CONTENTS

About the Author ................................................................. xi
About the Technical Reviewer ................................................ xiii
Acknowledgments ................................................................. xv
Introduction ................................................................. xvii

Chapter 1: Flex Basics ................................................................. 1
  The Internet, then and now ...................................................... 2
  HTML and dynamics .............................................................. 2
  Flex and RIA ............................................................... 6
  Flex, Flex Builder, and ActionScript 3.0 ................................... 8
    Eclipse and Flex Builder 3 .................................................. 8
  Installing Flex Builder 3 ..................................................... 9
    Installing Flex Builder as an Eclipse plug-in ......................... 10
    Installing Flex Builder 3 .................................................. 12
  Summary ................................................................. 20

Chapter 2: Flex and Flex Builder 3 ............................................. 23
  Starting out in Flex Builder 3 .................................................. 24
    Creating a Flex project .................................................... 27
    Creating a Flex application .............................................. 33
    Changing the properties ................................................... 36
  Anatomy of a Flex application ............................................. 39
    MXML ............................................................... 39
    Getting help ............................................................. 43
  Going behind the scenes ...................................................... 47
    Deploying the files ....................................................... 48
    Looking at generated ActionScript code ................................ 49
  Summary ................................................................. 52
Components ........................................... 177
  Adding a component .................................... 182
   Components and data .................................... 185
Custom events ......................................... 189
  Declaring a custom event .................................. 190
   Creating the event ...................................... 192
    Dispatching the event .................................... 193
     Passing data ......................................... 196
Summary ............................................. 199

Chapter 6: Flex and XML .................................... 201
Data sources .......................................... 202
XML: A very brief introduction ................................. 203
Using XML in Flex .................................. 205
  Using the HTTPService tag .................................. 207
   Displaying data ........................................ 209
XML and ActionScript 3.0 .................................. 211
  The ArrayCollection class .................................. 212
When good code goes bad ................................... 214
Flash Player security ....................................... 218
Introducing E4X ......................................... 220
The Model tag .......................................... 225
The DataGrid control ...................................... 226
Modifying DataGrid columns .................................. 227
The DateFormatter class ..................................... 229
Editing and rendering data ................................ 232
Images in the DataGrid container ................................. 241
The Tree control ......................................... 247
The AdvancedDataGrid component.............................. 258
Summary ............................................. 262

Chapter 7: Formatting and Cascading Style Sheets .............. 265
Flex and CSS ........................................... 266
  CSS: The basics ........................................ 267
   Working with CSS in Flex .................................. 268
Flex 3 and CSS ........................................ 276
  Flex Style Explorer ..................................... 276
   Converting to CSS .................................... 280
Validating and formatting ................................ 285
  NumberValidator ...................................... 286
ActionScript and validation ................................ 289
  The StringValidator ..................................... 296
Summary ............................................. 298
Using PHP or ASP.NET as the application server ........................................ 481
  Preparing the database ........................................................................... 481
  Building the application .......................................................................... 481
Summary ...................................................................................................... 487

**Chapter 13: Printing** ........................................................................... 489
  The printing process ................................................................................ 490
  Creating a separate Print container ....................................................... 493
  Printing and components ...................................................................... 499
Summary ...................................................................................................... 502

**Chapter 14: Charting** ......................................................................... 505
  Using the PieChart component ............................................................... 506
  Using the ColumnChart component ....................................................... 514
  Animating the chart ............................................................................... 519
  Using the BarChart component .............................................................. 522
Summary ...................................................................................................... 523

**Chapter 15: Adobe Integrated Runtime (AIR)** .................................... 525
  Understanding AIR ................................................................................ 526
  Hello World .......................................................................................... 528
  The Application Descriptor .................................................................... 536
  Deploying an AIR application ............................................................... 538
    Digital signatures ................................................................................ 541
  Converting an existing Flex application ............................................... 545
  Conclusion ............................................................................................. 548

**Appendix** ............................................................................................. 550

**Index** .................................................................................................. 569
ABOUT THE AUTHOR

Charles E. Brown is one of the most noted authors and teachers in the computer industry today. His first two books, Beginning Dreamweaver MX and Fireworks MX Zero to Hero, have received critical acclaim and were consistent bestsellers. In early 2004, Charles coauthored a book on VBA for Microsoft Access—VBA Access Programming.

