

Covers EXT JS version 4

EXT JS IN ACTION

SECOND EDITION

Jesus Garcia
Jacob K. Andresen
Grgur Grisogono



 MANNING



**MEAP Edition
Manning Early Access Program
Ext JS in Action, Second Edition, version 4**

Copyright 2012 Manning Publications

For more information on this and other Manning titles go to
www.manning.com

©Manning Publications Co. Please post comments or corrections to the Author Online forum:
<http://www.manning-sandbox.com/forum.jspa?forumID=794>

Table of contents

Part 1 Introduction to Ext JS 4.0

1. A framework apart
2. Back to basics
3. Components and containers

Part 2 EXT Js components

4. A place for components
5. Exploring layouts
6. Form it up
7. Data stores
8. GridPanels
9. Charting
10. Taking root with trees
11. Remote method invocation with Ext Direct
12. Drag and drop

Part 3 Building an application

- 13. Class system foundations
 - 14. Developing an Ext JS application
 - 15. Wiring up interaction models
- A. Basic data management

1

A framework apart

This chapter covers

- A holistic view of Ext JS
- Learning about what's new in Ext 4.0
- Downloading and unpacking the framework source code
- Exploring an Ajax-based "Hello world" example

Envision a scenario where you're tasked to develop an application with many of the typical UI (user interface) widgets such as menus, tabs, data grids, dynamic forms, and styled pop-up windows. You want something that allows you to programmatically control the position of widgets, which means it has to have layout controls. You also desire detailed and organized centralized documentation to ease your learning curve with the framework. Finally, this application needs to look mature and go into beta phase as quickly as possible, which means you don't have a lot of time to toy with HTML and CSS. Before entering the first line of code for the prototype, you need to decide on an approach for developing the frontend. What are your choices?

You do some recon on the common popular libraries on the market and quickly learn that all of them can manipulate the DOM, but only two of them have a mature UI library: YUI (Yahoo! User Interface) and Ext JS.

At first glance of YUI, you might think you need not look any further. You toy with the examples and notice that they look mature but are not exactly professional quality, which means you'll need to modify CSS. No way. Next, you look at the documentation. It's centralized and technically accurate, but it's far from user friendly. You look at all of the scrolling required to locate a method or class. Some classes are even cut off because the left navigation pane is too small.

In this chapter, we'll take a good look at Ext JS, and you'll learn about some of the widgets that compose the framework. After we finish the overview, you'll download Ext JS and take it for a test drive.

1.1 Looking at Ext JS

Having to develop an RIA with a set of rich UI controls, you turn to Ext JS and find that, out of the proverbial box, Ext JS provides a rich set of DOM utilities and widgets. Although you can get excited about what you see in the examples page, it's what is under the hood that's most exciting. Ext JS comes with a full suite of layout management tools to give you full control over organizing and manipulating the UI as requirements dictate. One layer down exists what's known as the Component model and Container model, each playing an important role in managing how the UIs are constructed.

Component and Container models

The Component and Container models play a key role in managing UIs with Ext JS and are part of the reason Ext JS stands out from the rest of the Ajax libraries and frameworks. The Component model dictates how UI widgets are instantiated, rendered, and destroyed in what's known as the component lifecycle. The Container model controls how widgets can manage (or *contain*) other child widgets. These are two key areas for understanding the framework, which is why we'll be spending a lot of time on these two topics in chapter 3.

Almost all UI widgets in the framework are highly customizable, affording you the option to enable and disable features, override functions, and use custom extensions and plug-ins. One example of a web application that takes full advantage of Ext JS is Conjoon. Figure 1.1 shows a screenshot of Conjoon in action.

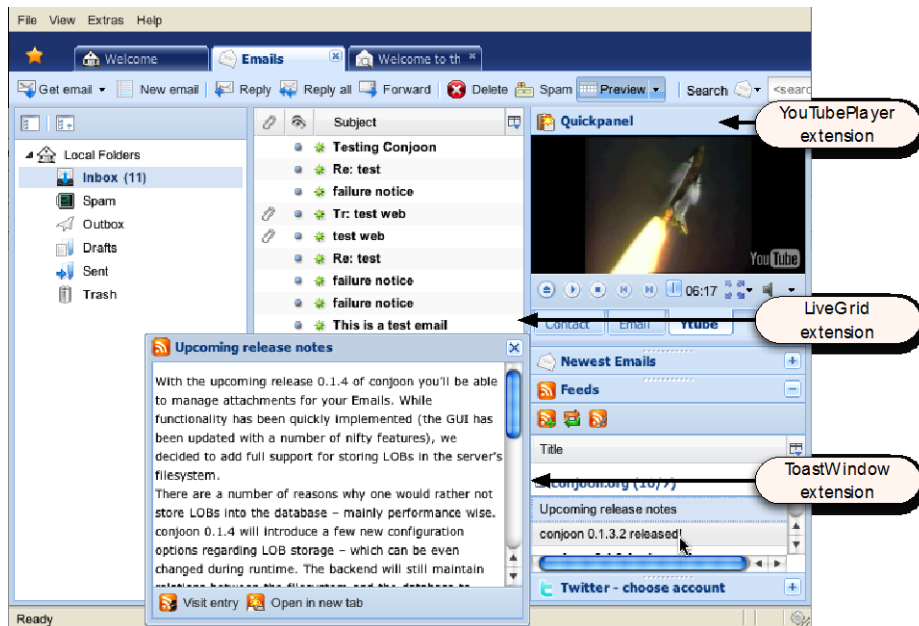


Figure 1.1 Conjoon is an open source personal information manager that's a great example of a web application that uses the framework to manage a UI that leverages 100 percent of the browser's viewport. You can download it at <http://conjoon.org/>.

Conjoon is an open source personal information manager and can be considered the epitome of web applications developed with Ext JS. It uses just about all of the framework's native UI widgets and demonstrates how well the framework can integrate with custom extensions such as YouTubePlayer, LiveGrid, and ToastWindow. You can get a copy of Conjoon by visiting <http://conjoon.org>.

You've learned how Ext JS can be used to create a full-page web application. But what if you have an application that's already in production? Next, you'll learn how Ext JS can be integrated into existing applications or websites.

1.1.1 Integration with existing sites

Any combination of widgets can be embedded inside an existing web page and or site with relative ease, giving your users the best of both worlds. An example of a public-facing site that contains Ext JS is the Dow Jones Indexes site, <http://djindexes.com>. Figure 1.2 shows Ext JS integrated with a page at djindexes.com.

