



# Authentication with Devise

Juan “Harek” Urrios  
@xharekx33

# What is “authentication”

The process of determining whether someone or something is who/what it claims to be.

Authentication deals with establishing WHO you are.



# What is Devise

Devise is an authentication solution for Rails:

- Modular: 10 different modules.
- Flexible: Only use what you need.



# Devise's 10 modules

- Authenticable: encrypts and stores a password in the database.
- Omniauthable: adds OmniAuth support.
- Confirmable: sends emails with confirmation instructions.
- Recoverable: resets the user password.
- Registerable: handles signing up users + editing & destroying their accounts.

...



# Devise's 10 modules

...

- Rememberable: handles remembering the user from a saved cookie.
- Trackable: tracks sign in count, timestamps and IP address.
- Timeoutable: expires sessions after X time with no activity.
- Validatable: provides validations of email and password.
- Lockable: locks an account after X number of failed sign-in attempts.



# How to install Devise

Add gem, run installation and run the devise initializer generator.

```
$ echo "gem 'devise'" >> Gemfile
```

```
$ bundle install
```

```
$ rails generate devise:install
```



@xharekx33

# Add Devise to your User model

Run the generator for the model

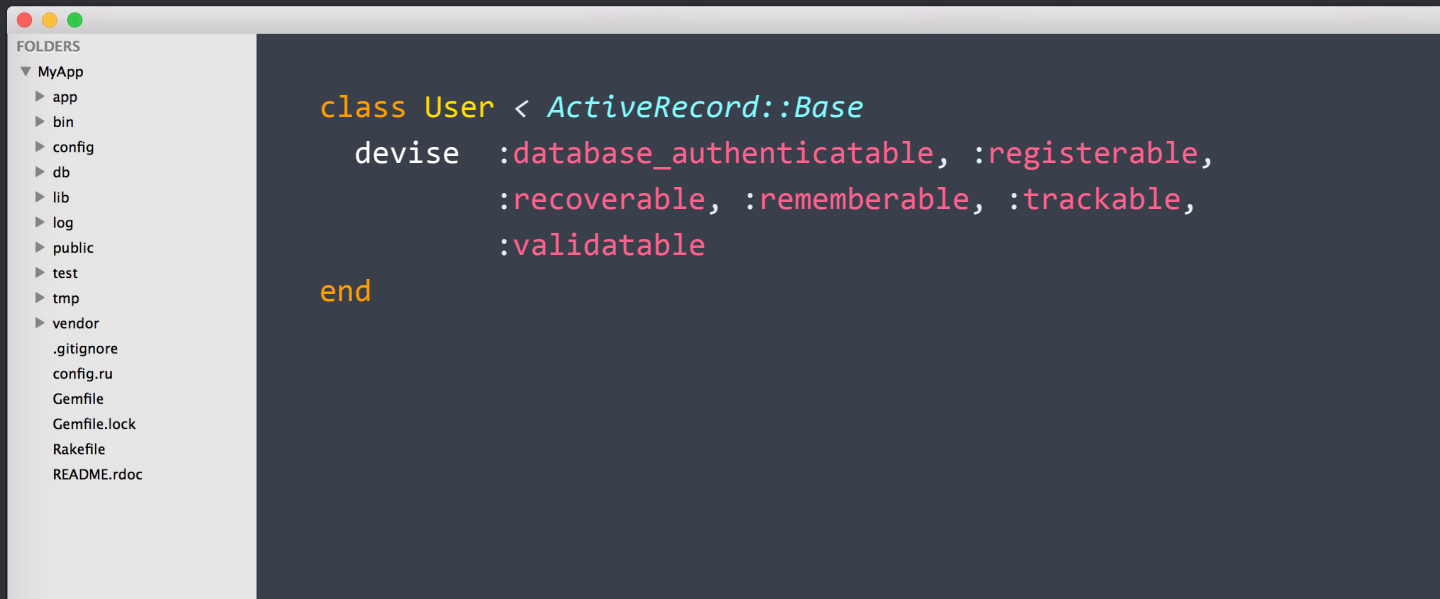
```
$ rails generate devise User
  invoke active_record
  create db/migrate/20141109155703_add_devise_to_users.rb
  insert app/models/user.rb
  route devise_for :users
```



@xharekx33

# The new and improved User Model

The new User model includes a call to the devise method, with different modules as parameters. Add or remove them as needed.

