Ext JS 6 By Example

Ext JS is one of the most famous JavaScript frameworks used to create rich interactive web applications using techniques such as Ajax, DHTML, JSON, and DOM scripting.

This book starts with a clear step-by-step installation guide for Ext JS 6 and the additional tools required for development on both the Mac and Windows operating systems.

We move on to focus on multiple basic GUI controls and a sample project that covers the fundamental and basic GUI controls. We proceed to show you how to organize your code by using the MVVM application architecture.

You'll also learn about the API required for AJAX and REST communication. A working RESTful API sample project is included to help you explore the data packages. Finally, we end with a discussion on responsive design, theming your application, and the available plugins.

Who this book is written for

If you’re a front-end web developer who is looking to learn a new JavaScript framework, or if you already know about Ext JS and are looking for a practical resource with multiple example projects in Ext JS, then this guide will be a great resource for you. A basic understanding of HTML, CSS, and JavaScript is expected.

$ 44.99 US
£ 28.99 UK

Prices do not include local sales tax or VAT

What you will learn from this book

- Install Ext JS Sencha Cmd 6 and debugging tools such as Illumination, App Inspector, and Sencha Fiddle
- Understand and apply the core concepts and querying capabilities of Ext JS 6
- Dive deep into powerful components such as grid, data view, and trees, and different kinds of charts including 3D charts, as well as bar, area, pie, and line charts
- Discover how an Ext JS application communicates with the server side through data packages such as stores and proxies
- Discover how to create a responsive web application
- Acquire and implement knowledge on accessibility, localization, and the drag-and-drop and theming capabilities of Ext JS 6

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Getting Started with Ext JS'
- A synopsis of the book’s content
- More information on Ext JS 6 By Example
Anand Dayalan is a software engineer who resides in the U.S. He currently works for Microsoft in Seattle. Anand has extensive experience and passion in developing scalable and cool-looking enterprise web applications.

He started building web applications in 2005 and has built numerous enterprise web applications, including an e-commerce web application, using Ext JS and Microsoft technologies.

Anand enjoys watching movies, listening to music, reading, cooking delicious food, and (most importantly) spending time with his family.
Preface

Ext JS is one of the most famous JavaScript frameworks available on the market. Ext JS is almost like a one-stop shop for developing rich UI applications. This book is all about the powerful Ext JS application framework and is written for developers who like to see less theory and more code examples and sample projects.

This book will help you to understand the concepts quickly and reduce your learning curve. The working source code and explanation of projects will help you to learn code quickly and easily.

The book starts with an introduction to Ext JS and talks about how to set up the development environment with the installation guide for Ext JS 6, Sencha Cmd, and other required tools. Each chapter starts with some concepts of Ext JS 6 and ends with a sample project using the concepts you have already learned.

What this book covers

Chapter 1, Getting Started with Ext JS, introduces you to Ext JS and Sencha Cmd. It provides a clear, step-by-step installation guide for Ext JS 6, Sencha Cmd, and other tools required for development in both Mac and the Windows operating system.

Chapter 2, Core Concepts, discusses some of the core concepts of Ext JS 6. Understanding these concepts is very important before we start building a sample application. You’ll learn about the class system and how to create and extend classes, events, layouts, and containers.

Chapter 3, Basic Components, covers some of the basic components available in Ext JS 6. It uses the concepts learned in the previous chapters and in this chapter to create a simple project.
Chapter 4, Data Packages, explores the tools available in Ext JS 6 to handle data and the communication between the server and client. It concludes with a sample project using a RESTful API.

Chapter 5, Working with Grids, talks about the different types of grid components in Ext JS 6 at an advanced level. It covers concepts such as pagination, sorting, filtering, searching, row editing, cell editing, grouping, docking toolbars, buffered scrolling, column resizing and hiding, grouped headers, multiple sort grids, row expanders, and so on. It also tells you how to build a sample project called company directory.

Chapter 6, Advanced Components, covers advanced components in Ext JS 6, such as trees and data views. It introduces you to a sample project called picture explore that is built with trees and data view components.