The screenshot shows the Dow Jones Indexes website. On the left, there is a sidebar with navigation links. The main content area is titled "Dow Jones-AIG Commodity Indexes" and includes a "Learn More" link. Below this is an "Index Information" section with a table of indices. A callout box labeled "Tab Panel" points to the "Alternative" tab in the table. Another callout box labeled "Grid Panel" points to the table itself. Below the table is a line chart for the "Dow Jones Wilshire 5000 Composite Index" with a callout box labeled "Grid Panel" pointing to the chart area. The table data is as follows:

Index	Benchmark	Alternative	Date Time	Last	Chg.
Dow Jones Wilshire 5000 Compos			13 Mar 17:25	7616.92	52.91
Dow Jones Wilshire 4500 Comple			13 Mar 17:24	328.01	1.26
Dow Jones Wilshire Global Total M			13 Mar 17:25	1447.10	21.95
Dow Jones Wilshire Asia/Pacific In			13 Mar 17:25	743.75	23.95
Dow Jones Wilshire Europe Index			13 Mar 17:25	1582.60	24.42
Dow Jones Wilshire REIT Index			13 Mar 17:24	80.89	-2.22
Dow Jones Wilshire exUS Real Es			13 Mar 17:24	1290.40	47.81

Figure 1.2 The Dow Jones Indexes website, <http://djiindexes.com>, is one example of Ext JS embedded in a traditional Web 1.0 site.

This Dow Jones Indexes web page gives its visitors rich interactive views of data by utilizing a few of the Ext JS widgets such as the tab Panel, grid Panel, and Window (not shown). Its visitors can easily customize the view of the stocks by selecting a row in the main grid Panel, which invokes an Ajax request to the server, resulting in an updated graph below the grid. The non-Ext JS graph view can be modified as well by clicking one of the time period buttons below it.

You now know that Ext JS can be leveraged to build single-page applications or can be integrated into existing multipage applications. But we still haven't satisfied the requirement of API documentation. How does Ext JS solve this?

1.1.2 Rich API documentation

With this version of the framework, the API documentation is all new and improved. When opening the API documentation for the first time (figure 1.2), you get a sense of the polish that the framework has. Unlike competing frameworks, the Ext JS API Documentation leverages its own framework to present a clean and easy-to-use documentation tool that uses Ajax to provide the documentation.

We'll explore all of the features of the API and talk about some of the components used in this documentation tool. Figure 1.3 illustrates some of the components used in the Ext JS API Documentation application.

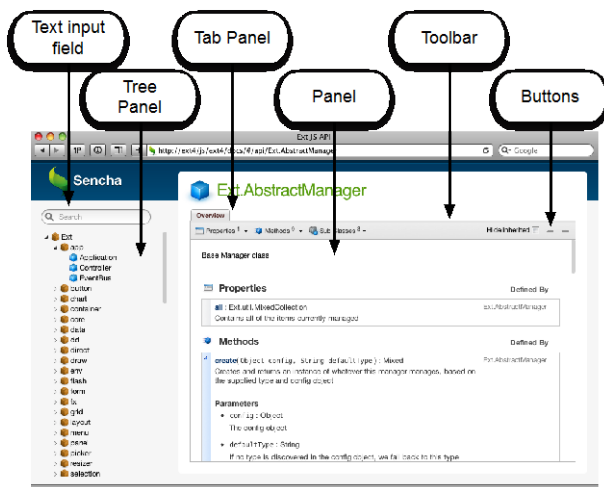


Figure 1.3 The Ext JS API Documentation contains a wealth of information and is a great resource for learning more about components and widgets.

The API Documentation tool is chock-full of gooey GUI goodness and uses six of the most commonly used widgets, which include the Text input field, tree Panel, tab Panel, Panel and Toolbar with embedded Buttons.

History support

The Ext JS 4.0 documentation now includes browser history support. This means that you can use the browsers forward and backward buttons to walk up and down your API documentation breadcrumbs.

I'm sure you're wondering what all of these are and what they do. Let's take a moment to discuss these widgets before we move on.

The Text input field, is a widget that wraps the native browser text input form control, adding features such as validation. In the API documentation, it is used to perform live searches against the tree Panel and is custom styled. We'll be talking more in about TPanels in Chapter 10.

The tree Panel widget displays hierarchical data visually in the form of a tree much like Windows Explorer displays your hard drive's folders. The tab Panel provides a means

to have multiple documents or components on the canvas but allows only one to be active at a time, though in the API documentation, only displays one item.

Panel is a workhorse of Ext JS, where it contains a lot of flexibility, containing many areas to display content, known as the dock and the content body. The dock is where items like Toolbars are typically placed, and the content body is the area where content or child widgets are typically rendered. In the case of the API documentation, the content body contains the documentation for the framework.

The `Toolbar` class provides a means to present commonly used UI components such as Buttons and Menus, but it can also contain, as in this case, any of the `Ext.form.Field` subclasses. I like to think of the `Toolbar` as a place for the common file-edit-view menus that you see in popular operating systems and desktop applications.

Using the API is a cinch. To view a document, click the class node on the tree. This will invoke an Ajax request to fetch the documentation for the desired class. Each document for the classes is an HTML fragment (not a full HTML page).

So the documentation is thorough. But what about rapid application development? Can Ext JS accelerate your development cycles?

1.1.3 Rapid development with prebuilt widgets

Ext JS can help you jump from conception to prototype because it offers many of the required UI elements already built and ready for integration. Having these UI widgets prebuilt, instead of having to engineer them, saves you a lot of time. In many cases, the UI controls are highly customizable and can be modified to your application needs.

1.2 What you need to know

Although being an expert in web application development isn't required to develop with Ext JS, developers should have some core competencies before attempting to write code with the framework.

The first of these skills is a basic understanding of Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS). It's important to have some experience with these technologies because Ext JS, like any other JavaScript UI library, uses HTML and CSS to build its UI controls and widgets. Although its widgets may look like and mimic typical modern operating system controls, it all boils down to HTML and CSS in the browser.

Because JavaScript is the glue that holds Ajax together, a solid foundation in JavaScript programming is suggested. Again, you need not be an expert, but you should have a good grasp of key concepts such as arrays, reference, and scope. It's a plus if you're familiar with object-oriented JavaScript fundamentals such as objects, classes, and prototypal inheritance. If you're new to JavaScript, you're in luck. JavaScript has existed since nearly the dawn of the internet. An excellent place to start is W3Schools.com, which offers a lot of free online tutorials and even has sandboxes for you to play with JavaScript online. You can visit them at <http://w3schools.com/JS/>.

If you're required to develop code for the server side, you're going to need a server-side solution for Ext JS to interact with as well as a way to store data. To persist data, you'll need to know how to interact with a database or file system with your server-side language of choice.

Naturally, the range of solutions available is quite large. For this book, we won't focus on a specific language. Instead, we'll use online resources at <http://extjsinaction.com>, where I've done the server-side work for you. This way, all you have to focus on is learning Ext JS.

We'll begin our exploration of Ext JS with a bird's-eye view of the framework, where you'll learn about the categories of functionality.

1.3 A bird's-eye view of the framework

The story of Ext JS's main codebase begins early 2010, during the development of Sencha Touch, the world's first HTML5 mobile framework (released November 2010). Sencha Touch brought forth the base underpinnings, known as Sencha Platform, which contains many of the critical features that Ext JS and Sencha Touch both use. Such common features include DOM and event Management, the Component Model, layouts and so forth, all of which we'll be progressively diving into later in this book.

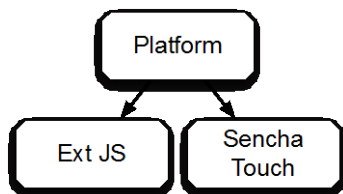


Figure 1.4 Ext JS 4.0 and Sencha Touch both branch off of Sencha Platform, a common base for the Sencha family of HTML5 frameworks.

