**Chapter 3**

**Using static assets**

· transitioning from server-rendered views to a static-asset SPA

· examining the differences between server-rendering and client-side rendering

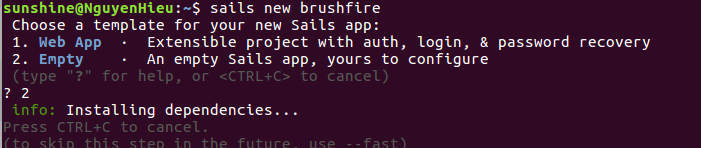
· exploring how Sails uses asset routes

· understanding the static asset pipeline

· setting up “fake” responses and loading states for front-end requests

## **3.1 Introduction to static routing**

Create the folder where brushfire will live



move into the root of the project by typing:



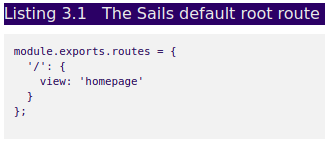
start the Sails by typing

****

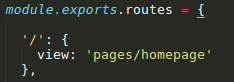
Our current homepage is not in the assets folder. Instead, by default, Sails created an explicit root route and a server-rendered homepage view when we created the project. We’ll explore both in the next section.

### **3.1.1 The default homepage**

When Sails generates a new project, by default, it sets up an explicit route to handle **GET** requests to the root path (/) of a project. In Sublime, open **brushfire/config/routes.js**



**In lastest Sails version 1.0.2**

****

### **3.1.2 Replacing the default homepage.**

We’re going to use a generator to modify our existing Sails project files changing Brushfire’s default behavior to use ***asset routes***as the entry point for our application.



The **brushfire/node\_modules/** folder is the folder where all **Brushfire dependency files** are installed. This makes each project self contained and very portable.

By using the **--save** parameter npm has also added the generator as a dependency in our **brushfire/package.json** file

The **brushfire/package.json** file has a section that contains the names and versions of all of your Brushfire module dependencies. Inevitably modules are upgraded with breaking changes. Having a file that tracks the working versions of the modules you use in your application is a lifesaver when it comes to debugging.

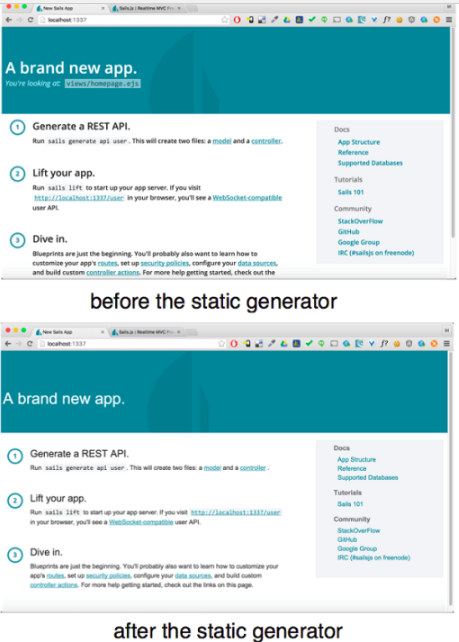
Now that the module is installed we can use the generator by typing



The generator removes the explicit route in **brushfire/config/routes.js** and copies most of the contents of the current homepage located in the **brushfire/views/homepage.ejs** (**brushfire/views/page/hompage.ejs** in version 1.0.2) to a file located in **brushfire/assets/index.html.**

