Chapter 12 Rails Tutorial Notes - Password Reset

12.1 Password Resets Resource

- Generate Password Resets controller: *rails generate controller PasswordResets new edit --no-test-framework*
- Create new entries in Routes.rb for password reset forms: new, create, edit, and update.
- Add link to the login page (in views/sessions/new.html.erb), new reset_digest to db and do a migrate.
- Create new password reset view, then in the controller create new *create* action. Finally, create password reset methods in User.rb.

12.2 Password Reset Emails

- In user_mailer.rb, create a method to email the user the reset instructions. Then, we set up a template for said instructions at password_reset.text.erb and a preview for reset_password just like for activation in CH 11.
- Use the server log to observe sending a password reset email.

12.3 Resetting the Password

- Add a hidden field to the edit page (views/password_resets/edit.html.erb) to store the user's email.
- Put in new *before_action* filters in the password_resets_controller.rb to make sure user exists/is valid.
- Consider 4 use cases and implement them in our *update* action in the controller as well:
    - An expired password reset
    - A failed update due to an invalid password
    - A failed update (which initially looks "successful") due to a blank password and confirmation
    - A successful update
- Create password reset methods in the user model to make sure the activation is fresh.
- Generate integration test with *rails generate integration_test password_resets*, then prove test status = green.

12.4 Email in Production (take two)

- As in CH 11, we use the SendGrid add-on for Heroku to handle our email services in production.
- Finish up with the usual merge > commit > push to our repo and then Heroku.

12.5 Conclusion

- Similar to sessions and activations, password resets can be modeled as a resource.
- Action Mailer can do both plaintext and html emails
- Password resets use a generated token to create a unique URL for resetting passwords.
- Password resets use a hashed reset digest to securely identify valid reset requests.

12.6 Proof of Expiration Comparison

`reset_sent_at < 2.hours.ago`

$\Delta t_r > \Delta t_e$

$\Delta t_r = t_N - t_r$ and $\Delta t_e = t_N - t_e$

$$\begin{aligned} \Delta t_r &> \Delta t_e \\ t_N - t_r &> t_N - t_e \\ -t_r &> -t_e \end{aligned}$$

$t_r < t_e$

[Proof Key]

Δt_r = time interval since sending the password reset

Δt_e = expiration time limit

t_N = time now