**Bootstrap and Rails**

**by Daniel Kehoe**

*Last updated 3 December 2013*

Twitter Bootstrap 3.0 and Rails 4.0. Shows how to set up a Rails application to use Bootstrap (formerly Twitter Bootstrap). [Bootstrap](http://getbootstrap.com/) provides CSS stylesheets and JavaScript code for the visual design of websites. CSS stylesheets are used for design and layout. JavaScript code combines user interaction elements and visual effects for features such as tabs, modal windows, and image carousels. Bootstrap is a framework for client-side (browser-based or “front end”) development, much like Ruby on Rails is a framework for server-side development.

**If You Are New to Rails**

If you’re new to Rails, see [What is Ruby on Rails?](http://railsapps.github.io/what-is-ruby-rails.html), the book [Learn Ruby on Rails](http://learn-rails.com/learn-ruby-on-rails.html), and recommendations for a [Rails tutorial](https://tutorials.railsapps.org/rails-tutorial).

**What is the RailsApps Project?**

This is an article from the RailsApps project. The [RailsApps project](http://railsapps.github.io/) provides example applications that developers use as starter apps. Hundreds of developers use the apps, report problems as they arise, and propose solutions. Rails changes frequently; each application is known to work and serves as your personal “reference implementation” so you can stay up to date. Each is accompanied by a tutorial so there is no mystery code. Support for the project, including the example applications and the [Rails Composer](http://railsapps.github.io/rails-composer/) tool, comes from subscribers to the [RailsApps tutorials](https://tutorials.railsapps.org).

**What’s Here**

This article covers:

* Installation of Bootstrap
* Application Layout
* Flash Messages
* Navigation Links
* Form Builders

In summary, here are the steps for adding Bootstrap to a Rails application:

* add a gem to the **Gemfile**
* modify the file **app/assets/javascripts/application.js** to add Bootstrap’s Javascript files
* add the file **app/assets/stylesheets/framework\_and\_overrides.css.scss** to add Bootstrap’s CSS files
* modify the file **app/views/layouts/application.html.erb** to change the application layout

Complete instructions are below.

**Choose a Bootstrap Gem**

[Several Ruby gems](http://rubysource.com/twitter-bootstrap-less-and-sass-understanding-your-options-for-rails-3-1/) offer Bootstrap packaged for the [Rails asset pipeline](http://guides.rubyonrails.org/asset_pipeline.html). Bootstrap can be installed using either its native [LESS CSS](http://lesscss.org/) language or the [Sass](http://sass-lang.com/) language. Sass is a default for Rails, so I recommended using the gem [bootstrap-sass](https://github.com/thomas-mcdonald/bootstrap-sass) from Thomas McDonald.

We’ll add the [rails\_layout](https://github.com/RailsApps/rails_layout) gem to generate files for an application layout, navigation links, and flash messages styled with Bootstrap CSS classes and layout.

In your **Gemfile**, add:

|  |
| --- |
| gem 'bootstrap-sass'  group :development do  gem 'rails\_layout'  end |

You don’t need the [rails\_layout](https://github.com/RailsApps/rails_layout) gem deployed to production, so put it in the development group.

Run $ bundle install in the Terminal.

We’ll use the [rails\_layout](https://github.com/RailsApps/rails_layout) gem to set up Bootstrap and create the files we need.

**Rails Layout Gem**

The generator provided by the [rails\_layout](https://github.com/RailsApps/rails_layout) gem will set up Bootstrap and add the necessary files. Run:

|  |
| --- |
| $ rails generate layout bootstrap3 --force |

With the --force argument, the rails\_layout gem will replace existing files.

The rails\_layout gem will rename the file:

* **app/assets/stylesheets/application.css**

to:

* **app/assets/stylesheets/application.css.scss**

It will create the file:

* **app/assets/stylesheets/framework\_and\_overrides.css.scss**

and modify the file:

* **app/assets/javascripts/application.js**

The gem will create or replace four files:

* **app/views/layouts/application.html.erb**
* **app/views/layouts/\_messages.html.erb**
* **app/views/layouts/\_navigation.html.erb**
* **app/views/layouts/\_navigation\_links.html.erb**

Let’s examine the files to see how our application is configured to use Bootstrap.

