Stuart Ashworth

Ext JS 6: Getting Started

A practical guide to understanding and getting started with Ext JS 6

Ext JS 6: Getting Started

Contents

Chapter 0
About the author

Stuart Ashworth

Web & Mobile Engineer

Stuart Ashworth is a freelance web and mobile developer and an all-round web geek currently living in Glasgow, Scotland, with his fiancé, Sophie, and wee dog, Meg.

Since graduating with a first class honors degree in design computing from the University of Strathclyde, Stuart has worked in the IT industry creating software for both large and small companies across the UK and around the world.

Stuart has worked with Sencha technologies for over 5 years, creating numerous web applications, mobile applications, and framework plugins along the way. He co-authored Ext JS Essentials and Ext JS 4 Web Application Development Cookbook both published by Packt Publishing and also curates content for the fortnightly Sencha Insights e-mail newsletter (www.senchainsights.com).

Stuart enjoys playing football, snowboarding, and traveling. He blogs about Sencha and web technology on his website (www.stuartashworth.com), and can be contacted through Twitter at @StuartAshworth9, through e-mail at stuart@stuartashworth.com, or through the Sencha forums.

Stuart Ashworth

Chapter 1
Introduction

As Ext JS 6, the next evolution of Sencha's flagship framework, is released, another revolutionary approach to developing cross-platform web applications is upon us. This time it focuses on a unified framework for all platforms and is hailed as the iteration that will finally bring some cohesion to our development for desktop platforms and for mobile.

With Ext JS leading the way for single-page web applications, providing a hugely impressive interactive user experience, and Sencha Touch becoming one of the most popular frameworks for developing web and hybrid mobile applications, it didn't take a big stretch of the imagination to see them eventually coming together. The result is Ext JS 6 and it marks a big step in front-end framework technology.

Who is this book for?

This book is aimed at new and existing Ext JS developers who are looking for a great resource to help them get up to speed on Ext JS 6 with clear explanations of the new concepts and terminology; simple tutorials of how to get an application up and running with the new architecture and details of how to go about upgrading to the latest version.

This book won't focus on learning Ext JS or Sencha Touch as a framework, but instead will assume a reasonable knowledge of the framework and its main concepts. It will build upon these to show how Ext JS 6 makes use of these.

What will you learn?

We will cover a lot of topics surrounding Ext JS 6 and by the end you should have a solid understanding of the latest framework version, including:

- What new features have been incorporated in Ext JS 6.
- The new key concepts to understand when developing with Ext JS 6.
- How to create a new project using Sencha Cmd 6.
- How to architect Universal Applications.
- Strategies for upgrading Ext JS and Sencha Touch applications into Ext JS 6.

Chapter 4
Diving into Ext JS 6

Sample Chapter

This is a sample chapter from the Ext JS 6: Getting Started book. To purchase the entire book along with the bonus features, head over to www.stuartashworth.com/ext-js-6-getting-started

Now that we've gone over the main changes in structure of our apps we'll dive in and start playing around with a Universal Application of our own.

We will be building a simple To Do application....

Installing Sencha Cmd 6

Before we get started we need to install the latest version of Sencha Cmd. Head to the Sencha website and download Sencha Cmd 6 and follow the instructions to install it. The default settings of the installation wizard should be fine to accept.

This shouldn't disrupt your previously installed versions but will mean that to use a previous version you will have to use the command along with its full name (e.g. `sencha-5.1.0.26 app build`*).*

Generating a new Project

Now we've got Sencha Cmd installed we can use it to create new applications. We can generate apps in different modes based on the Toolkits we want to use, either Classic, Modern or Universal. We will start with a Universal app.

First, download and extract the Ext JS 6 framework from the Sencha website.

Open up Terminal or Command Prompt and navigate to the extracted framework folder.