The Ext JS framework not only provides UI widgets but also contains a host of other features. These fall into six major areas of purpose: Core, UI components, web remoting, data services, drag and drop, and general utilities. Figure 1.5 illustrates the six areas of purpose.

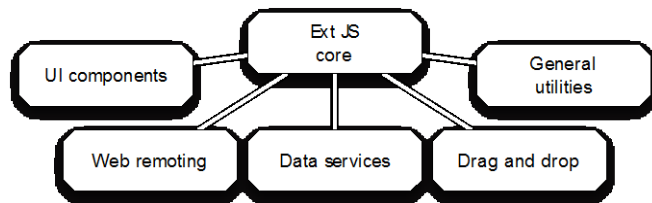


Figure 1.5 The six areas of purpose for Ext JS classes: Ext JS Core, UI components, web remoting, data services, drag and drop, and general utilities

Knowing what the different areas of purpose are and what they do will give you an edge when developing applications, so we'll take a moment to discuss them.

CORE

The first feature set is the Ext JS Core, which comprises of many of the basic features such as Ajax communication, DOM manipulation, and event management. Everything else is dependent on the Core of the framework, but the Core isn't dependent on anything else.

UI COMPONENTS

The UI components contain all of the widgets that interface with the user.

WEB REMOTING

Web remoting is a means for JavaScript to (remotely) execute method calls that are defined and exposed on the server, which is commonly known as a Remote Procedure Call, or RPC. It's convenient for development environments where you'd like to expose your server-side methods to the client and not worry about all of the fuss of Ajax method management. This package is known as Ext Direct.

DATA SERVICES

The data services section takes care of all of your data needs, which include fetching, parsing, and loading information into stores. With the Ext JS data services classes, you can read Array, XML, and JSON (JavaScript Serialized Object Notation), which is a data format that's quickly becoming the standard for client-to-server communication. Stores typically feed UI components.

GET YOUR JSON ON!

Even though JSON has been around for many years, if this is the first time you've heard of it, I encourage you to visit <http://json.org>, which happens to be the go to source for information on this ubiquitous data exchange format. If you are interested in learning how to implement JSON in your server side language of choice, there are a ton of JSON implementations, most of which are documented and explained online. I suggest searching Google using a query like "PHP JSON".

DRAG AND DROP

Drag and drop is like a mini framework inside Ext JS, where you can apply drag-and-drop capabilities to an Ext JS component or any HTML element on the page. It includes all of the necessary members to manage the entire gamut of all drag-and-drop operations. Drag and drop is a complex topic. We'll spend the entirety of chapters 13 and 14 on this subject alone.

UTILITIES

The utilities section comprises cool utility classes that help you perform some of your routine tasks easier. An example would be `Ext.util.Format`, which allows you to format or transform data easily. Another neat utility is the CSS singleton, which allows you to create,

update, swap, and remove style sheets as well as request the browser to update its rule cache.

Now that you have a general understanding of the framework's major areas of functionality, let's take some time to look at some of the more commonly used UI widgets that Ext JS has to offer.

1.3.1 Containers and layouts at a glance

Even though we'll cover these topics in detail in chapter 3, we should spend a little time here talking about containers and layouts. The terms *container* and *layout* are used extensively throughout this book, and I want to make sure you have at least a basic understanding of them before we continue. Afterwards, we'll begin our exploration of visual components of the UI library.

CONTAINERS

Containers are widgets that can manage one or more child items. A child item is generally any widget or component that's managed by a container or parent, thus the -parent-child paradigm. You've already seen this in action in the API. The `tab Panel` is a container that manages one or more child items, which can be accessed via tabs. Please remember this term, because you'll be using it a lot when you start to learn more about how to use the UI portion of the framework.

LAYOUTS

Layouts are implemented by a Container to visually organize the child items in the container's content body. Ext JS has whopping 33 layouts in the library! The good news is that you only have to learn 13 of them, which we'll go into in great detail in chapter 5 and shows the ins and outs of each layout. Now that you have a high level of understanding of containers and layouts, let's look at some containers in action.

In the figure 1.6, you see two subclasses of Container, Panel and Window, each engaged in parent-child relationships, demonstrating the power of the Container class and various layouts.

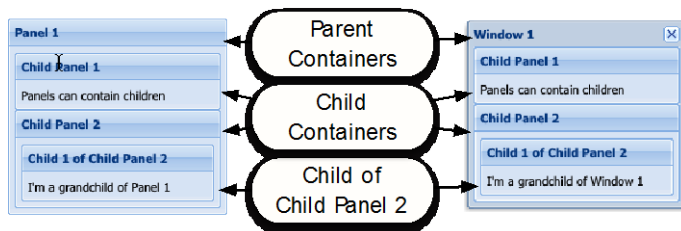


Figure 1.6 Here, you see two parent Containers, Panel (left) and Window (right), managing child items, which include nested children.

The Panel (left) and Window (right) in figure 1.5 each manage two child items. Child Panel 1 of each parent container *contains* HTML, whereas the children with the title Child Panel 2 manage one child panel each using the simple `AutoLayout`, which is the default Container Layout. This parent-child relationship is the crux of all of the UI management of Ext JS and will be reinforced and referenced repeatedly throughout this book.

You learned that Containers manage child items and use layouts to visually organize them. Now that you have these important concepts down, we'll move on to see and discuss other Containers in action.

1.3.2 Other Containers in action

You saw Panel and Window being used when you learned about Containers. Figure 1.7 shows some other commonly used subclasses of Container.

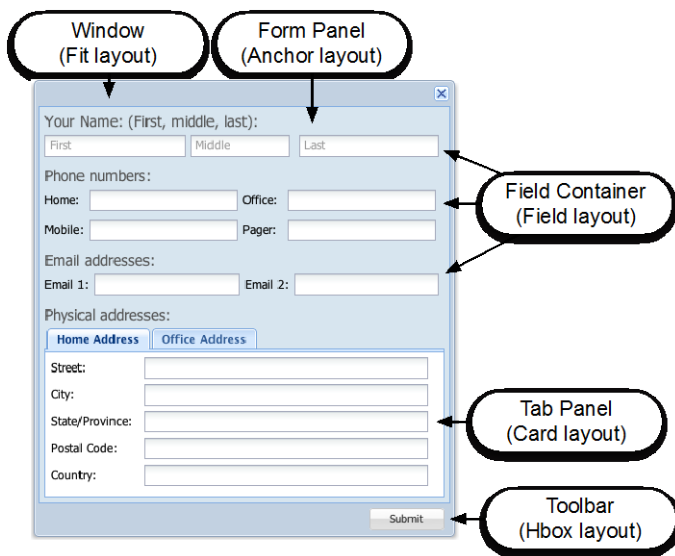


Figure 1.7 Commonly used subclasses of Container—FormPanel, tab Panel, FieldContainer, and Toolbar—and the layouts used to compose this UI Window. We'll build this in chapter 6, where you learn about forms.