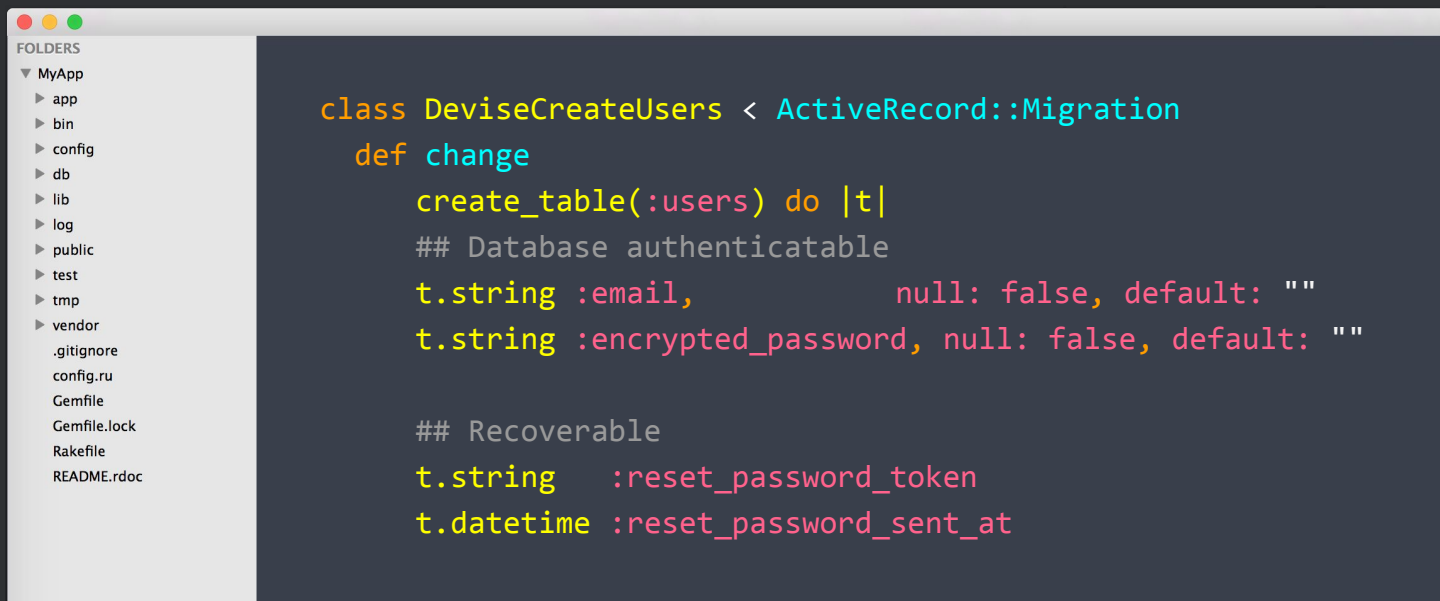


@xharekx33



# The migration file

Depending on the modules you want to use, you might have to comment or uncomment sections of it.



The image shows a code editor window with a sidebar on the left displaying a file tree under the heading 'FOLDERS'. The tree includes 'MyApp' with subfolders 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', and 'vendor', as well as files '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'. The main editor area contains the following Ruby code:

```
class DeviseCreateUsers < ActiveRecord::Migration
  def change
    create_table(:users) do |t|
      ## Database authenticatable
      t.string :email,           null: false, default: ""
      t.string :encrypted_password, null: false, default: ""

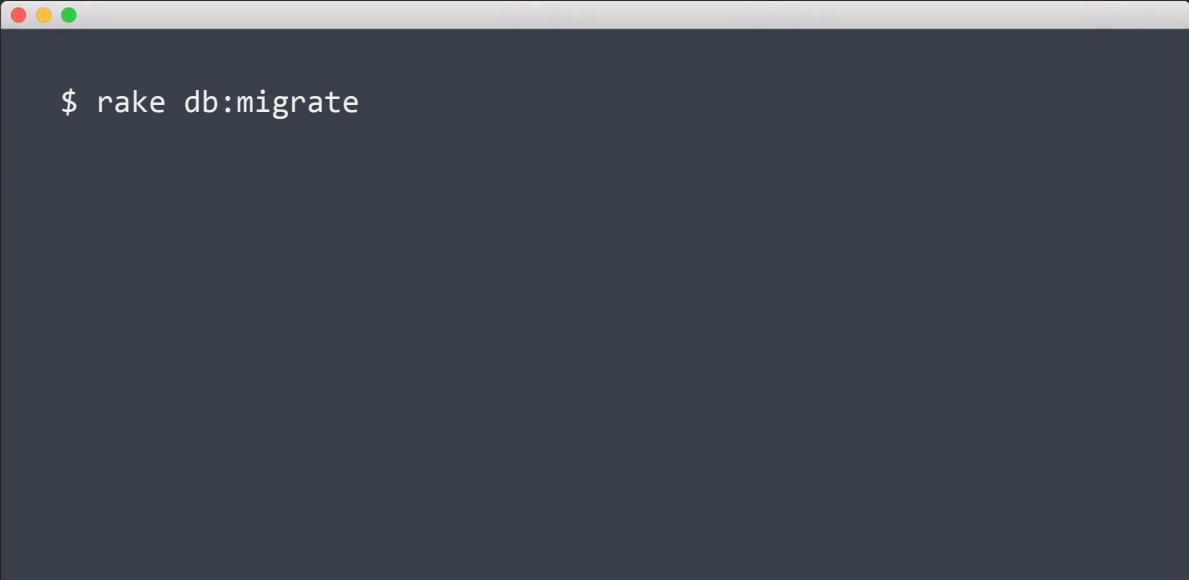
      ## Recoverable
      t.string   :reset_password_token
      t.datetime :reset_password_sent_at
```



@xharekx33

# Apply the migration

Once any necessary changes to the Model and the migration are done run the rake db:migrate command

A terminal window with a dark blue background and a light gray title bar. The title bar has three colored window control buttons (red, yellow, green) on the left. The terminal displays the command `$ rake db:migrate` in a white monospaced font.

```
$ rake db:migrate
```

