Object Oriented WEB Programming Using Ruby

Day 8: 01/June/2017

Login User Registration

Changes: Removal of LESS

I guess we are not going to use LESS, so I have removed

gem ‘less-rails’

Add login authentification

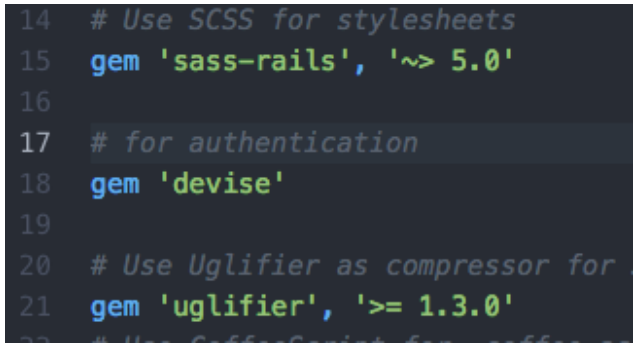
Today, we add the authentication.

We use devise, and we follow the same procedure we introduced last week.

Open Gemfile, add gem ‘devise,

Then, bundle install.

Try this on your trial project.



Install Devise to the project

Type the following command,

rails generate devise:install

to install devise to your project, at the project root directory.

```
[root@cisnote Chirpy]# rails generate devise:install
      create  config/initializers/devise.rb
      create  config/locales/devise.en.yml
===============================================================================

Some setup you must do manually if you haven't yet:

  1. Ensure you have defined default url options in your environments files. Her
e
     is an example of default_url_options appropriate for a development environm
ent
     in config/environments/development.rb:

       config.action_mailer.default_url_options = { host: 'localhost', port: 300
0 }
```

Read the message from the system carefully

```
===============================================================================

Some setup you must do manually if you haven't yet:

  1. Ensure you have defined default url options in your environments files. Here
     is an example of default_url_options appropriate for a development environment
     in config/environments/development.rb:

       config.action_mailer.default_url_options = { :host => 'localhost:3000' }

     In production, :host should be set to the actual host of your application.

  2. Ensure you have defined root_url to *something* in your config/routes.rb.
     For example:

       root :to => "home#index"

  3. Ensure you have flash messages in app/views/layouts/application.html.erb.
     For example:

       <p class="notice"><%= notice %></p>
       <p class="alert"><%= alert %></p>

  4. If you are deploying on Heroku with Rails 3.2 only, you may want to set:

       config.assets.initialize_on_precompile = false

     On config/application.rb forcing your application to not access the DB
     or load models when precompiling your assets.

  5. You can copy Devise views (for customization) to your app by running:

       rails g devise:views

===============================================================================
```

3 steps to use devise

1. Ensure you have defined default url options in your your environments files.

   cancels if it causes an error!

2. Ensure you have defined root_url to *something* in your config/routes.rb.

3. Ensure you have flash messages in app/views/layouts/application.html.erb.

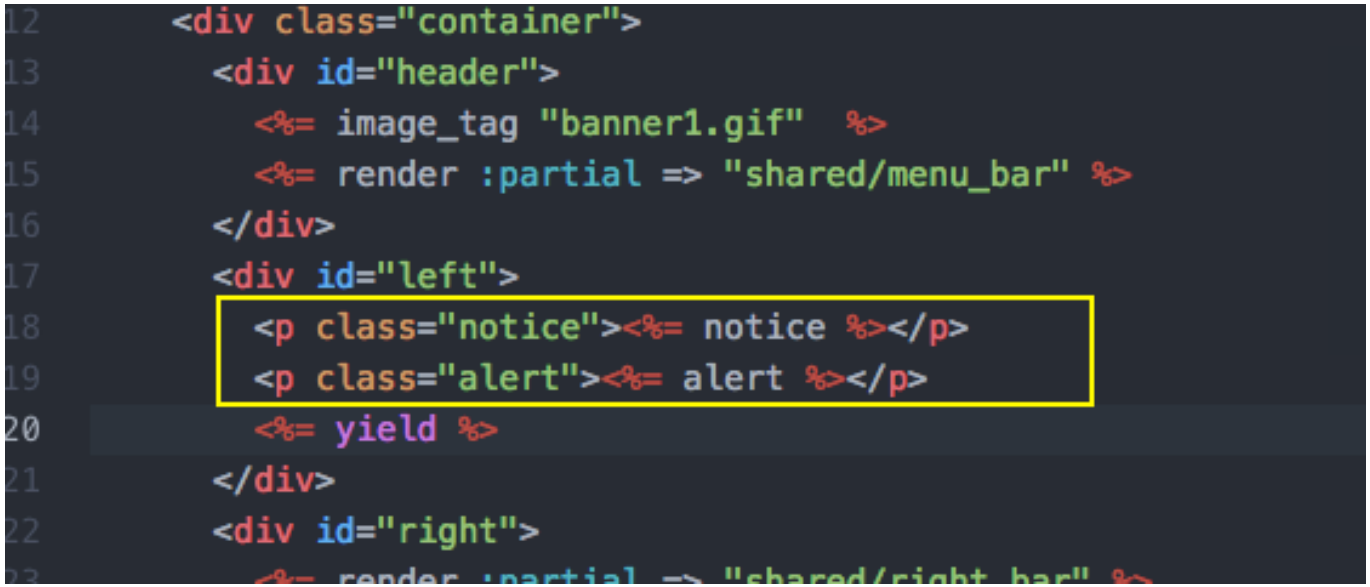
Add two lines for login result message display (Step 3)