Chapter 7, Working with Charts, talks about the different types of chart components in Ext JS 6. It concludes with a sample project called expense analyzer.

Chapter 8, Theming and Responsive Design, focuses on the basics of how to theme your Ext JS application and responsive design. It also introduces you to SASS.
Gone are the days when you used plain Vanilla JavaScript. JavaScript is a great, powerful language, but many find it difficult to maintain the code as the web application gets bigger and bigger. So, it's very difficult and time consuming to handle everything in plain Vanilla JavaScript.

When it comes to JavaScript frameworks, there are client-side JavaScript frameworks as well as server-side JavaScript frameworks. Ext JS is a client-side JavaScript framework.

This chapter is pretty much about setting the development environment with the required tools, and we'll see the basic introduction of Ext JS. In this chapter, we will cover the following topics:

- The advantages of using Ext JS
- An introduction to Ext JS
- Setting up Ext JS and Sencha Cmd
- Scaffolding the Ext JS application with Sencha Cmd
- The application architecture
- Exploring Sencha Cmd commands
- Debugging an Ext JS application
- The development IDE

Why Ext JS?
Now, let's take a look at some of the advantages of using Ext JS over plain Vanilla JavaScript in your web application.
Cross-browser support
You may have spent several hours or even days solving the cross-browser bugs. Why should you spend time for this instead of focusing on your business functionality? Rather, if you use a good JavaScript framework, such as Ext JS, it will take care of most of these tasks, and you can focus on the business functionality.

Rich UI components
Ext JS comes with a huge number of rich UI controls, such as data grid, tab panels, tree controls, date picker, charts, and so on, and these should reduce your development time a lot.

Two-way binding
Two-way binding means that when the view data changes, your model gets updated automatically. Also, when your application updates the model, the data will be propagated to the view automatically.

For example, take the edit page. When you load the page, you have to render the data from the model to the HTML, and when the user updates the data in the view, you have to update your model. You don't have to do these programmatically yourself if you use Ext JS.

Architectural pattern for JavaScript
As more and more code is moved to the client, maintaining the client side, JavaScript becomes difficult. By bringing MVC (Model View Controller)/MVVM (Model View, View Model) to the client side, it becomes easier to maintain the client-side JavaScript code, and it increases the productivity. MVC and MVVM are explained later in this chapter.

Simplifying a complicated operation
Think about writing the AJAX call in plain JavaScript and making sure to support all the browsers as well. Take a look at the source code of the AJAX call method in any of the JavaScript framework. Think about creating a complex control like grid in plain JavaScript with features, such as pagination, sorting, filtering, grouping, keyboard navigation, editable fields, and so on.
Easy DOM access
In plain JavaScript, you can access the DOM, but it is a bit complicated.

Client-side routing
In web applications, routing means mapping of the URL to web pages and the logic behind it. Routing can be on the server side as well as on the client side. Typically, client routing is used in a Single-Page Application (SPA).

Support for accessibility
Accessibility means that the content of application must be easily accessible to people who are visually impaired and depend on assistive technologies (such as screen readers). Developing an application with very good support for accessibility is very difficult.

In USA, if you are developing software that will be used by federal and state employees, then in most cases, you have to make sure that your application is accessible as per Section 508. Very few JavaScript frameworks provide very good support for accessibility. Ext JS provides excellent support for accessibility.

World Wide Web Consortium (W3C) has created a technical specification called WAI-ARIA (Web Accessibility Initiative - Accessible Rich Internet Applications). This defines ways to make web application accessible for people with disabilities. Ext JS has excellent support for this, and all the controls/widgets in Ext JS are accessible and don't require any additional code or work from you.

An introduction to Ext JS
Ext JS is almost a one-stop shop to develop rich UI applications. It provides MVC, MVVM, two-way binding, cross-browser compatibility, routing, an extensive set of rich UI components, charts, and so on. Ext JS also has an excellent documentation for all the APIs in the framework. Ext JS is originally built as an add-on library extension of YUI by Jack Slocum, it is now a product of Sencha Inc.

