CHAPTER 2

Overview of Ext JS 4

Let's get started with Ext JS 4. In this chapter you'll learn Ext JS 4 from scratch. We'll install it, configure it, and churn out some code to get a feel for the API.

Downloading Ext JS 4

Getting started with Ext JS 4 is really easy. You can download the latest stable version of Ext JS 4 from http://www.sencha.com/products/extjs/download/. You have two options that you can choose for downloading.

- *Commercial trial version*: You can download a 45-day commercial evaluation version of Ext JS 4. You can buy commercial licenses for Ext JS 4, based on your project needs, from https://www.sencha.com/store/extjs/.
- *GPL version*: Ext JS 4 is available under the General Public License (GPL) version 3. You can use this for building open source projects. Note, however, that Ext JS 4 applications built using the GPLv3 license require the release of the source code.

Just visit http://www.sencha.com/products/extjs/license/ to read more about Ext JS 4 licensing options. No matter which version you choose, you'll get a zip file of the entire Ext JS 4 library.

The examples in this book were tested against version 4.2 of Ext JS 4.

Getting Started With Ext JS 4

Extracting the Ext JS 4 zip will give you the contents shown in Figure 2-1.

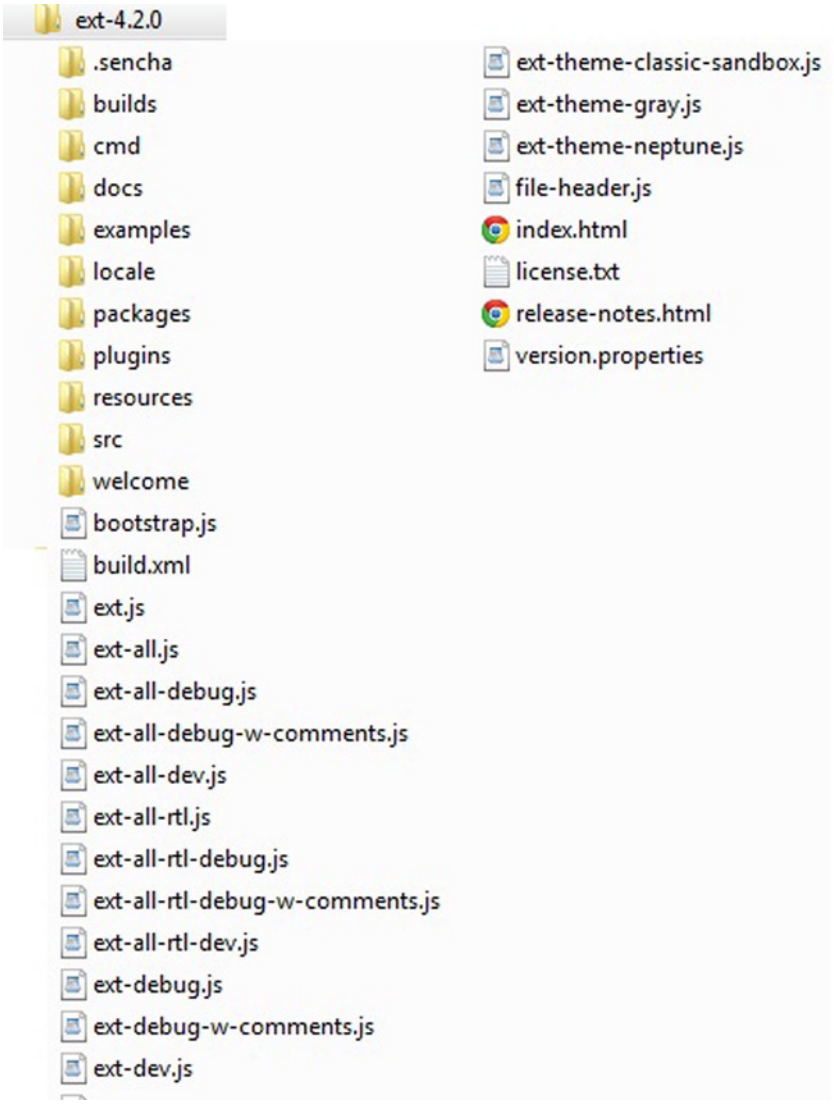

Figure 2-1. *Contents of the Ext JS 4 folder*

Let me give you a quick overview of the contents of the folder.

- The *resources* folder contains the standard set of themes that you can use in your application.
- The *src* folder contains the entire API organized into numerous JavaScript files.
- The *docs* folder contains the API documentation and guides.
- The *ext.js* file contains the core API.
- The *ext-all.js* file contains the complete API in a compressed or minified format. It's a large-sized file that is suitable for development purposes only.

- The *ext-all-dev.js* contains the complete API with comments.
- The *ext-all-debug-w-comments.js* contains the complete API with comments and console warnings. This file is meant to be used for development.