Execute the following command to generate a new application named Todo,

making sure you alter the path so it is generated in the correct place.

```
sencha generate app Todo path/to/your/workspace
```

Alternatively, you can do it in reverse and navigate to your project folder and point Sencha Cmd to where the Ext JS framework is located.

```
sencha -sdk=path/to/ext generate app Todo .
```

Note the . tells Sencha Cmd to generate the app in the current folder.

Classic Only App

If you aren't interested in mobile devices you can create a simpler Classic Toolkit application by modifying the `sencha generate` command slightly by adding the `--classic` option.

```
sencha generate app --classic Todo path/to/your/workspace
```

This will create a more familiar looking project structure and will only include the Classic (i.e. desktop) part of the framework.

Modern Only App

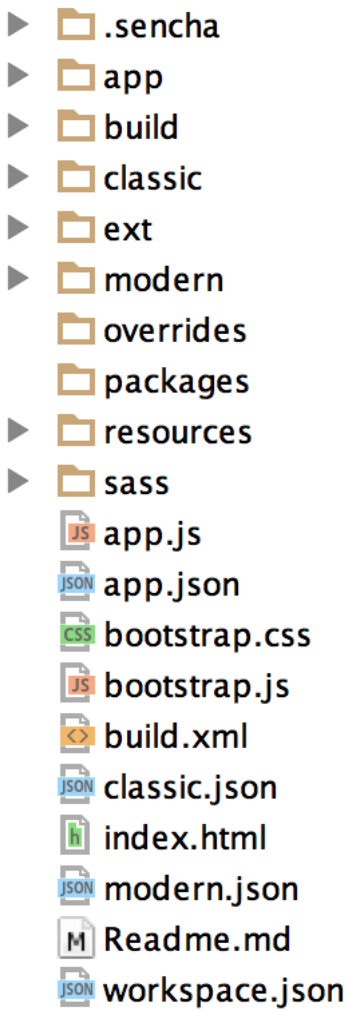
Similarly if you only want to target phones and tablets then you can create a Modern app by adding the `--modern` option.

```
sencha generate app --modern Todo path/to/your/workspace
```

Explanation of our New Project Structure

Now that we have generated a new app with Sencha Cmd let's take a look at its structure and see what's changed since Ext JS 5.

Our Universal App project looks like this:



Toolkit Apps

At first glance this doesn't look too different from what we're used to from Ext JS 5 or Sencha Touch, but for a few important additions.

You will notice that there are two new folders named `classic` and `modern`. These folders relate to the two Toolkits we discussed earlier and hold the Classic and Modern toolkit code.

Within each of these directories there are `src` and `sass` folders, like we are used to seeing in a traditional Ext app. This is because these 'toolkit builds' are essentially their own standalone app - building on either the Classic API (i.e. Ext JS) or the Modern API (i.e. Sencha Touch). You can almost think of these as additional Sencha applications projects within a Workspace.

More often than not it will be only view code that is placed in these toolkit folders, unless there is specific business logic required that is not shared between the Classic and Modern apps.

You will also notice that there is a `classic.json` and a `modern.json` file at the root of the project. These two files are auto-generated and are the equivalent of the `bootstrap.json` you are used to seeing in Ext JS apps. These files are indexes of all the classes used in the app and where they are found.

Common Code

Finally we get round to the million dollar topic of how to share code between these two toolkit apps and join the dots of how these universal apps work.

So, you will notice that the project's other folders and files look identical to what we're used to - there's folders for `overrides`, `packages`, `resources` and `sass`, and also an `app` folder. This is because this is essentially a full Sencha app that the toolkit apps are built upon.

Everything that is shared between the Classic and Modern toolkit apps should be placed in these folders. For example, the common Models, Stores and Controllers should be placed in the `app` folder and the common images and fonts can be put in the `resources` folder.

Familiar Ground

Once you have digested these new additions the rest of the structure looks familiar and shouldn't cause any problems when creating your applications. The usual `packages`, `overrides` and `build` folders still exist and can be used in exactly the same way.