In Ext JS, you'll write most of your code in JavaScript. Mostly, you do not need to write HTML. Ext JS ships with a huge set of rich UI components, which is a huge time save in your development.
All the example code and sample projects code in this book will use the latest Ext JS Version 6, but still the majority of the code are compatible with the previous Ext JS Version 5. Most concepts in Ext JS 5 and Ext JS 6 are the same. So, if you are using Ext JS 5, you can still get a great benefit out of this book. However, remember some of the code in this book will not run in Ext JS 5 and may need some minor modification to make it run in Ext JS 5.

The most important change in Ext JS 6 is that it merges two frameworks: Ext JS and Sencha Touch into one single framework. Ext JS 6 also brought a new SASS compiler called Fashion, 3D chart improvements, and so on.

To understand why the merge of Ext JS and Sencha Touch happened, we need to look back a bit.

Sencha Touch was a separate product that specialized in creating touch applications for mobiles and tablets. It leverages hardware acceleration techniques to provide high-performance UI components for mobile devices.

Ext JS 4 and Ext JS 5 are used mainly to develop web applications for the desktop. If you have created a web application for the desktop in Ext JS 5 or Ext JS 4, which will still work in a mobile and tablet but it won’t have some of the touch-specific functionalities and won’t leverage hardware acceleration techniques to provide high-performance UI components for mobile devices. So, to better support mobile devices, Sencha developers were told to use Sencha Touch.

There are many advantages of using Sencha Touch. Applications written in Sencha will have the native look of the mobile platform, and the performance will be better. However, many developers had a complaint about this because they were forced to maintain two set of code base for the same application.

Although Sencha Touch and Ext JS are totally a different product with many differences, they had a common shared code; the concepts and ideas of both the frameworks are very similar. If you know Ext JS, then it is extremely easy to learn Sencha Touch.

For long, manyExt JS and Sencha Touch users were asking why not merge both the products into a single product and bring the touch capabilities to Ext JS. In Ext JS 6, Sencha made the decision to merge both the products into a single product.

Now, in Ext JS 6, you can maintain a single code. For some of the views, you may need to have a separate view code, but there will be a lot of shared code.
In Ext JS 6, they merged the common code and put them as a core framework, and they brought a concept called toolkit. A toolkit is a package with visual components, such as button, panels, and so on. There are two toolkits: classic and modern. Ext JS visual components are placed in the classic toolkit, and Sencha Touch components are placed in the modern toolkit.

Now, in Ext JS 6, you can simply choose the toolkit that you want to target. If you are writing an application that only targets mobile devices, you can choose modern, and if you are targeting only for desktop, then you can choose the classic toolkit.

**The universal application**

If you want to target both desktop and mobile devices, then in Ext JS 6, you can create a universal application, which will use both the toolkits. Instead of adding the preceding toolkit config mentioned before, you have to add the following builds config section that specifies which build uses which toolkit and theme:

```json
"builds": {
  "classic": {
    "toolkit": "classic",
    "theme": "theme-triton"
  },
  "modern": {
    "toolkit": "modern",
    "theme": "theme-neptune"
  }
}
```

The basic idea here is to have two set of toolkits in a single framework in order to target the desktop and mobile devices.

If you are totally new to Ext JS, these may be a bit confusing for you now, but don't worry about it now much. These will make more sense later when we work on the samples and example code.

Ext JS 6 ships with two set of themes for the classic toolkit and the modern toolkit. There are specific themes in Ext JS, which provides native looks for Windows, Android, and iPhone. You'll learn about theming later in *Chapter 8, Theming and Responsive Design*. 

---

[5]
Setting up Ext JS
To make your Ext JS application development easy, you have to install a tool called Sencha Cmd. It's available for Windows, Mac, and Linux.

Sencha Cmd is not a must for Ext JS' development application, but using it makes your life easier. So, it's highly recommended to use Sencha Cmd.

