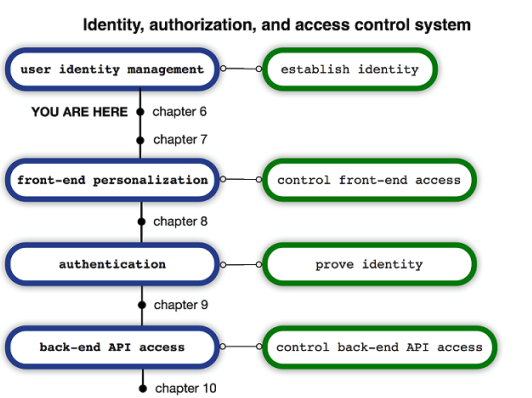
**CHAPTER 6**

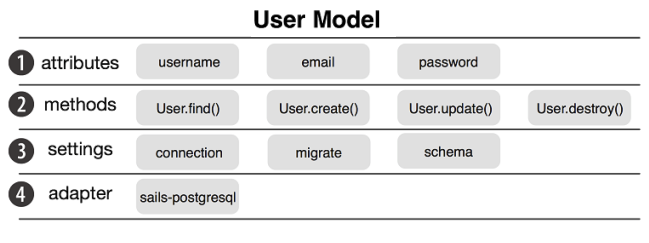
**USING MODELS**

****

**Figure 6.1 The four components of an identity, authentication, personalization and access control system.**

We’ll then transition Brushfire from using the default sails-disk database to a PostgreSQL database.

## **6.1 Understanding Sails models**

****

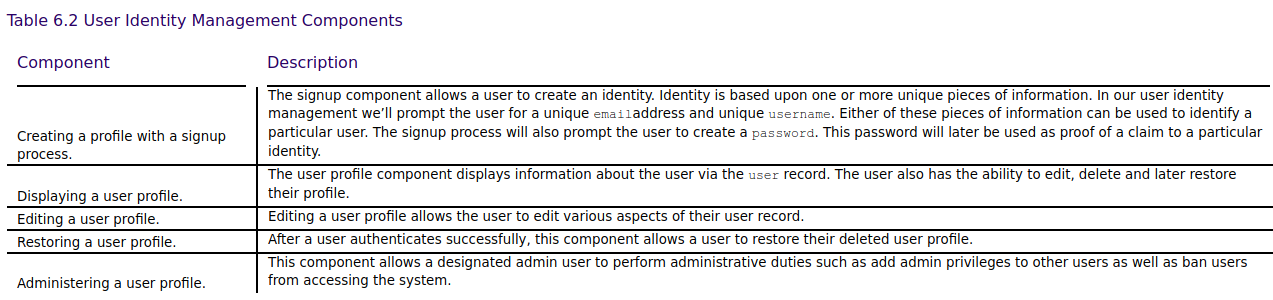
1) Model attributes are the properties of a user like the ***username***,***email***, and ***password****.*

2) Model methods are functions we use to find and manipulate database records.

3) Model settings are configuration settings for the model.

4) **Adapters are NPM packages** which you can install in your project in order to add support for a particular database. Behind the scenes the adapter is what allows Sails to provide one unified way of configuring, accessing and managing a model. Sails takes care of translating this unified approach to the specific requirements of each database system.

## **6.2 Managing user data**



### **6.2.1 Obtaining the example materials for this chapter.**

Copy the clone URL from the repo page. From the terminal window type the following command

~/brushfire$ git clone https://github.com/sailsinaction/brushfire-ch6-start

Change into the **brushfire-ch6-start** folder.

~/brushfire $ cd brushfire-ch6-start

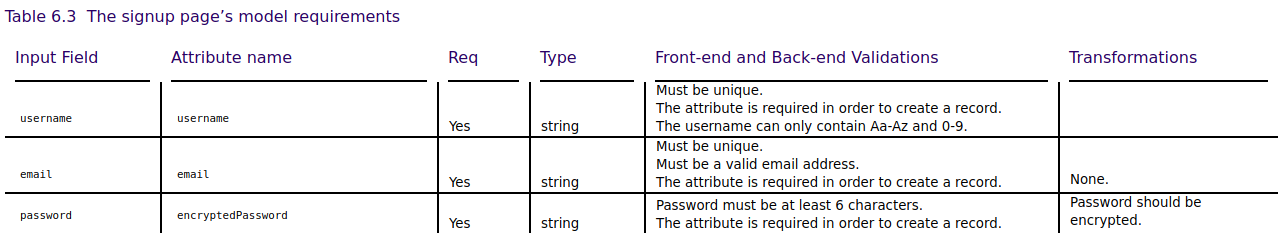
Next, we’ll install the node modules listed in the **brushfire-ch6-start/package.json** file. From the terminal window type:

~/brushfire-chp6-start $ npm install

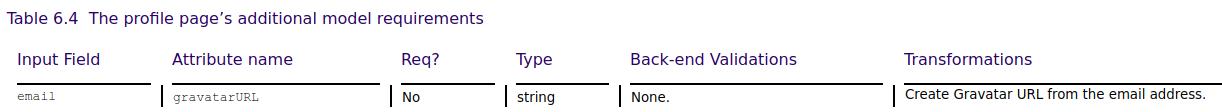
In Sublime, copy the **brushfire/config/local.js** file you created in chapter 5

### **6.2.2 A front-end first approach to data modeling**

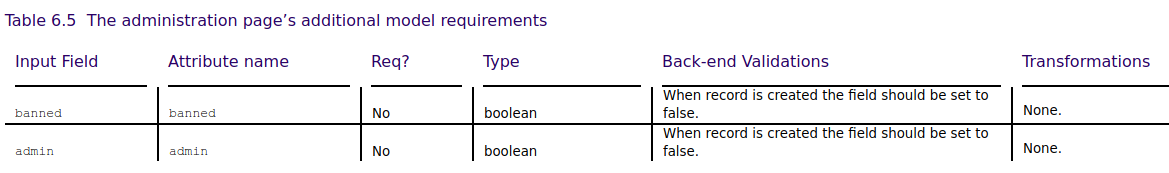
### **6.2.3 Building a signup page**



### **6.2.4 Building a user profile page**

****

### **6.2.5 Building an admin interface**

****

### **6.2.6 Recovering data after a soft delete**

## **6.3 Creating a new model**

### **6.3.1 Running the generator**

Let’s generate an API

~/brushfire-chp6-start $ sails generate api user

**6.3.2 Creating our first record**

1) Our ***User***model is currently empty and doesn’t contain a **connection** => find **connection**

2) Because there’s no **connection**, the default ***localDiskDb* connection** is used.

3) ***localDiskDB*** is set to use the ***sails-disk* adapter**.

- Typically the **connection** dictionary will contain *host*, *username,* *password*, *name* of the *database*, and *adapter.* In this case, *adapter* is ‘**sails-disk**’

Why did we use the hash **(#)** symbol in the path of our browser request? HTTP will ignore anything after the hash **(#)** symbol. This allows other frameworks like Angular to come up with their own routing strategy. So the **/signup** path of **/#/signup** is actually being processed by Angular’s router and not by the backend Sails router. **The Angular router then determines which template file to display**. In this case it’s our **brushfire/assets/templates/signup.html** file.

Sails looks for the database connection it will use to store and manipulate records for a particular model in ① ***brushfire/api/models/User.js***. If it doesn’t find a connection it then looks to ② ***model settings.*** If no connection exists Sails looks to the internal *core default connection* ③ ***localDiskDb****.* The default connection uses the ④ ***sails-disk***adapter to access the ***sails-disk database****.*

The **database name** we’re referring to above isn’t the actual name of a database system like PostgreSQL, MySQL, or MongoDB. The database name we’re referring to is whatever arbitrary name you provide for your database like brushfire, mydatabase, etc.

Adapter: Sails will use it to connect to database.

An adapter is a bit of code that maps model methods like ***find()*** and ***create()*** to a lower-level syntax like SELECT \* FROM and INSERT INTO.   
 So **sails-disk** is an adapter that talks directly to the **sails-disk** database.