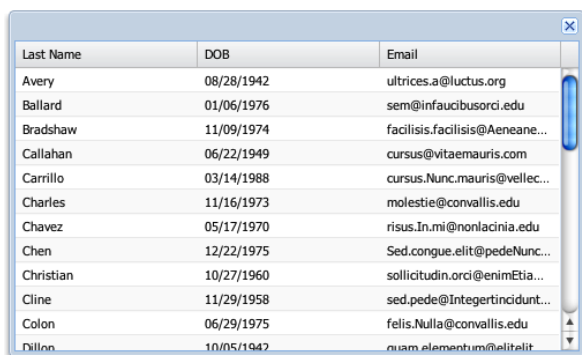
In figure 1.6, you see the FormPanel, tab Panel, Window, Toolbar and FieldContainer widgets. The FormPanel works with the `BasicForm` class to wrap fields and other child items with a form element. All of these widgets are contained by an instance of `Ext.window.Window`.

We'll spend some time building a complex UI in chapter 6, where you'll learn more about FormPanels. For now, let's move on to see what data presentation widgets the framework has to offer.

1.3.3 Data-bound views

You've already learned that the Data Services portion of the framework is responsible for the loading and parsing of data. Ext JS 4 has a lot of widgets that are bound to data Stores, known as Views. Many of the views that you will be deploying include the data View, grid Panel, and tree Panel. If your application requires charts, then you'll be pleased to learn that all of the charts in the framework are also considered views and are bound to data Stores.

Figure 1.8 is a screenshot of the Ext JS grid Panel in action.



Last Name	DOB	Email
Avery	08/28/1942	ultrices.a@luctus.org
Ballard	01/06/1976	sem@infaucibusorci.edu
Bradshaw	11/09/1974	facilisis.facilisis@Aeneane...
Callahan	06/22/1949	cursus@vitaemauris.com
Carrillo	03/14/1988	cursus.Nunc.mauris@vellec...
Charles	11/16/1973	molestie@convallis.edu
Chavez	05/17/1970	risus.In.mi@nonlacinia.edu
Chen	12/22/1975	Sed.congue.elit@pedeNunc...
Christian	10/27/1960	sollicitudin.orci@enimEtia...
Cline	11/29/1958	sed.pede@Integertincidunt...
Colon	06/29/1975	felis.Nulla@convallis.edu
Dillon	10/05/1942	quam.elementum@ielitlit

Figure 1.8 The grid Panel as seen in the Buffered grid example in the Ext JS SDK

The newly refactored grid Panel is a subclass of Panel and presents data in a table-like format, but its functionality extends far beyond that of a traditional table, offering sortable, resizable, and movable column headers and selection models such as RowSelectionModel and CellSelectionModel. You can customize its look and feel as you desire and couple it with a PagingToolbar to allow large datasets to be segmented and displayed in pages. It contains many features and plugins allowing you to do things like editing by row or cell or lock a column.

The data View as shown in figure 1.9 renders photos and other bits of data for various phones on the market.



Figure 1.9 The data View as demonstrated in the Ext JS SDK examples

The `DataView` class consumes data from a store, paints it onscreen using a class known as `XTemplate`, and provides a simple selection model. The Ext JS `XTemplate` is an HTML fragment generation utility that allows you to create a *template*, with placeholders for data elements, which can be filled in by individual records in a store and stamped out on the DOM.

Gone is the ListView widget!

If you're coming from Ext JS 3.0, you may wonder where the `ListView` widget is. The simple answer is that the `ListView`, providing faster table rendering in Ext JS 3.0, was removed from 4.0, in favor of refactoring the `grid Panel` for much faster performance!

The `grid Panel` and `DataView` are essential tools for painting data onscreen, but they do have one major limitation. They can only show lists of records. They can't display hierarchical data. This is where the `tree Panel` fills the gap.

1.3.4 Make like a tree Panel and leaf

The `tree Panel` widget is an exception to the list of UI widgets that consume data, in that it doesn't consume data from a data store. Instead, it consumes hierarchical data via the usage of the `TreeStore` class. Figure 1.9 shows an example of an Ext JS `tree Panel` widget.

In figure 1.10, the `tree Panel` is being used to display the parent-child data inside the directory of an installation of the framework.

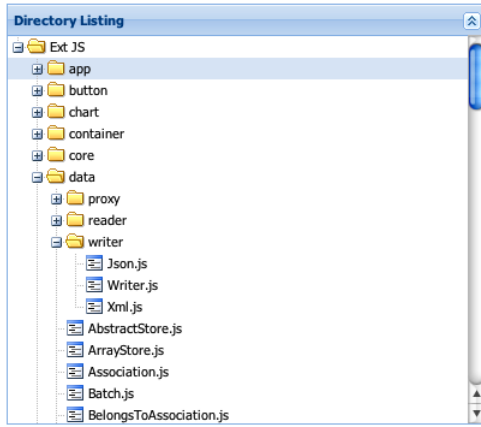


Figure 1.10 An Ext JS tree, which is an example from the Ext JS SDK

For Ext JS 4.0, it has been completely rebuilt, and is now a close cousin to the grid Panel, rather than not being in the same family! Figure 1.11 demonstrates the versatility of the new tree Panel.

Project	Teams	Lead	Hours
Financial magazine	Project teams	Pat Sheridan	
	UI/UX	Jay Garcia	
	Jay Garcia		40
	Pat Sheridan		40
	Server side PHP	Anthony De Moss	
	Jordan Laramy		40
	Database Architecture	Nury Sword	
	TBD		40
	UI Development	Jay Garcia	
	Cemal Can Efe		80
Asim Safa		80	

Figure 1.11 A tree Panel with columns.

You already saw TextFields in a form when we discussed containers a short while ago. Next, we'll look at some of the other input fields that the framework has to offer.

1.3.5 Form input fields

Ext JS has a palette of eight input fields. They range from simple `Text` Fields, as you've already seen, to complex fields such as the `ComboBox` and the `HtmlEditor`. Figure 1.12 is an illustration of some of the available Ext JS form field widgets that come out of the box.

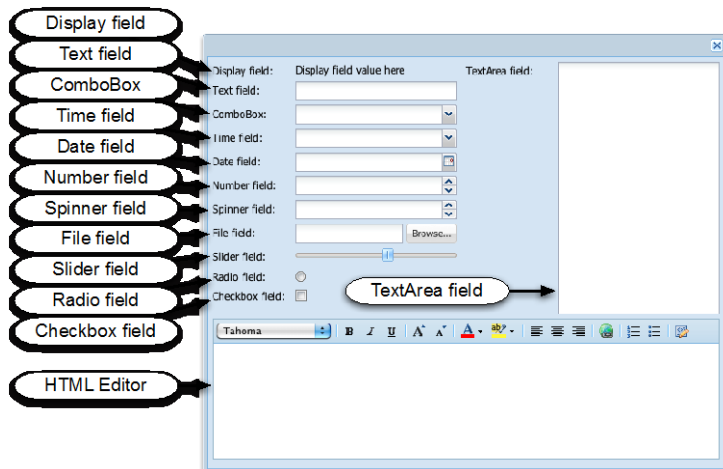


Figure 1.12 All of the out-of-the-box form elements displayed in an encapsulating Window

As you can see in figure 1.12, some of the form input fields look like stylized versions of their native HTML counterparts. The similarities end here, however. With the Ext JS form fields, there's much more than meets the eye!

