Chapter 1

Getting to Know Flex

In This Chapter

- Understanding what Rich Internet Applications (RIAs) are
- Comparing Flex to other RIA technologies: Flash, AJAX, and Silverlight
- Taking Flex applications offline by using Adobe Integrated Runtime (AIR)

Adobe Flex is an application development platform that you can use to build Rich Internet Applications (RIAs). Flex applications are Web-based, but they provide immersive levels of interactivity and rich media experiences that make them seem more like computer desktop programs than traditional Web applications. Flex is the ideal technology to use when you want to create complex data visualization, performance dashboards, multimedia experiences, and countless other interactive applications. RIAs are raising the bar for Web applications, and Flex is leading the way. In this chapter, we discuss what Flex is, what it isn't, and how it compares to other technologies.

Using Flex to Develop Rich Internet Applications

The computer world has come a long way from static HyperText Markup Language (HTML) Web pages. Over the past two decades, rich online experiences have gradually evolved into RIAs. Flex is on the forefront of technology that allows you to create such engaging Web-based applications, but it has taken nearly 20 years since the first HTML page was created to get to where we are now. (See the nearby sidebar for more on the journey from HTML to RIA.)

Understanding what an RIA is

The term *Rich Internet Application* (RIA) was coined in a Macromedia white-paper written in 2002 to describe a new model for application development that separated the back-end data services from a rich front-end client. One of the cornerstones of RIA development is the ability to asynchronously

load data within the application. Simple HTML Web pages require a full page refresh to load new data. RIAs, on the other hand, load data *asynchronously,* which means they can load chunks of data without requiring page refreshes and they keep track of the application state in memory. Flex applications are *stateful clients*, which means they store data about the current state of the application, such as the content of a shopping cart, in memory in the client.

RIAs usually load data by using eXtensible Markup Language (XML). Asynchronously loading XML is an integral part of all RIA technologies (not only Flex). Version 4 of Flash, which was released in 1999, was the first version of Flash that let developers load external XML data into Flash applications.

Taking a look at the rise of Flex

Macromedia, which Adobe later acquired, introduced the first version of Flex in March of 2004. Initially, the first two major releases, Flex 1 and 1.5, were expensive server-based products. A license for Flex 1.5 cost about $15,000, and you had to deploy a server application that would compile your Flex applications and serve them to the user. These initial versions of Flex were based on Flash Player 7 and ActionScript 2, and the code editor was based on the Macromedia Dreamweaver editor.

The release of Flex 2 marked a dramatic shift in the product line. Flex 2 was no longer a server technology at all; instead, Flex was a completely client-side product. The cost dropped dramatically, and Adobe rewrote the entire Flex framework and the Integrated Development Environment (IDE) from the ground up. Flex 2 was based on Flash Player 9 and ActionScript 3, which brought serious performance gains.

Flex 3 added additional functionality to Flex Builder, such as refactoring and enhanced styling support, as well as new data visualization components in the Flex framework. Flex 3 also marked the official open-source release of the Flex SDK and Flex compiler. For more on the open-source aspect of Flex, visit `http://opensource.adobe.com`.

Defining Flex

Defining exactly what Flex is can be confusing because Flex actually includes a combination of different technologies. Flex is not a single software product, but instead includes the following four main pieces:

- **Languages:** ActionScript 3 and MXML

  You use a combination of ActionScript, a scripting language, and MXML, a markup language, to create Flex applications. MXML is similar to HTML, so if you have experience creating HTML Web pages, then you should be able to figure out MXML pretty easily.

- **Component framework:** Flex SDK

  The *Flex SDK* (also known as the *Flex framework*) is a set of user interface components, such as lists, buttons, and charts, that you use to build Flex applications. The Flex SDK (with the exception of the charting package) is free and open source.

- **Integrated Development Environment (IDE):** Flex Builder

  You use Flex Builder to edit your code, compile your applications, debug your code, and profile performance. Flex Builder is an integrated development environment (IDE) sold by Adobe.

- **Cross-browser runtime:** Flash Player

  You deploy Flex applications in a Web browser with the Flash Player plug-in. You can also deploy Flex applications as standalone desktop applications by using the Adobe Integrated Runtime (AIR).