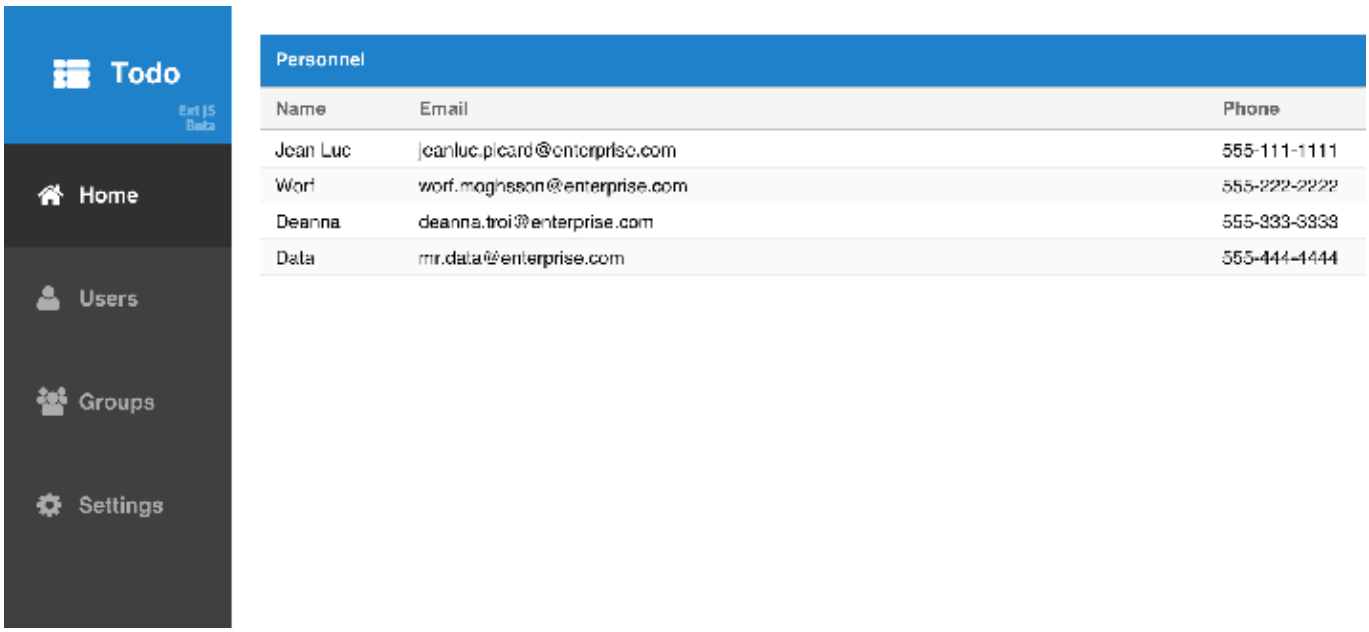
Running the New Project

Now we have a better understanding of the project structure let's get the basic app running and see what it looks like. When using the `generate` command of Sencha Cmd a simple app is created to show off the structure and some sample classes, so when we run our app we will see this and be able to confirm all is working as expected.

When developing we run our application as we always have done, by opening the top level `index.html` and leaving the Microloader to load all our app's resources as required. Let's navigate our browsers to the `index.html` file within your project directory - ensuring it is running on a web server.

If you are using JetBrains' IntelliJ like me, you can right click the HTML file and choose Open in Browser which will fire it up in a temporary local server - very handy. For another approach, see the section on the watch command in Chapter 5.

