Rails-Bootstrap

Last updated 28 Jun 2014

GitHub Repository (https://github.com/RailsApps/rails-bootstrap) · Issues (https://github.com/RailsApps/rails-bootstrap/issues)

Bootstrap Quickstart Guide

Introduction

This guide shows how to integrate Bootstrap (http://getbootstrap.com/) with Rails. Versions:

- Bootstrap 3.2
- Rails 4.1

A complete rails-bootstrap (https://github.com/RailsApps/rails-bootstrap) example application is available from the RailsApps project.

You will need:

- The Ruby language – version 2.1
- The Rails gem – version 4.1

See the article Installing Rails (http://railsapps.github.io/installing-rails.html) for instructions about setting up Rails and your development environment.

Is It for You?

This guide is for experienced Rails developers who want to integrate Bootstrap with Rails. It shows the quickest path to set up Bootstrap with Rails. For a full tutorial, with additional example code and a complete application, see:

- Rails and Bootstrap Tutorial (https://tutorials.railsapps.org/tutorials/rails-bootstrap)

What's Here

You'll find basics here:

- installation of Bootstrap
- the application layout
- flash messages
- navigation links

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

This guide uses the rails_layout (https://github.com/RailsApps/rails_layout) gem to set up Bootstrap and the necessary files.

Rails Composer

You can use the Rails Composer (http://railsapps.github.io/rails-composer/) tool to build a Bootstrap-based starter app in minutes.

To build the example application, Rails 4.1 must be installed in your development environment. Run the command:

```
$ rails new rails-bootstrap -m https://raw.github.com/RailsApps/rails-composer/master/composer.rb
```

Follow the prompts. The README (https://github.com/RailsApps/rails-bootstrap) from the rails-bootstrap example application has more information about the options.

If you'd like to add Bootstrap to an existing application, or learn how to integrate Bootstrap with Rails, read the rest of this guide.

Gems

The bootstrap-sass (https://github.com/thomas-mcdonald/bootstrap-sass) gem adds Bootstrap CSS and JavaScript files to the Rails asset pipeline (http://guides.rubyonrails.org/asset_pipeline.html). The bootstrap-sass gem is the official port of Bootstrap for Sass.

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.

Rails Layout

A 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:install bootstrap3
```

You can add the `--force` argument to replace existing files.

The rails_layout generator will rename the file:

- app/assets/stylesheets/application.css

to:

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

This will allow you to use Sass syntax in your application stylesheet. Stylesheets can use variables, mixins, and nesting of CSS rules when you use Sass.

The rails_layout generator 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

Next we'll examine each of these files and explain their purpose.

Application CSS

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/styleshee
ts,
 * or vendor/assets/stylesheets of plugins, if any, can be referenced here using a relative pat
h.
 *
 * You're free to add application-wide styles to this file and they'll appear at the bottom of
the
 * compiled file so the styles you add here take precedence over styles defined in any styles
 * defined in the other CSS/SCSS files in this directory. It is generally better to create a ne
w
 * file per style scope.
 *
 *= require_tree .
 *= require_self
 */
```

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 makes 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.

Application Layout

Rails uses 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) : "Rails Bootstrap" %></title>
    <meta name="description" content="<%= content_for?(:description) ? yield(:description) : "R
ails Bootstrap" %>">
    <%= 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 Sass mixins in the app/assets/stylesheets/framework_and_overrides.css.scss file will apply Bootstrap classes to the HTML element `main` . You can add Bootstrap classes directly to the application layout instead of using Sass mixins.

Bootstrap Grid

You can add Bootstrap classes directly to the application layout.

You can organize your layout in horizontal sections using `row` classes. Rows should be nested within a `container` class.

A row can contain a single column or you can split it into multiple columns. A row contains a maximum of 12 columns.

The width of columns varies with the width of the browser window. Here's a table that shows the width of the page on different devices:

	Phones	Tablets	Desktops	Desktops
Container width	none (auto)	750px	970px	1170px
Class prefix	.col-xs-	.col-sm-	.col-md-	.col-lg-
Column width	Auto	60px	78px	95px

Here's how to add Bootstrap classes so all your pages display within a single full-width column:

```
<main role="main" class="container">
  <div class="row">
    <div class="col-lg-12">
      <%= render 'layouts/messages' %>
      <%= yield %>
    </div>
</main>
```

