CH5 Notes “Filling In the Layout”

This chapter is about adding CSS style sheets to our Sample App and using Bootstrap – an open source web design framework. Continuing on with sample_app

5.1 Adding some structure

First, we created a new git branch:

git checkout –b filling-in-layout

Added additional HTML to the site layout file /app/views/layout/application.html.erb :

- Internet Explorer HTML5 shim
- **classes** (have special meaning to Bootstrap)
  - nav
  - navbar-nav
  - navbar-right
  - container
  - navbar
  - navbar-fixed-top
  - navbar-inverse
- Use # for a **stub link** – a placeholder link

Edited other views to add more HTML and CSS classes (to aid Bootstrap)
Add images to app/assets/images directory

- *image_tag* helper pulls images to be used on pages
- Paid cat tax

BOOTSTRAP (custom CSS)

- Open source from Twitter
- Makes web pages **responsive** (able to adjust to changing screen size)
- Added **bootstrap-sass** gem to gemfile (don’t forget *bundle install*!)
- Create custom CSS file at /app/assets/stylesheets/custom.scss (NOTE: scss extension means **sassy css file**)
  - use *@import “bootstrap-sprockets”;* and *@import “bootstrap”;* to include Bootstrap css framework in the custom.scss file
  - Add universal styling elements to custom.scss too

PARTIALS
Naming convention for partials: *_partialname.html.erb*
Use render to cause code from partial to be inserted into page

- *<%= render 'layouts/header' %>*

5.2 Sass and the Asset Pipeline

Assets:

- **app/assets**: assets specific to the present application
- **lib/assets**: assets for libraries written by your dev team
- **vendor/assets**: assets from third-party vendors

Asset files in /images, /javascript, and /stylesheets get combined into application.css and application.js, **minified** into quick-loading browser files.

Syntactically Awesome Style Sheets (Sass)

Sass allows nesting and variables in css files

$variable_name: #value;

5.3 Layout Links

Changed routes.rb file from: *get 'static_pages/help'* to use **named routes** by saying:

get '/help', to: 'static_pages#help'

This allows us to use Rails functionality **_path** and **_url** in HTML pages like:

<%= link_to "About", about_path %>

Changed stub links (#) to named route links in header and footer partials

Created tests for layout links

5.4 User Signup: A First Step

rails generate controller Users new to create new controller Users and users/new.html.erb view

Changed routes.rb to *get '/signup' to: 'users#new'* (this adds the named route)

Changed home.html.erb to use signup_path for the signup button:

<%- link_to "Sign up now!", signup_path, class: "btn btn-lg btn-primary"%>

5.5 Conclusion

What we learned: HTML5, header, footer, body layout; rails partials; CSS classes and ids; Bootstrap; Sass and asset pipeline; named routes and custom routing rules.

NOTE: bundle install before running rails test after git merge!