Each of the Ext JS fields (minus the `HtmlEditor`) includes a suite of utilities to perform actions like get and set values, mark the field as invalid, reset, and perform validations against the field. You can apply custom validation to the field via `regex` or custom validation methods, allowing you complete control over the data being entered into the form. The fields can validate data as it's being entered, providing live feedback to the user.

TEXT FIELD AND TEXTAREA

The `Text` field and `TextArea` classes can be considered extensions of their generic HTML counterparts that bring forth extra features like validation. The `Text` field class is the base for many other complex widgets, such as the `ComboBox`, `Number` field and `Time` field.

RADIO AND CHECKBOX

Like `Text` field, the `Radio` and `Checkbox` input fields are extensions of the plain old `Radio` and `Checkbox` but include all of the Ext JS `Element` management goodness and have convenience classes to assist with the creation of `Checkbox` and `RadioGroups` with automatic layout management. Figure 1.13 shows a small sample of how the Ext JS `Checkbox` and `RadioGroup` classes can be configured with complex layouts.

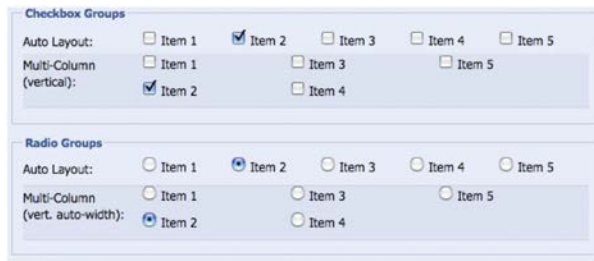


Figure 1.13 An example of the `Checkbox` and `RadioGroup` convenience classes in action with automatic layouts.

HTMLEditor

The HTML editor is WYSIWYG, like the `TextArea` on steroids. The `HtmlEditor` leverages existing browser HTML editing capabilities and can be considered somewhat of a black sheep when it comes to fields. There's much more to discuss about this field, which we're going to save for chapter 6. But for now, let's circle back to the `ComboBox` and its subclass, the `TimeField`.

TRIGGER FIELD FAMILY OF FIELDS.

The `Trigger` field widget is the base class responsible for rendering a button to the right of a `Text` field. Its subclasses are broken up into two groups, `Pickers` and `Spinners`. Included in the list of `Pickers` are the `ComboBox`, and `Date` field. The `spinners` include the `Spinner` and `Number` fields.

The `ComboBox` is easily the most complex and configurable form input field. It can mimic traditional option drop-down boxes or can be configured to use remote datasets via the data store. It can be configured to autocomplete text, known as *type-ahead*, entered by the user and perform remote or local filtering of data. It can also be configured to use your own instance of an `Ext JS XTemplate` to display a custom list in the drop-down area, known as the `BoundList`. Figure 1.14 is an example of a custom `ComboBox` in action.

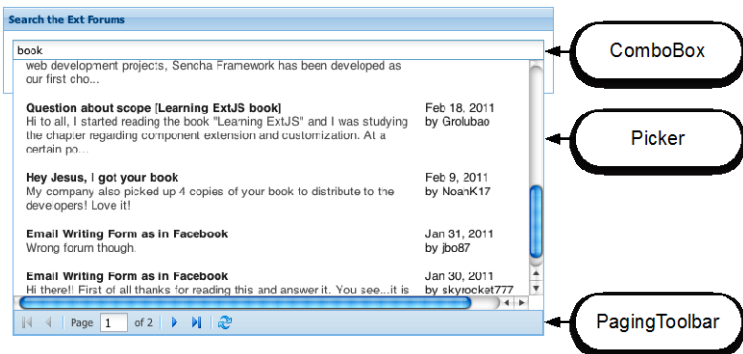


Figure 1.14 A custom `ComboBox`, which includes an integrated paging toolbar, as shown in the downloadable Ext JS examples

In figure 1.12, a custom combo box is being leveraged to search the Ext JS forums. The `ComboBox` here shows information like the post title, date, author, and a snippet of the post in the list box. Because some of the dataset ranges are so large, it's configured to use a paging toolbar, allowing users to page through the resulting data. Because the `ComboBox` is so configurable, we could also include image references to the resulting dataset, which can be applied to the resulting rendered data.

Here we are, on the last stop of our UI tour. Now we'll take a peek at some of the other UI components that work anywhere.

1.3.6 Other widgets

A bunch of UI controls stand out, which aren't major components but play supporting roles in a grander scheme of a UI. Look at the illustration in figure 1.15 for a palette of the various widgets rendered on screen.

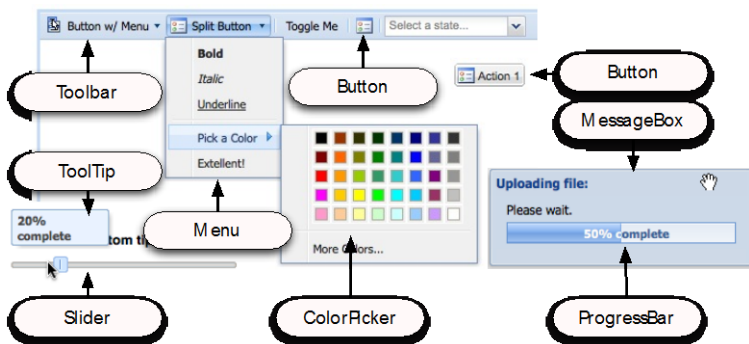


Figure 1.15 Miscellaneous UI widgets and controls

You've learned how Ext JS can help you get the job done through a large palette of widgets. You've learned that you could elect to use Ext JS to build an application without touching an ounce of HTML or integrate it with existing sites. You also got a top-down view of the framework, which included a UI tour. All of the material discussed thus far existed for Ext JS 3.0. Let's take a moment to discuss what's new in Ext JS 4.0.

1.4 Ext JS 4.0 – what's new?

I feel that I am not exaggerating when I say that Ext JS 4.0 is a revolution for JavaScript frameworks! There are so many enhancements to the framework that it is sometimes hard to grasp all that has changed. This is mainly because a lot of the changes are beneath the presentation layer, deepest darkest caverns of the Ext JS codebase, a place where I rarely venture due to its sometimes mind-bending complexity.

Next, we'll look at some of the most drastic transformations that the framework has undergone. If you have experience in version 3.0, you may have wondered why the size of the framework has grown, you'll learn why in the next few subsections.

1.4.1 Poof goes the Adapter layer!

Through the use of an adapter layer, Ext JS 2.0 and 3.0 were able to ride on top of the jQuery and Prototype and YUI libraries. With Ext JS 4.0, this is no longer the case.

While heralded by developers migrating from those libraries, the adapter layer has always been a source of contention for a number of reasons. The main issues being that the versions of the base libraries would change, and introduce bugs into Ext JS.

Another well-known issue is the problem of framework namespace collision. Ext JS 1.0 – 3.0 added to JavaScript by injecting methods into the function, String and Array prototypes. Being that other libraries did the same with similar method names, meant that Ext JS would trample on the changes that the base libraries made.