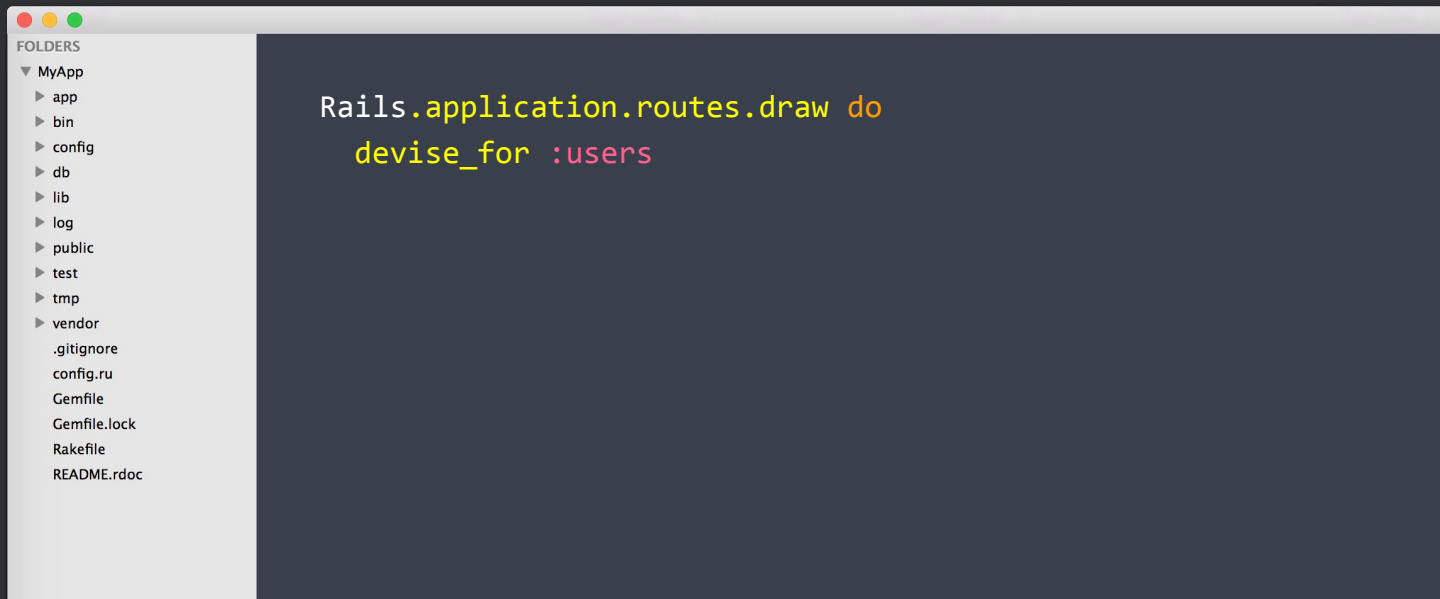


@xharekx33

# The routes file

---

The routes file now includes a line to manage three resources: Sessions, Passwords and Registrations



@xharekx33

# Routes

Check the new available routes with the `rake routes` command.

```
$ rake routes
```

Prefix	Verb	URI Pattern	Controller#Action
new_user_session	GET	/users/sign_in(.:format)	devise/sessions#new
user_session	POST	/users/sign_in(.:format)	devise/sessions#create
destroy_user_session	DELETE	/users/sign_out(.:format)	devise/sessions#destroy
user_password	POST	/users/password(.:format)	devise/passwords#create
new_user_password	GET	/users/password/new(.:format)	devise/passwords#new
edit_user_password	GET	/users/password/edit(.:format)	devise/passwords#edit
	PATCH	/users/password(.:format)	devise/passwords#update
	PUT	/users/password(.:format)	devise/passwords#update

....



@xharekx33

# Customizing views

---

These routes already display pages but they are very basic. To customize them simply run **rails generate devise:views**