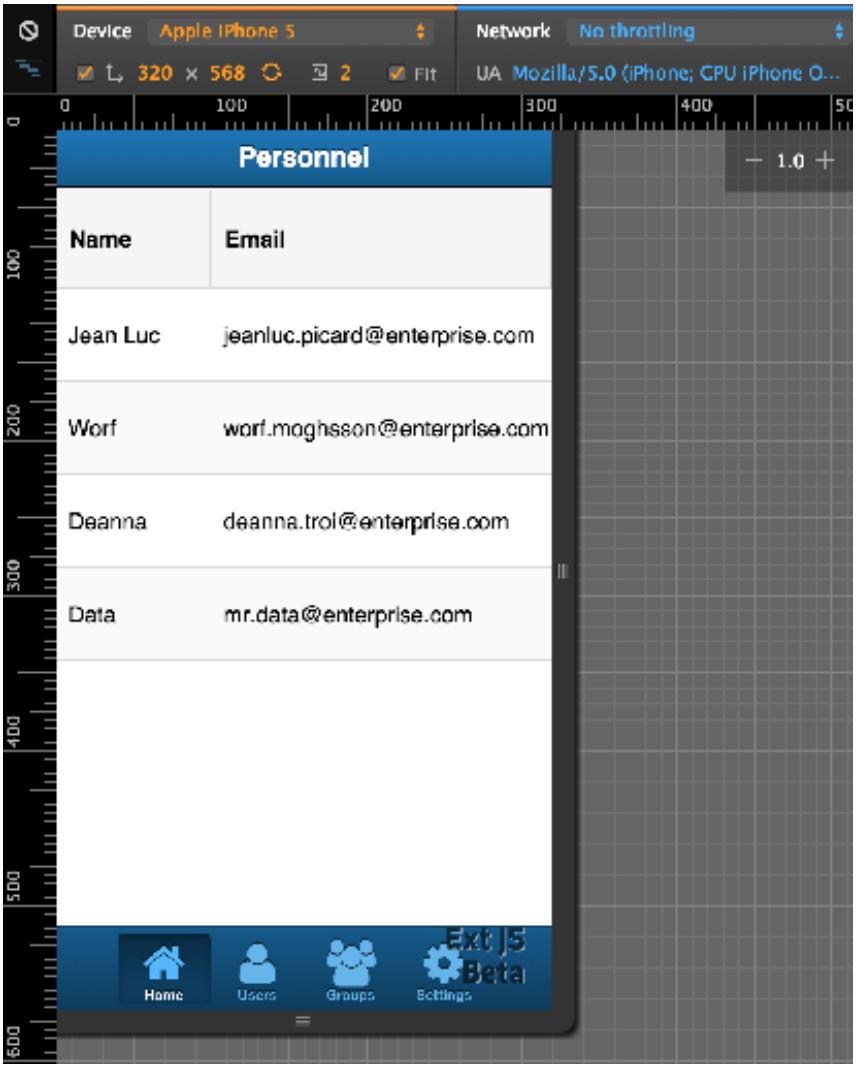
Assuming you're running your browser in 'normal' mode (i.e. not using Device Mode) then you will see a desktop formatted web application (like the one in the image below). This is the Classic Toolkit application running with, essentially, Ext JS 5.



This application is loading in the code from the `classic` folder, combined with any shared code from the root `app` folder.

To view the Modern Toolkit app you can enable Device Mode (if you are using Google Chrome) and then refresh the page. To enable Device Mode, open Dev Tools (CMD-ALT-I or F12) and click the mobile phone icon at the top left of the Dev Tools pane.

When you refresh the page you should see the Modern Toolkit app, which looks suspiciously like Sencha Touch because, well, it is.



When the framework detects it is a mobile device it switches to the Modern toolkit and loads the code found in the `modern` folder, combining it with the common code from the `app` folder.