The Ext JS 4 zip file can be extracted to a web server's directory to access the documentation as a web application. I have a Microsoft Internet Information Services (IIS) server running on my Windows machine. I extracted the zip archive into the IIS root directory C:\inetpub\wwwroot folder. Figure 2-2 shows the index.html file, one of the documents loaded when you access it from a web browser.

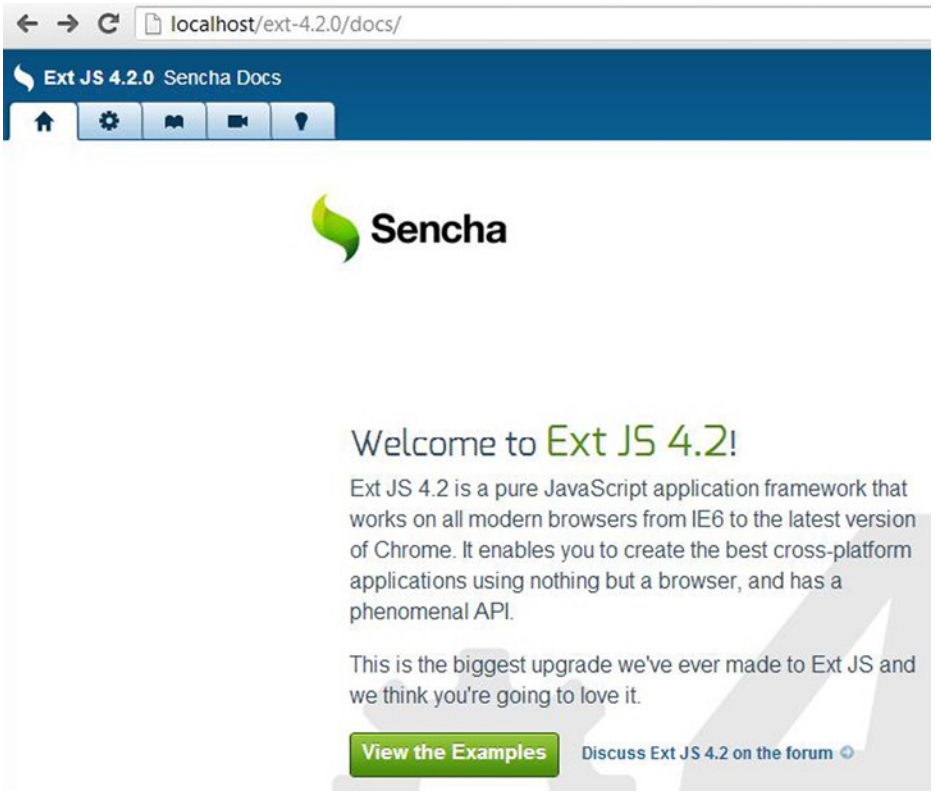

Figure 2-2. *Documentation*

IDE for Ext JS 4 Development

Sencha has a product known as Sencha Architect that can be used to design Ext JS 4 applications. Sencha Architect is not a full-fledged editor, but it is very useful to design the UI and then use the generated code in the main copy of the application.

You can use any simple JavaScript editor for working with Ext JS 4. The choice usually depends on the server side technology you decide to use while working with Ext JS 4 applications. If you're a Java developer, you can use Eclipse or Net Beans or IntelliJ IDEA. If you're a .NET developer, Visual Studio is all that you need. The same applies to Rails or Grails as well.

Ext JS 4 applications run on almost all major browsers. Chrome, Safari, or Firefox is generally preferred during development because of the debugging facilities they provide.

Let's create an Ext JS 4 application and crack some Ext JS 4 code.

Hello World With Ext JS 4

Let's create an Ext JS 4 application called *Chapter02*. We'll create a folder called *extjs* in Chapter02. The *extjs* folder will contain the Ext JS 4 library files that we need for developing the application. We'll copy the necessary resources from the Ext JS 4 folder that we've extracted earlier and put them in the *Chapter02/extjs* folder. The resources include the js files, css files, and the default set of images. You don't really have to call the folder extjs, though; it's just a convention. You can name it scripts or lib or resources, or anything that you feel is appropriate.

Let's create an index.html file and write some Ext JS 4 code in it. The structure of the *Chapter02* application is shown in Figure 2-3.

