCS), and controller files.

4 Database

Okay, now that we have a model, let's create a database so we can persist any information that the application might want to store. Without creating a database, no changes we make will be recorded in our database schema. You can think of this step as building the connection between Model and Database of the Overview diagram. Still within the root of the "todo_app" directory, type and enter the following in the command prompt:

```
>> bundle exec rake db:migrate
```

This command looks a little different from our previous `rails` prefixed commands. What's going on here? `bundle exec` is a Bundler command that executes a script in the context of the current bundle. By context, we're referring to your directory's Gemfile. `rake` is the script, and `db:migrate` is <namespace>:<defined task name> (There are a lot of new terms here, feel free to pause and read more online documentation before proceeding!). In this case, we're running the migrate task on our database. You should output similar to the below:

```
johnbyang@john-yangs-mbp todo_app % bundle exec rake db:migrate
== 20200528014227 CreateTodos: migrating ======================================
-- create_table(:todos)
   -> 0.0028s
== 20200528014227 CreateTodos: migrated (0.0028s) =============================
```

Ok. Now that we have our database, let's insert some records! Open `db/seeds.rb` in a text editor of your choice, then paste the following two lines at the bottom of the file, below the comments:

```
Todo.create(description: "start␣the␣cs169␣discussion␣section")
Todo.create(description: "release␣the␣third␣homework")
```

Then, run the folloowing command in your command prompt to insert the data into the database:

```
>> rake db:seed
```

To check if the records are inserted into the database successfully, you can run `rails console` (docs) in your command prompt, which opens up an interactive session that allows you to access `app` and `helper` instance variables and their values.

5 Poking Around

You can run the command `rails server` to start the server and run your application. If it executes correctly, there should be a hyperlink that you can copy and paste in a browser to see your application!

Type `rake routes` within your command prompt to see all the routes generated by the scaffold command. We recommend making requests to these routes so you can see how your application responds, both in terms of visual behavior along with what code is called or modified.

At this point, feel free to play around a bit with the application. We encourage you to change parts of the codebase and see how things change!

6 Extending the App

6.1 More Migrations

Our first migration allowed us to create the database. Let's say the customer wants each "to do" to have a due date associated with it. How do we make that happen? Time to use another generator, specifically the `rails generate migration AddDueDateToTodo due_date:datetime` command! In the command prompt, run the following:

```
>> rails generate migration AddDueDateToTodo due_date:datetime
```

This time, we'll leave it up to you to figure out what new files or changes to the existing files may have occurred. The answer can also be found in the documentation.

For these changes to persist, make sure to run `rake db:migrate` one more time.

6.2 A New Route

At some point, you may want to add a new service to your existing application. For it to be accessed, you'll have to create a new route. Let's get some practice.

1. Add a route in `config/routes.rb` and map it to a controller action. (Place this line inside the `routes.draw` body).

```
get '/hello', to: 'todos#hello'
```

2. Add a new view corresponding to the route. If someone navigates to the "hello" page of your application (i.e. `http://localhost:<port number>/hello/`), there will be a web page displayed.
   - Create a file in the `app/views/todos` directory called `hello.html.erb`.
   - You can enter whatever HTML you'd like to display your web page, but for now, you can get away with just putting `<h1>Hello!</h1>`.
   - Add the below method to `app/controllers/todos_controller.rb`.

```
def hello
    respond_to do |format|
        format.html { render :hello }
        format.json { render json: "hello world!" }
    end
end
```

7 Extras

Think about how you might go about doing the following. The solutions to these are in the example codebase and the solution worksheet.

- Add a new attribute to the Todo model and update the views to display and edit the new field. For practical purposes, let's say we want to add a new boolean field call "done" with a default value of false.
  Solution:
  1. In command prompt: `rails generate migration add_done_to_todos done:boolean`
  2. change the migrate file generated by the above to the following:

```
class AddDoneToTodos < ActiveRecord::Migration
    def change
        add_column :todos, :done, :boolean, default: false
    end
end
```

  3. In command prompt: `bundle exec rake db:migrate`
  4. Use `rails console` command to check if the change has been integrated.
  5. Add the following code at the bottom of the `app/views/todos/_form.html.erb` file.

```
<%= f.label :done %>
<%= f.check_box :done %><br>
```

  6. Change `index.html.erb` and show.html.erb respectively
  7. Change the `todo_params` method in `todos_controller.rb` to the following:

```
def todo_params
    params.require(:todo).permit([:description, :due_date,:done])
end
```

- Change the routing schema. Suppose we want a new route `new_todo` to go to a page that creates a new Todo item.
  Solution: Add the following code to the `config/routes.rb` file

```
get '/new_todo, to: 'todos#new'
```

- Set todos#index as the homepage for the app. Solution: Place the following in the `config/routes.rb` file

```
root 'todos#index'
```

8 Miscellaneous DB Notes

- `rake db:migrate VERSION=0` - Command for reversing back to a previous version of the database
- `rake db:migrate:status` - List all migrations. Should show a set of logs similar to the bottom image. The Migration ID corresponds to the version

```
 Status   Migration ID    Migration Name
--------------------------------------------------
   up     20160211013318  Create todos
   up     20160211025643  Add done to todos
   up     20160211043246  Add due date to todo
```