Chapter 1

A Brief Tour of Cocoa Development

In This Chapter

- Programming for Mac OS X
- Discovering the Cocoa development process
- Exploring the tools for programming Cocoa applications

These are exciting times for Macintosh users. When Apple unleashed Mac OS X upon the world, it ushered in a new era of computing for the Mac faithful. Besides the rock-solid stability of UNIX, Mac OS X offered functionality and features that Mac users could have only dreamt of a few years earlier. Along with this great operating system, Apple saw fit to remember Macintosh developers and have done so ever since. Principal among Apple's achievements is Cocoa, the subject of this book. This chapter introduces you to the world of Mac OS X programming and, in particular, Cocoa programming.

Mac OS X Is a Programmer's Dream

Macintosh programming has never been as easy or as accessible as it is with Mac OS X. For starters, Apple, the friendly folks that they are, thought it'd be a great idea to give away the development tools. For free. Apple provides the Xcode Developer Tools as a free download on the Apple Developer Connection Web site. By installing the Xcode Developer Tools download, you instantly gain access to a complete collection of tools, utilities, documentations, and example source codes to get you started programming for the Mac OS. In the past, a developer bundle this comprehensive would have cost hundreds of dollars. Today, Apple provides it for no additional charge.

Some older versions of Mac OS X ship with a Developer Tools disc; newer versions don't. If you're searching for the disc and can't find it, you may have an installation of Mac OS X that doesn't include the Developer Tools disc. You needn't worry, however, because you can download the Xcode Developer Tools by signing up for a free ADC membership at Apple's developer site (`https://connect.apple.com`). In fact, even if you already have a Developer Tools disc, check Apple's developer site for updates because each

version of Xcode Developer Tools is specific to a particular OS X release. ***Note:*** Xcode Developer Tools installations can total in the hundreds of megabytes, so you'll probably want a fast Internet connection to download them.

Apple Developer Connection (ADC) is Apple's support program for developers. You can register at different tiers (and pay different prices) for membership, which gives you varying amounts of support and other perks, such as Worldwide Developers Conference (WWDC) tickets. The lowest tier is completely free, so it doesn't cost you anything to download the Xcode Developer Tools.

Just because the Xcode Developer Tools is a free download doesn't mean that the software is second-rate. On the contrary, Xcode Developer Tools are world class. When developing software for the Macintosh with these tools, you can take advantage of the following benefits:

- Write code in a variety of programming and scripting languages: C, Objective-C, Python, Ruby, Java, or even AppleScript.
- Create beautiful interfaces that follow Apple's Human Interface Guidelines.
- Develop applications with rich features, some of which you can add to your project without writing a single line of code.

Further, because Mac OS X has a UNIX flavor at its core, you can take advantage of the decades of work by UNIX users. For example, most open-source software run on different varieties of UNIX, so you can leverage thousands of compatible source-code examples for use in your own Mac OS X applications as well.

Why Program with Cocoa?

Cocoa is one kind of programming that you can perform with the Apple Xcode Developer Tools. Cocoa is a collection of tools and libraries (or *frameworks*) that allows you to get the most out of Mac OS X programming. Many features make Cocoa great; some include

- Modular object-oriented design
- Use of frameworks
- Visual interface design

Object-oriented programming is in common use these days, and for good reason. By programming with an object-oriented design, your code can more closely model items in the real world. This book isn't an object-oriented text; in fact, you should come to Cocoa with at least an idea of how to program in an object-oriented fashion. This book does, however, discuss the object-oriented nature of Cocoa and examines its primary language: Objective-C. Objective-C, as you might induce from its name, is an object-oriented superset of the C language. It permits you to program in an object-oriented fashion without some of the messy baggage that C++ has. Because Objective-C is a superset of C, you can also take advantage of the C that you know. Everything that you can do in C is valid code to the Objective-C compiler.

The use of frameworks is another great aspect of Cocoa development. Experienced programmers may be tempted to call frameworks by another name — libraries. *Frameworks* are collections of classes that provide you, the Cocoa developer, with a specific type of functionality. Mac OS X ships with several frameworks for you to choose from, but two big ones stand out: AppKit and Foundation. The *AppKit Framework* provides you with scores of classes and functions for working with interfaces, and the *Foundation Framework* gives you utilitarian functions relating to data manipulation and program execution. You use them a lot when writing Cocoa software.

