Chapter 12 - Password Reset

- 12.1
 - General sequence for password reset:
 - When a user requests a password reset, find the user by the submitted email address.
 - If the email address exists in the database, generate a reset token and corresponding reset digest.
 - Save the reset digest to the database, and then send an email to the user with a link containing the reset token and user's email address.
 - When the user clicks the link, find the user by email address, and then authenticate the token by comparing it to the reset digest.
 - If authenticated, present the user with the form for changing the password.
 - Generate a controller for the Password Reset Resource make both a new and edit action
 - \$ rails generate controller PasswordResets new edit --no-test-framework
 - Note the flag for skipping the test generation
 - Add the resources for new, create, edit, and update to the config/routes.rb:
 - resources :password_resets, only: [:new, :create, :edit, :update]
 - Now add a link on the new.html.erb view for the forgot password action
 - <%= link to "(forgot password)", new password reset path %>
 - Now we need to add two attributes to the Users model:
 - Reset_digest : for storing a password reset digest
 - Reset sent at: for expiring the reset request
 - Do this with the following migrate command:
 - \$ rails generate migration add_reset_to_users reset_digest:string \
 reset_sent_at:datetime
 - Then regular rails db:migrate
 - Now we build the Forgot password page; app/views/password_resets/new.html.erb
 - See tutorial for html that's added at this step (12.1.2)
 - When a user submits this new page we must find the user by email, and update the password reset token and sent-at timestamp. Then we redirect to the root URL.
 - Some of this work will be handled in the app/controllers/password resets controller.rb
 - Code will be added to the create method. See tutorial for code (fig 12.5)
 - Additional code must be added to the app/model/user.rb
 - attr_accessor :remember_token, :activation_token, :reset_token

 Also add the create_reset_digest & send_password_reset_email methods (see tutorial for code)

12.2

- Using the existing app/mailers/user_mailer.rb that we generated in chapter 11 we need to update the password_rest method that was already defined:
 - def password_reset(user)@user = usermail to: user.email, subject: "Password reset"

End

- We also need to update the two associated email templates (html and text). See tutorial listing 12.8 & 12.9 for details
- We can add details to our test/mailers/previews/user_mailer_preview.rb file to preview our reset emails. This is nearly identical to how we did it in chapter 11
- Now we can set up a test for the password reset by adding the code listed in Listing 12.12. This is added to the test/mailers/user mailer test.rb

12.3

- Now we need to create the password reset page/form
 - The first step to resetting the password is to add a hidden field to the app/views/password_resets/edit.html.erb for holding the user's email. This is because we need it to be available for the update action when the form is submitted
 - <%= hidden field tag :email, @user.email %>
- To get this form to render we need do a bunch of *before_action* stuff to validate the user. We add the following code to the controllers/password_resets_controller.rb:
 - before_action :get_user, only: [:edit, :update] before_action :valid_user, only: [:edit, :update]
 - Then define the get_user and valid_user methods. See Listing 12.15 for details
- Now that we can submit the reset we need to validate the reset
 - To check that the reset has not expired add another before_action check to the password resets controller.rb:
 - before_action :check_expiration, only: [:edit, :update] # Case (1)
 - Also add the associated check_expication method. See Listing 12.16 for details
 - To check that the password is valid, is not empty, and to handle successful submission of the update add the given code in Listing 12.16 to the *update* method
 - Now we add the password_reset_expired? Method to the models/user.rb
 - # Returns true if a password reset has expired.
 def password reset expired?

reset_sent_at < 2.hours.ago end

- We can now test valid and invalid submissions. Start by generating integration tests for the password reset:
 - \$ rails generate integration_test password_resets
 - Next ass ALL the code he supplied for the test/integration/password_resets.test.rb. See Listing 12.18 (it is a lot of code)

• 12.4

 Like chapter 11 we'll use SendGrid on Heroku for this email service. If you already did this in chapter 11 then there's nothing left to do but the normal wrap up procedure