Sencha Cmd
Sencha Cmd is a powerful command-line tool for Sencha's application development. It helps in increasing the productivity by automating many tasks. Some of its features are scaffolding, package management, JS compiler, build scripts, theming, and so on.

Before installing Sencha Cmd 6, you need JRE, and if you're going to use ExtJS 5 and Sencha Cmd 5, then you'll also need Ruby.

Java Runtime Environment (JRE)
To check whether Java is running on your machine, type the following command in the terminal (Mac) or the command window (Windows):

```
java -version
```

If you have Java already running on your machine, then you should see something similar to the following code; otherwise, download and install the JRE or JDK:

```
java -version
java version "1.8.0_25"
Java(TM) SE Runtime Environment (build 1.8.0_25-b17)
Java HotSpot(TM) 64-Bit Server VM (build 25.25-b02, mixed mode)
```

If you still get an error, then you may need to add the installed path to the PATH variable.

Ruby
Note that you do not need Ruby if you are going to use Sencha Cmd 6, but if you are going to use Ext JS 5 and Sencha Cmd 5, then you will need Ruby. To check whether Ruby is installed on your machine, type the following command in the terminal (Mac) or the command window (Windows):

```
ruby --version
```
If you have Ruby already installed on your machine, then you should see something similar to the following code; otherwise, download and install Ruby:

```
ruby --version
ruby 2.0.0p481 (2014-05-08 revision 45883) [universal.x86_64-darwin14]
```

If you still get an error, then you may need to add the installed path to the PATH variable.

### Installing Sencha Cmd

Now, download and install Sencha Cmd from the Sencha website. Once installed, make sure that Sencha Cmd is available for use from the terminal or the command window. To check, run the following command:

```
 sencha which
```

If available in the terminal or the command window, then you should have something similar to the following code:

```
 Sencha Cmd v6.0.0.92
 /bin/Sencha/Cmd/6.0.0.92/
```

If you get an error instead, then you may need to add the installed path to the PATH variable. On Mac, run the following command in the terminal to add the installed path to the PATH variable:

```
 export PATH=~/bin/Sencha/Cmd/6.0.0.92:$PATH
```

In the preceding command, change `/bin/Sencha/Cmd/6.0.0.92` with the correct installed path.

On Windows, run the following command in Command Prompt to add the installed path to the PATH variable:

```
 set PATH=%PATH%;C:\Sencha\Cmd\6.0.0.92
```

In the preceding command, change `C:\Sencha\Cmd\6.0.0.92` with the correct installed path.
Generating the Ext JS application using Sencha Cmd

Open the terminal (Mac)/command (Windows) and type the following command:

```
sencha generate app --ext MyApp ./myapp
```

The preceding command will scaffold an Ext JS application called MyApp and place all the files in the subfolder called myapp.

Note that the preceding command will generate an app containing code for both toolkits: classic and modern. If you need either the classic or modern toolkit alone, then use `--modern` or `--classic`, as shown in the following command:

```
sencha generate app --ext --modern MyApp ./myapp
```

When you run this command for the first time, it should automatically download Ext JS 6. If it does not download Ext JS automatically, then you can manually download Ext JS 6, extract it to your local machine, and use the following command to specify the SDK path:

```
sencha -sdk /path/to/sdk generate app MyApp /path/to/myapp
```

Sencha Cmd supports Ext JS 4.1.1a and higher and Sencha Touch 2.1 and higher. You can have multiple versions of the SDK in your machine. The preceding is the format of the command to generate an application based on a specific Sencha SDK.

The following command is an example that will generate the Ext JS 6 application called MyApp under the /projects/extjs/myapp path:

```
sencha -sdk /bin/Sencha/ext/6.0.0/ generate app MyApp /projects/extjs/myapp
```

Now, to see the Ext JS application created, run the following command in the terminal or Command Prompt:

```
cd sample
sencha app watch
```
Now, this will perform a bunch of build-related tasks, and at the end, on the terminal window, you'll see something similar in Figure 1.1:

```
css build complete
loading widget definitions...
rendering widgets...
css build complete for ../../../build/temp/development/MyApp/sass/MyApp-all.sass[1]
[INF] Appending content to /Users/ananddayalan/projects/ExtJs/myapp/bootstrap.js
[INF] Writing content to /Users/ananddayalan/projects/ExtJs/myapp/classic.json
[INF] Waiting for changes...
```

Figure 1.1

The watch monitors any code changes made, and as soon as the code changes are saved, it will refresh the browser to include the updated code changes.