Here's a footer presented as a row with two sections:

```
<footer class="container">
  <div class="row">
    <section class="col-sm-4">
      Copyright 2014
    </section>
    <section class="col-sm-8">
      All rights reserved.
    </section>
  </div>
</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. By specifying `col-sm-` , the footer will contain side-by-side columns for all desktops and tablets. On phones, the columns will collapse and display as stacked rows.

To see the grid in action, change `col-sm-` to `col-lg-` and watch what happens when you resize the browser on a desktop computer. For any width narrower than 1170px, the columns will collapse and display as stacked rows.

See the documentation for the Bootstrap Grid (http://getbootstrap.com/css/#grid) to learn about the Bootstrap grid classes.

Flash Messages

Rails provides a standard convention to display alerts (including error messages) and other notices (including success messages), called a *flash message*.

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 Bootstrap 3.0 %>
<% flash.each do |name, msg| %>
  <% if msg.is_a?(String) %>
    <div class="alert alert-<%= name.to_s == 'notice' ? 'success' : 'danger' %>">
      <button type="button" class="close" data-dismiss="alert" aria-hidden="true">&times;</butt
on>
      <%= content_tag :div, msg, :id => "flash_#{name}" %>
    </div>
  <% end %>
<% end %>
```

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.

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://getbootstrap.com/components/#alerts) if you want to change the styling of the alert boxes.

Beyond Red and Green

For simplicity, it’s wise to stick with the Rails convention of using only “alert” and “notice.” However, if you wish, you can accommodate an additional “success” class.

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 accommodate a Rails flash message with a third name, such as `:success`. Here's an example.

You can replace `<%= name.to_s == 'notice' ? 'success' : 'danger' %>">` with this:

```
<%= name.to_s == 'notice' ? 'success' : (name == 'info' ? 'info' : 'danger' )%>
```

This will style any message with the red `alert-danger` class unless it is "notice" (blue) or "success" (green).

Navigation Bar

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 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-c
ollapse">
        <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.

Troubleshooting

The behavior of the navigation bar will show you if Bootstrap JavaScript is working correctly.

When you test your application, reduce the browser window to a narrow width. The navigation links should be replaced by an icon. If clicking the menu icon doesn't reveal a drop-down menu, the application may not be loading the Bootstrap JavaScript library. For more troubleshooting, open the browser JavaScript console and look for error messages.

Form Helpers

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.

SimpleForm version 3.1 (and newer) is compatible with Bootstrap 3.

In your Gemfile, add:

```
gem 'simple_form'
```

If SimpleForm version 3.1.0 final is not yet out, use the prerelease:

```
gem 'simple_form', '>= 3.1.0.rc1'
```

Run `$ bundle install .`

Run the generator to install SimpleForm with a Bootstrap option:

```
$ rails generate simple_form:install --bootstrap
```

which installs several configuration files:

```
config/initializers/simple_form.rb
config/initializers/simple_form_bootstrap.rb
config/locales/simple_form.en.yml
lib/templates/erb/scaffold/_form.html.erb
```

Here the SimpleForm gem uses the `rails generate` command to create files for initialization and localization (language translation). SimpleForm can be customized with settings in the initialization file.

Resources for Bootstrap

Bootstrap's large and active developer community offers hundreds of additional components, tools, and themes that can enhance your application.

Look for Bootstrap resources at the Bootstrap Hero site:

- Big Badass List of Useful Twitter Bootstrap Resources (http://www.bootstraphero.com/the-big-badass-list-of-twitter-bootstrap-resources/)

Bootstrap Components

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

Themes for Bootstrap

You can find free themes on sites such as:

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

You can purchase themes from marketplaces such as:

- WrapBootstrap (https://wrapbootstrap.com/)
- Themeforest (http://themeforest.net/search?utf8=%E2%9C%93&term=bootstrap)
- Creative Market (https://creativemarket.com/tag/bootstrap)

Bootstrap Carnival (http://bootstrapcarnival.com/) aggregates themes from many vendors.

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

If you'd like to design your own theme, or tinker with all the Bootstrap variables, try this Bootstrap themes generator:

- Bootstrap Magic (http://pikock.github.io/bootstrap-magic/app/index.html)

Adapting Themes for Rails

Here's an article on How to integrate a WrapBootstrap theme into a Rails app (http://pnthr.com/integrating-a-wrapbootstrap-theme-into-a-rails-app/).

Online Design Tools

You can design your pages using Bootstrap with drag-and-drop design tools such as:

- Easel (https://www.easel.io/)
- Divshot (http://www.divshot.com/)
- Jetstrap (https://jetstrap.com/)
- LayoutIt (http://www.layoutit.com/)

Comments

Credits

Daniel Kehoe implemented the application and wrote the tutorial.

Did You Like This Guide?

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/).

Any issues? Please create an issue (http://github.com/RailsApps/rails-bootstrap/issues) on GitHub. Reporting (and patching!) issues helps everyone.