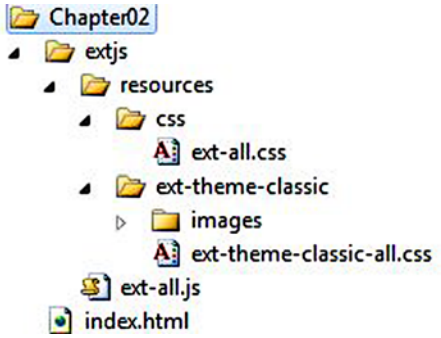

Figure 2-3. *Structure of Chapter02 application*

The application contains an *extjs* folder where we have copied the ext-all.js file from the Ext JS 4 library folder. The ext-all.js file contains the complete Ext JS 4 API code. While you may not really need the entire API in a project, this is adequate to get started with the coding. The *resources* folder contains the css files with the default theme folder `ext-classic-theme`. The classic theme is the standard theme that gives a bluish look and feel to the entire application. You can change it if you want, but let's not worry about that right now.

Let's play with the index.html file by adding references to the ext-all.css and ext-all.js files. And let's display a Hello world in an alert box. Listing 2-1 shows the code for index.html.

Listing 2-1. index.html File

```
<!DOCTYPE html>

<html>
<head>
    <link href="extjs/resources/css/ext-all.css" rel="stylesheet" type="text/css" />
    <script src="extjs/ext-all.js" type="text/javascript"></script>
    <script>
        Ext.onReady(function () {
          Ext.Msg.alert("Hello World","All set!!!");
        });
    </script>
</head>
<body>
</body>
</html>
```

Running index.html page in the browser will pop up a window with the title *Hello World*, as shown in Figure 2-4.

Figure 2-4. *index.html*

The starting point of the Ext JS4 application is an `Ext.onReady()` function. The `onReady()` function takes a function as an argument. The `onReady()` function is called when the document is loaded and DOM is ready. We've created a simple dialog box using `Ext.Msg.alert()` method.

If you want to display hello world in a label, let's modify the `onReady()` method in Listing 2-1 as shown below. We can create an object of the Label class provided by Ext JS 4 as shown below.

```
Ext.onReady(function () {
  Ext.create("Ext.form.Label", {
    text: "Hello World",
    renderTo: Ext.getBody()
  });
});
```

We create an object of the class `Ext.form.Label` and render it to the document body. If you have to display a *Hello World* button, we'll create an object of the `Ext.Button` class and render it to the document body as shown below.

```
Ext.onReady(function () {
  Ext.create("Ext.Button",{
    text : "Hello World",
    renderTo : Ext.getBody()
  });
});
```

One noteworthy aspect of Ext JS 4 code is its object-oriented approach. All the UI components are available as classes, and we just need to create instances of the classes and render them.

Let's add an event handler to the button we created. Clicking the button will pop up a message as shown in Listing 2-2.

Listing 2-2. Button With the Click Event Handler

```
Ext.create("Ext.Button",{
  text : "Hello World",
  handler : function(){
    Ext.Msg.alert("You clicked the hello world button");
  },
  renderTo : Ext.getBody()
});
```

Now, let's create a textbox and a button and render them to the body of the document. We can create a container such as a panel and add textbox and button components as shown in Listing 2-3.

Listing 2-3. Panel With a Textbox and Button

```
Ext.create("Ext.Panel",{
  title : "Hello World Panel",
  items : [
            Ext.create("Ext.form.field.Text",{
              fieldLabel : "Name"
            }),
            Ext.create("Ext.Button",{
              text : "Click"
            })
  ],
  renderTo : Ext.getBody()
});
```

The `Ext.Panel` class has an array property called `items`. The `items` array holds the list of components that need to be added to the panel. The output of Listing 2-3 is shown in Figure 2-5.

Figure 2-5. *Panel with a textbox and a button*

If you want to display the value of the textbox, when the button is clicked, let's modify the code in Listing 2-3 and add an id to the textbox and access it like this.

```
Ext.create("Ext.form.field.Text",{
 fieldLabel : "Name",
 id:"nametext"
}),
Ext.create("Ext.Button",{
 text : "Click",
 handler : function(){
   Ext.Msg.alert(Ext.getCmp("nametext").getValue());
 }
})
```

We've used the `Ext.getCmp()` method to access the UI component by specifying the id. You can then invoke the `getValue()` method on the text box to fetch the value.

I hope this short introductory chapter has given you a basic feeling for Ext JS 4 coding style. The intent is to get you started with some Ext JS 4 code without really worrying about the syntax. You'll learn more about the API in the subsequent chapters. As we delve deeper into the API, you'll find various different options available for creating even a simple component like a button.

At this point if you are able to view an Ext JS 4 textbox and button in your browser you're good to go to the next chapter.

Summary

In this chapter you got started working with Ext JS 4. We extracted the Ext JS 4 library and created a simple Ext JS 4 application by copying the necessary CSS and JavaScript files. `Ext.onReady()` is the starting point in an Ext JS 4 application. All the UI components are available as classes. You saw how to create simple UI components such as a textbox, a button, and a label. I introduced you to adding simple event handlers to components such as buttons. I also discussed container components like Panel where you can add child components as items.

In the next chapter you'll learn about the structure and syntax of the Ext JS 4 API.