If you open the application with the default URL (http://localhost:1842), as shown before, the application will look similar to Figure 1.2:

![MyApp Personnel Table]

Figure 1.2
By default, when you navigate to the URL (http://localhost:1842) from a desktop computer, the app to show is automatically detected, and it shows you the classic toolkit. If this is accessed from a mobile browser, it will show you the modern toolkit. To see the modern app on the desktop computer, append ?profile=modern to the URL, and you'll see the following screenshot:

![Personnel Table]

The contents of MyApp will look as shown in Figure 1.4. We'll take a look at some of the important files of this sample application.

The app contains model, store, and application.js. Consider a store to be like a collection of instances of the model. The store loads data using the proxy and provides functionalities, such as sort, filter, paging, and so on. You'll learn more about the store later.
In the following screenshot, see the modern and classic folders. These folders contain the application code that uses the respective toolkits: modern and classic.

![Figure 1.4](image)

Figure 1.4 shows the content of the classic folder and the modern folder. The classic folder and the modern folder contains the src folder that contains application views. The main.scss file contains the styles specific to mobile devices and the desktop. There is the sass folder at the root that contains the common application style.
SASS (Syntactically Awesome Stylesheets) is a stylesheet language. SASS is heavily used in Ext JS. You'll learn more about styling and theming later in Chapter 8, Theming and Responsive Design.

Note that these are not the framework's toolkit code, but these are application code. You can find the framework's modern and classic toolkit code in the ext folder:

![Figure 1.5](image)

In the next section, we'll take a look at MVC and the content that some of these files generated using Sencha Cmd in the MyApp sample application.

**The application architecture**

Ext JS provides support for both MVC and MVVM application architectures.
Model
This represents the data layer. The model can contain data validation and logics to persist the data. In Ext JS, mostly model is used along with a data store.

View
This represents the user interface. So, components such as button, form, and message box are views.

Controller
This handles any view-related logic, event handling of the view, and any app logic.

View model
This encapsulates the presentation logic required for the view, binds the data to the view, and handles the updates whenever the data is changed.

Now, let's examine some of the files created by Sencha Cmd for the view, controller, and view model.

If you open app.js, you'll see the following code, which is the starting code for your Ext JS application:

```javascript
Ext.application({
   name: 'MyApp',

   extend: 'MyApp.Application',

   requires: [ 'MyApp.view.main.Main',
   'MyApp.view.main.Main'
   ],

   mainView: 'MyApp.view.main.Main'
});
```

**Downloading the example code**

You can download the example code files for all Packt books you have purchased from your account at [http://www.packtpub.com](http://www.packtpub.com). If you purchased this book elsewhere, you can visit [http://www.packtpub.com/support](http://www.packtpub.com/support) and register to have the files e-mailed directly to you.
Getting Started with Ext JS

In the preceding code, the first line defines the name of the application, and the following line extends MyApp.Application, which is declared in Application.js in the app folder:

```javascript
extend: 'MyApp.Application'
```

The list of classes required for this class has to be specified in the requires section. These will be loaded first before instantiating this class. The last line specifies the name of the initial view to create.

Next, if you check the app folder, you will see Application.js, and model, view, store, and so on.

In the application.js file, you'll see some code as follows:

```javascript
Ext.define('MyApp.Application', {
    extend: 'Ext.app.Application',
    name: 'MyApp',
    stores: [],
    launch: function () {
        // TODO - Launch the application
    }
});
```

Here, you can see that MyApp.Application extends Ext.app.Application. The launch function is defined in Ext.app.Application. This function is called after the page is loaded.