The Sencha development team made sure to prevent such collisions and sources of tension with other libraries by moving said features into the `Ext.util` namespace as `String`, `Function` and `Array` singletons. With such changes, the Sencha team decided to remove the adapter layer and make Ext JS work along side any other library, allowing you to use any version of those libraries without any fear that an upgrade of those libraries would cause problems with your Ext JS code.

1.4.2 New class system

Ext JS 4.0 comes with an entirely new class system that brings forth features such as dependency injection and on-the-fly class loading, a must have for internet-facing RIAs built with Ext JS 4.0.

Along with dynamic class loading comes the concept of Mixins, a modern Object Oriented Programming pattern that allows for multiple inheritance. This allowed the Sencha development team to be much more creative when developing the framework, reducing the amount of duplicate code, while increasing the level of functionality and sometimes the ease of use for some classes and widgets.

Learn about Mixins!

If you're new to the concept of Mixins, the following article explains this programming concept very well: <http://en.wikipedia.org/wiki/Mixin>.

While the new class system brings forth many new features, it comes at a cost however - new patterns. The new class system promotes vastly different patterns compared to that of Ext JS 3.0, when it comes to instantiation of a class as well as defining. While these new patterns can steepen the learning curve for Ext JS 4.0, rest assure that these new patterns will allow you to be more creative with your application code.

Speaking of classes, Ext JS 4.0 has a completely refactored data class system, which we should discuss.

1.4.3 Data package

The all-new data package in Ext JS 4.0 can trace its origins back to Sencha Touch, which brought forth terms like Model in place of Record. The changes to the data Package bring functionality and organization far beyond that of Sencha Touch, however.

The Ext JS 4.0 data package brings forth an explosion of classes, and include new members such as the `LocalStorage` proxy and `TreeStore`. The `LocalStorage` proxy allows data to be stored and retrieved using the browsers local storage feature, while the `TreeStore` replaces the Ext JS 3.0 tree loader, allowing you to do a lot more with Trees than ever before!

The data package comes with added features, such as associations and validations as well as a well-thought out reorganization of functionality. Figure 1.16 illustrates how features and functions of the data package are segmented and related.

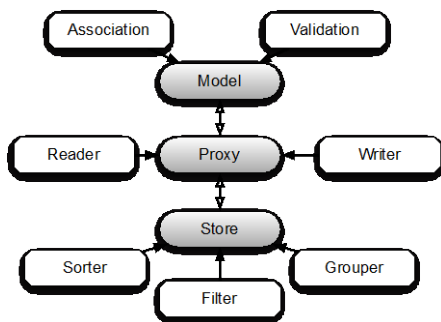


Figure 1.16 The Ext JS 4.0 data package

While we'll be going over this in greater detail later on, I figure it would make sense to wet your appetite with some detail.

In Ext JS 4.0, Models can directly use Proxies, where they could not in previous versions of the framework. Likewise, validations and associations are now performed at the model level.

While the data package has seen a lot of attention, the layout namespace has seen a lot of refactoring love as well.

1.4.4 Layouts – an explosion of code

As we discussed earlier, Ext JS 4.0 comes jam packed with 33 new layout managers, but only 13 of them are ones that you need to be aware of. This is because layouts are broken up into two main areas of functionality - Component and Container layouts.

The Component and Container layouts play two completely different roles in the framework. Component layouts are responsible for arranging the HTML for components, while Container layouts are responsible for managing the location and size of child components.

Since we're on the topic of layouts, I should shed some light on the new docking system that Ext JS 4.0 brings to the table.

1.4.5 New docking system

Originating in Sencha Touch, Panels in Ext JS can have widgets such as Toolbars arranged on the outside of the area known as the Content body, affording more UI arrangement flexibility than ever before with this widget.

Figure 1.18 shows us three Toolbars docked on the outside of a Panel.

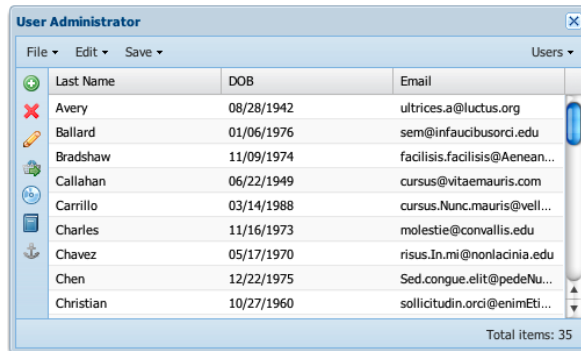


Figure 1.18 Demonstrating the new docking feature of the Ext JS Panel

Figure 1.18 illustrates three Toolbars rendered on the top, bottom and left of a Panel, which wasn't possible with any previous versions of Ext JS without deep nesting of containers and layouts. This is all made possible via the Component layout known as Dock.

While using the Dock layout is something that you might be able to envision taking full advantage of, if your application uses grid Panels, what we're about to discuss might get you a bit excited.

1.4.6 Grid Panel improvements

The Sencha development team literally worked night and day on features like the grid Panel and the results show, especially after taking good look at what's changed since Ext JS 3.0.

New features to the grid Panel include what is known as the "Infinite Grid", allowing you paginate through large datasets without the need to include a paging toolbar. Other features for the grid package include a reorganization of the namespace to better group classes.

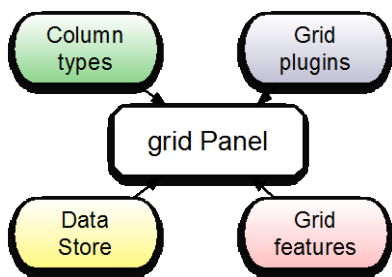


Figure 1.19 The grid Panel supporting areas of functionality.

The grid area of code has been segmented by groups of code, including Column types, plugins, and features. Even though not technically in the grid namespace, the data Store is a supporting class for the grid Panel, so I included it in Figure 1.19.

This level of organization of the grid package means that you have more flexibility in configuring grid Panels, allowing Ext JS to only implement code that is required. For instance, if you want to allow cell editing, you simply include the CellEditing plugin in your grid Panel configuration. Likewise, if you want to include Drag and Drop functionality, simply include the DragDrop plugin.

Other bits of functionality were migrated to the so-called feature namespace, which is similar to plugins, but different. I don't want to muddy the water with details of how features work, but it's good to note that grid goodies like row Grouping or providing a Summary row of your data can be engaged only if you desire it so.

As we just learned, the grid Panel endured a lot of changes. The story of major change does not end here. As we'll learn, the Tree Panel has undergone some serious changes as well!

1.4.7 Tree Panel now closer to Grids

While the code for the Ext JS tree Panel has relatively stayed the same for Ext JS versions 1.0 through 3.0, the Ext JS 4.0 tree Panel has completely been rewritten. If we could put a

family tree analogy to the difference between prior versions of the grid and Tree Panels, I would say that they were at best 3rd cousins. In Ext JS 4.0, they are siblings!

To emphasize this, I've put together a simple diagram.