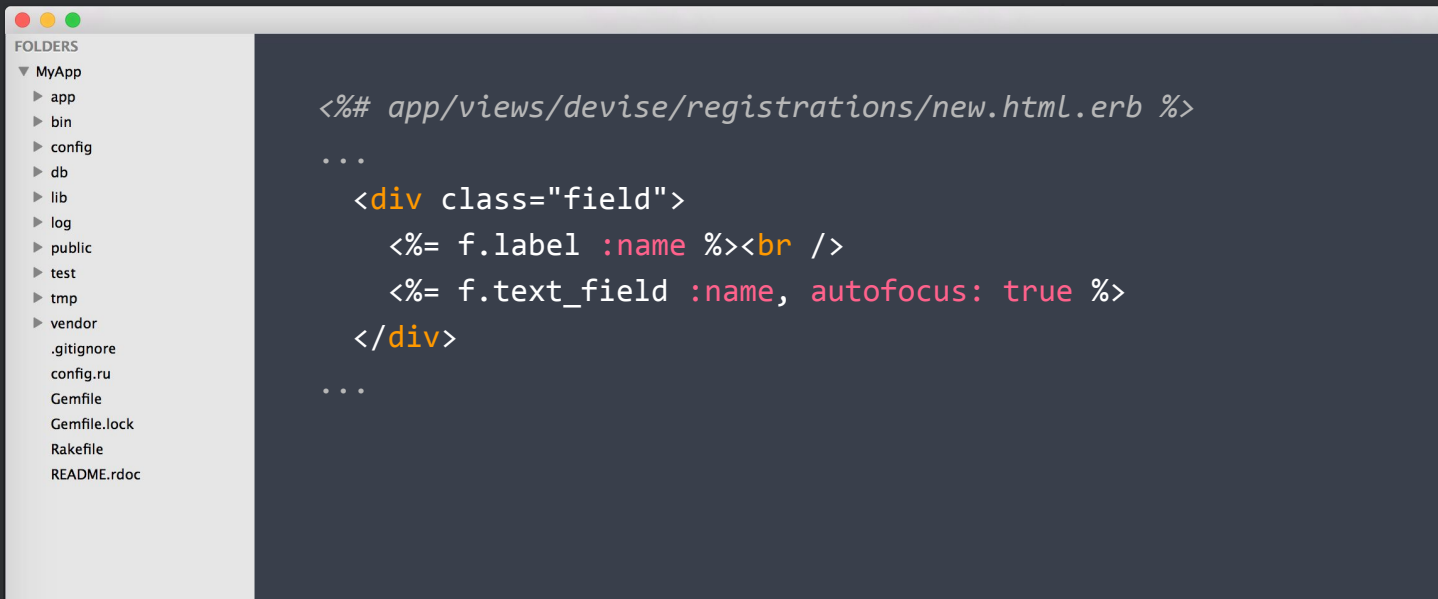
```
$ rails generate devise:views
  invoke  Devise::Generators::SharedViewsGenerator
  create   app/views/devise/shared
  create   app/views/devise/shared/_links.html.erb
  invoke  form_for
  create   app/views/devise/confirmations
  create   app/views/devise/confirmations/new.html.erb
  create   app/views/devise/passwords
  create   app/views/devise/passwords/edit.html.erb
  create   app/views/devise/passwords/new.html.erb
  create   app/views/devise/registrations
```



@xharekx33

# Add fields to the Sign up form

Modify the form for registrations and add a new field



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like app, bin, config, db, lib, log, public, test, tmp, and vendor, and files like .gitignore, config.ru, Gemfile, Gemfile.lock, Rakefile, and README.rdoc. The code editor displays the content of the file `app/views/devise/registrations/new.html.erb`. The code is an ERb template snippet for a registration form. It starts with a comment `<%= # app/views/devise/registrations/new.html.erb %>`, followed by three dots. Then, it opens a `<div class="field">` tag. Inside the div, there are two lines of code: `<%= f.label :name %><br />` and `<%= f.text_field :name, autofocus: true %>`. The div is closed with `</div>`, followed by three dots.

```
<%= # app/views/devise/registrations/new.html.erb %>
...
<div class="field">
  <%= f.label :name %><br />
  <%= f.text_field :name, autofocus: true %>
</div>
...
```



@xharekx33

# What's happened?

Devise took care of everything and created a new user and logged you in. But there is a problem...

```
$ rails console
  Loading development environment (Rails X.X.X)
  X.0.0-pXXX :001 > User.last
    User Load (0.3ms)  SELECT  "users".* FROM "users"   ORDER BY
"users"."id" DESC LIMIT 1
  => #<User id: 2, email: "joe@joe.com", encrypted_password:
"$2a$10$mJrlbfd2FRd11QiqWPVB4.IHWD1TgEHPeoMBaiyato...",
reset_password_token: nil, reset_password_sent_at: nil,
remember_created_at: nil, sign_in_count: 1, current_sign_in_at: "2015-07-
31 19:47:49", last_sign_in_at: "2015-07-31 19:47:49", current_sign_in_ip:
"127.0.0.1", last_sign_in_ip: "127.0.0.1", created_at: "2015-07-31 19:47:
49", updated_at: "2015-07-31 19:47:49", name: nil>
```



@xharekx33

# Strong parameters

There three actions in Devise that require parameters to be sanitized. By default they accept:

- `sign_in:` email
- `sign_up:` email, password, password\_confirmation
- `account_update:` email, password, password\_confirmation, current\_password