**Install PostgreSQL for Ubuntu**

Navigate to this website: <https://www.postgresql.org/download/linux/ubuntu/>

**1**. Open the terminal, Create the file ***/etc/apt/sources.list.d/pgdg.list***

and add a line for the repository:

**deb http://apt.postgresql.org/pub/repos/apt/** **xenial-pgdg main**

**2**. - Access **sources.list.d** folder.



- Import the repository signing key:

**wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -**



If the terminal appears “OK”, that means success

- Update the package lists

**sudo apt-get update**

****

**3.** Install PostgreSQL 10

**sudo apt-get install postgresql-10**



**Use PostgreSQL in Ubuntu**

**1. Access postgresql:**





**2. Create Database**

****

**3. Exit postgresql:**

****

So to start the transition we need to install the **PostgreSQL adapter**

****

### **6.4.2 Configuring a database**

**Access postgresql:**





**Create Database**

****

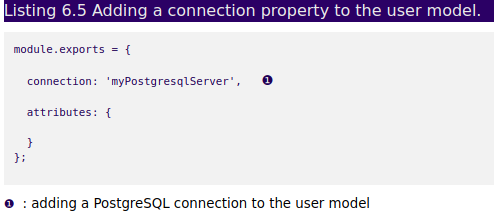
So to start the transition we need to install the **PostgreSQL adapter**

****

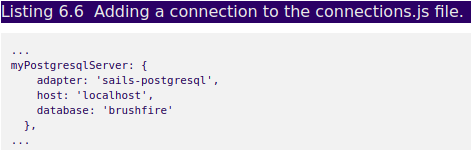
In Sublime open ***brushfire-chp6-start/package.json*** and take a look at the dependencies property

****

In Sublime open the ***user*** model located in ***brushfire-ch6-start/api/models/User.js***. Add a ***connection*** property in listing 6.5.

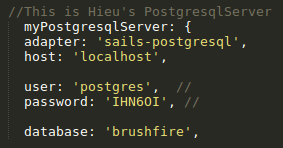


In Sublime, open **/brushfire-chp6-start/config/connections.js** and add the following configuration information to **myPostgresqlServer** shown in listing 6.6.



**==>** You **CAN NOT sails lift** if you config like above image.

You **MUST** config Sails like this:

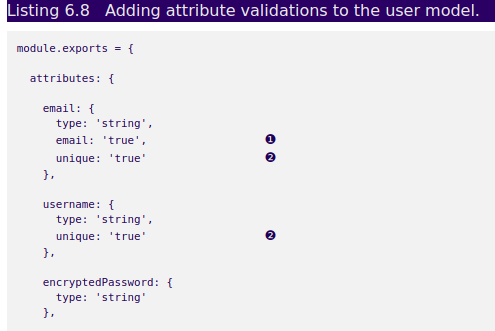
==> you need to **add** 2 fields: **user** and **password** of postgresql

### **6.4.3 Defining attributes**

From Sublime open **brushfire-chp6-start/api/models/User.js** and add the following model attributes in listing 6.7



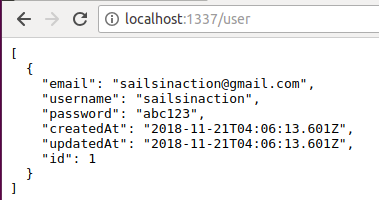
### **6.4.3 Attribute validation**

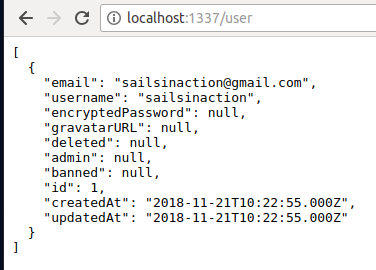
Our first attribute validation is a requirement for both the **email** and **username** attributes to be **unique**. What that means is that no record can contain an identical **email** or **username** in the database

❶ setting the **email** option to true

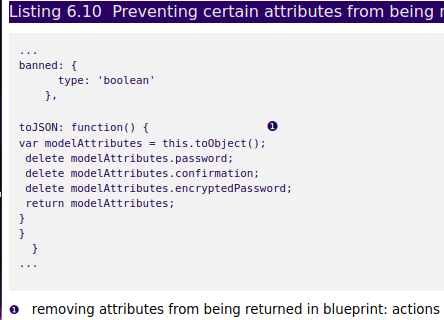
❷ requiring each attribute value to be unique

When you use **sails-disk,** which didn’t require that an attribute be defined before it could be used.

