Rails Demo 1: Working with Models & Deploying to Heroku

So we've done hello world and it was fun and easy. It turns out that Rails can do a LOT for us right out of the box. Let's take a look at how we can generate database backed web pages.

1. Let's assume we want to create a toy catalog. We begin by generating scaffolding for our Toys.

    ```
        rails generate scaffold Toy name:string description:text manufacturer:string
    price:decimal
    ```

2. This created a whole bunch of stuff:

    - views, controllers ...
    - routes (rake routes)
    - models: however, note that these models are backed by the database (sqlite by default).

3. Take a look in db/migrate - you see a file that creates a table in our db. In order to effect this we must run the following command:

    ```
    rake db:migrate
    ```

4. Notice now that we have a db/scheme.rb. The table has been created. Let's try to load the page. (https://toy-jengelsma.c9users.io/toys). We can now create toy records / add change delete.

Deploying Your Rails App to Heroku

You've worked so hard on your app, and it is now wonderful and ready to change the world --- once you can figure out how to deploy it. Let's take a look at how easy it is to deploy your app to Heroku. Before we do this we first need to push our app out to a repo on github.com.

1. First, go out to github.com (or bitbucket.com) and create a new empty repo.

2. Next, if you haven't done this yet, setup git in your dev environment (Those of you using cloud IDEs will need to do this for sure):

```
git config --global user.name "YOUR NAME"
git config --global user.email "YOUR EMAIL ADDRESS"
git config --global core.editor vim
```

3. When you created your Rails project, Rails already initialized an empty git repo in your project directory; but, you still need to stage / commit your updates. Do this by typing in these commands in the top level directory of your project:

   ```
   git add .
   git commit -m "initial commit"
   ```

4. Make sure SSH is selected on github.com and copy/paste these commands on your dev shell (note the github.com repo details need to be replaced with your own repo info):

   ```
   git remote add origin git@github.com:jengelsma/cautious-eureka.git
   git push -u origin master
   ```

5. Now that we are fully setup with git, it is time to deploy our app to Heroku. First go out and register for a free account on heroku.com.

6. Next, note that Heroku uses a postgres database and our app is using sqlite by default. Update our Gemfile so that it downloads the postgres gem for the production environment Note also that the sqlite3 gem was listed at the global level in the Gemfile by default. Move that line down under the `development` and `test` groups to avoid trying to install it on Heroku when you deploy. It will fail!

```
group :development do
  # Access an interactive console on exception pages or by calling 'console' anyw
here in the code.
  gem 'web-console', '>= 3.3.0'
  gem 'listen', '>= 3.0.5', '< 3.2'
  # Spring speeds up development by keeping your application running in the backg
round. Read more: https://github.com/rails/spring
  gem 'spring'
  gem 'spring-watcher-listen', '~> 2.0.0'
  # Use sqlite3 as the database for Active Record
  gem 'sqlite3'
end

group :test do
  # Adds support for Capybara system testing and selenium driver
  gem 'capybara', '>= 2.15'
  gem 'selenium-webdriver'
  # Easy installation and use of chromedriver to run system tests with Chrome
  gem 'chromedriver-helper'
  # Use sqlite3 as the database for Active Record
  gem 'sqlite3'
end

group :production do
  gem 'pg'
  gem 'rails_12factor'
end
```

7. Next, on Heroku, you will be using the postgres database instead of the default sqlite, so in the `config/database.yml` file you need to change the production database settings to be the following:

```
production:
  adapter: postgresql
  encoding: unicode
  # For details on connection pooling, see Rails configuration guide
  # http://guides.rubyonrails.org/configuring.html#database-pooling
  pool: <%= ENV.fetch("RAILS_MAX_THREADS") { 5 } %>
  database: workspace_production
  username: workspace
  password: <%= ENV['WORKSPACE_DATABASE_PASSWORD'] %>
```

8. Then we need to run `bundle install` to update `Gemfile.lock`. the `--without production` asks bundle to ignore production gems for now:

```
bundle install --without production
```

9. Since we updated `Gemfile` and `Gemfile.lock` we need to commit and push them.

   ```
   git commit -am "updated gem and db for production"
   git push origin master
   ```

10. Now, make sure the Heroku CLI tools are installed in your dev environment.

    ```
    heroku version
    ```

    This is the command to install Heroku in the CodeAnywhere environment

    ```
    wget -O- https://toolbelt.heroku.com/install-ubuntu.sh | sh
    ```

    (This page has instructions on installing Heroku locally: https://devcenter.heroku.com/articles/heroku-cli)

11. Are you having fun yet? Next we need to tell the Heroku toolbelt to login to our Heroku account

    ```
    heroku login -i
    ```

    (The `-i` is for "interactive". Interactive mode is necessary in CodeAnywhere. If you are installing Heroku locally, you can omit the `-i` and log in through your browser.)

12. Then we add our ssh keys to our Heroku account:

    ```
    heroku keys:add
    ```

13. And, of course, we create and rename the app instance from Heroku's craftily generated name to our preferred name. Mine is jre-toys1; you can pick your own and substitute it in the command below.

    ```
    heroku create
    heroku rename jre-toys1
    ```

14. And at long last, we are finally ready to deploy our app to Heroku using a good old fashioned `git push`.

    ```
    git push heroku master
    ```

15. And to cap off our amazing Heroku party, nothing beats a satisfying `rake db:migrate`. If you don't do this, your app is not going to work because your DB scheme will not be up-to-date!

```
heroku run rake db:migrate
```

16. At this point, you should be able to point your browser to your Heroku url (you can find that from the heroku portal or the command below) and be amazed at the ease in which you can now deploy a Rails app.

    ```
    heroku info -s | grep web_url
    ```