In addition to his busy writing schedule, Charles conducts frequent seminars as an Adobe Certified Trainer. His topics include Flex, Flash, Dreamweaver, ActionScript programming, and After Effects. He also does seminars about Java and web design, and he is frequently called in as a consultant for major websites.

Charles is a noted classical organist, pianist, and guitarist, and studied with such notables as Vladimir Horowitz, Virgil Fox, and Igor Stravinsky. It was because of his association with Stravinsky that he got to meet, and develop a friendship with, famed twentieth-century artist Pablo Picasso.

Charles can be contacted through his website, a continuous work in progress, at www.charlesbrown.com.

David turned his hand to writing and teaching about web technologies after a successful career spanning nearly 30 years in BBC radio and television as a reporter, producer, and editor. He lived in Japan for nine years, first on loan from the BBC to the Japan Broadcasting Corporation (NHK) as an advisor on English-language broadcasting, and later as BBC correspondent in Tokyo reporting on the rise and collapse of the bubble economy. In 1991–92, he was President of the Foreign Correspondents’ Club of Japan.

When not pounding the keyboard writing books or dreaming of new ways of using PHP and other programming languages. David enjoys nothing better than visiting his favorite sushi restaurant. He has also translated several plays from Japanese.
ACKNOWLEDGMENTS

I couldn’t have done this book without the help of a lot of people.

Every time I thought I wrote the perfect chapter, David Powers, my incredible technical editor, brought me back to reality. His wisdom and guidance took this book in some slightly different directions from the first edition. I also want to thank him for his contributions regarding the use of the PHP technology.

I have to thank my project manager, Sofia Marchant, for developing more than a few gray hairs with an ever-changing production schedule. Working in a beta testing environment is not the easiest of things to do, and she was great about keeping everything moving smoothly.

I want to thank all of my many friends and co-developers (including some students at my training classes) for their invaluable suggestions and insights.

Finally, I want to thank the many kind readers who wrote words of encouragement on Amazon.com as well as other services (including e-mailing me). Their many words gave me some great ideas for this book.
INTRODUCTION

I can’t believe that we have now reached the second generation of Flex. It seemed like I had just finished the first edition and, within a few weeks, we were in a long and ever-changing series of betas for Flex 3. In the course of that period, many of the chapters you read in this book were rewritten three or four times.

Let me begin by thanking the many readers who took the time to write kind reviews for Amazon.com and other places. I read nearly every suggestion and incorporated them into this edition. I cut down a bit on the technical ActionScript explanations and focused on the features of Flex itself.

After years of doing technical training, where I have only a couple of days to cover large topics, I have learned to substitute shorter, and more pointed, explanations that clarify a concept in place of larger, more technical (and often confusing) explanations. In other words, I often like to get right to the heart of the matter, without taking circuitous routes.

Please keep a few things in mind when reading this book. First, you will find that the techniques I show you are techniques that reflect my style of programming and design. Certainly, there are many alternative ways of arriving at the same point. It is impossible for any one book to cover all possible variations, especially with topics as large as I cover here. If you find a different way of doing something, by all means use it if it works for you.

Second, I very purposely kept my examples simple in order to illustrate a point. I do not want you, the reader, to get caught up in just following recipe-like instructions that do little more than test your ability to read and follow instructions. While I have a case study in the book, each chapter will stand on its own, without reliance on exercises done in previous chapters. For that reason, you can open to nearly any chapter and just work on the subject of that chapter.

Third, I am assuming that you already have at least a cursory knowledge of object-oriented programming concepts. While I do intersperse many of these concepts throughout the chapters, it is only a very basic introduction. OOP is a very large subject in which huge volumes have been written.

OK, enough of the warning and disclaimers.
What I hope this book does is give you enough of a taste of Flex and the ActionScript 3.0 environment that you will be able to solve the unique problems your own situations will require. I spend a great deal of time discussing how to find help by using the ActionScript 3.0 Language Reference.

I had to make a decision as to what server technology to show the dynamic side of Flex in. Since I use ColdFusion in my own work, I decided to use that technology. My wonderful technical editor, David Powers, is a world-leading authority on PHP, and he has written many books on the subject. He was kind enough to write an example of using PHP in Flex to show as an alternative to ColdFusion, and for that I thank him profusely.

I hope you will walk away from this book with the same sense of excitement that I have about Flex 3. I really encourage you to take the many examples in this book and experiment with them. Look upon this book as the beginning, not the end.