**Renaming the application.css File**

The rails\_layout gem renamed the **app/assets/stylesheets/application.css** file as **app/assets/stylesheets/application.css.scss**. Note the **.scss** file extension. This will allow you to use the advantages of the Sass syntax for your application stylesheet. Stylesheets can use variables, mixins, and nesting of CSS rules when you use Sass. For more on the advantages of Sass and how to use it, see the [Sass](http://sass-lang.com/) website or the [Sass Basics RailsCast](http://railscasts.com/episodes/268-sass-basics) from Ryan Bates.

Sass has two syntaxes. The most commonly used syntax is known as “SCSS” (for “Sassy CSS”), and is a superset of the CSS syntax. This means that every valid CSS stylesheet is valid SCSS as well. You can use Sass in any file by adding the file extension **.scss**. The asset pipeline will preprocess any **.scss** file and expand it as standard CSS.

Before you continue, make sure that the rails\_layout gem renamed the **app/assets/stylesheets/application.css** file as **app/assets/stylesheets/application.css.scss**. Otherwise you won’t see the CSS styling we will apply.

**The application.css.scss File**

The Rails asset pipeline will concatenate and compact CSS stylesheets for delivery to the browser when you add them to this directory:

* **app/assets/stylesheets/**

The asset pipeline helps web pages display faster in the browser by combining all CSS files into a single file (it does the same for JavaScript).

Let’s examine the file **app/assets/stylesheets/application.css.scss**:

|  |
| --- |
| /\*  \* This is a manifest file that'll be compiled into application.css, which will include all the files  \* listed below.  \*  \* Any CSS and SCSS file within this directory, lib/assets/stylesheets, vendor/assets/stylesheets,  \* or vendor/assets/stylesheets of plugins, if any, can be referenced here using a relative path.  \*  \* You're free to add application-wide styles to this file and they'll appear at the top of the  \* compiled file, but it's generally better to create a new file per style scope.  \*  \*= require\_self  \*= require\_tree .  \*/ |

The **app/assets/stylesheets/application.css.scss** file serves two purposes.

First, you can add any CSS rules to the file that you want to use anywhere on your website. Second, the file serves as a *manifest*, providing a list of files that should be concatenated and included in the single CSS file that is delivered to the browser.

**A Global CSS File**

Any CSS style rules that you add to the **app/assets/stylesheets/application.css.scss** file will be available to any view in the application. You could use this file for any style rules that are used on every page, particularly simple utility rules such as highlighting or resetting the appearance of links. However, in practice, you are more likely to modify the style rules provided by Bootstrap. These modifications don’t belong in the **app/assets/stylesheets/application.css.scss** file; they will go in the **app/assets/stylesheets/framework\_and\_overrides.css.scss** file.

In general, it’s bad practice to place a lot of CSS in the **app/assets/stylesheets/application.css.scss** file (unless your CSS is very limited). Instead, structure your CSS in multiple files. CSS that is used on only a single page can go in a file with a name that matches the page. Or, if sections of the website share common elements, such as themes for landing pages or administrative pages, make a file for each theme. How you organize your CSS is up to you; the asset pipeline lets you organize your CSS so it is easier to develop and maintain. Just add the files to the **app/assets/stylesheets/** folder.

**A Manifest File**

It’s not obvious from the name of the **app/assets/stylesheets/application.css.scss** file that it serves as a *manifest file* as well as a location for miscellaneous CSS rules. For most websites, you can ignore its role as a manifest file. In the comments at the top of the file, the \*= require\_self directive indicates that any CSS in the file should be delivered to the browser. The \*= require\_tree . directive (note the Unix “dot operator”) indicates any files in the same folder, including files in subfolders, should be combined into a single file for delivery to the browser.

If your website is large and complex, you can remove the \*= require\_tree . directive and specify individual files to be included in the file that is generated by the asset pipeline. This gives you the option of reducing the size of the application-wide CSS file that is delivered to the browser. For example, you might segregate a file that includes CSS that is used only in the site’s administrative section. In general, only large and complex sites need this optimization. The speed of rendering a single large CSS file is faster than fetching multiple files.

**Bootstrap JavaScript**

Bootstrap provides both CSS and JavaScript libraries.

Like the **application.css.scss** file, the **application.js** file is a manifest that allows a developer to designate the JavaScript files that will be combined for delivery to the browser.

The rails\_layout gem modified the file **app/assets/javascripts/application.js** to include the Bootstrap JavaScript libraries:

|  |
| --- |
| //= require jquery  //= require jquery\_ujs  //= require turbolinks  //= require bootstrap  //= require\_tree . |

It added the directive //= require bootstrap before //= require\_tree ..