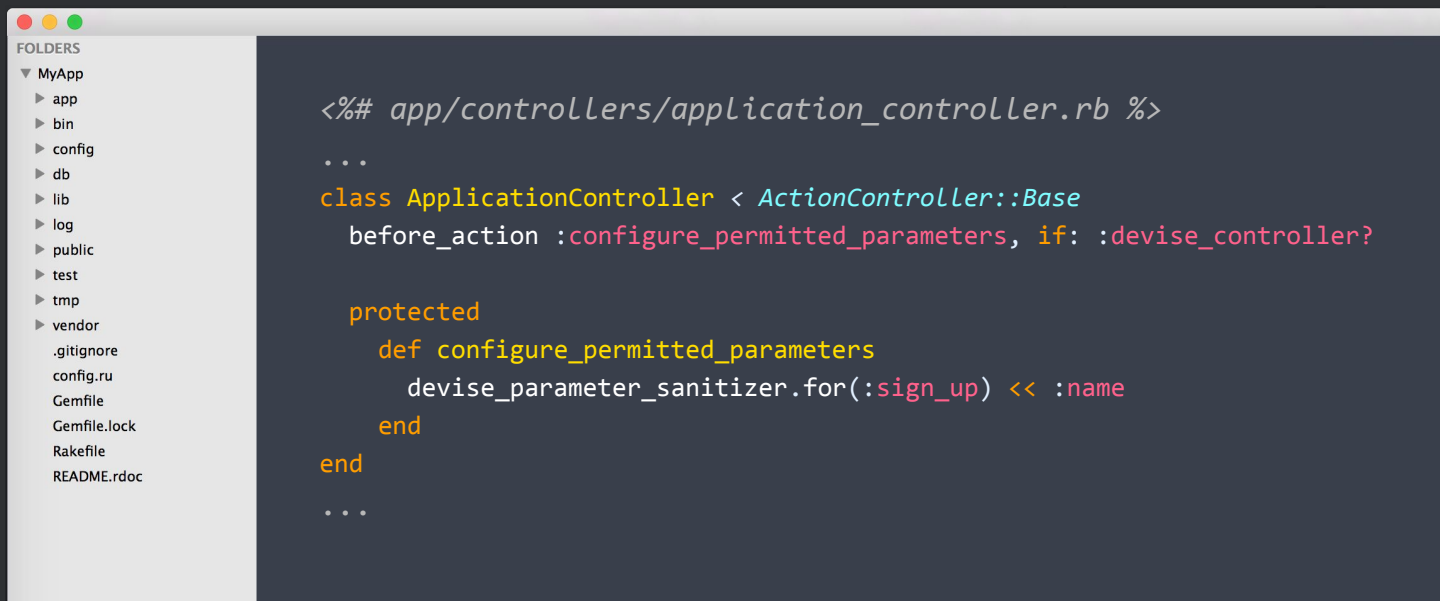
To accept other parameters sanitization will have to be added to the controller





# Sanitize the user name

Add the user's name to the list of permitted parameters



The image shows a code editor window with a sidebar on the left displaying the project structure under 'FOLDERS'. The main area contains a Ruby code snippet for a Rails application controller.

```
FOLDERS
▼ MyApp
  ► app
  ► bin
  ► config
  ► db
  ► lib
  ► log
  ► public
  ► test
  ► tmp
  ► vendor
  .gitignore
  config.ru
  Gemfile
  Gemfile.lock
  Rakefile
  README.rdoc
```

```
<%= app/controllers/application_controller.rb %>

...
class ApplicationController < ActionController::Base
  before_action :configure_permitted_parameters, if: :devise_controller?

  protected
    def configure_permitted_parameters
      devise_parameter_sanitizer.for(:sign_up) << :name
    end
end

...
```



@xharekx33

# Controller filters & helpers

Until now, our whole app is the same for everybody. No access control.

Devise gives us methods to control flow in controllers:

- `before_action :authenticate_user!`

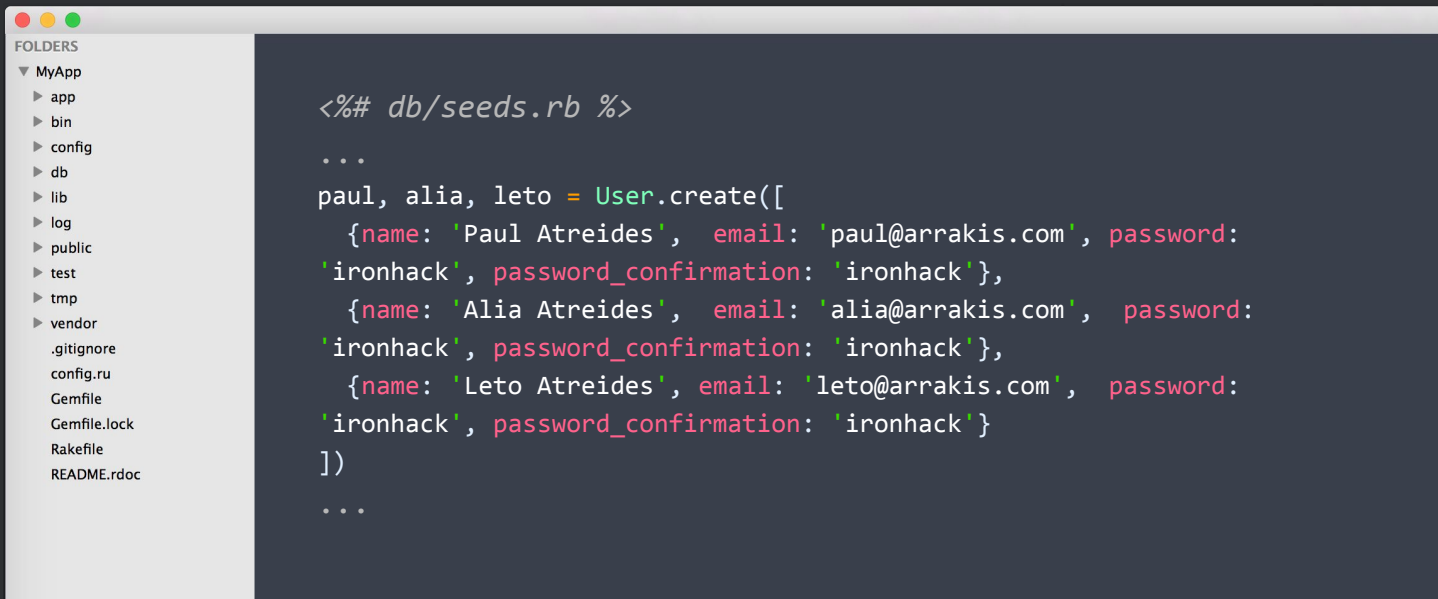
and views:

- `user_signed_in?`
- `current_user`
- `user_session`



# Seed database

Let's add some users to the seed file and run `rake db:seed`



The screenshot shows a code editor with a sidebar on the left displaying a file tree under the name 'FOLDERS'. The tree includes 'MyApp' with subfolders 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', and 'vendor', as well as files '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'. The main editor area shows the content of 'db/seeds.rb' with the following code:

```
<%= db/seeds.rb %>

...
paul, alia, leto = User.create([
  {name: 'Paul Atreides', email: 'paul@arrakis.com', password:
'ironhack', password_confirmation: 'ironhack'},
  {name: 'Alia Atreides', email: 'alia@arrakis.com', password:
'ironhack', password_confirmation: 'ironhack'},
  {name: 'Leto Atreides', email: 'leto@arrakis.com', password:
'ironhack', password_confirmation: 'ironhack'}
])

...
```



# Users controller

Generate the necessary User controller where the profile action will go.

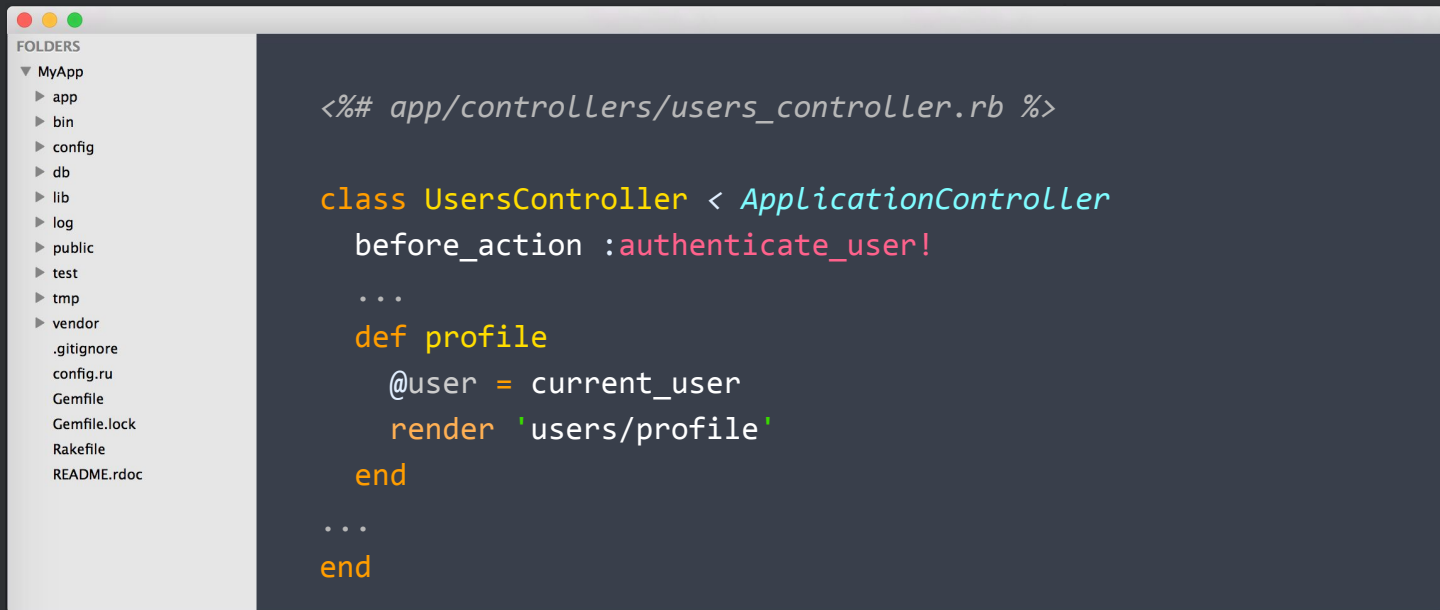
```
$ rails g controller users --skip-test-unit
create  app/controllers/users_controller.rb
       invoke  erb
       create  app/views/users
       invoke  helper
       create  app/helpers/users_helper.rb
       invoke  assets
       invoke  coffee
       create  app/assets/javascripts/users.js.coffee
       invoke  scss
       create  app/assets/stylesheets/users.css.scss
```