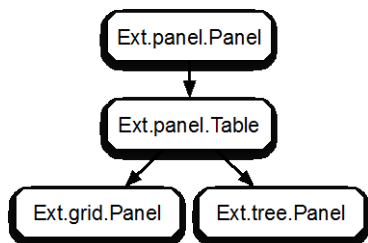


Figure 1.20 The tree and grid Panels share the same superclass!

As illustrated in Figure 1.20, the grid and Tree Panels are siblings because they share the same superclass, meaning they share the same base code. The good news is that once you learn one of the, the learning curve is reduced for the sibling class. Having the tree Panel share the same base code means that you can have things like columns in your tree views.

All of this said, the tree Panel does not contain a lot of the functionality that the grid Panel sports, such as the Summary row plugin or column locking. The tree Panel also must use the `TreeStore` class from the data package to manage and display hierarchical data.

We just covered two of the major data-bound views that have been with the framework since its early days. What we're going to learn about next is the all-new charting package.

1.4.8 Draw and Charts

Charts were first introduced in Ext JS 3.0 with relatively little fanfare. This had to do with a couple of facts. The first being that they were Flash-based charts repackaged from the YUI library. The second being that upgrades to the YUI packaged charts often lagged behind a few revisions, frustrating developers.

With Ext JS 4.0, the YUI charting package was tossed and built from scratch in two major sections. The first is Ext Draw, which is a mini-framework inside of Ext JS that has its roots from lessons learned by Raphael JS, a Sencha labs project for drawing in the browser using VML, SVG or Canvas.

EXT DRAW NOT COVERED IN THIS BOOK

Due to the complex nature of SVG, VML and Canvas, Ext Draw is not covered in this manuscript. There are numerous resources on the web, such as http://www.w3schools.com/svg/svg_path.asp to learn these technologies.

The second is the charting package, which uses Ext Draw as a base. With the new charting package come two new graphs, Scatter and Radar. Figure 1.21 illustrates the radar chart in action.

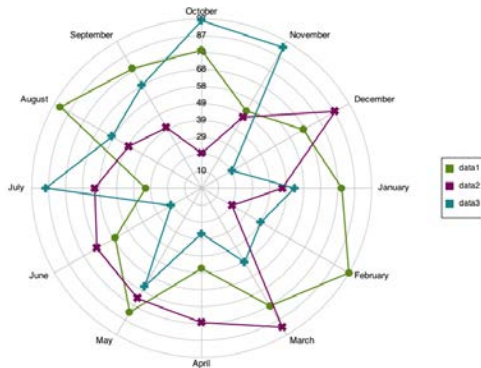


Figure 1.21 Ext JS contains new charts that do not use Flash.

There are lots of elements from the UI widgets that we've talked about, but there are others that are under the hood that are worth mentioning.

1.4.9 New CSS Styling architecture

Ext JS uses SASS (Syntactically Awesome Style Sheets) to allow both the Sencha development team as well as us to create custom themes. This means if you want to change your entire color scheme, you can do so with relative ease if you know Sass.

LEARN MORE ABOUT SASS

Sass has taken the world of style sheet management by storm and has arguably revolutionized how people style their web pages and apps. To learn more about this utility, check out SASS and Compass in Action via <http://www.manning.com/netherland/>.

Custom style sheets and widgets are play their roles to enable you to develop applications with Ext JS. They need something to tie them together, and with Ext JS 4.0, Sencha has delivered such a tool.

1.4.10 New MVC Architecture

One of the things that Ext JS has lacked is a solid pattern for developing applications with the framework. This is not the case with Ext JS 4.0! Using the lessons learned with the Sencha Touch, Ext JS 4.0 comes with a solid MVC architecture allowing you to develop code using the tried and true MVC pattern. We'll be going over this in great detail in the last two chapters of this book.

The new stuff for Ext JS 4.0 does not just apply to what can be used in the browser. There are other tools that the framework comes with that you could leverage in your application build process.

1.4.11 Bundled packaging tool

Earlier, we learned that Ext JS 4.0 comes with a dynamic class loading system. While the class loader is a great solution for internet based Ext JS applications, intranet-based applications often have higher demands on response times, which is why Sencha now includes their popular jsBuilder packaging and minification tool, the same tool that they use to build and package Ext JS and Sencha Touch!

We've spent a lot of time looking at what's new in the framework. I think it's time that we download it and begin using it.

1.5 Downloading and configuring

Even though downloading Ext JS is a simple process, configuring a page to include Ext JS isn't as simple as referencing a single file in HTML. In addition to configuration, you'll learn about the folder hierarchy and what folders are and what they do.

The first thing you need to do is get the source code. Visit the following link: <http://www.sencha.com/products/extjs/download/>.

The downloaded file will be the SDK in a zip file, which weighs in at over 30 MB in size. I'll explain why this file is so large in a bit. Now, extract the file in a place where you serve JavaScript. In order to leverage Ajax and view the documentation without having to visit sencha.com, you're going to need a web server. I typically use Apache configured locally on my computer, which is free and cross--platform, but IIS for Windows will do. Let's peek at what you just extracted.

1.5.1 Looking at the SDK contents

If you're like me, you probably checked the size of the files extracted from the downloaded SDK zip file. If your jaw dropped, feel free to pick it back up. Yes, over 30 MB is rather large for a JavaScript framework. Pay no attention to the size for now; figure 1.22 shows what was extracted.

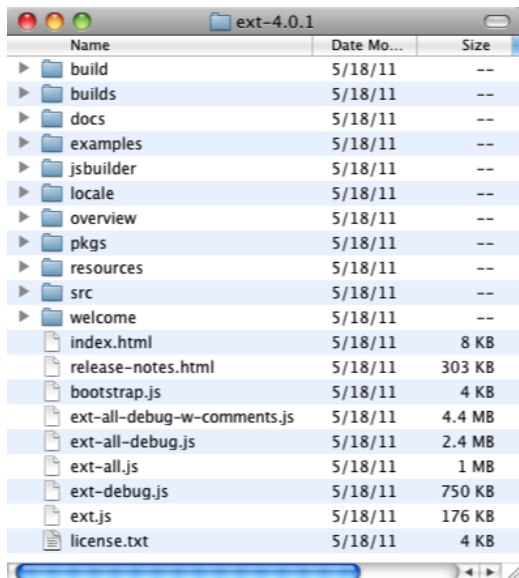


Figure 1.22 A view of the Ext JS SDK contents

Looking at the contents of the SDK, you see a lot of stuff. The reason there are so many folders and files is that the downloadable package contains a few copies of the entire code base and CSS. It's this way because you get the freedom to build or use Ext JS any way you see fit. Table 1.1 explains what each of the folders is and what it does.