From HTML to RIA

When programmers began tinkering with the Web, they used HTML to create Web pages that looked like actual pages of a book. These pages contained a bunch of text that someone wrote and published for the world to read. Those were the good old days of choppy animated images and heavy use of the `<blink>` tag, which was about as "rich" as the Internet got in the early '90s. Then, Web servers became more sophisticated and started serving up dynamic content, depending on what users were looking for or how they interacted with the Web site. For the first time, Web pages started turning into *Web applications.*

Server-side languages, such as Java, PHP, and ColdFusion, increased the capabilities of Web applications on the back-end. An e-commerce site could let you keep track of items in your shopping cart while you browsed the retailer's site, but each time you added an item, the full Web page would have to reload so that the number of items shown in your cart would stay updated. The state of the application was stored completely on the server, either in memory or in a database. Whenever the server wanted to show you something new, you were sent to a new static HTML page.

But then those static Web pages became animated when a small company called FutureWave Software released FutureSplash Animator, which was purchased by Macromedia and renamed to Flash in 1996. A whole new breed of silly animated movies and impressive visual effects appeared. The Flash platform brought a greater level of interactivity to the Web than ever before. Designers could create interactive visual experiences that went far beyond what was possible with simple HTML.

Flash 5 included the ActionScript programming language, which turned Flash into much more than a simple animation tool. Developers started creating complex Flash interactive experiences by combining the visual and programming capabilities to produce full-blown applications within a Web browser. The first RIAs were being created before the term RIA was widely used or understood. Then in 2004 Macromedia released the first version of Flex. Finally developers had a tool that was specifically for creating RIAs.

What's next?

Adobe is actively developing Flex 4, codenamed Gumbo. This version will focus on an improved workflow between developers and designers, and will likely include an improved component framework model to more gracefully separate the visual design of your Flex applications from the underlying code. Toward a similar goal, an additional product in the Flex product line (code-named Thermo) will focus specifically on allowing designers to create complex RIA user interfaces and interactions by using a visual editor, which will create Flex application code that the designer and developer can share.

Flex has been largely in a class of its own when it comes to RIA development platforms, but that reign is now being challenged because some serious competitors are entering the RIA market. Microsoft's Silverlight platform and Sun's JavaFX are two RIA development products aimed directly at taking on Flex. Over the next few years, the competition will get serious, which can only be a good thing for all the developers out there. RIA development is an exciting field; you've chosen wisely!

Comparing Flex to Flash, AJAX, and Silverlight

Because you bought this book, we assume you've decided that Flex is the right choice for your project. But, if you need to sell the decision to your boss, he or she will probably ask you how Flex compares to related technologies. The following sections summarize some of the key differences between Flex and a few other RIA technologies.

Flex versus Flash

Whenever Doug tries to explain what he does for a living, someone always asks whether Flex is Flash. This question quickly leads to a heated discussion about Flash advertisements, and he has to calm everyone down and explain that, fundamentally, Flex is a Flash-based technology, but no, he doesn't make those annoying ads — he makes applications. Flex is an application development framework and toolset that you can use to create RIAs. These applications are delivered over the Internet by using the Flash Player.

So, what's the difference between Flex and Flash? The following list identifies a few of the most important features that Flex offers that are not available in Flash:

- **The Flex framework:** Flash has its own component set that has some of the same functionality as the Flex SDK, but it does not provide as many components and does not include charting components, layout containers, and other framework features that are very useful for developing large applications.
- **MXML:** You can use MXML markup to create your Flex applications, but this markup language is not available in Flash. Flash does use the same ActionScript 3 scripting language, however.
- **A powerful Integrated Development Environment (IDE):** Flex Builder was designed specifically to build applications, as opposed to the Flash Authoring tool, which was originally designed to create animations. You can use both tools to create RIAs, but Flex Builder has features like code-hinting, a powerful debugger, and a profiler that make it a more powerful development tool.

You can use Flash, rather than Flex, to create RIAs, but you have to work in the Flash Authoring environment, which means you don't get the benefits of Flex Builder (such as code hinting), and you can't use the MXML markup language. You may find the Flash Authoring tool really helpful if you're creating animated movies. It uses the timeline metaphor when you create animations and includes drawing tools. Flex Builder, on the other hand, is designed for application development, not animation creation. Flex Builder doesn't include any drawing tools, and it has no timeline.

If you have to decide between Flex and Flash for a specific project, think about exactly what kind of project you're working on. Flex excels when you're creating large desktop-like applications, sharing work among a team of developers, or visualizing data with charts and graphs. For other kinds of projects that require complete control over the visual experience, such as games or advertisements, Flash might be a more appropriate choice.