**Bootstrap CSS**

The rails\_layout gem added a file **app/assets/stylesheets/framework\_and\_overrides.css.scss** containing:

|  |
| --- |
| // import the CSS framework  @import "bootstrap";  .  .  . |

The file **app/assets/stylesheets/framework\_and\_overrides.css.scss** is automatically included and compiled into your Rails application.css file by the \*= require\_tree . statement in the **app/assets/stylesheets/application.css.scss** file.

The @import "bootstrap"; directive will import the Bootstrap CSS rules from the Bootstrap gem.

You could add the Bootstrap @import code to the **app/assets/stylesheets/application.css.scss** file. However, it is better to have a separate **app/assets/stylesheets/framework\_and\_overrides.css.scss** file. You may wish to modify the Bootstrap CSS rules; placing changes to Bootstrap CSS rules in the **framework\_and\_overrides.css.scss** file will keep your CSS better organized.

**Overriding Bootstrap Classes**

The file **app/assets/stylesheets/framework\_and\_overrides.css.scss** shows how to customize Bootstrap classes.

|  |
| --- |
| // make all images responsive by default  img {  @extend .img-responsive;  margin: 0 auto;  }  // override for the 'Home' navigation link  .navbar-brand {  font-size: inherit;  } |

The first style rule make all images responsive by default. With this rule, all images will resize to accommodate browser windows of varying widths. We use the Sass @extend directive to add the Bootstrap class img-responsive to the HTML img element.

The second style rule will force the font size of a navbar-brand navigation link to be the same size as the fonts specified in an enclosing container.

**Using Sass Mixins with Bootstrap**

In addition to the simple @import "bootstrap"; directive, the **app/assets/stylesheets/framework\_and\_overrides.css.scss** contains a collection of Sass mixins. These are examples that you can remove.

You can use Sass mixins to map Bootstrap class names to your own semantic class names. The rails\_layout gem provides examples of Sass mixins that apply CSS style rules to the default application layout. In doing so, the default application layout is free of framework-specific code and can be used with Bootstrap, Zurb Foundation, or other front-end frameworks. The book [Learn Ruby on Rails](http://learn-rails.com/learn-ruby-on-rails.html) explains how to use Sass mixins.

**Using Bootstrap CSS Classes**

Now that you’ve installed Bootstrap, you have a rich library of interactive effects you can add to your pages.

Take a look at the [Bootstrap documentation](http://getbootstrap.com/) to see your options. Here are just a few examples:

* [buttons](http://getbootstrap.com/css/#buttons)
* [modal dialogs](http://getbootstrap.com/javascript/#modals)
* [carousel](http://getbootstrap.com/javascript/#carousel)

At a simpler level, Bootstrap provides a collection of carefully-crafted styling rules in the form of CSS classes. These are building blocks you use for page layout and typographic styling. For example, Bootstrap gives you CSS classes to set up rows and columns in a grid system.

Let’s take a closer look at the Bootstrap grid system.

**Bootstrap Grid**

By default, the Bootstrap grid is 940 pixels wide. Two grids are available; “small” for browsers less than 768 pixels in width, and “large” for all others. Start by designing for the small screen with the classes prefixed “small”; then add classes prefixed “large” to change the layout for a large screen. The layout will change when the browser width is less than 768 pixels wide.

The grid gives you 12 columns by default. You can organize your layout in horizontal and vertical sections using row and columns classes.

For example, you could use Bootstrap grid classes to set up an application layout with a footer as a row with two sections:

|  |
| --- |
| <footer class="row">  <section class="col-md-4">  Copyright 2013  </section>  <section class="col-md-8">  All rights reserved.  </section>  </footer> |

The Bootstrap row class will create a horizontal break. The footer will contain two side-by-side sections. The first will be four columns wide; the second will be eight columns wide.

To better understand the grid system with all its options, see the [documentation for the Bootstrap Grid](http://getbootstrap.com/css/#grid).

**Application Layout with Bootstrap**

Generating a new Rails application with the rails new command will create a default application layout. Rails will use the layout defined in the file **app/views/layouts/application.html.erb** as a default for rendering any page.

Let’s look at the application layout file created by the rails\_layout gem:

Examine the contents of the file **app/views/layouts/application.html.erb**:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title><%= content\_for?(:title) ? yield(:title) : "Learn Rails" %></title>  <meta name="description" content="<%= content\_for?(:description) ? yield(:description) : "Learn Rails" %>">  <%= stylesheet\_link\_tag "application", media: "all", "data-turbolinks-track" => true %>  <%= javascript\_include\_tag "application", "data-turbolinks-track" => true %>  <%= csrf\_meta\_tags %>  </head>  <body>  <header>  <%= render 'layouts/navigation' %>  </header>  <main role="main">  <%= render 'layouts/messages' %>  <%= yield %>  </main>  </body>  </html> |

The book [Learn Ruby on Rails](http://learn-rails.com/learn-ruby-on-rails.html) covers the application layout file in detail.