The object-oriented nature of Cocoa and its rich set of frameworks form an unbeatable code-reuse duo. Computer programmers can be a lazy bunch, not wanting to repeat a single task. To aid developers in their pursuit of reusable code, Cocoa offers a wide array of reusable classes. After you complete some programming tasks, you can even store the results in your own framework for use in other projects. Apple gives you reusable code out of the box, and you can reuse your own code as well. The object-oriented design of Cocoa makes this reuse possible.

Reusable code is good for a variety of reasons: It lets you create software quickly, it reduces the number of bugs in your code, and it prevents you from reinventing the wheel each time you sit down to program. By reusing the frameworks that Apple provides with the Xcode Developer Tools, you gain all sorts of great functionality without having to know how it works under the hood.

Besides the geekier benefits, you'll love many other aspects of Cocoa programming. For starters, the frameworks that accompany Mac OS X provide a rich set of interface elements that you can use to build sophisticated interfaces demanded by professional software. Moreover, Cocoa programming gives you instant access to a wide range of free classes. Whether you need an About box, a spell-checker, or QuickTime movies in your application, Cocoa has a solution for you.

The Tools You Need

To facilitate your Cocoa development, Apple was nice enough to provide you with a large selection of tools and utilities. With these tools, you can begin creating Cocoa software from the ground up. When you're finished programming, the tools will even build the application, prepare it for distribution, and put together an installer.

To begin programming with Cocoa, find the development tools. If you installed them in the default location, they reside in the following directory on your hard drive:

```
/Developer/Applications
```

If you discover that you don't have the development tools on your system, visit `developer.apple.com` to download the latest version.

You won't need all the applications that Apple provides in the `/Developer/Applications` directory. In fact, for many tasks you can probably get away with using only two: Xcode and Interface Builder.

Xcode

Xcode is the main application that you'll use for all your Cocoa projects. Xcode serves a number of roles in the Cocoa development process:

- **Xcode acts as the central repository for all the files in your Cocoa projects.** Using a familiar document approach, Xcode lets you organize the components of a Cocoa project in one easy-to-use document. Figure 1-1 shows a Cocoa project opened in Xcode.
- **You also use Xcode to write and edit Cocoa source code.** When you write code for a project, Xcode guides you by coloring the syntax, indenting code automatically, and providing auto-completion features to reduce the amount of typing (and remembering) that you have to do. It also offers convenient one-click access to all the functions in your code, as shown in Figure 1-2.
- **Your Cocoa project may have other types of files beyond code, and Xcode is prepared to help you work with them.** For example, if you want to include images in your project, Xcode lets you view them in the main project window without skipping a beat. You don't need to use another application to view those images. Xcode displays them right in the code editor, as shown in Figure 1-3.

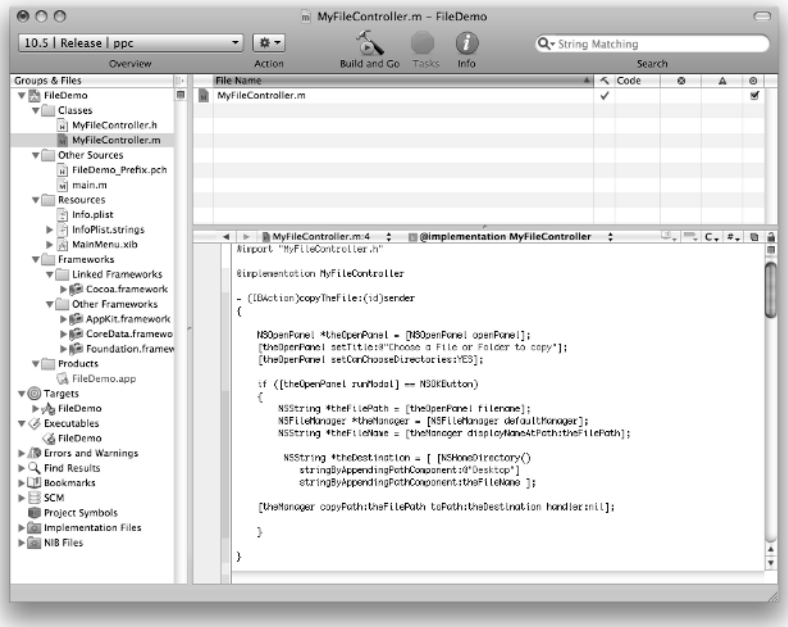

Figure 1-1: Xcode acts as your primary tool for writing Cocoa software.

