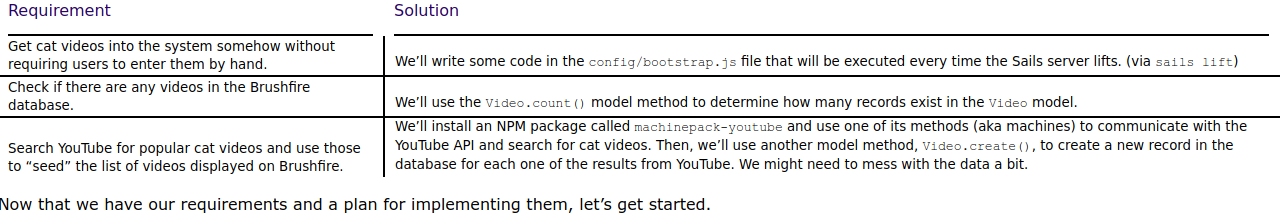
**CHAPTER 5**

The problem is all of Brushfire’ videos **have to be entered by hand**. To over come this issue we must find a way to “seed” the database.

Be careful not to confuse the Sails bootstrap function with the popular Bootstrap CSS Framwork. The bootstrap function in Sails is where you can put custom code if you want it to run when the Sails server lifts.



## **5.3 A deeper understanding of model methods**

Remember back in chapter 4, when we ran **sails generate video** from the command-line? We already saw how this created a model definition file in **brushfire/api/models/**, and we witnessed the effect that it had on the blueprint API. But there was another handy side effect: When we lift our app, Sails builds a JavaScript dictionary called Video and exposes it as a global variable.

Besides **Video.count()**,the **Video** dictionary provides a smorgasbord of other methods for fetching, analyzing, and manipulating the videos stored in our database.. We’ll look at another one of those, **Video.create()**

**5.4 Introducing machinepacks**

**machinepack-youtube** will allow us to interact with the YouTube API.

**5.4.2 Installing a machinepack**

**~/brushfire $npm install machinepack-youtube --save**

Dowload source files and install them in **/brushfire/node\_modules/**

Since we added – **flag,** the dependency was also recorded in **/brushfire/package.json**

### **5.4.4 Using machines**

There are two ways to delete records individually altogether: in the **Sails REPL** (sails console) or simply by wiping the database altogether.

~**/brushfire $ rm –rf .tmp/localDiskDb.db**: wipe all of date in **default disk database.**

### **5.4.6 Setting your own custom configuration in local.js**

We could have just pasted our API key inline in the **bootstrap.js** file. But there is a better way: custom configuration

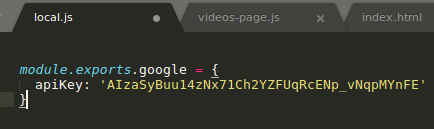
( **you must enable Youtube Data API to access to Youtube data, such as videos, playlists, and channels.**)

**local.js file:** a configuration file with special meaning. This file is designed for use in development, and it has two useful properties:

- First, any configuration we provide in **brushfire/config/local.js** **takes precedence** over the rest of the configuration files in our app.

- Second, this file is explicitly excluded from version control by the default **.gitignore** file that was created when we first generated our new Sails app

Let’s add our Google API key to the **local.js** file. In Sublime, create **brushfire/config/local.js** and add the following code in listing 5.4. Be sure to paste in your own Google API key.



Since we exported **google**, we’ll be able to access this dictionary as **sails.config.google** from anywhere in our application. Now let’s change our code in **bootstrap.js** to use grab the configured API key and pass it in to **.searchVideos()**.

### **5.4.7 Using custom configuration in your code**

### Instead of pasting our API key directly into **bootstrap.js.** We configured the API key using the **local.js** file

### Open **brushfire/config/bootstrap.js** in Sublime, call **searchVideos()**, and plug in the configured API key as shown in listing 5.5.

### 

### Open **brushfire/config/bootstrap.js**

## 

## If you’re thinking that Sails would refuse to lift, because there’s a bug in our code, then you’re right!

## Notice how we don’t call our own **cb()** (the callback from the bootstrap function) in either **successor** or **error**?

## **5.4.8 Understanding machine exits**

### **5.4.9 Using callbacks**

### Open brushfire/config/bootstrap.js

### 

### **5.4.10 Marshaling data**

### Open **brushfire/config/bootstrap.js**

### **\_.each()** is lodash function.

## **5.5 Creating multiple records**

## Open **brushfire/config/bootstrap.js**

## Use **Video.create()** to create multiple records to DATABASE

## **5.6 Summary**

When you implement asynchronuos functions, there are new rules:

- Trigger your callback.

- Attach callback function(s) to receive either the result or the error.

- bootstrap function which executes every time you lift your app, specifically, just before the Sails server start up, this is particularly useful for seeding initial records in your database.

- A special file in your app’s **config/** folder, called **local.js,** allow you to set custom configuration without checking it into version control. This is particularly useful for plugging in sensitive credentials, like API keys.

- By default, model methods are automatically exposed on global variables that correspond with the names of each of your models. This allow you to fetch, analyze, and manipulate records in your database, from custom code anywhere in your application.

\*) **bootstrap.js**: put custom code if you want it to run when the Sails server lifts.

\*) in your app’s **config/ folder**, called **local.js:** This is particularly useful for plugging in sensitive credentials, like API keys.