- Modify

(project)/app/views/layouts/application.html.erb

- Add the following two lines before <%= yield %>

```
<p class="notice"><%= notice %></p>
<p class="alert"><%= alert %></p>
```



View for devise

- Here we generate views for devise. Type

  rails generate devise:views

```
                :ScopsOwl                $ rails generate devise:views
Running via Spring preloader in process 5891
Expected boolean default value for '--markerb'; got :erb (string)
      invoke  Devise::Generators::SharedViewsGenerator
      create    app/views/devise/shared
      create    app/views/devise/shared/_links.html.erb
      invoke  form_for
      create    app/views/devise/confirmations
      create    app/views/devise/confirmations/new.html.erb
      create    app/views/devise/passwords
      create    app/views/devise/passwords/edit.html.erb
      create    app/views/devise/passwords/new.html.erb
      create    app/views/devise/registrations
      create    app/views/devise/registrations/edit.html.erb
      create    app/views/devise/registrations/new.html.erb
      create    app/views/devise/sessions
      create    app/views/devise/sessions/new.html.erb
      create    app/views/devise/unlocks
      create    app/views/devise/unlocks/new.html.erb
      invoke  erb
      create    app/views/devise/mailer
      create    app/views/devise/mailer/confirmation_instructions.html.erb
      create    app/views/devise/mailer/email_changed.html.erb
      create    app/views/devise/mailer/password_change.html.erb
      create    app/views/devise/mailer/reset_password_instructions.html.erb
      create    app/views/devise/mailer/unlock_instructions.html.erb
```

User model

User model for authentication could be used to register the user.

So, this can contain the handle name, real name, and such. Add the necessary attributes for it. If you think you do not need to keep the real name for your system, use the following sample.

rails generate scaffold user email:string handle:string

```
[root@cisnote chirpy]# rails generate devise user handle_name:string
      invoke  active_record
      create    db/migrate/20131128171402_devise_create_users.rb
      create    app/models/user.rb
      invoke    test_unit
      create      test/models/user_test.rb
      create      test/fixtures/users.yml
      insert    app/models/user.rb
       route  devise_for :users
[root@cisnote chirpy]#
```

What is Scaffold?

Necessary files for DB maintenance.

Generated Files

```
                :ScopsOwl                    $ rails generate scaffold user email:s
tring handle:string
Running via Spring preloader in process 5918
      invoke  active_record
      create    db/migrate/20170526050804_create_users.rb
      create    app/models/user.rb
      invoke    test_unit
      create      test/models/user_test.rb
      create      test/fixtures/users.yml
      invoke  resource_route
       route    resources :users
      invoke  scaffold_controller
      create    app/controllers/users_controller.rb
      invoke    erb
      create      app/views/users
      create      app/views/users/index.html.erb
      create      app/views/users/edit.html.erb
      create      app/views/users/show.html.erb
      create      app/views/users/new.html.erb
      create      app/views/users/_form.html.erb
      invoke    test_unit
      create      test/controllers/users_controller_test.rb
      invoke    helper
      create      app/helpers/users_helper.rb
      invoke      test_unit
      invoke    jbuilder
      create      app/views/users/index.json.jbuilder
      create      app/views/users/show.json.jbuilder
      create      app/views/users/_user.json.jbuilder
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/users.coffee
      invoke    scss
      create      app/assets/stylesheets/users.scss
      invoke  scss
      create    app/assets/stylesheets/scaffolds.scss
```

Migration

Every time you generated database tables, you need to migrate, to create tables.

rake db:migrate

```
                    :ScopsOwl                    $ rake db:migrate
== 20170526050804 CreateUsers: migrating ======================================
-- create_table(:users)
   -> 0.0118s
== 20170526050804 CreateUsers: migrated (0.0120s) =============================
```

By migrating, what happened?