Let’s get started learning.

Layout conventions

To keep this book as clear and easy to follow as possible, the following text conventions are used throughout.

Important words or concepts are normally highlighted on the first appearance in **bold type**.

Code is presented in **fixed-width font**.

New or changed code is normally presented in **bold fixed-width font**.

Menu commands are written in the form Menu ➤ Submenu ➤ Submenu.

Where I want to draw your attention to something, I’ve highlighted it like this:

```
Ahem, don’t say I didn’t warn you.
```

Sometimes code won’t fit on a single line in a book. Where this happens, I use an arrow like this: ➤.

```
This is a very, very long section of code that should be written all ➤ on the same line without a break.
```
1 FLEX BASICS
Let's begin with a couple of assumptions:

Your knowledge of Internet design doesn’t go past HTML pages.

You haven’t got the foggiet idea what Flex is.

Using this paradigm, we can start right at the very beginning. In this chapter, we are going to look at where Flex fits into the evolution of the Internet. From there, we will examine what exactly Flex is and how it is different from traditional web technologies.

Finally, before you can roll up your sleeves and get to work, you need to install Flex and its related technologies. I will walk you through that process.

The Internet, then and now

Before you can appreciate the benefits of Flex, you need a general understanding of the history of the Internet up to this point. I say historical because the various technologies we see today came about at various points in the timeline of the Internet’s evolution. As I just stated in the short introduction to this chapter, it is important to understand this evolution in order to see where Flex fits.

HTML and dynamics

The earliest websites were just conveyors of text data. Frequently, they would have hyperlinks to other pages. Because of very slow Internet connection speeds (anyone remember 28K connection speeds?), graphics were kept to a minimum. You can still find some examples if you look, such as the contact page for the publisher of this book:

www.friendsofed.com/contact.html

Figure 1-1 shows this contact page.

This is the traditional HTML (Hypertext Markup Language) site. Notice that there are just a few simple graphics (earlier websites had even simpler graphics) with the rest being text and hyperlinks. Also notice the file extension of .html. This web page will never change unless someone physically goes in and changes it.

HTML pages, such as this one, are referred to as static or unchanging pages. Unchanging is perhaps an unfair word. More specifically, it only changes when someone goes into the XHTML code and makes changes manually.

A word is in order to demonstrate how static pages are called.
When you type www.friendsofed.com into your browser, the request is sent out over a series of routers on the World Wide Web until it arrives at the host web server. The web server searches its root for the requested HTML page, packages the HTML page up, stamps a return address on it, and sends it back to your browser. Your browser then reads the HTML code and displays the page as you see it here. A popular misconception, which I still hear in my training seminars, is that web pages are being viewed “over the Internet.” The web pages are downloaded to your computer and viewed in your computer. Once the web server sends you the HTML page, its job is completed. More specifically, we say that you are viewing the pages on the client machine. You, being the consumer or viewer of the web page, are the client.

Of course, this discussion is really simplified. A detailed discussion of building and distributing HTML pages is out of the focus of this book. There are any number of books that discuss these details. I recommend The Essential Guide to CSS and HTML Web Design by Craig Grannell (friends of ED, 2007).

Let’s evolve this to the next level.

Go to the following website:

www.adobe.com/cfusion/webforums/forum/index.cfm?forumid=60
This web address takes you to the Adobe Flex Support Forums, shown in Figure 1-2.

Figure 1-2. The Adobe Flex Support Forums

This is an example of what has been traditionally called a **dynamic website**. Let's discuss the mechanics here. They add a couple of additional steps to the preceding scenario for static pages.

Once you type the URL (web address) into your browser, it once again goes over a series of routers over the World Wide Web until it finds the Adobe web server. Here is where things change a bit.

Notice the letters `cfm` in the address. These letters tell the web server to send the request to another piece of software called an **application server**. There are five types of application servers that handle dynamic technology (actually, there are several more, but these are the most popular):

- **CFM**: ColdFusion
- **ASP**: Classic Microsoft Active Server Pages
- **ASPX**: Microsoft .NET Active Server Pages
- **JSP**: Java Server Pages
- **PHP**: A scripting language whose letters stand for nothing
All five of these technologies ostensibly do the same thing with various degrees of ease and complexity. They receive the request from the web server and then reach out to a database server using SQL code in the request.