If you use the Sass mixins provided by the rails\_layout gem in the **app/assets/stylesheets/framework\_and\_overrides.css.scss** file, there’s no need to use Bootstrap classes directly in the application layout (Bootstrap classes will be applied to the HTML element main).

If you don’t use Sass mixins, you can add Bootstrap classes directly to the application layout. For example, you might set the main element to be twelve columns wide using a Bootstrap class:

|  |
| --- |
| <main role="main" class="col-md-12"> |

See the [documentation for the Bootstrap Grid](http://foundation.zurb.com/docs/components/grid.html).

**Flash Messages with Bootstrap**

Rails provides a standard convention to display alerts (including error messages) and other notices (including success messages), called a *flash message*. The name comes from the term “flash memory” and should not be confused with the “Adobe Flash” web development platform that was once popular for animated websites.

Rails uses :notice and :alert as flash message keys. Bootstrap provides a base class alert with additional classes alert-success and alert-danger. A bit of parsing is required to get a Rails “notice” message to be styled with the Bootstrap alert-success style. Any other message, including a Rails “alert” message, will be styled with the Bootstrap alert-danger style.

By default, Bootstrap applies a green background to alert-success and a red background to alert-danger. Bootstrap provides additional classes alert-info (blue) and alert-warning (yellow). With a little hacking, it’s possible to create a Rails flash message with a custom name, such as :info, that will display with the Bootstrap info class. However, it’s wise to stick with the Rails convention of using only “alert” and “notice.”

You can include code to display flash messages directly in your application layout file or you can create a [partial template](http://guides.rubyonrails.org/layouts_and_rendering.html#using-partials) – a “partial” – to better organize the default application layout.

The application layout file includes a messages partial:

|  |
| --- |
| <%= render 'layouts/messages' %> |

Examine the file **app/views/layouts/\_messages.html.erb**:

|  |
| --- |
| <%# Rails flash messages styled for Twitter Bootstrap 3.0 %>  <% flash.each do |name, msg| %>  <% if msg.is\_a?(String) %>  <div class="alert alert-<%= name == :notice ? "success" : "danger" %>">  <button type="button" class="close" data-dismiss="alert" aria-hidden="true">&times;</button>  <%= content\_tag :div, msg, :id => "flash\_#{name}" %>  </div>  <% end %>  <% end %> |

We use each to iterate through the flash hash, retrieving a name and msg that are passed to a block to be output as a string. The expression if msg.is\_a?(String) serves as a test to make sure we only display messages that are strings. We construct a div that applies Bootstrap CSS styling around the message. Bootstrap recognizes a class alert to construct an alert box. A class of either alert-success or alert-danger styles the message. Rails notice messages will get styled with the Bootstrap alert-success class. Any other Rails messages, including alert messages, will get styled with the Bootstrap alert-danger class.

We use the Rails content\_tag view helper to create a div containing the message.

Finally, we create a “close” icon by applying the class close to a link. We use the HTML entity &times; (a big “X” character) for the link; it could be the word “close” or anything else we like. Bootstrap’s integrated JavaScript library will hide the alert box when the “close” link is clicked.

Bootstrap provides [detailed documentation](http://foundation.zurb.com/docs/components/alert-boxes.html) if you want to change the styling of the alert boxes.

**Navigation Partial with Bootstrap**

You’ll likely need navigation links on every page of your web application.

The layout and styling required for the Bootstrap navigation bar are in the navigation partial file.

The application layout file includes a navigation partial:

|  |
| --- |
| <%= render 'layouts/navigation' %> |

Examine the file **app/views/layouts/\_navigation.html.erb**:

|  |
| --- |
| <%# navigation styled for Twitter Bootstrap 3.0 %>  <nav class="navbar navbar-inverse navbar-fixed-top">  <div class="container">  <div class="navbar-header">  <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">  <span class="sr-only">Toggle navigation</span>  <span class="icon-bar"></span>  <span class="icon-bar"></span>  <span class="icon-bar"></span>  </button>  <%= link\_to 'Home', root\_path, class: 'navbar-brand' %>  </div>  <div class="collapse navbar-collapse">  <ul class="nav navbar-nav">  <%= render 'layouts/navigation\_links' %>  </ul>  </div>  </div>  </nav> |

The navigation partial includes layout and Bootstrap classes needed to produce a responsive navigation bar.