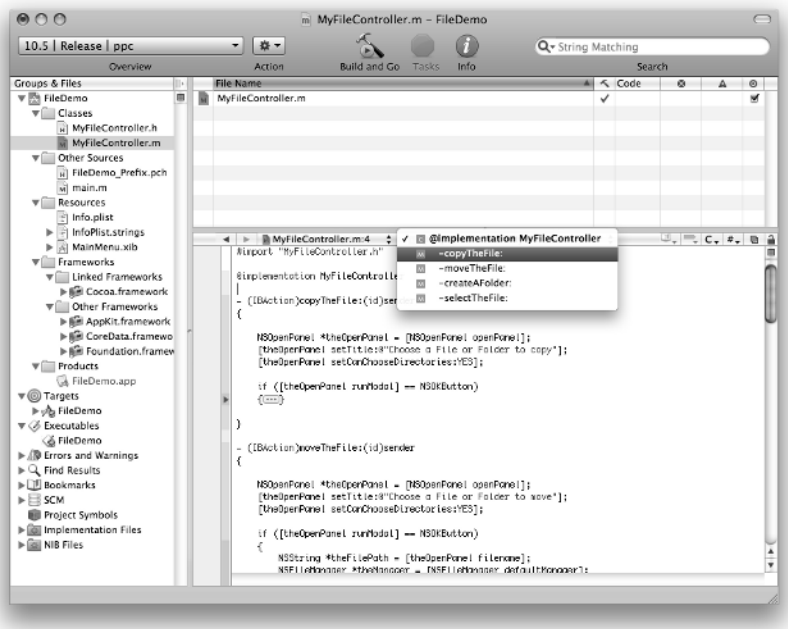

Figure 1-2: Edit your code in Xcode.

Figure 1-3: You can view other types of files in Xcode.

- **When you get stuck, Xcode gives you access to the complete collection of Cocoa, Xcode, and other developer documentation.** You can view and navigate the documentation with Xcode in much the same way as you would a Web browser. Figure 1-4 shows what the screen looks like when documentation is loaded into Xcode.

After you complete your Cocoa project, you use Xcode to compile, link, and build a final application. You can then distribute the application to friends, co-workers, and even the world (as long as they use Mac OS X).

Xcode wears many hats. If you're accustomed to other development environments, you may be surprised to discover that Xcode performs tasks that require multiple tools in other environments. For example, Xcode functions as a

- **Project organizer,** managing files and resources in your Cocoa projects
- **Code editor,** allowing you to write and edit Cocoa code
- **Browser,** displaying built-in documentation or other kinds of resources in your Cocoa projects
- **Compiler and linker,** spitting out a complete Cocoa application at the end of the development process

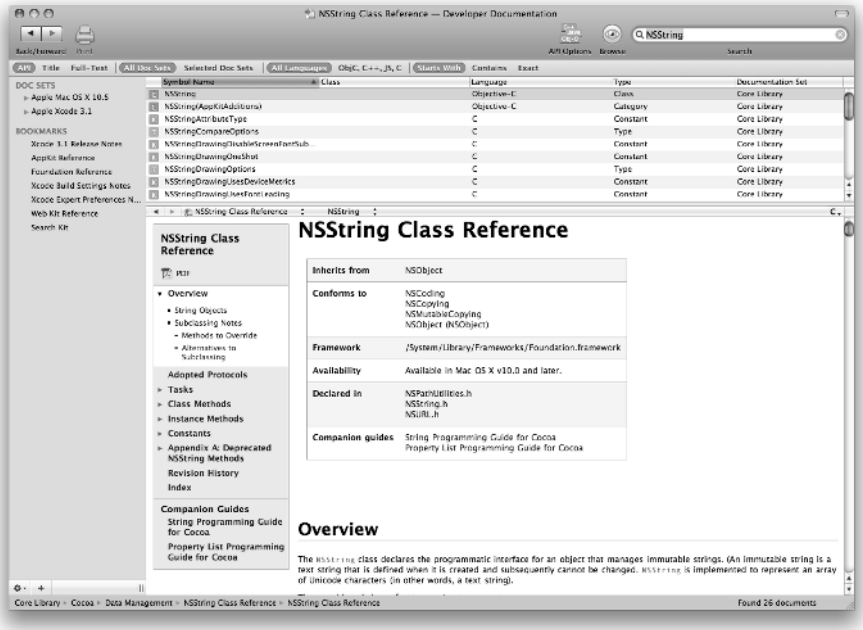

Figure 1-4: You can view the Cocoa documentation in Xcode.

Interface Builder

Interface Builder is a constant companion to Xcode. As you can probably guess, Interface Builder's main purpose is to create interfaces. With it, you can build interfaces that adhere to Apple's interface guidelines.