Database tables are generated.

```
[redacted]:ScopsOwl [redacted]$ cd db
[redacted]:db [redacted]$ ls
development.sqlite3     schema.rb
migrate                 seeds.rb
[redacted]:db [redacted]$ sqlite3 development.sqlite3
SQLite version 3.18.0 2017-03-28 18:48:43
Enter ".help" for usage hints.
sqlite> .schema
CREATE TABLE IF NOT EXISTS "schema_migrations" ("version" varchar NOT NULL PRIMARY KEY);
CREATE TABLE IF NOT EXISTS "ar_internal_metadata" ("key" varchar NOT NULL PRIMARY KEY, "
value" varchar, "created_at" datetime NOT NULL, "updated_at" datetime NOT NULL);
CREATE TABLE IF NOT EXISTS "users" ("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "em
ail" varchar, "handle" varchar, "created_at" datetime NOT NULL, "updated_at" datetime NO
T NULL);
sqlite> .exit
[redacted]db [redacted]$
```

How to see table conditions

Type command:

rake db:migrate:status

```
[redacted]:db [redacted]$ rake db:migrate:status
(in /Users/[redacted]/Documents/atom.workspace/ScopsOwl)

database: /Users/[redacted]/Documents/atom.workspace/ScopsOwl/db/development.sqlite3

 Status   Migration ID    Migration Name
--------------------------------------------------
   up     20170526050804  Create users

[redacted]:db [redacted]$
```

Do not remove tables manually

Schema_migrations table is maintained by rails.

So, if you remove db table manually, rails will fail to find the table which is registered in the maintenance record, to cause an error.

So, how to cancel table creation

Cancel command of rails generation is "destroy." I.e., to cancel "rails generate scaffold user email:string handle:string, you need to type;

```
rails destroy scaffold user
```

Before you remove the scaffold, you need to type;

```
rake db:rollback
```

to let rails remove the table.

Use ‘user’ table for authentication

Generate devise to manage the login authentication for ‘users’ table

rails generate devise user

```
[redacted]:db [redacted]$ rails generate devise user
Running via Spring preloader in process 6281
      invoke  active_record
      create    db/migrate/20170526053634_add_devise_to_users.rb
      insert    app/models/user.rb
       route  devise_for :users
```

