Lab 8.2: Rails and MongoDB
IN705 Databases Three

Introduction

In the last lab we installed and tried out MongoDB, a *document database*. In this lab we will convert our Rails application to use MongoDB rather than a relational database as it does now. The remarkable thing is how little we actually have to do.

1 Create a new Git branch

We want to work on our new version of splatter without interfering with development of our original relational database version. We will create a *branch* in our Git repository to support this.

Make a new working directory your home direstory on your EC2 server. Inside that directory, execute the following

```
git clone git@github.com:<your username>/db3.git
cd db3
git checkout -b mongo
```

Inside your `splatter` directory, edit your `Gemfile`, adding the follwoing lines:

```
gem 'mongoid', github: 'mongoid/mongoid'
gem 'bson_ext'
```

Then, commit your changes on the new branch.

```
git add .
git commit -m "added deps for mongo to Gemfile"
git push origin mongo
```

Check the Github web site to verify that your new branch is present.

2 Reconfigure your Rails application

We need to make a some configuration changes to our Rails application. Carry out the following tasks in your `splatter` directory.

First, update your Ruby gems by running `bundle update`.

Next, generate your MongoDB configuration file with the command `rails g mongoid:config`. This will write the file `config/mongoid.yml`. It's not necessary to make any changes right now, but you should take a minute to look at this file.

Edit the file `config/application.rb`. Comment out the line that says `require 'rails/all'` and add the following lines:

```
require "action_controller/railtie"
require "action_mailer/railtie"
require "sprockets/railtie"
```

Edit the file `config/environments/development.rb`, commenting out the line

```
config.active_record.migration_error = :page_load
```

3 Modify the models

Since the model classes are the part of our code that interacts with the database, we need to change those classes.

First, remove the original user.rb file and replace it with a new one:

```
rm app/models/user.rb
rails g model user name:string email:string password:string blurb:string
```

Take a look at the new model and note the differences. Since a MongoDB database is schemaless, we have to specify our data fields in the model.

We don't need the splatt model file for now, so remove it from the your repository with the command

```
git rm app/model/splatt.rb
```

Now try running you server with the command `rails server`.

4 Test your application

You can use curl to test your user CRUD functions:

```
 curl -H "Content-type: application/json" -X POST http://localhost:3000/users \
        -d '{"name": "Mongo User", "email":"a@b.c", "password":"foo", "blurb":"bar"}'

 curl -H "Content-type: application/json" -X GET http://localhost:3000/users
```

Note the id of your new user and substitute it in the requests below..

```
curl -H "Content-type: application/json" -X GET http://localhost:3000/users/540f3f736269623441000000

curl -H "Content-type: application/json" -X PUT http://localhost:3000/users/540f3f736269623441000000 \
        -d '{"name":"Bob"}'
```

Once you have tested your application, commit and push your changes.