Kyle Loveless
Chapter 9 Notes

In Chapter 9 we complete the REST actions for the Users resources by adding the edit, update, index, and destroy actions.

9.1 Updating users
Updating is much like creating new Users, however instead of a new action to render a view for new Users there is a edit action to render a view for editing Users. Unlike create responding to a POST request there is a update action responding to a PATCH request. The major difference is that anyone can signup as a new user, but only existing Users(and the current User) should be able to update their information.

~See Listing 9.2 to view the similarities between the edit page and the new users page~

Listing 9.4

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

We add a Settings option that links to the edit page of the current user.

Handle unsuccessful signups in Listing 9.5 with the use of strong parameters

```
if @user.update_attributes(user_params)
    # Handle a successful update.
  else
    render 'edit'
```

Still receive error messages existing User model validations and the error-messages partial in Listing 9.2

Listing 9.6 -> More testing

Listing 9.8 - Listing 9.10 Using TDD to handle successful signups

9.2 Authorization

Even though in the previous section we created the edit and update functions they have a major security flaw, anyone can update anyone's information. This section deals with correcting that and using the authentication tools created in Chapter 8.

In Listing 9.12 we use a before filter to require users to be logged in before they can access the edit and update actions.

```
before_action :logged_in_user, only: [:edit, :update]
```

```
def logged_in_user
    unless logged_in?
      flash[:danger] = "Please log in."
      redirect_to login_url
    end
  end
```

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 appropriate **:only** options hash.

Listing 9.14 -> More testing.

Listing 9.16 - Listing 9.18 We can see that our test suite is still passing when we comment out the before filter. This means that we need to add some tests that will fail in the test unless the before filter is there. This is just a security hole that we need to be aware of.

Listing 9.20 - Listing 9.22 We are now testing for the correct user to be logged in now before we try to use the edit or update actions.

- Listing 9.20 We add a second user to the test/fixtures/users.yml
- Listing 9.21 -> More testing. ***Should get some type of error because you're trying to use a method that we never created due to skipping 8.4***
- Listing 9.22 We use another before filter but this time we are making sure that the user is correct and restricting the access to the edit and update actions.

```
before_action :correct_user,   only: [:edit, :update]

def correct_user
    @user = User.find(params[:id])
    redirect_to(root_url) unless @user == current_user
  end
```

Friendly Forwarding

If a non-logged in user attempts to visit their edit page, gets redirected to the login page, then they'll be redirected back to their profile page. It would be a lot nicer if instead we redirected them back to the page they initially wanted to visit.

Kyle Loveless
Chapter 9 Notes

9.3 Showing All Users

Listing 9.32 We add to the first before filter we created so that only logged in users can access the index action.

```
before_action :logged_in_user, only: [:index, :edit, :update]
```

Faker gem used to populate the users index so that we aren't lonely

Pagination makes the page look more appealing by grouping the users into pages of 30(in this example)

9.4 Deleting Users

Adding a admin boolean to the database

Listing 9.53 Adding a destroy action

```
before_action :logged_in_user, only: [:index, :edit, :update, :destroy]

def destroy
  User.find(params[:id]).destroy
  flash[:success] = "User deleted"
  redirect_to users_url
 end
```

Listing 9.54 we add another before filter to restrict destroy actions to admins only.

```
before_action :admin_user,     only: :destroy

def admin_user
    redirect_to(root_url) unless current_user.admin?
  end
```