When the database returns the requested information, the application server actually writes a brand-new XHTML page based on a template. The page contains the latest version of the data. From there, the application server returns the newly created XHTML page to the web server which, in turn, sends it back to your browser as before.

The only difference between the first and second examples is when the XHTML page is being written. In the first case, it was written by a developer and it doesn’t change until that developer, or someone else, makes the changes. In the second case, it is written on the fly and reflects the latest data in the database.

In both cases, every time new data is requested, the entire process has to start over again. Since this all happens in a fraction of a second, and works most of the time, it may not seem like much to you. However, in the background, this requires a tremendous amount of server time and tremendous use of resources on the various servers and your own client computer. All the graphics need to be downloaded separately and held in your computer’s memory, and all of the downloaded pages get stored in a folder in your computer.

Let’s move forward again. Go to the following Adobe website:


As you look at the site, the differences from the earlier two pages should be quite obvious. Notice that when you click the tabs, you move from page to page smoothly without the reloading process you saw in the previous examples. Also, in the Products tab, if you change the price range of the cell phones, you will see the cell phones animate while re-arranging themselves.
This is the prototype of a Flex site, and the mechanics will be, of course, the subject of this book. However, in its simplest form, all you really loaded was one file, a Flash SWF (pronounced “swif”) file. From there, when the information needs to change, what you refresh is what gets changed instead of the entire page. This means fewer potential errors, faster data display, and a much nicer user experience. Also, as you will see as you progress through this book, it will take fewer resources and be ideal for today's emerging portable Internet devices.

How exactly is this technology different?

**Flex and RIA**

In order for you to fully understand what is going on with the last example, you might need to change your thinking a bit.

As you saw in the first two examples, a traditional web page goes from page to page by sending another request back to the server and going through the process just discussed. In the case of a dynamic page, the web server takes the request and sends it out to one of the five application servers discussed, which in turn sends it out to the database server. The data is then assembled by the application server, and a new HTML page is written, sent back to the web server, and, finally, sent back to your web browser for display. If you go to
five different pages on a site, like Amazon.com, you end up going through that entire process five times. I think most would agree that, in retrospect, this isn’t a terribly efficient way of doing things.

What’s more, I think most people can easily distinguish between an Internet application, like the first two examples shown, and a desktop application like Microsoft Word. The whole look and feel is different.

Wouldn’t it be nice if the whole process ran much more efficiently? And wouldn’t it be even nicer if desktop and web applications had more or less the same look and feel?

Did the Flex prototype in the last example feel like an Internet application? Or did it feel closer to a desktop application?

To address these questions, Macromedia (now Adobe), with the introduction of Flash MX, introduced a new term: **rich Internet application (RIA)**. This Flash-based technology overcomes many of the limitations of traditional HTML in that it is nearly indistinguishable from a desktop application.

RIA applications, as you have seen in the last example shown, do not need to be rebuilt completely. Only the requested data is returned and plugged in where needed. As I stated in the last section, this means decreased demands on the server and much smaller file sizes (which lends itself nicely to emerging mobile Internet technologies).

Also, in a traditional HTML environment, user interactivity is limited to forms and just a few buttons. Desktop functions, like menus and smooth transitions from section to section, often perform poorly and could add significantly to file sizes. Also, while developers use JavaScript for this functionality, browser security programs often prevent JavaScript from functioning. As a result, even more functionality is often lost.

Flash MX addressed these issues by giving the web developer a whole new set of programming tools that allowed for greater interactivity without the issues that HTML/JavaScript presented. Suddenly, in an RIA environment, users could have the same interactive experience as in a desktop application environment. As a bonus, this additional interactivity could be added without dramatically increasing the file size.

The release of Flash MX also saw the arrival of the first Flash server: **Flash Remoting MX**. This new server gave RIA environments a greater ability to interact quickly and smoothly with data transfer technologies such as XML files and web services. In addition, it could interact with the popular Java and .NET environments. This meant that Flash could now work as a presentation tool over a variety of programming environments. Many developers started to find this as a welcome alternative to the less-than-ideal Java and .NET presentation containers.

Flash MX, however, presented a few new and unique problems.

After the release of Flash MX, Macromedia introduced ActionScript 2.0 as an update. ActionScript 1.0 had been a rather primitive procedural language to assist Flash in creating animations. To address the newer needs of RIA, ActionScript 2.0 was a semi–object-oriented programming (OOP) language.
While it followed some of the rules of OOP syntax, it also had to support the previous non-OOP ActionScript 1.0. The results were not always favorable, and many complained that the debugging tools were all but nonexistent.