@xharekx33

# Set up controller

Make sure users are logged in and set up the new profile action



The image shows a code editor window with a sidebar on the left displaying a file tree under the heading 'FOLDERS'. The tree includes 'MyApp' with subfolders 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', and 'vendor'. Below these are files: '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'. The main editor area contains Ruby code for a Rails controller. The code starts with a comment line, followed by a class definition for 'UsersController' inheriting from 'ApplicationController'. Inside the class, there is a 'before\_action' call to ':authenticate\_user!', followed by three dots indicating more code. Then, a 'def profile' method is defined, which sets '@user = current\_user' and calls 'render :users/profile'. The method ends with 'end', followed by three dots, and the class ends with 'end'.

```
<%= app/controllers/users_controller.rb %>

class UsersController < ApplicationController
  before_action :authenticate_user!

  ...

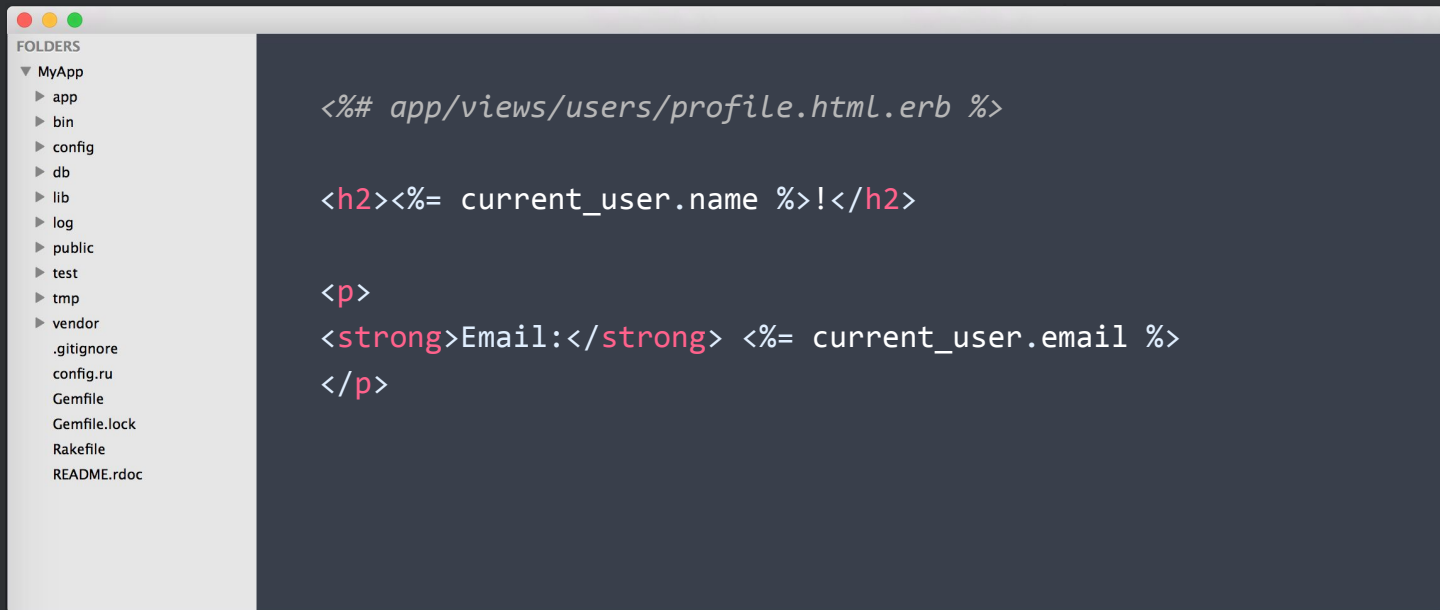
  def profile
    @user = current_user
    render :users/profile
  end

  ...
end
```



# Profile page

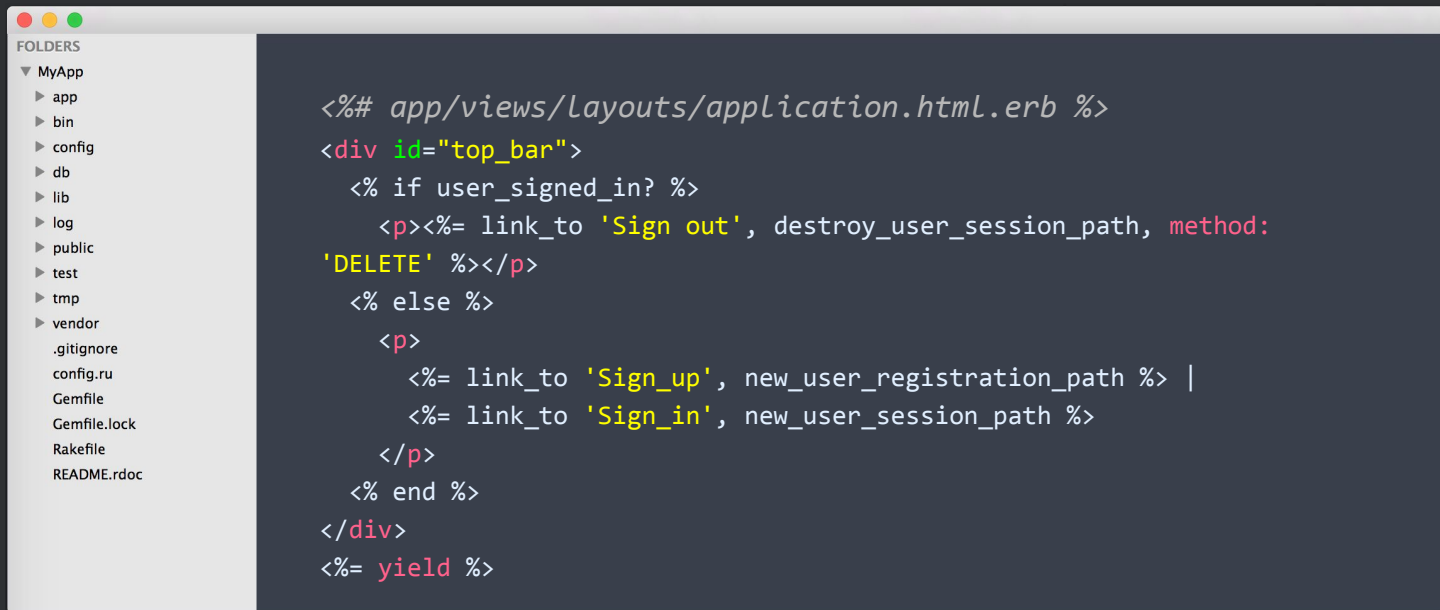
Create the corresponding view for the profile action