The store in application.js is nothing, but data stores. You'll learn about stores in detail later in the upcoming chapters.

View model – MainModel.js

Take a look at MainModel.js under \app\view\main\. This class is the view model for the Main view of the application. The view model extends from Ext.app.ViewModel, as shown in the following code:

```javascript
Ext.define('MyApp.view.main.MainModel', {
    extend: 'Ext.app.ViewModel',
    alias: 'viewmodel.main',

    data: {
        name: 'MyApp',
    }
});
```
Controller – MainController.js

This class is the view controller for the main view of the application. In the following code, you can see that the onItemSelected function is defined; this will be called when an item is selected from the grid in the view.

```javascript
Ext.define('MyApp.view.main.MainController', {
    extend: 'Ext.app.ViewController',

    alias: 'controller.main',

    onItemSelected: function (sender, record) {
        Ext.Msg.confirm('Confirm', 'Are you sure?', 'onConfirm', this);
    },

    onConfirm: function (choice) {
        if (choice === 'yes') {
            //
        }
    }
});
```

There are two types of controllers: Ext.app.Controller and Ext.app.ViewController.

You'll learn about its difference and usage later in the upcoming chapters.

View – main.js

If you have used Sencha Cmd 6, and if you generated the app only for classic or modern toolkits with `--modern` or `--classic`, then you'll find the main.js file under the `\app\view\main\` folder, but if you have used Sencha Cmd 6 to generate a universal application, then you can find two main.js files located under two paths: `\modern\src\view\main\` and `\classic\src\view\main\`. 
Before we see the contents of this file, let's go through the background behind these two main.js files under two different paths.

Earlier in this chapter, you learned how Ext JS 6 merges Sencha Ext JS and Sencha Touch into one framework. As a result, a single framework is created with two toolkits.

The core of both these frameworks are moved to a common library, and they split the rest of the code into two parts: classic and modern. The traditional Ext JS code moved to the classic toolkit, and the modern code that supports touch and HTML5 are moved to the modern toolkit.

Applications that share core resources and logic and utilize both the toolkits are called universal applications.

Now, let's take a look at the main.js view file under modern:

```javascript
Ext.define('MyApp.view.main.Main', {
    extend: 'Ext.tab.Panel',
    xtype: 'app-main',

    requires: [
        'Ext.MessageBox',
        'MyApp.view.main.MainController',
        'MyApp.view.main.MainModel',
        'MyApp.view.main.List'
    ],

    controller: 'main',
    viewModel: 'main',

    defaults: {
        styleHtmlContent: true
    },

    tabBarPosition: 'bottom',

    items: [
        {
            title: 'Home',
            iconCls: 'fa-home',
            layout: 'fit',
            items: [
                {
                    xtype: 'mainlist'
                }
            ]
        },
    ],

    // More code...
```
This sample view defines controller, viewModel, and other dependency classes required, creates four tabs, and binds the loremIpsum property of ViewModel. You'll learn more about this in detail in the upcoming chapters.

Now, let's take a look at main.js under `\classic\src\view\main\`

```javascript
Ext.define('NewApp.view.main.Main', {
    extend: 'Ext.tab.Panel',
    xtype: 'app-main',

    requires: [
        'Ext.plugin.Viewport',
        'Ext.window.MessageBox',
        'NewApp.view.main.MainController',
        'NewApp.view.main.MainModel',
        'NewApp.view.main.List'
    ],

    controller: 'main',
    viewModel: 'main',

    ui: 'navigation',
});
```
tabBarHeaderPosition: 1,
titleRotation: 0,
tabRotation: 0,

header: {
  layout: {
    align: 'stretchmax'
  },
title: {
    bind: {
      text: '{name}'
    },
    flex: 0
  },
  iconCls: 'fa-th-list'
},

 tabBar: {
  flex: 1,
  layout: {
    align: 'stretch',
    overflowHandler: 'none'
  }
},

responsiveConfig: {
  tall: {
    headerPosition: 'top'
  },
  wide: {
    headerPosition: 'left'
  }
},

defaults: {
  bodyPadding: 20,
  tabBar: {
    plugins: 'responsive',
    responsiveConfig: {
      wide: {
        iconAlign: 'left',
        textAlign: 'left'
      },
      tall: {

In the preceding code, you can see that the content of items is almost the same as in the modern toolkit. Additionally, you can see that this file has some config that is specific to responsive design. The following line in the preceding code tells the framework to use the ui style component called navigation:

```javascript
ui: 'navigation'
```

You'll learn more about the UI config and responsive design later in Chapter 8, Theming and Responsive Design.