Flex versus AJAX

Asynchronous JavaScript and XML (AJAX) is a technique that you can use to load data into HTML Web pages without refreshing the full page. Instead of sending a new HTML page to the user's Web browser for every change, AJAX applications send XML asynchronously and update relevant portions of the screen when needed. By using AJAX, you often end up with a much more responsive user interface and a more desktop-like application experience. Flex applications also asynchronously load XML data.

JavaScript and ActionScript are very similar languages; both are based on a language specification called *ECMAScript*, which is developed by Ecma International, a collective group of representatives from various technology companies (Adobe is a member). The ECMAScript specification has had few revisions. ECMAScript Edition 3 is the latest published edition, and Ecma

is currently developing Edition 4. JavaScript is currently an implementation of ECMAScript Edition 3. The current version of JavaScript most closely resembles ActionScript 2, which also implemented ECMAScript Edition 3. Adobe released ActionScript 3 in 2006, which is based on the preliminary draft of ECMAScript Edition 4. After Ecma officially releases a new version of ECMAScript Edition 4 and Web browsers support an updated version of JavaScript, the JavaScript programming language should become more like ActionScript 3.

Because of the similarities between the ActionScript and JavaScript languages, and their similar approaches to asynchronously loading XML data, the fundamental benefits of Flex have to do with its underlying Flash Player technology. Here's a rundown of some of those benefits:

- **Multimedia capabilities:** Flash Player allows you to create a whole range of rich multimedia experiences that you simply can't achieve by using HTML and JavaScript. Flash has powerful graphics capabilities that can do complex drawing and image manipulation. In addition, Flash supports audio and video streaming, so many leading online video sites use it to play video on the Web.
- **Cross-browser support:** You can be sure that any Flex application you develop will look and behave the same way in all browsers on all platforms. Web browsers all have their own quirks and idiosyncrasies when it comes to how they render HTML and even how they run JavaScript. When you develop AJAX applications, you need to test your application in multiple Web browsers to make sure your application is compatible with them all. However, because Flex applications rely on the Flash Player, you can be assured that your application will look the same, pixel for pixel, and behave the same across all browsers.

Flex versus Silverlight

Microsoft's competitive RIA technology is Silverlight, a browser plug-in, like Adobe's Flash Player. Just like you have to install the Flash Player plug-in to run Flex applications, you need the Silverlight plug-in to run Silverlight applications. Because Silverlight is fairly new, the Silverlight plug-in isn't nearly as common as Flash Player. Silverlight will likely become more widely used in the future, but right now, the Flash Player plug-in has a strong advantage because of the large number of computers on which it is installed.

The first release of Silverlight 1.0 included the browser plug-in and focused on streaming video on the Web. This release certainly competed with Flash Player because it focused on some of the media features of Flash, but it didn't really threaten Flex's position because Silverlight 1.0 didn't contain a set of user interface controls that you could use to build RIAs. Silverlight 1.0 provided all the low-level graphics capabilities but none of the application framework pieces.

In early 2008, Microsoft released the first beta version of Silverlight 2.0 (originally named Silverlight 1.1), which included a set of UI controls, such as a DataGrid, CheckBox, and Slider. These new controls make Silverlight a closer competitor with Flex. The competition between Flex and Silverlight has just begun, and it's too early to draw any firm conclusions about how serious a competitor Silverlight will become.

Taking Flex to the Desktop with AIR

Adobe developed Adobe Integrated Runtime (AIR), previously code-named Apollo, to let you deploy Flex applications as computer desktop applications. By using AIR, you can create your own desktop applications that can run natively on Windows, Mac, and Linux operating systems. You can create AIR applications as Flex applications or AJAX applications, so if you know how to create Flex or AJAX applications, you can create desktop applications, too.

When deciding whether AIR is the right technology for your application, consider these three main features that AIR provides:

- **Local file-system access:** One of the main reasons for moving from a Web-based Flex application to a desktop AIR application is the integrated local file-system access that AIR offers. If you build a Flex-based AIR application, you get a few extra tools that don't come in the normal Flex framework. These tools let you read and write to the user's local file system.
- **Integrated Web browser:** The AIR runtime includes a built-in Web browser: the open-source WebKit browser. This Web browser allows you to load full HTML pages right into your application, something that you can't do in a Web-based Flex application. The AIR framework also lets you display PDF files within your application.
- **Embedded SQLite database:** AIR applications can access an embedded database for offline database access. So, you can build applications that can connect to a server database (like a typical Flex application can) or to an offline database if your application is only occasionally connected to the Internet.

Using the AIR-specific framework controls falls outside the scope of this book. If you're interested in finding out more about AIR, visit `www.adobe.com/products/air`.