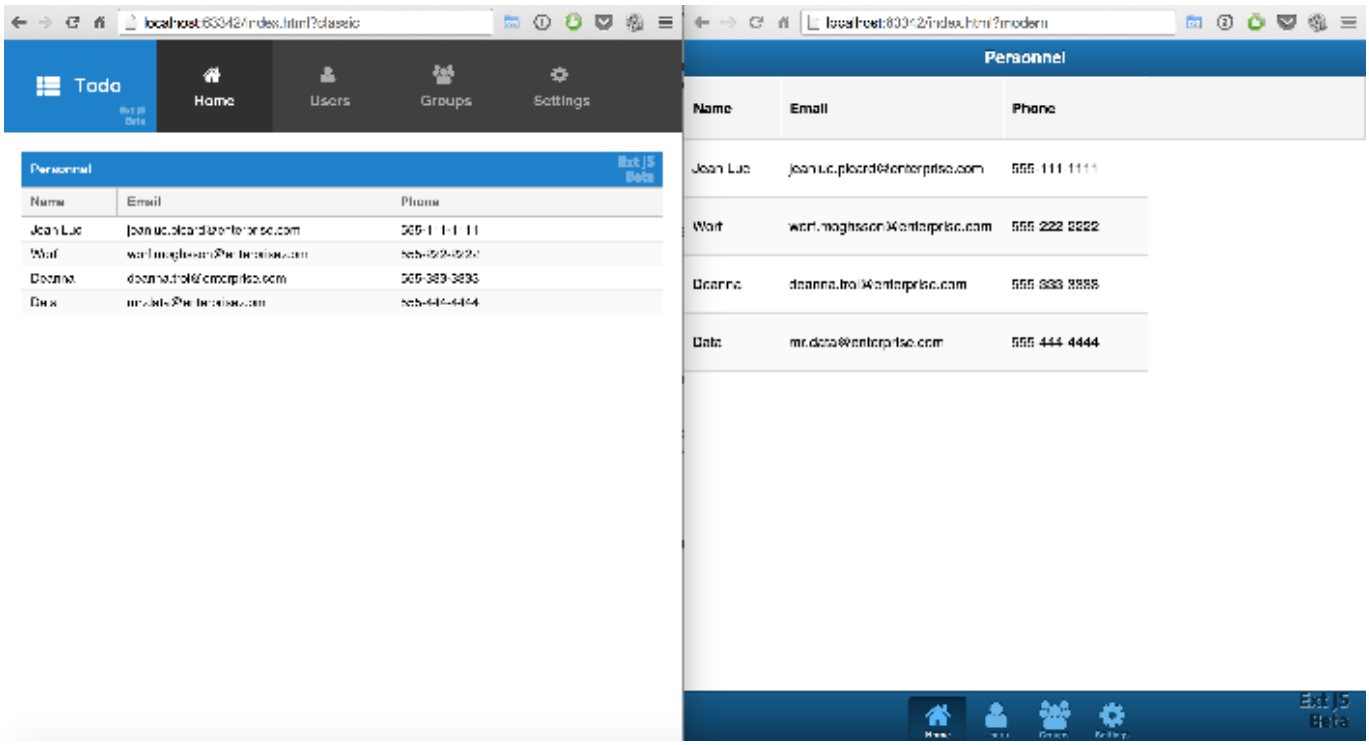
We will discuss how to force a specific toolkit app on users that have a particular setup (i.e. tablet users get Classic app but phone users get Modern) in a later section.

Forcing a toolkit

It is possible to force an app to launch with a particular app toolkit which is very handy for development where you aren't able to mock the device profile.

To do this, simply add a query string of `?modern` or `?classic` to the URL and refresh the page. This will be picked up by the `Ext.beforeLoad` function that is defined in the `index.html` (go have a look and see how it works).

You should see the page load with the different toolkit apps depending on the name added.



This function is executed *before* the Microloader starts loading in all the app's files but *after* it has done some device and platform detection. Within the default code it looks for the strings `classic` and `modern` in the query string and, if found, forces the app to use that particular profile. If none is found then it shows the Classic app if it is Desktop and the Modern app if it isn't.

Stuart Ashworth

Ext JS 6: Getting Started

Buy the book and video tutorial packages today!

www.extjs6gettingstarted.com

To learn more about the author and how he might be able to help with your project, visit:

www.stuartashworth.com