The responsive navigation bar adjusts to different browser widths. At small sizes, the navigation links will disappear and be replaced by a menu icon. Clicking the icon will reveal a vertical menu of navigation links. The navigation menu is a great demonstration of the ability of Bootstrap to adjust to the small screen size of a tablet or smartphone.

If you’d like to add a site name or logo to the tutorial application, you can replace the link helper <%= link\_to 'Home', root\_path, class: 'navbar-brand' %>.

We wrap the nested partial render 'layouts/navigation\_links' with Bootstrap layout and classes to complete the navigation bar.

**Navigation Links Partial**

The file **app/views/layouts/\_navigation\_links.html.erb** is very simple:

|  |
| --- |
| <%# add navigation links to this file %> |

You can add links to this file, for example:

<%# add navigation links to this file %>

|  |
| --- |
| <li><%= link\_to 'About', page\_path('about') %></li>  <li><%= link\_to 'Contact', new\_contact\_path %></li> |

The navigation links partial is simply a list of navigation links. It doesn’t require additional CSS styling. By separating the links from the styling that creates the navigation bar, we segregate the code that is unique to Bootstrap. In the future, if the Bootstrap layout or CSS classes change, we can make changes without touching the navigation links.

**SimpleForm with Bootstrap**

Rails provides a set of view helpers for forms. They are described in the [RailsGuides: Rails Form Helpers](http://guides.rubyonrails.org/form_helpers.html) document. Many developers use an alternative set of form helpers named SimpleForm, provided by the [SimpleForm gem](https://github.com/plataformatec/simple_form). The SimpleForm helpers are more powerful, easier to use, and offer an option for styling with Bootstrap 2.3.

As of early December 2013, SimpleForm 3.0 does not fully support Bootstrap 3.0. See the SimpleForm issue [Bootstrap 3 support](https://github.com/rafaelfranca/simple_form-bootstrap/pull/28).

**Themes for Bootstrap**

Frameworks such as Bootstrap are intended to provide the building blocks for a custom website design. If you’ve got strong design skills, or can partner with an experienced web designer, you can develop a custom design that expresses the purpose and motif of your website. If you don’t have the skill or resources to customize the design, you can use the generic Bootstrap design.

An alternative is to find or purchase a pre-designed theme for your website. [Themes for Bootstrap](http://www.themesforbootstrap.com/) aggregates themes from many sites.

You can adapt free themes from sites such as:

* [Themestrap](http://code.divshot.com/themestrap/)
* [Start Bootstrap](http://startbootstrap.com/)
* [Bootswatch](http://bootswatch.com/)

You can adapt themes that you purchase on marketplaces such as:

* [WrapBootstrap](https://wrapbootstrap.com/)

Here’s more on [How to integrate a WrapBootstrap theme into a Rails app](http://pnthr.com/integrating-a-wrapbootstrap-theme-into-a-rails-app/).

A few consultants offer Bootstrap themes that are specifically designed for Rails projects:

* [Dressed Rails Themes](http://dresssed.com/) from Marc-André Cournoyer
* [Railsview.com](http://railsview.com/) from Richardson Dackam

You can design your own themes with drag-and-drop design tools such as:

* [Divshot](http://www.divshot.com/)
* [Jetstrap](https://jetstrap.com/)
* [LayoutIt](http://www.layoutit.com/)

**Resources for Bootstrap**

The [Bootsnipp](http://bootsnipp.com/) gallery offers dozens of Bootstrap-based design elements and code examples.

You can find many more resources for Bootstrap on the [Big Badass List of Useful Twitter Bootstrap Resources](http://www.bootstraphero.com/the-big-badass-list-of-twitter-bootstrap-resources/) at the Bootstrap Hero site.

**Getting Help for Bootstrap**

[Stack Overflow](http://stackoverflow.com/questions/tagged/twitter-bootstrap) is a popular site for questions and answers about Bootstrap.

**Did You Like the Article?**

Was the article useful to you? Follow [@rails\_apps](http://twitter.com/rails_apps) on Twitter and tweet some praise. I’d love to know you were helped out by the article.

You can also find me on [Facebook](https://www.facebook.com/daniel.kehoe.sf) or [Google+](https://plus.google.com/+DanielKehoe/).