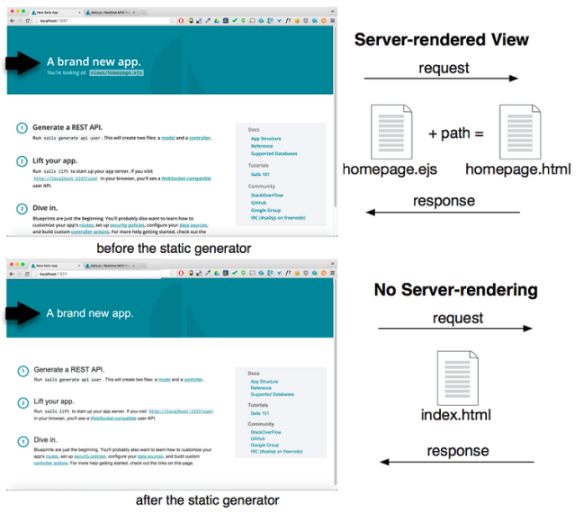
Restart the Sails server by typing



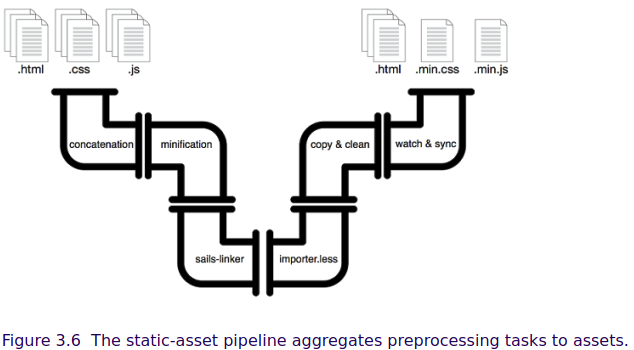
****

**-** The first difference is not apparent from viewing each page. When your browser made a **GET** request to the root route, Sails tried to match it to an explicit route that contains the ***route address*** '**GET /**'. Instead, Sails used another type of route, an ***asset*** *route*, to deliver the home page located at **brushfire/assets/index.html**. Moving the contents of the homepage view to **brushfire/assets/index.html**, automatically created an asset route. Therefore a **GET** request to **/** now responds with the contents of **brushfire/assets/index.html** a.k.a. the home page.

The second difference is that the path to the homepage is not displayed like it was for the default **server-rendered** homepage. When Sails used the **explicit route**, the path was added to **brushfire/views/homepage.ejs** before the server responded with the view as shown in figure 3.5.



## **3.2 The asset pipeline**



### **3.2.1 A quick look at the** .tmp/ **folder**

With all of this talk of the assets folder, you might think Sails was serving up **brushfire/assets/index.html** from the **brushfire/assets/** folder. Out of the box, the Sails asset pipeline pulls files from brushfire/assets/, does transformations in the **brushfire/.tmp/** folder, and then dumps the final result in the **brushfire/.tmp/public/** folder

### **3.2.2 Grunt: The other white meat**

*Grunt* calls itself a “*JavaScript task runner”. We think* that’s pretty accurate. It allows Sails to set up repetitive asset pipeline management tasks, which are then executed automatically or manually depending on the task. We saw earlier that there's a Grunt task which looks for changes in **brushfire/assets/** and syncs them to **brushfire/.tmp/public/**.

### **3.2.3 Putting it all together: Chad’s sweet homepage**

### Based on Chad’s mockups, let’s add our own markup to the generated homepage in **brushfire/assets/index.html** Open **brushfire/assets/index.html** in Sublime and replace the current markup after the body tag with the HTML in listing 3.2.

### 

### Restart the server and sails-lift. we’ll see the page like this:

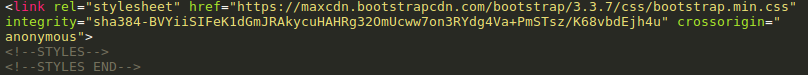
### Why is this page not beautiful? **Because we didn’t install CDN**

### This markup uses classes from the popular **Bootstrap styling framework, which we’ll include in our project using a content delivery network (CDN) to deliver the dependencies.**

### **3.2.4 Using a CDN**

First let’s head over to **[http://getbootstrap.com/getting-started/#download](http://getbootstrap.com/getting-started/" \l "download)** and grab the link from the “Bootstrap CDN” section.

Copy the link has “**bootstrap.min.css**”



Restart the server and you can see the change:

### 

### **3.2.5 Why index.html?**

### Sometimes you may want to link to a folder instead of directly to a file. As a convenience, files named **index.html** have a special status within the **brushfire/assets/** folder. If a request is made to a ***sub-folder***of **brushfire/assets**, the Sails server will look for and display a file named index.html in that sub-folder. If **index.html** doesn’t exist, the server will respond with a **404** **“**not found” status.