Table 1.1 The contents of the Ext JS SDK

Folder	What it does
build	Contains the necessary scripts to utilize jsBuilder to concatenate and minify your application code.
builds	This directory contains three different build of the Ext JS framework. First is the sandbox version, where you can run Ext JS 4.0 inline with Ext JS 3.0 to mitigate migration risk. The second is the core of the library. The core contains DOM management and various utilities in the framework. The last item is Ext JS foundation, which is the very base of the Ext JS framework.

docs, overview and welcome	The docs folder holds the full API documentation, while the overview directory contains a quick introduction to the framework. The welcome folder contains the necessary resources to support the framework's splash screen, which is visible by double clicking on index.html.
examples	Holds all of the example source code, which is great source to learn by example (no pun intended).
jsbuilder	This directory contains the binaries and source code for jsBuilder.
locale	Contains 45 spoken language translations to replace various texts in the framework.
pkgs	This is where the entire framework is broken up into minified and concatenated chunks to allow for browser consumption over slower connections. It's broken up into foundation, DOM, classes and extras. Classes is by far the largest of the set containing all of the widgets and store code, while the extras contains utilities like JSON and the Ext JS MVC application.
resources	This is where the CSS, images and SASS source code are located.
ext*.js	There are various ext*.js files in the root of the Ext JS distribution. Know that anything with "-debug" in the name is a non-minified version of that file. These can be broken down into two groups. First are ext.js and ext-debug.js. These two contain the foundation of the framework. Include these when you want to use the Ext class Loader. The ext-all* files are the entire library packaged into one file. We'll be using ext-all-debug for our exercises.

Although there are quite a few files and folders in the distribution, you need only a few of them to get the framework running in your browser. Now is a good time to talk about how to set up Ext JS for use.

1.6 *Take it for a test drive*

For this exercise, we're going to create an instance of `Ext.form.Panel`, which will be rendered inside of an `Ext.window.Window`. The `form Panel` will contain two text input fields, and a button to provide some feedback once pressed.

Listing 1.1 demonstrates how we will bootstrap our application code.

Listing 1.1 Creating our hello_world.html

```

<link rel="stylesheet" type="text/css"
      href="extjs/resources/css/ext-all.css" />           #1
<script type="text/javascript"
      src="/extjs/ext-all-debug.js"></script>           #2
<script type="text/javascript" src='hello_world.js'>
</script>                                               #3
#1 Include ext-all.css
#2 The Ext JS code
#3 The soon-to-be-created hello_world.js

```

Listing 1.1 includes the HTML markup for a typical Ext-only setup, which includes the concatenated CSS file, ext-all.css #1 and the required JavaScript file, ext-all-debug.js #2. Last, you include our soon-to-be-created hello_world.js file #2.

If you haven't noticed it, we're using /extjs as the absolute path to our framework code. Be sure to change it if your path is different. Next, you create a script tag pointing the hello_world.js file, which will contain your main JavaScript code.

Moving forward, we're going to create the hello_world.js file in two phases. The first being the construction of form Panel and its related child components and the second being the Window that will render the form Panel.

Listing 1.2 Creating hello_world.js

```

var tpl = Ext.create('Ext.Template', [                    // 1
    'Hello {firstName} {lastName}!',
    ' Nice to meet you!'
]);
var formPanel = Ext.create('Ext.form.FormPanel', {       // 2
    itemId      : 'formPanel',
    frame       : true,
    layout      : 'anchor',
    defaultType : 'textfield',
    defaults    : {
        anchor      : '-10',
        labelWidth  : 65
    },
    items       : [                                       // 3
        {
            fieldLabel : 'First name',
            name        : 'firstName'
        },
        {
            fieldLabel : 'Last name',
            name        : 'lastName'
        }
    ],
    buttons : [
        {
            text      : 'Submit',                          // 4
            handler : function() {
                var formPanel = this.up('#formPanel'),
                    vals      = formPanel.getValues(),
                    greeting  = tpl.apply(vals);
            }
        }
    ]
});

```

```

        Ext.Msg.alert('Hello!', greeting); // 5
    }
}
});
#1 Create an instance of Template
#2 Configure the form Panel
#3 Setup two input fields
#4 Configure a button for feedback
#5 Show an Ext.Msg alert dialog

```

Listing 1.2 contains the necessary code to configure a form Panel that contains two input fields and a Button. Since we're going to cover all of this material in much greater detail as we progress through this book, I'm going to just gloss over how this is all put together.

First, we create an instance of Ext.Template #1, which we'll use later on to create a dynamic dialogue text body. Next, we move on to create an instance of Ext.form.FormPanel #2, which contains two text input fields #3 and a Button #4. The Button is configured with a handler that uses the Template we configured before and values from the form Panel to display an Ext.Msg alert dialogue.

We're almost done with our hello-world example. Our form has yet to be rendered on screen. For this, we need to call Ext.onReady. We'll also wrap the form Panel inside of Window to demonstrate the flexibility of the framework.

Listing 1.3 Putting it all together

```

Ext.onReady(function() { // #1

    Ext.create('Ext.window.Window', { // #2
        height : 125,
        width  : 200,
        closable : false,
        title   : 'Input needed.',
        border  : false,
        layout  : 'fit',
        items   : formPanel // #3
    }).show();

});
#1 Call Ext.onReady
#2 Render the Window
#3 Include our formPanel

```

Listing 1.3 contains code to render our form Panel inside of an Ext JS Window. Here is how it works.

We first call Ext.onReady and pass in an anonymous function, which gets executed when Ext JS deems that the browser is ready to have the DOM manipulated. Inside of this anonymous function, is where we create our Ext.window.Window instance #2, which contains our form Panel instance #3.

Here is what it looks like on screen.



The screenshot shows a light blue window titled "Input needed." with a white background. It contains two text input fields: "First name:" and "Last name:". Below the fields is a "Submit" button.

Figure 1.23 Our example Window rendering our form Panel.

Figure 1.23 shows our hello-world example rendered on the screen. To exercise the Submit Button handler, we need to enter data into the two input fields and press the button. If we've done everything correctly, we should see the `Ext.Msg` alert dialog using the data that we placed in the form.

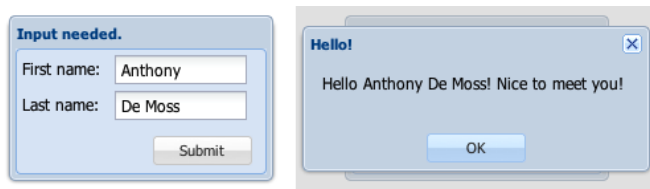


Figure 1.24 The final result of our hello-world example.

There you have it! We just used Ext JS to render a `form Panel` with related input fields and a button inside of an Ext JS Window. While this example was simple in nature, I think it shows you the power of Ext JS.

1.7 Summary

In this introduction to Ext JS, you learned how it can be used to build robust web applications or integrated into existing websites. You also learned how it measures up against other popular frameworks on the market and that it's the only UI-based framework to contain UI-centric support classes such as the `Component`, `Container`, and `Layout` models.

We explored many of the core UI widgets that the framework provides and showed that the number of prebuilt widgets helps rapid application development efforts. In doing that, we talked about some of the changes that Ext JS 4.0 has brought forth, such as all new non-Flash charts and the MVC package.

Last, we discussed where to download and how to set up the framework with each individual base framework. We created a "Hello world" example of how to use an Ext JS Window to render a `form Panel` with a `Button` that displays an `Ext.Msg` alert dialog with some simple JavaScript.

In the chapters to follow, we'll explore how Ext JS works from the inside out. This knowledge will empower you to make the best decisions when building well--constructed UIs and better enable you to leverage the framework effectively. This will be a fun journey.