Interface Builder provides a complete set of controls that you can add to your application. From windows and drawers to buttons and sliders, Interface Builder gives you drag-and-drop access to a full suite of interface elements to make your software the best it can be. Don't forget that Interface Builder is an Apple product. No one knows the Macintosh user interface better than Apple, because they created it, so you can be certain that the controls in Interface Builder follow the strictest Apple guidelines.

Figure 1-5 shows an example interface with many different types of controls available to you in Interface Builder. The interface won't win any design awards, but it does show you the range of elements that you can use in your own Cocoa software.

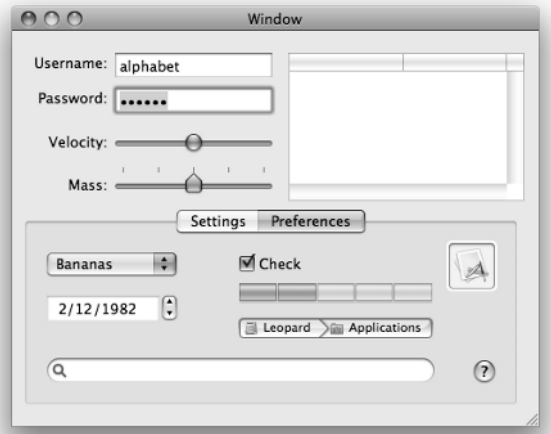

Figure 1-5: Interface Builder has a complete set of interface controls for you to use in your Cocoa projects.

Interface Builder's features aren't limited to WYSIWYG (what you see is what you get) interface editing. You can also create classes that have no visual representation. Although you don't actually write the code in Interface Builder for your classes, you do define the basic structures and methods for them there. You can also connect the interface to your classes with simple drag-and-drop techniques, as shown in Figure 1-6.

Do you speak the language?

Cocoa programming (like most kinds of computer programming) requires the use of a programming language. To create Cocoa applications, you need to know Objective-C, Python, Ruby, Java, or AppleScript. This book uses Objective-C because it's the "native language" of Cocoa. Objective-C is a superset of the traditional C programming language. If you have experience with C, you're well on your way to understanding Objective-C. All the C functions you know and love are available to you in Cocoa. Objective-C, however, goes one step further and enhances C by adding object-oriented features to the language.

Objective-C has a syntax that may look a little foreign to you at first, unless you're also familiar with SmallTalk. But after you get the hang of it, you'll find that it isn't hard to understand at all. Chapter 6 goes into the details of Objective-C, but you start using it in Chapter 2 to build your first Cocoa project.

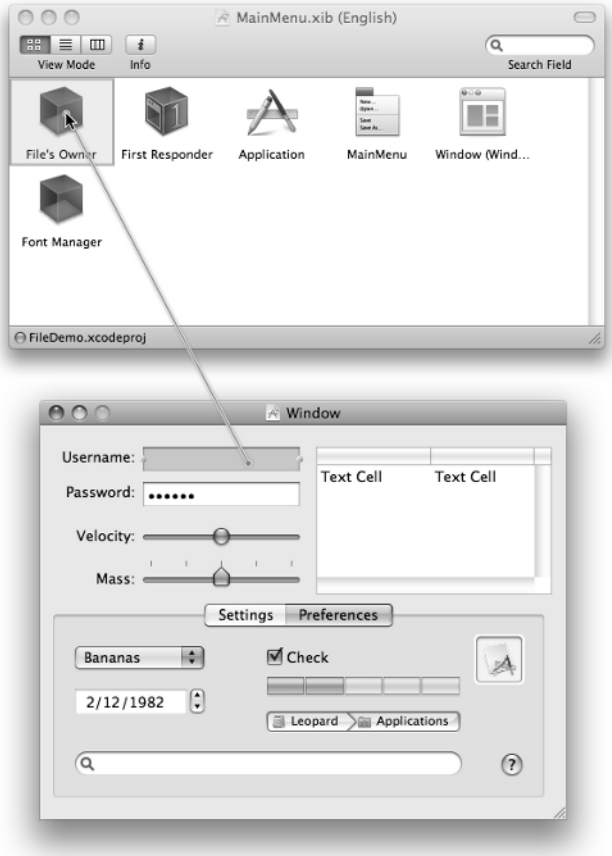

Figure 1-6: Connecting an interface element to a class is a simple drag-and-drop operation.

After you complete an interface, Interface Builder goes the extra mile and creates the header and implementation files for you and then inserts them into the desired Xcode project. Although Interface Builder's strongest features pertain to designing and creating great-looking interfaces, many other features make it much more than an interface-building tool. It plays a big part in the Cocoa programming experience.