Similarly, if you take a look at List.js under classic and modern, you'll only find some minor differences.
Exploring Sencha Cmd commands

Now, let’s explore some of the useful commands in Sencha Cmd.

The Sencha command format

Sencha commands take the following format:

```
sencha [category] [command] [options...] [arguments...]
```

There are many commands and options available in Sencha Cmd. Let’s take a look at some of the important commands.

Help

If you simply type the following command, you’ll get a list of categories, a list of top-level commands, and options that are available:

```
sencha help
```

To get help on a particular category type, the category name, followed by help, for example, to get help on a category app, run the following command:

```
sencha help app
```

This will produce the following output:

```
~ $sencha help app
Sencha Cmd v6.0.0.92
sencha app
This category contains various commands for application management.

Commands
* build  - Executes the build process for an application
* clean  - Cleans the application for a build
* emulate - Builds the application via a Packager then executes in the simulator/emulator
* explain - Resolves a reference path from the application’s entry file to the specified symbol
* prepare - Builds the application then the Packager prepares the app for native build (cordova only)
* publish - Publishes an application version to Sencha Space.
* refresh - Updates the application metadata (aka “bootstrap”) file
* run  - Builds the application via a Packager then executes the on a device
* upgrade - Upgrade the current application to the specified SDK
* watch  - Watches an application for file system changes and rebuilds.
```

If you further want to get help on subcategory commands under app, you can just add the command at the end for clean, as shown in the following code:

```
sencha help app clean
```
If you want to check whether there are any updates available for Sencha Cmd, use the following command:

`sencha upgrade --check`

If you want to upgrade Sencha Cmd, just remove the `--check` option, as shown in the following code:

`sencha upgrade`

### Generating an application

Sencha Cmd supports Ext JS 4.1.1a and higher and Sencha Touch 2.1 and higher. You can have multiple versions of the SDK on your machine. Here is the format of the command to generate an application based on a Sencha SDK, such as Ext JS or Sencha Touch:

```
 sencha -sdk /path/to/sdk generate app [---modern/classic] MyApp /path/to/ myapp
```

This is an example that will generate the Ext JS 6 application called `MyApp` under the `/Users/SomeUser/projects/extjs/myapp` path:

```
 sencha -sdk /Users/SomeUser/bin/Sencha/Cmd/repo/extract/ext/6.0.0/ generate app MyApp /Users/SomeUser/projects/extjs/myapp
```
Building the application
The following command will build HTML, JS, SASS, and so on:

`sencha app build`

In Sencha Cmd 6 and Ext JS 6, you can also run one of the following code to choose either modern or classic:

`sencha app build modern`
`sencha app build classic`

Here, `modern` and `classic` refers to the builds config in `app.json`. By default, Sencha Cmd puts two builds config: classic and modern in `app.json`; you can add the additional build config if required.

Launching the application
The watch command can be used to rebuild and launch the application. This will not only launch the application, but also monitor any code changes made, and as soon as the code changes are saved, it will refresh the browser to include the updated code changes as follows:

`sencha app watch`

In Sencha Cmd 6 and Ext JS 6, you can also run one of the following code to choose either modern or classic:

`sencha app watch modern`
`sencha app watch classic`

The code generation
With Sencha Cmd, you can generate the Ext JS code, such as views, controller, and model:

`sencha generate view myApp.MyView`
`sena generate model MyModel id:int,fname,lname`
`sena generate controller MyController`

If the field type is not specified when you generate the model, the default field type `string` will be used.
Upgrading your application
Sencha Cmd makes upgrading from one version of the SDK to another version easy. Use the upgrade command in the app category to upgrade to the new framework:

```
sencha app upgrade [ path-to-new-framework ]
```

Debugging an Ext JS application
You can use the browser's default debugger to debug the Ext JS code, but the debugging Ext JS code is much easier with a Firefox plugin called Illumination or the App Inspector plugin for Chrome.