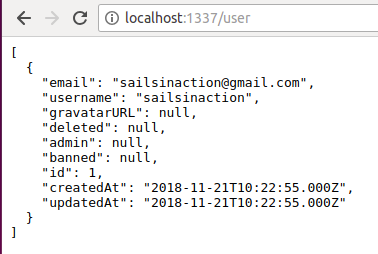
  
and when you use PostgreSQL, any parameter not defined as an attribute will not be stored in the **user** record.



We can limit the attributes defined in the model and returned by a blueprint action by overriding the **.toJSON()** method in the model. Open **brushfire-chp6-start/api/models/User.js** in Sublime and add the following **toJSON** method in listing 6.10.

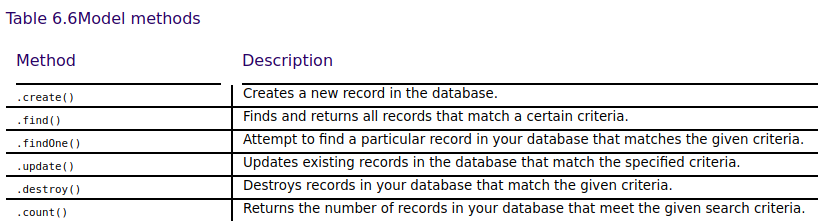


After overriding the **toJSON()** method, the blueprint create action will no longer return the **password**, **confirmation**, or **encryptedPassword** ,**parameters** to the front end.



## **6.5 Understanding model methods**

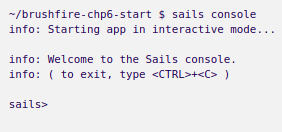
In chapter four we used the **create()** and **find()**methods indirectly to list and create **video** records via the blueprint create action and blueprint find action. In chapter five, we used the **Video.create()** and **Video.count()** methods directly. Let’s now look at the model methods we’ll use most in Brushfire and listed in table 6.6.



The **Sails console** is a way to start the Sails server in a project and then interact with it in the Node **Read-Eval-Print-Loop (REPL)**.

A REPL is an interactive tool that allows you to interact with a programming environment. In this case Node and Sails.

This means you can access and use all of your models to try out various queries during development without having to add them in a controller action and restart the Sails server each time. If your Sails server is currently running, close it by typing **ctrl-c** (twice). To start the Sails console, open a terminal window and from the root of your project type



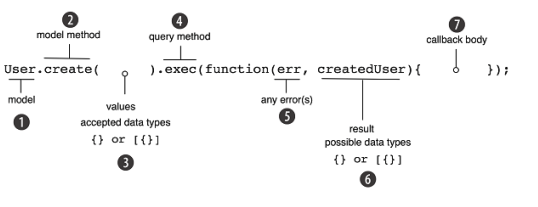
### **6.5.1 Anatomy of a Sails model method**

### First, we’ll take a look at the essential syntax of a Sails model method as illustrated in figure 6.16.

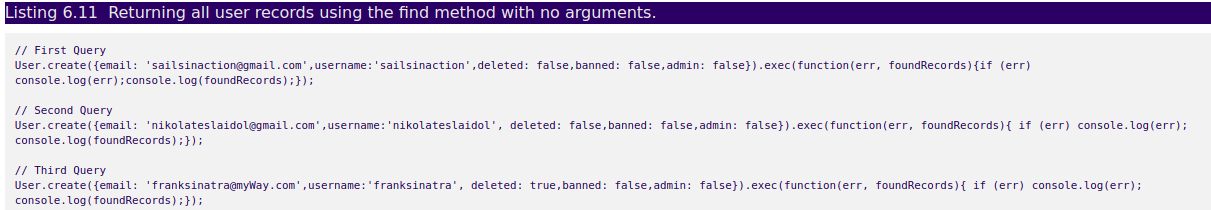
### 

Figure 6.16 The generic syntax of a model method includes ① the model name, ② model method, ③ criteria, ④ values, ⑤ query method, the callback method with ⑥ error and ⑦ result arguments and ⑧ callback body.

