Mark Prucha

Chapter 8 Notes Login, Log Out

- 8.1 Sessions
  - 8.1.1 Sessions Controller
    - HTTP is stateless – meaning it has no way to remember a user's identity from page to page.
    - To combat this, we must use sessions
      - create sessions Controller
      - Sessions follow REST architecture

```
get    'login'   => 'sessions#new'
post   'login'   => 'sessions#create'
delete 'logout'  => 'sessions#destroy'
```

  - 8.1.2 Login Form
    - Use form_for (:session, url: login_path)
  - 8.1.3 Finding and Authenticating a User
    - :session is the key to a nested params hash, and is also a hash itself.
    - So `params[:session]` is equivalent to:

```
{session:{password: "foobar", email: "user@example.com" }}
```

    - Remember, when we post to the login path, we create the session with user information stored in the session. To authenticate a user, we must access the user from the database and compare the information of the user with the information entered on the login form.
    - We access the user from the database by email address using the find_by method
    - We then use the authenticate method to make sure the user from the database has the same password as the user logging in

LISTING 8.5

```
class SessionsController < ApplicationController

  def new
  end

  def create
    user = User.find_by(email: params[:session][:email].downcase)
    if user && user.authenticate(params[:session][:password])
      # Log the user in and redirect to the user's show page.
    else
      # Create an error message.
      render 'new'
    end
  end

  def destroy
  end
end
```

- 8.1.4 Render with Flash Message
  - If the password is incorrect, we need an error message. So we use flash object.

    flash [:danger] = 'invalid email/password combination'

  - Flash object is already styled via CSS from previous chapter.
  - Problem is flash message persists for one request and re-rendering the page (as we are doing when the login fails) does not count as a request.
    - So flash message appears on the next page accessed after the re-rendered login form. That page then counts as the request.
- 8.1.5 Flash Test
  - To fix the flash issue we use flash.now, which is a variant of flash, used to display a flash message specifically on re-rendered pages.
  - Can develop a test to make sure flash appears only on re-rendered login page as in Listing 8.7

- 8.2 Logging In
  - 8.2.1 log_in Method
    - Temporary session cookie used. When session controller is created, so is SessionHelper. The helper is a module where you can put further methods for sessions such as the log_in method written below.

```
# Logs in the given user.
def log_in(user)
  session[:user_id] = user.id
end
```

  - 8.2.2 Current User
    - Create current_user method so we can keep track of the current user's information on subsequent pages without having to constantly access the database. Can do:

```
def current_user
     if @current_user.nil?
       @current_user = User.find_by(id: session[:user_id])
     else
       @current_user
     end
end
```

      - In Ruby, you can also use ||=

        i.e. `@current_user ||= User.find_by(id: session[:user_id])`

      - `||= is` like a boolean version of +=
      - If the current_user is null then set it to `User.find_by(id: session[:user_id])`
        - Otherwise just return the current_user
  - 8.2.3 Changing the Layout Links
    - Bootstrap stuff to change links depending on whether or not a user is logged in.
    - For example if a user is logged in, you're going to want a logout link.
    - Need a logged_in boolean method to determine if there is a user logged in.

        - Can determine logged_in by finding out if current_user is null. (Listing 8.15)

    - 8.2.4 Testing Layout Changes
        - Can use fixtures to create hypothetical user data in order to test login.
        - In login tests, we define a setup method which refers to the fixture file (see Listing 8.20)
        - New test methods used:
            - `assert_redirected_to @user`
            - `follow_redirect!`
            - `assert_select "a[href=?]", login_path, count: 0`
        -
    - 8.2.5 Login Upon Signup
        - Call the log_in method when creating the user, so that once a new user is created, he or she will be logged in (Listing 8.22).
- 8.3 Logging Out
    - Create log_out method in sessions helper which deletes the user id from the session and sets the current user to null (See below).

```
# Logs out the current user.
def log_out
  session.delete(:user_id)
  @current_user = nil
end
```

    - Define destroy in the sessions controller. It calls the log_out method and redirects to the home page.

```
def destroy
  log_out
  redirect_to root_url
end
```