Illumination
Illumination is a third-party tool. It is not a product of Sencha, and right now, it is only available for Firefox, and it requires Firebug.

The features of Illumination
Here are the some of the features of the Illumination plugin. This will reduce the amount of time you spent in debugging.

Object naming
Illumination will recognize the Ext JS components easily, so in the illumination tab you'll see Ext JS component names such as `Ext.panel.Panel` instead of showing `Object` as in Firebug's DOM tab.

Element highlighting
If you hover over any of the objects in the Illumination window, it will highlight the whole component in the HTML page.

The contextual menu
An Ext JS component is composed of several HTML elements. If you right-click on the page and select the Firebug context menu, you'll be taken to the element nested in the Ext JS component, but if you select the Illumination context menu, it will show you the Ext JS component that makes it easier to examine the component and its methods, properties, and events.
Check the Firebug DOM tab in Figure 1.8 and see how the objects are represented:

![Figure 1.8](image)

Now, check the Illumination tab in Figure 1.9 and see how the objects are represented. You can see that Illumination recognizes all the Ext JS component, as shown in the following screenshot:

![Figure 1.9](image)

Although Illumination makes the debugging of the Ext JS application easier, it is not a must. Illumination is not a free tool. So, if you do not want to pay for it, you can still use Firebug to debug, but you may need to spend a little bit of extra time to debug, or you may need to take a look at App Inspector or Sencha Fiddle.
App Inspector
App Inspector is a free Chrome plugin developed by Sencha. It also provides all the features provided by Illumination. Some of the features provided are component inspector, store inspector, and layout profiles.

Some information is easier to find in App Inspector than Illumination, and debugging with the Illumination Ext JS application takes a long time to load than debugging with App Inspector.

*Figure 1.10 and Figure 1.11 shows couple of tabs in App Inspector:*

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>classic</td>
<td>6.0.0</td>
</tr>
<tr>
<td>Core Version</td>
<td>5.0.0</td>
</tr>
<tr>
<td>Ext JS Version</td>
<td>5.0.0</td>
</tr>
<tr>
<td>Application Name</td>
<td>MyApp</td>
</tr>
<tr>
<td>theme-base</td>
<td>5.0.0</td>
</tr>
<tr>
<td>theme-neptune</td>
<td>5.0.0</td>
</tr>
<tr>
<td>theme-neutral</td>
<td>5.0.0</td>
</tr>
</tbody>
</table>

*Figure 1.10*

*Figure 1.11*
Sencha Fiddle

This is another debugging tool that may be helpful. It is also an online web-based IDE that provides some debugging capabilities, as shown in Figure 1.12:

![Sencha Fiddle](image)

Figure 1.12

The development IDE

Although you can use any simple text editors to write the Ext JS code, using the IDEs definitely makes it a bit easier. Sencha provides the Sencha JetBrains plugin for JetBrains products, such as IntelliJ, WebStorm, PHPStorm, and RubyMine.
If you're looking for some free and simple IDE, then take a look at Visual Studio Code and Brackets.io. Both of these are extremely lightweight and available for Mac, Windows, and Linux. *Figure 1.13* shows **Visual Studio Code**:

![Visual Studio Code](image)

**Summary**

In this chapter, we looked at some of the advantages with JavaScript frameworks instead of using plain JavaScript. We also looked at some of the famous JavaScript frameworks. You learned how to set up a development environment for the Ext JS application, and we scaffolded an Ext JS application with Sencha Cmd.

In the next chapter, you’ll learn the core concepts and basics of Ext JS.
Where to buy this book

You can buy Ext JS 6 By Example from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.