@xharekx33

# Application layout

Add links to the layout so users can sign up, sign in and sign out

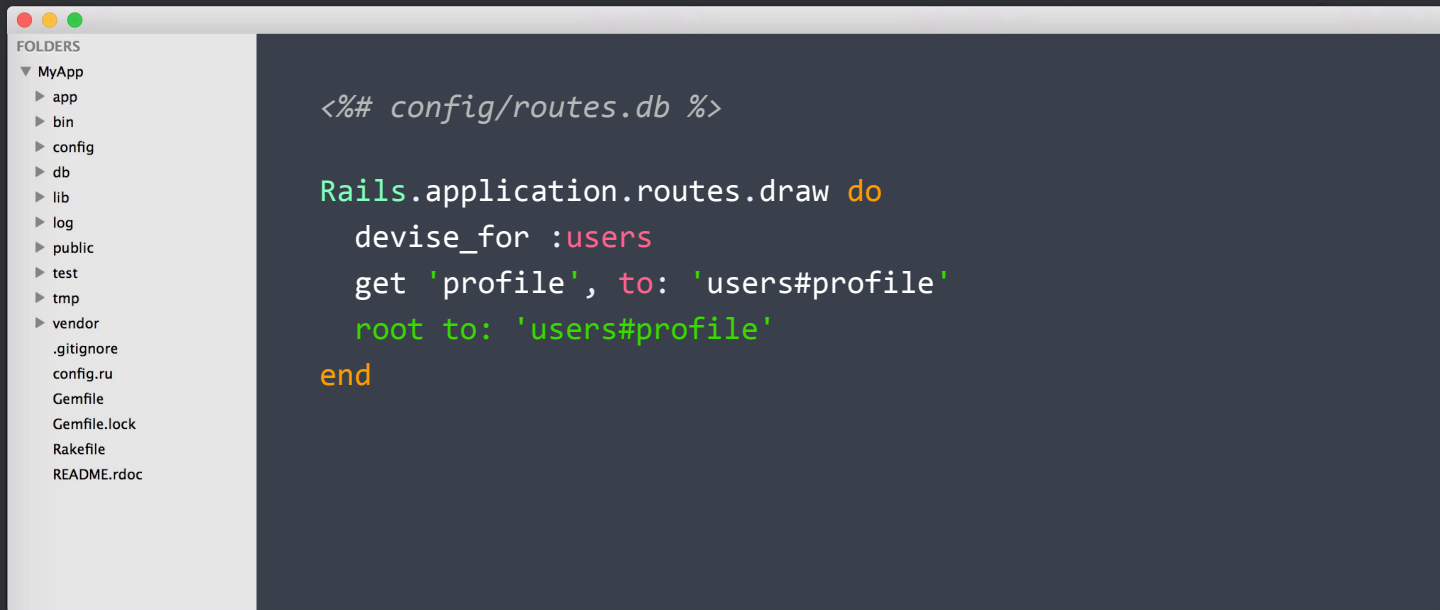


@xharekx33

# Add a route for the profile action

---

Add the required route to be able to access this new action



The image shows a code editor window with a sidebar on the left displaying a file tree under the name 'FOLDERS'. The tree includes a 'MyApp' directory with sub-items: 'app', 'bin', 'config', 'db', 'lib', 'log', 'public', 'test', 'tmp', 'vendor', '.gitignore', 'config.ru', 'Gemfile', 'Gemfile.lock', 'Rakefile', and 'README.rdoc'. The main editor area displays the content of 'config/routes.rb'. The code defines a new route for the 'profile' action within the 'users' namespace.

```
<%=# config/routes.rb %>

Rails.application.routes.draw do
  devise_for :users
  get 'profile', to: 'users#profile'
  root to: 'users#profile'
end
```



@xharekx33



# Check how it works

Now, if you log in using any of the users we created (or create a new one via the registration page) you should be taken to the profile page, that will display the current user name and email.

At the top there will be a link to Sign out

Sign out and you'll be taken to the Sign in page. If you try to access the profile page without signing back in, Devise will kick you out because only logged users can access the User profile action.

