Rails Demo 0: Getting to Know Rails!

Let's create our first hello world web applications!

1. Create a new rails app:

   ```
   rails new toystore
   ```

2. Review the basic directory structure of a Rails app:

   - app: contains most of our app code, models, views, and controllers as well as assets: javascript, css, etc.
   - bin: contains the script needed to start your app, and other setup scripts as well.
   - config: specifies the app's routes, database, and more configuration related info.
   - db: contains the app's database schema and migrations.
   - Gemfile: define's which Ruby gems your app depends on. RubyGems is a ruby package management system.
   - lib: extended modules for your app.
   - log: your app's log files.
   - public: the only directory seen by the world as is. Contains any static files as well as compiled assets.
   - Rakefile: locates and loads tasks that can be run from the commandline.
   - test: unit tests, test fixtures, and other text apparatus.
   - tmp: tmp files go here.
   - vendor: all third party code goes here (gems)

3. Wow that was fun. Let's get our toy web app to say hello world for us! We do this by creating a new controller: (make sure you are in the top dir!)

   ```
   rails generate controller welcome index goodbye
   ```

4. That created a whole bunch of files (controller and views.) But before we go any further we need to setup the applications home page. We do this by editing the config/routes.rb file. Add the following line:

   ```
   root 'welcome#index'
   ```

5. Note also that it added a couple of GETs to the routes. The routing file basically defines how incoming requests get mapped to controllers in your app. (Give example of this.) We can display all of the routes defined by our app with the rake command:

```
rake routes
```

6. Next, we need to get our app to display something. Let's add hello / good bye messages to our views/welcome/*.erb files for index and goodbye.

7. Now let's try our app out and see how it works. We can start it with the following commandline. Manually load the two pages and demonstrate they are loading. (Note that the -binding option assumes you are running in CodeAnywhere. If you have a local installation of rails you can ignore them!)

```
rails s --binding=0.0.0.0
```

8. Now let's add some dynamic content to our view! Modify welcome/index.html.erb so it does this:

```
Hello World at time=<%= Time.now %>
```

9. However, we need to be careful that we don't put too much code in the views. If we had to internationalize the date/time it could take a lot of logic, and that would be better done in the controller. Let's go ahead and add the following to the index method of our controller:

```
    @curr_time = Time.now
```

10. then modify the index.html.erb as follows:

```
    <h1>Welcome#index</h1>
    <p>Hello World at time=<%= @curr_time %> </p>
```

11. Now let's link our pages together! We can modify our index page to include the following html:

```
    Say <a href="/welcome/goodbye"> goodbye</a>!
```

12. This approach is a bit fragile though. If we moved our app's location on the server filesystem, our links might break. Rails gives us helpers that let us do this much easier:

```
    Say <%= link_to "Goodbye", welcome_goodbye_path %>!
```