CHAPTER 24

Pages and Navigation

Most traditional Windows applications are arranged around a window that contains toolbars and menus. The toolbars and menus *drive* the application—as the user clicks them, actions happen, and other windows appear. In document-based applications, several equally important “main” windows may be open at once, but the overall model is the same. The users spend most of their time in one place, and jump to separate windows when necessary.

Windows applications are so common that it’s sometimes hard to imagine different ways to design an application. However, the Web uses a dramatically different page-based navigation model, and desktop developers have realized that it’s a surprisingly good choice for designing certain types of applications. In a bid to give desktop developers the ability to build weblike desktop applications, WPF includes its own page-based navigation system. As you’ll see in this chapter, it’s a remarkably flexible model.

Currently, the page-based model is most commonly used for simple, lightweight applications (or small feature subsets in a more complex window-based application). However, page-based applications are a good choice if you want to streamline application deployment. That’s because WPF allows you to create a page-based application that runs directly inside Internet Explorer or Firefox. This means that users can run your application without an explicit installation step—they simply point their browsers to the correct location. You’ll learn about this model, called XBAP, in this chapter.

Finally, this chapter wraps up with a look at WPF’s WebBrowser control, which lets you host HTML web pages in a WPF window. As you’ll see, the WebBrowser not only shows web pages, but also allows you to programmatically explore their structure and content (using the HTML DOM). It even allows your application to interact with JavaScript code.

Page-Based Navigation

The average web application looks quite a bit different from traditional rich client software. The users of a website spend their time navigating from one page to another. Unless they’re unlucky enough to face pop-up advertising, there’s never more than one page visible at a time. When completing a task (such as placing an order or performing a complicated search), the user traverses these pages in a linear sequence from start to finish.

HTML doesn’t support the sophisticated windowing capabilities of desktop operating systems, so the best web developers rely on good design and clear, straightforward interfaces. As web design has become increasingly more sophisticated, Windows developers have also begun to see the advantages of this approach. Most important, the web model is simple and streamlined. For that reason, novice users often find websites easier to use than Windows applications, even though Windows applications are obviously much more capable.
In recent years, developers have begun mimicking some of the conventions of the Web in desktop applications. Financial software such as Microsoft Money is a prime example of a weblike interface that leads users through set tasks. However, creating these applications is often more complicated than designing a traditional window-based application, because developers need to re-create basic browser features such as navigation.

**Note** In some cases, developers have built weblike applications by using the Internet Explorer browser engine. This is the approach that Microsoft Money takes, but it’s one that would be more difficult for non-Microsoft developers. Although Microsoft provides hooks into Internet Explorer, such as the WebBrowser control, building a complete application around these features is far from easy. It also risks sacrificing the best capabilities of ordinary Windows applications.

In WPF, there’s no longer any reason to compromise, because WPF includes a built-in page model that incorporates navigation. Best of all, this model can be used to create a variety of page-based applications, applications that use some page-based features (for example, in a wizard or help system), or applications that are hosted directly in the browser.

### Page-Based Interfaces

To create a page-based application in WPF, you need to stop using the Window class as your top-level container for user interfaces. Instead, it’s time to switch to the System.Windows.Controls.Page class.

The model for creating pages in WPF is much the same as the model for creating windows. Although you could create page objects with just code, you’ll usually create a XAML file and a code-behind file for each page. When you compile that application, the compiler creates a derived page class that combines your code with a bit of automatically generated glue (such as the fields that refer to each named element on your page). This is the same process that you learned about when you considered compilation with a window-based application in Chapter 2.

**Note** You can add a page to any WPF project. Just choose Project ä Add Page in Visual Studio.

Although pages are the top-level user interface ingredient when you’re designing your application, they aren’t the top-level container when you *run* your application. Instead, your pages are hosted in another container. This is the secret to WPF’s flexibility with page-based applications, because you can use one of several containers:

- The NavigationWindow, which is a slightly tweaked version of the Window class
- A Frame that’s inside another window
- A Frame that’s inside another page
- A Frame that’s hosted directly in Internet Explorer or Firefox

You’ll consider all of these hosts in this chapter.