Chapter 10 Updating, showing and deleting users

Editing users is like creating users but instead of the “new” action, we use the “edit” action.

To turn the mockup into a working page we need to fill in both the Users controller action and the user edit view.

NOTE: The code from `app/views/users/edit.html.erb`

Is exactly the same as `app/views/users/new.html.erb`

Rails knows which one to use because it knows whether a user is new or already exists in the database.

After changing the Users controller action and the user edit view we need to fill in the URL of the settings link in the site navigation using:

```
<%= link_to "Settings", edit_user_path(current_user) %>.
```

Unsuccessful edits

Unsuccessful edits are much like unsuccessful signups. In this case we create an “update” action. Which uses update_attributes to update the user based on the submitted params hash.

We create a test for unsuccessful edits by the command:

```
$ rails generate integration_test users_edit
```

Successful edits

To get the edit test green we need to edit the /users_edit_test.rb

And the app/controllers/users_controller.rb and we need to change the app/model/user.rb to accept empty passwords.

Authorization

In the context of web applications, authentication allows us to identify users of our site.

Using the before filter in the user controller, we can make sure that a user is logged in before accessing the page.

By default, before filters apply to *every* action in a controller, so here we restrict the filter to act only on the :edit and :update actions by passing the only: potions hash.

Requiring the right user

First we need to add a new user. Now we can log in as by editing the test/controllers/users_controller_test.rb.

To redirect users trying to edit another user's profile, we can add a method called "correct_user". And using another before filter. We do this by editing the app/controller/users_contoller.rb.

Friendly forwarding

When a user is trying to access a page, they are not redirected to where they want to go. This section will redirect them to where they want to go.

To direct users to the intended place, we need to find where they want to go and store it, then redirect to that location instead of the default. We do this by editing the app/helpers/session_helper.rb.

Then we need to add that location to the …/users_controller.rb. THEN we need to redirect back using the "redirect_back_or user" command. In the sessions_controller.rb.

All users/ Users index

To start with the users index we need to first implement a security model.

To display the users, we need to make a variable containing all the site's users and render each one by iterating through them in the index view. We use User.all by editing the …/users_controller.rb

Then we make a view app/views…index.html.erb. Then add code to the scss. Then add to the …./_header.html.erb

Sample users

In this section we will git our sample user some company (See what I did there… git).

If you haven't do so already from chapter 3 or 4, you will need to add to the gemfile, I already did earlier, so I'm not going over it again.

First we need to add a Ruby program to seed the database with sample users using the db/seeds.rb. The tutorial says the code is advanced, so don't worry about it, so I'm not.

Yeah! That code we didn't worry about just created a bunch of new users.

To apply the "don't worry "code, you need to

$ rails db:migrate:reset Then

$ rails db:seed.

Now we have 100 users!

Pagination

Now there are too many users and they all show up on the same page, we will now only put 30 per page. We do this by adding more to the Gemfile, which, I already did, so I'm not going over it again.

We need to add the index to the app/views/users/index.html.erb

Now we have 30 per page!

Administrative users

We will use an admin Boolean for this. First we use the :$ rails generate migration add_admin_to_users admin:Boolean

We add this to the db/migrate/[ts]_add_admin_to_user.rb. we set the Boolean to false(we don't need to but if we don't, it will be nil, which is false anyway)

Then we do a $ rails db:migrate

Then we edit our db/seeds.rb file to make the first user an admin by default.

Then we do a $ rails db:migrate:reset   an

$ rails db:seed

Destroy action

The last step is to add delete links and destroy action. We do this by editing the _user.html.erb file

To get the delete links to work, we need to add a destroy action in the …/users_controller.rb

We want only admins to be able to delete so we restrict access to the destroy action to admins in the …/users_controller.rb file with this code: before_action :admin_user,   only: :destroy