### Create a new **subfolder** named **videos/** within **brushfire/assets**



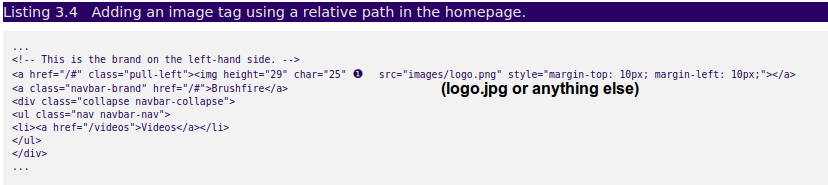
You can access the HTML for the page from the following gist: <https://gist.github.com/sailsinaction/b77772769f28112247bc>

Copy the source from the gist into **brushfire/assets/videos/index.html**

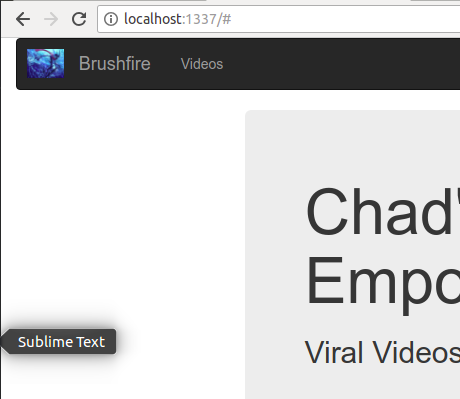
### Ensure that Sails is running, then navigate to **localhost:1337/videos** in the browser and you should see the videos page displayed.

### **3.2.6 An <img>is worth a thousand words**

### Save your image in the **brushfire/assets/images/** folder, and name it **logo.jpg** (or .png,…). In Sublime, open **brushfire/assets/index.html** and add the following tags similar to listing 3.4*.*

**

With the Sails server running via ***sails lift***, navigate your browser to **localhost:1337** and you should see something like this.

****

**3.2.7 Relative paths**

The logic behind relative paths is not very intuitive so let’s dig a bit deeper. Earlier, we used **images/logo.png** as the source path for the **img** tag in the navigation bar. This is a ***relative*** path. That is, **images/logo.png** will be appended to the location of the page that contains the **img** tag, **brushfire/assets/index.html**. So in this case **images/logo.png.** will be appended to **brushfire/assets/**, resolving to **brushfire/assets/images/logo.png**.

### Let’s look at another example to show you what we mean. In Sublime, open **brushfire/assets/videos/index.html** and add the same **img** tag to the videos page similar to listing 3.5.

### 

### Make sure Sails is running via **sails lift** and navigate your browser to **localhost:1337/videos**. You should see a broken image link in your browser similar to this figure.

### 

### The solution is to add a leading **“*/”*** (slash) in front of the source path. Placing a leading **“/”** (slash) before the folder name changes the relative path to an ***absolute*** path. That is, the source path **/images/logo.png** will now be appended to the ***asset root*** or ***web root*** folder. Since the web root folder is **brushfire/assets/**, this will resolve to **brushfire/assets/images/logo.png**, which will display the image properly. Let’s propogate this change to the videos page in **brushfire/assets/videos/index.html** similar to listing 3.6.

### 

### And the result look like this

### 

## **3.3 Managing scripts and stylesheets**

***Sails linker***. The Sails linker will automatically add CSS and JavaScript file links to the asset folder, which can save a lot of time compared to manually adding them. To illustrate how sails-linker works we’ll trade out our **Bootstrap CDN** link for a local copy of the Bootstrap library.

To do this, first remove the CDN links from **brushfire/assets/index.html** and **brushfire/assets/videos/index.html**. Next, go back to [http://getbootstrap.com/getting-started/ - download](http://getbootstrap.com/getting-started/" \l "download) and download the bootstrap distribution ZIP file. Uncompress the ZIP file and copy the **bootstrap.min.css** and **bootstrap.css.map** files into the **brushfire/assets/styles/** folder. Make sure Sails is running via ***sails lift*** and navigate to ***localhost:1337***. The home page doesn’t seem to have changed, butlet’s take a closer look at the page source.

The links are added in **.tmp/public** folder.

****

But how did these files get linked to the home page? (*hint*: insert grunting sound). That’s correct, another **Grunt task** to the rescue. In fact, we’ve grouped a special collection of Grunt tasks into the asset pipeline we call the ***sails-linker***.

The Sails “**linker**” is one of the most powerful utilities in the built-in asset pipeline. It’s one of the Grunt tasks: **brushfire/tasks/config/sails-linker.js.**

For example, any files with a **.css** extension placed in **brushfire/assets/styles** will automatically be linked to any page that contains the special Sails linker tags

****

The same is true for files with a .jsextension placed in the brushfire/assets/js/folder. The script tags for those JavaScript files will be automatically included on any page with the Sails linker tags

****

A quick look at the home page reveals that a script has also been added to the page.