Many developers also complained that to develop an RIA, they needed knowledge of many of the complexities of the Flash environment (timelines, scenes, and so on).

To address these many issues, Macromedia introduced Flex in 2004. This gave the developer a more traditional programming environment without many of the design complexities of Flash. It even had its own Dreamweaver-like development tool called Flex Builder. However, it never had the popularity hoped for due to the limitations of ActionScript 2.0.

It was clear a major overhaul was needed.

**Flex, Flex Builder, and ActionScript 3.0**

Flex 2 was introduced in the summer of 2006. It was not just an update from the original Flex, but a complete top-to-bottom overhaul. Central to the change was the introduction of ActionScript 3.0.

As you will be seeing as you progress through this book, ActionScript 3.0 is a full-fledged, open-source programming language similar to C++ and Java. While you may still associate ActionScript with Flash, its relationship is now only incidental. In other words, if you wanted to, you could build a complete application with just ActionScript alone without going near Flash.

If you were forced to describe what Flex is in just a couple of words, you could easily say that it is a presentation server. Chapter 2 examines this concept in detail. However, at the moment, you just need to know that in its simplest form, it sits over any of the application servers discussed earlier and takes the place of XHTML/JavaScript in presenting your data. Thus, rather than presenting your data as XHTML, it can be presented using the dynamic abilities of a Flash (SWF) file.

To accommodate this powerful new set of development tools, Adobe decided to not upgrade the Dreamweaver-like Flex Builder 1. Instead, Adobe turned to a development environment familiar to many programmers: **Eclipse**.

**Eclipse and Flex Builder 3**

Eclipse is a free programming development environment (we use the term IDE or integrated development environment) used extensively by many programmers, especially Java
developers. It allows a developer to work in multiple programming environments simultaneously. You can find Eclipse at

www.eclipse.org

While Eclipse is heavily used by Java developers, its real power is the ability to accommodate plug-ins for a variety of programming languages. For instance, there are plug-ins for C++, PHP, and even an increasingly popular free plug-in for doing ColdFusion development. We will be using that plug-in a little later on in this book when we integrate Flex with ColdFusion.

While many of the plug-ins for Eclipse are free, Flex Builder 2 was not. However, Flex Builder 2 allowed developers to work in a traditional IDE with many of its powerful programming and debugging tools.

Flex Builder 3 presents even more powerful tools that, as you will see throughout this book, allow the developer to harness the power of Adobe’s other powerful design and development tools. Among those improvements are

- An enhanced Design View to take advantage of the powerful Adobe CS3 design tools. This can improve workflow between designers and developers.
- Easier ways to connect with data sources and servers with new and enhanced data components.
- The ability to construct and deploy the new Adobe Integrated Runtime (AIR) tools within the Flex Builder IDE.

As of this writing, Adobe announced that it was making Flex fully open source. This means that other developers could create competing IDEs for Flex development. For the purposes of this book, however, we will be using Flex Builder 3.

We will examine Flex Builder 3 with greater detail in Chapter 2. For now, however, we have to start installing this technology before we can use it. If you are ready to get it installed, let’s move on to the next section.

**Installing Flex Builder 3**

Flex Builder consists of three separate components:

- **The Flex Software Development Kit**: This is the collection of ActionScript classes (we will be discussing class files in Chapter 3) necessary to build, run, and deploy Flex applications.

- **The Eclipse plug-in integrated development environment**: This plug-in assists in building the applications.

- **Flash Player 9**: Flex applications will only run with Flash Player 9 or later.
Flex can be installed in one of two ways:

If you are an existing Eclipse user, you can install the plug-in version. As you are installing, you will be prompted to enter the location of Eclipse, and the installer will know what to do from there.

If you are not an existing Eclipse user, you can install the stand-alone version. This is Flex Builder and Eclipse packaged together.

Both versions will get you to the same place in the end. However, there is a slight difference. Eclipse uses a technique called Perspectives. We will be looking at this a bit more in Chapter 2. However, for now just know that a Perspective is an arrangement of tools and windows needed to develop in a particular programming language. The Perspective for Java programming would be different from that for C++, and Flex would require yet a different Perspective. If you install Eclipse from its site, the default Perspective is for Java development. However, if you install the stand-alone version of Flex Builder, the default Perspective is Flex. Also, I found that other plug-ins, such as the ones for ColdFusion (www.cfeclipse.org), were easier to install and use through the plug-in version of Flex Builder 3. For that reason, I strongly recommend that you use the plug-in version of Flex Builder 3.

