CS 416
Web Programming

Ruby on RAILS
Chapter 5 Filling out layout

Dr. Williams
Central Connecticut State University

Based on *Ruby on Rails Tutorial: Learn Web Development with Rails*.
Copyright © 2016 by Michael Hartl.

A look ahead

- Tag helpers
- Routes
- Partials
- Asset pipeline
- Incorporating CSS
  - CSS frameworks
  - Custom CSS
  - Sass preprocessing

- Getting started

```
git checkout -b filling-in-layout
```

Add common site navigation

Add to app/views/layouts/application.html.erb

```
<header class="navbar navbar-fixed-top navbar-inverse">
  <div class="container">
    <%= link_to "sample app", '#', id: "logo" %>
    <nav>
      <ul class="nav navbar-nav navbar-right">
        <li><%= link_to "Home",   '#' %></li>
        <li><%= link_to "Help",   '#' %></li>
        <li><%= link_to "Log in", '#' %></li>
      </ul>
    </nav>
  </div>
</header>
<div class="container">
  <%= yield %>
</div>
```

Link_to

```
<%= link_to "sample app", '#', id: "logo" %>
```

Create an anchor link

- Remember Ruby functions don't require parentheses
- 1st argument is text associated with link
- 2nd is the hyperlink (currently "#" is placeholder)
- 3rd is id to give anchor link tag

Home page

```
<div class="center jumbotron">
  <h1>Welcome to the Sample App</h1>

  <h2>
    This is the home page for the
    <a href="http://www.railstutorial.org/">
          Ruby on Rails Tutorial</a>
    sample application.
  </h2>

  <%= link_to "Sign up now!", '#',
        class: "btn btn-lg btn-primary" %>
</div>

<%= link_to image_tag("rails.png",
     alt: "Rails logo"),'http://rubyonrails.org/'
%>
```

Tag helpers

Link to helper

```
<%= link_to "Sign up now!", '#', class:
    "btn btn-lg btn-primary" %>
```

Results in:

```
<a href="#" class="btn btn-lg
    btn-primary">Sign up now!</a>
```

Layout links

- Could use hard coded path

```
<a href="/static_pages/about">About</a>
```

- Instead want to use ***named routes***

```
<%= link_to "About", about_path %>
```

This allows if route changes in future all dependent links get the update – fix once, fixes everywhere

Rails routes

- Creates a name that maps url to controller method
- Original entry in routes.rb:

```
get 'static_pages/help'
```

  - Matches GET to "**static_pages/help**" and maps to `static_pages_controller#help`

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

  - Matches GET to "**/help**" and maps to **`static_pages_controller#help`**

- Creates two named routes:

```
help_path -> '/help'
help_url  -> 'http://www.example.com/help'
```

Create for rest of routes (root special case)
Update rest of layout paths

Add contact page

- Add route to contact page

In *config/routes.rb* add:

```
get  '/contact', to: 'static_pages#contact'
```

Add action to controller *static_pages_controller.rb*

```
def contact
end
```

Create view contact.html.erb

```
<% provide(:title, 'Contact') %>
<h1>Contact</h1>
<p>
  Contact the Ruby on Rails Tutorial about the sample app at the
  <a href="http://www.railstutorial.org/contact">contact page</a>.
</p>
```

Update static pages controller test for new paths

Partials

- Like functions simplify common code, partials simplifies common HTML
- Name partials underscore, name, html.erb

- Move common header to:

*app/views/**layouts/_header.html.erb***

- Add call to partial in application.html.erb

```
<%= render 'layouts/header' %>
```

Partials cont.

- Create footer partial:

app/views/layouts/_footer.html.erb

```
<footer class="footer">
  <small>
    The <a href="http://www.railstutorial.org/">
              Ruby on Rails Tutorial</a>
  </small>
  <nav>
    <ul>
      <li><%= link_to "About",   about_path %></li>
      <li><%= link_to "Contact", contact_path %></li>
      <li><a href="http://news.railstutorial.org/">
      News</a></li>
    </ul>
  </nav>
</footer>
```

- Now do the same with rails_default includes

The asset pipeline

- **Asset pipeline**
  - Designed to simplify management of **static** assets such as: CSS, JavaScript, and images

- 3 standard asset directories
  - **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

- Each has subdirectories for each asset class:

```
images/     javascripts/     stylesheets/
```

Rails image assets

```
<%= link_to image_tag("rails.png",
      alt: "Rails logo"),'http://rubyonrails.org/' %>
```

Image tag helper

```
image_tag("rails.png", alt: "Rails logo")
```

- Rails will automatically find any images in the `app/assets/images/` directory using the asset pipeline
- To download a file in Linux such as Cloud 9:

```
curl -o app/assets/images/rails.png -OL
railstutorial.org/rails.png
```

Results in:

```
<img src="/assets/rails-9308b8f92fea4c19a3a0d8385b494526.png"
alt="Rails logo" />
```

Note src is generated file name appearing directly in assets directory

Manifest files

- Specify the order to build combined assets into single file (applies to CSS and Javascript not images)

app/assets/stylesheets/application.css

```
/*
 *  …
 *= require_tree .
 *= require_self
 */
```

app/assets/javascripts/application.js

```
//= require jquery
//= require jquery_ujs
//= require turbolinks
//= require_tree .
```

Preprocessor engines

- Runs preprocessing for:
  - Sass .scss
  - CoffeeScript .coffee
  - Embedded Ruby .erb

- Processes manifest files to build single combined file, and minifies (removes all extra spaces, line breaks, indentations) to make it efficient for production
  - *application.css*
  - *application.js*

CSS revisited

Revisit style sheet tag in layout

```
<%= stylesheet_link_tag 'application',
    media: 'all',
    'data-turbolinks-track': 'reload' %>
```

Outputs

```
<link data-turbolinks-track="true"
    href="/assets/application.css"
    media="all" rel="stylesheet" />
```

Add Bootstrap

- Add Bootstrap gem in Gemfile and install

```
gem 'bootstrap-sass', '3.3.6'
bundle install
```

Add a custom CSS file to asset pipeline

```
touch app/assets/stylesheets/custom.scss
```

".scss" extension is “Sassy CSS”

Now import Bootstrap into custom CSS

```
@import "bootstrap-sprockets";
@import "bootstrap";
```

See full custom.scss from text

Sass stylesheets

- Allows short hand and scripting in CSS file with .scss extension that is then preprocessed with Sass
- Generates out resulting basic CSS to be sent to browser
- Result is easier, more readable stylesheets

Sass nesting

```
.center {
  text-align: center;
}

.center h1 {
  margin-bottom: 10px;
}
```

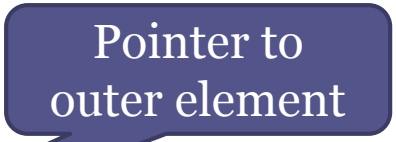
Becomes

```
.center {
  text-align: center;
  h1 {
    margin-bottom: 10px;
  }
}
```

Sass references

```
#logo {
  float: left;
  font-weight: bold;
}
#logo:hover {
  color: #fff;
  text-decoration: none;
}
```

Becomes




```
#logo {
  float: left;
  font-weight: bold;
  &:hover {
    color: #fff;
    text-decoration: none;
  }
}
```

Sass variables

```
$light-gray: #777;


h2{
  color: $light-gray;
}
...
footer{
  color: $light-gray;
}
```

Wrap up

```
git add .
git commit -m "Finish layout and routes"
git checkout master
git merge filling-in-layout

rails test
git push origin master
git push heroku
```

(might need to create if not done already)