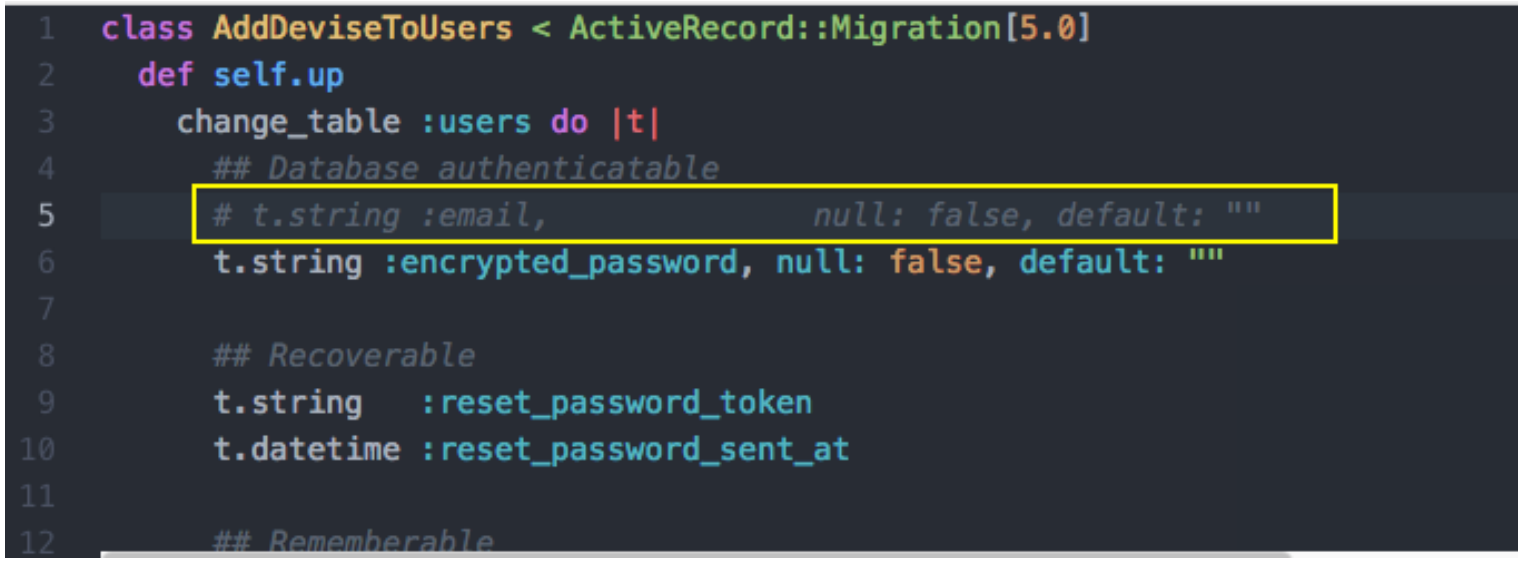
Modify migration file

Remove the following line

t.string :email, null: false, default: “”

in the file:

db/migrate/
2017xxxx_add_devise_to_users.rb



Check migration file, and migrate

Type

rake db:migrate

```
[redacted]:db [redacted]$ rake db:migrate
(in /Users/[redacted]/Documents/atom.workspace/ScopsOwl)
== 20170526053634 AddDeviseToUsers: migrating ================================
-- change_table(:users)
   -> 0.0144s
-- add_index(:users, :email, {:unique=>true})
   -> 0.0968s
-- add_index(:users, :reset_password_token, {:unique=>true})
   -> 0.0036s
== 20170526053634 AddDeviseToUsers: migrated (0.1161s) =======================
```

Request to login

Add the following line in ChatController

```
before_action :authenticate_user! only: [:index]
```

Authenticate method is provided by devise gem. "user" is the model name which is assigned for the authentication control.

So, if you created member table, and "rails generate devise member" the method name should be :authenticate_member!

```
1  class ChatController < ApplicationController
2    before_action :authenticate_user!, only: [:index]
3    def index
4    end
5  end
```

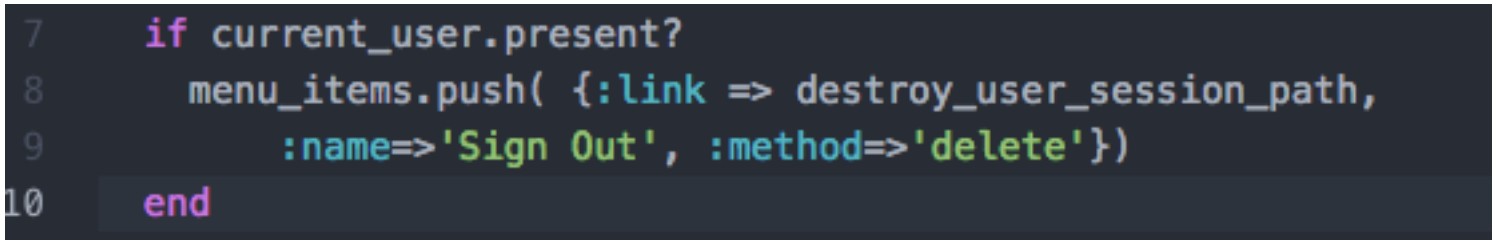
Sign Out path

Check by "rake routes"

You will find destroy_user_session_path in the routing list.

In _menu_bar.html.erb, three lines had been added.

```
if current_user.present?
  menu_items.push( {:link => destroy_user_session_path,
    :name=>'Sign Out', :method=>'delete'})
end
```



Apply bootstrap to Users

The following command will automacitaly apply bootstrap classes to the scaffold.

```
rails g bootstrap:themed Users
```

See the following page;

http://getbootstrap.com/components/

Bootstrap Arrange

Sample for menu_bar

```
12  <div class="btn-group">
13    <% menu_items.each_with_index do | item, index | -%>
14      <div class="btn btn-default">
15          <%= menu_link_to item -%>
16      </div>
17    <% end -%>
18  </div>
```

Result of Scaffold generation

CRUD of database are all ready for Users

Class name : User

Action Mailer

Generate Sign up mailer

```
rails g mailer SignupMailer
```

Then manually register

signup_email.html.erb, and

signup_email.text.erb

Signup Mailer

Mailers / signup_mailer.rb is generated.

Modify as the following

```
1  class SignupMailer < ApplicationMailer
2    default from: 'webdb.hosei@signalysis.co.jp'
3
4    def signup_email( user, url )
5      @user = user
6      @url  = url
7      mail(to: @user.email, subject: 'Register your Handle to ScapsOwl Project')
8    end
9  end
10
```

Now design your own portal

How many screens do you write?

What kind of data do you need to store?

Think about the screen links.

Next presentation:

- Screen proto-types,
- Database Schema