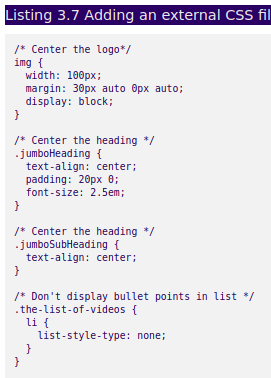
For now just know that any file added to **brushfire/assets/js/** will be linked to in HTML files that contain the special **<!—SCRIPTS-->** tags.

### **3.3.2 Built-in LESS support**

By default, a file named **importer.less** was created in the **brushfire/assets/styles/** folder.

In order for LESS files to be compiled and included automatically in the markup, you must first import them via **importer.less**

In Sublime, create a new file named **brushfire/assets/styles/custom.css** and add the styles in listing 3.7. ()

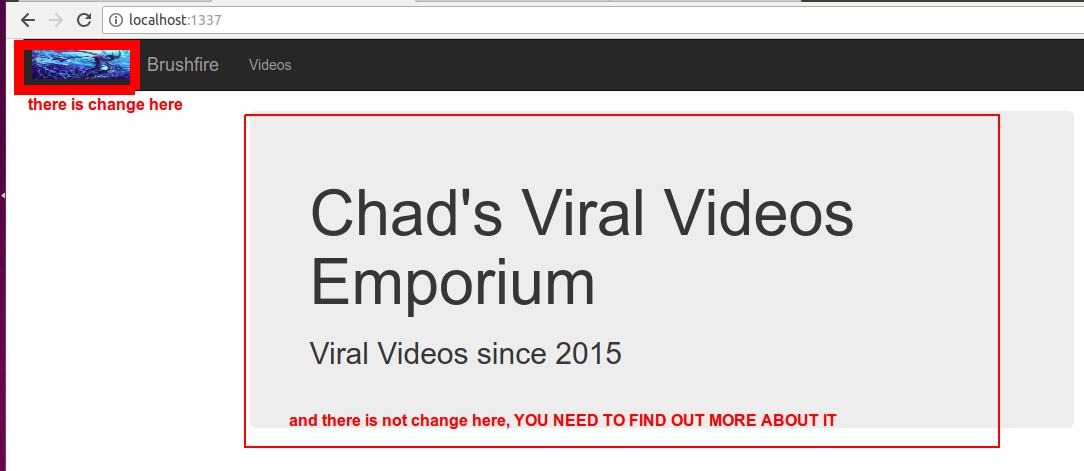
****

Next, open **brushfire/assets/importer.less** in Sublime and add

( you can NOT write this line in **brushfire/assets/importer.less**)



Restart sails and you can see:

****

## **3.4 Front-end first API design**

### **3.4.1 Identifying back-end requirements**

### the videos page has two requests that require us to simulate back-end responses—one for video submissions, and the other to load the initial video list.

### 

## **3.5 Using Sails with jQuery**

## Integrating jQuery as a dependency is easy. Head over to [**http://jquery.com/**](http://jquery.com/)and download the latest production/compressed version of jQuery. Copy that file into the **brushfire/assets/js/dependencies/** folder, name it **jquery.min.js**, and you’re done. The jQuery library will now be added automatically to **brushfire/assets/index.html** and **brushfire/assets/videos/index.html**. We’re also going to use a popular JavaScript utility library called ***lodash.js***.

## Navigate to [**https://raw.githubusercontent.com/lodash/lodash/4.5.0/dist/lodash.core.min.js**](https://raw.githubusercontent.com/lodash/lodash/4.5.0/dist/lodash.core.min.js)and copy the contents of the page to a new file in **brushfire/assets/js/dependencies/lodash.js***.*

### **3.5.1 Example: listing data with jQuery**

### Currently we have the initial videos hard-coded in the mark-up. Instead, we’ll load them from an array using jQuery.

### Open **brushfire/assets/videos/index.html.**

Next, we’ll create a new file for the jQuery code that will load the video list. In Sublime, create a new file named **brushfire/assets/js/videos-page.js** and add the jQuery in listing 3.9