### **6.5.2 The .create() model method**

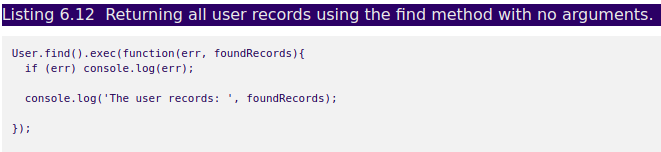


Let’s create a few records we can use to explore our model methods. Make sure the Sails console is running and type or copy each query in listing 6.11.



### **6.5.3 The .find() model method**

The find method returns all records that meet the criteria passed as the first argument of the method. The criteria can be a dictionary, a string or a number of the **id** you are trying to find. If no criteria argument is given, all records will be returned. Let’s give this a try. Make sure the Sails console is running and then type or copy and paste the query in Listing 6.12.

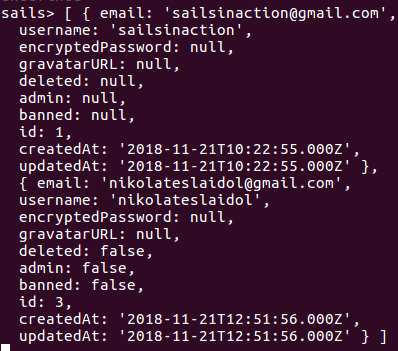


The terminal window should return results similar to listing 6.13.

  
Next, let’s find a particular **user** record by passing in a criteria dictionary as the first argument. Copy and paste the query in listing 6.14 into the sails console.

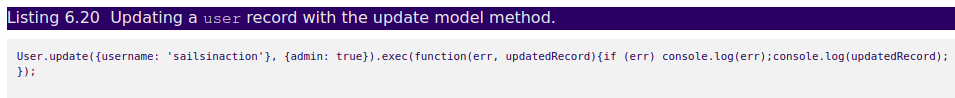


The console should return results similar to listing 6.15.

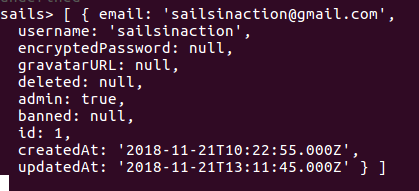


### **6.5.4 The .update() model method**

We already explored the syntax of the update model method at the beginning of this section. Now let’s see it in action. For example, we’ll make the **user** record with the **username sailsinaction** an administrator by updating the admin property to **true**. With the Sails console running type or copy and paste the following query in listing 6.20 into the Sails console.

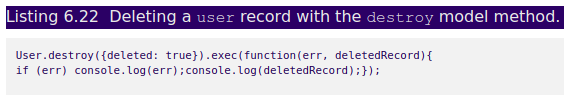


The console should return results similar to listing 6.21.

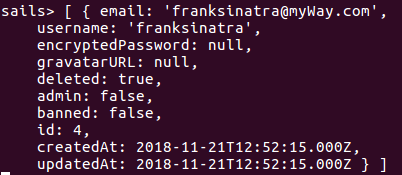


### **6.5.5 The .destroy() model method**

As its name implies, the destroy model method can delete one or more existing records in the model based upon the criteria provided as the first argument. The criteria can be a dictionary or an array of dictionaries. The criteria can also be a string or number of the **id** you are trying to destroy. Let’s say we want to delete any records that have their **deleted** property set to **true**. Assure that the Sails console is running and then type or copy and paste the following query in listing 6.22 into the Sails console.

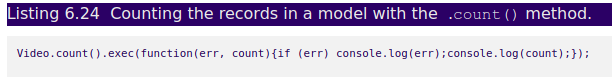


Your terminal window should look similar to figure 6.23.



### **6.5.6 The .count() model method**

The count model method returns the number of records in a particular model.  Assure that the Sails console is running and then type or copy and paste the following query in listing 6.24 into the Sails console.

  
**6.6 Summary**

· Sails models contain attributes, methods, settings, and an adapter named around a common resource.

· Model requirements consist of attributes, validations and transformations.

· Model requirements are identified using the front-end-first approach by reviewing interactive mockups.

· Models connect to a database using a connection that points to an adapter, which translates a common query interface into the specific syntax of the underlying database