In this section of the book, I will be showing you how to install the plug-in version. Once we get past a certain point, the installation will be exactly the same. For that reason, I divide the installation into two sections.

The following install instructions are valid as of the writing of this book. Some of the steps and screens shown could change as Adobe makes adjustments.

As of the writing of this book, there is an emerging bug regarding installation of the Flash Player ActiveX control: any existing players are not being fully uninstalled. This may be fixed by the time you read this section. However, to play it safe, it would be a good idea to go to

http://kb.adobe.com/selfservice/viewContent.do?externalId=tn_19254

and download the Flash Player uninstall tool to completely remove any instances of Flash Player you have installed. Flex will reinstall them properly.

Installing Flex Builder as an Eclipse plug-in

Before you can install the Flex Builder plug-in, you must first install Eclipse, so let’s start there:
1. Go to www.eclipse.org and click the Download Eclipse button. You will be presented with a screen similar to Figure 1-4.

![Eclipse download screen](image)

Figure 1-4. The Eclipse download screen

2. As you can see, the free Eclipse IDE is available for a variety of platforms. Click the link for the version you want.

3. Once it is fully downloaded, unzip the file to the directory of your choice. Because Eclipse is platform independent, there is no traditional install process.

4. If you installed in Windows, you will want to go into File Explorer, navigate to the folder that you installed Eclipse in, and right-click the EXE file associated with Eclipse. Select Send to ➤ Desktop as shown in Figure 1-5.
That is all that is involved with installing Eclipse. From here on in, the install process is very similar no matter which version of the program you are installing.

**Installing Flex Builder 3**

As of this writing, Adobe will sell Flex Builder 3 either on disk or as a download. The download is about 345MB in size. Within those two choices, you can install it (as I have stated several times) either as a stand-alone version or as an Eclipse plug-in. The differences for installing the versions are very minor.

1. Depending on the operating system you are installing for, start the install process. A program called InstallAnywhere should start. It may take a couple of minutes before the first screen appears.

2. The first screen will prompt you as to the language you are installing in. Choose the language and click OK.
3. It is a good idea to close all running programs and windows, especially browsers. This is because Flex Builder will install its own version of Flash Player 9. The next screen will prompt you for that. Once done, click Next.
4. This next screen, shown in Figure 1-8, is the licensing screen. Just accept the license agreement and click Next.

![Figure 1-8. The License Agreement screen](image)

5. The next screen, shown in Figure 1-9, prompts you for the default location. Unless you have a good reason to change it, just accept the location by clicking Next.

![Figure 1-9. The default install location on the Choose Install Folder screen](image)
6. The next screen, shown in Figure 1-10, will appear only if you are installing the plug-in version of Flex Builder 3. It will ask you for the location of the Eclipse installation (if you installed it earlier). You need to select Choose and navigate to the folder (in this example, my install directory was C:\Eclipse). Click Next when selected.

![Figure 1-10. Selecting the location of Eclipse](image)

7. The following screen, shown in Figure 1-11, is quite important. It prompts to install Flash Player in each of the installed browsers on your computer. This version of Flash Player, however, is not the one most end users download. This player has the ability to debug the SWF files you create in Flex, and it will play an important role as you progress through this book; you will have your first look at it in Chapter 2.

Earlier in the chapter, I mentioned an emerging bug with installing Flash Player 9. I strongly recommend reviewing that and uninstalling any existing versions of Flash Player. The installation shown here will reinstall everything properly. Once you do that, reinstall Flash Player for all browsers you have installed.

You are also prompted about installing additional Eclipse plug-ins if you want to do additional ColdFusion and JavaScript programming later on. Even if you don’t have ColdFusion installed now, I strongly recommend that you select both of these options, if you plan on using either of these technologies later on.
8. The final screen before installation allows you to review the installation parameters of the program folder and the Debugging Flash Player (see Figure 1-12).

9. Assuming all is well, go ahead and click the Install button.
You will see a progress screen similar to Figure 1-13.

Figure 1-13. Install progress screen

10. A final screen should appear as shown in Figure 1-14, letting you know all installed properly. Click Done.

Figure 1-14. The final screen showing the installation was successful