Chapter 1

Getting to Know the Ribbon

In This Chapter

- Getting the overview of the Office Ribbon
- Understanding the RibbonX elements
- Using the Ribbon in Office 2007

Microsoft has made significant changes to Office 2007. The most noticeable change is the new Ribbon, which appears across the top of applications in place of the menu-and-toolbar interface of old. Unfortunately, in creating this new interface, Microsoft also decided against backward compatibility. All of those well-ordered toolbars and menus you created in the past now appear on a single Ribbon tab, Add-Ins. Yes, your code will still work — but users will find it significantly more difficult to use your applications.

If you have a relatively simple application, using the Add-Ins tab might not result in a loss of productivity. Most applications, however, don't translate well to the Add-Ins tab. That's because they rely on the context of the old menu-and-toolbar system — which means you probably end up with a mess instead of the nice interface you used in the past. Microsoft hasn't provided a clear and easy method to overcome the mess they created, which is (presumably) why you're reading this book.

This chapter introduces you to the Ribbon. The new interface really does have an appeal from an ease-of-use perspective. The chapter also examines the few aids that Microsoft has provided to help you overcome the problems with the new interface and tells you how to use them.

You'll discover that the Ribbon does have a few redeeming features for the developer as you examine the Ribbon elements. Theoretically, once you transition your application to the Ribbon, users can become more productive because the Ribbon hides complexity and makes application features more visible. By using the new screen elements carefully, you can create amazing new interfaces. Not only do you have access to new controls, but you also use new grouping features to use those controls with greater success.

Finally, this chapter shows you how the Ribbon appears in the target applications for this book. Knowing how Microsoft has put the Ribbon together in the various applications can help you create better application elements of your own.

This book uses the term Ribbon to refer to the physical presentation of application elements such as tabs, groups, and controls. The Ribbon is the part of the application interface that you can see. A similar term, RibbonX, refers to the programming interface you use to create the Ribbon elements for your application. You define how the Ribbon appears in the application by using the RibbonX programming interface.

Understanding the Office Ribbon

The Ribbon is the new interface for Office. Microsoft doggedly pursues most of its innovations — so it's quite likely that you'll see the new Ribbon in most, or all, Microsoft applications of the future. Because of the change in interface, the options available to Office users who have a substantial investment in custom templates consist of the following:

- Use the Add-Ins tab to access custom features in existing templates, which isn't a viable option for custom templates of any complexity.
- Continue to use an older version of Office, which means that you won't have updates at some point to protect against real-world dangers such as viruses.
- Convert existing applications to use the Ribbon, which means writing the interface code from scratch.
- Scrap existing applications as they become outdated — which means you'll eventually incur substantial development costs, but have a completely updated application.

None of these options would be optimal in every situation. Yes, you can use the Add-Ins tab for less complex applications — but the moment you have more than one or two menu or toolbar entries, the Add-Ins tab quickly becomes unviable. Even if you do use the Add-Ins tab, users will require some level of retraining because the options won't appear in the same locations they occupied in the past.

It's tempting to think that you can simply ignore the Ribbon completely by holding on to your current version of Office. Certainly, some companies are still using Office 97 without a pressing need to update to obtain new features. However, the risks of this approach are many. The biggest risk is that a new virus will appear, and because Microsoft won't provide updates for your old

copy of Office, you can easily get it and lose all of your data. Keeping the old version of Office is really only viable if your office is completely closed to the outside world — especially the Internet — and few offices are in that position any longer.

The paragraphs that follow consider the last two options in a bit more detail. It's important to become familiar with the Ribbon, decide whether it meets specific needs in your organization, look at the tools that Microsoft provides to update your templates, and then perform all the required manual conversion. Of course, you may simply decide that the features the Ribbon provides are too significant to ignore, and create a completely new version of your application. The "new application" option is probably going to be a little challenging for many organizations, though, considering the amount of custom code they already have in place.

The techniques you discover in this book may help you in more than one way in the future. You may eventually find the Ribbon in other applications. Microsoft has decided to license the Ribbon interface to third parties free of charge. Obviously, Microsoft wants to entrench the new Ribbon interface into a broad range of applications as quickly as possible. Making RibbonX available to third parties is an efficient way to perform the task. You can discover more about third-party licensing at

`http://msdn2.microsoft.com/en-us/office/aa973809.aspx`

Considering Office support for the Ribbon

The first question anyone with a perfectly running Office application will ask is why Microsoft decided to change the interface completely. The old system of menus and toolbars had simply become too cumbersome for most users; some organizations have found that users have a hard time locating a particular command or feature. Even though the menu-and-toolbar system is straightforward, commands often appear several layers deep, or on a toolbar that the user doesn't have displayed. Microsoft designed the Ribbon to overcome these problems by making the interface simpler and options easier to find. For example, you can choose programmatically to display tabs when the user needs them, and then hide them when the user has finished performing a task. (Whether the new approach actually works remains to be seen.)

Figure 1-1 shows the old menu-and-toolbar approach used with Excel. Figure 1-2 shows the same view, using the new Ribbon. As you can see, the new Ribbon does tend to make features easier to find by grouping them together and reducing the view to just the current task. By organizing the application features and focusing user attention, Microsoft contends that it's possible to obtain a significant improvement in user performance of common tasks.

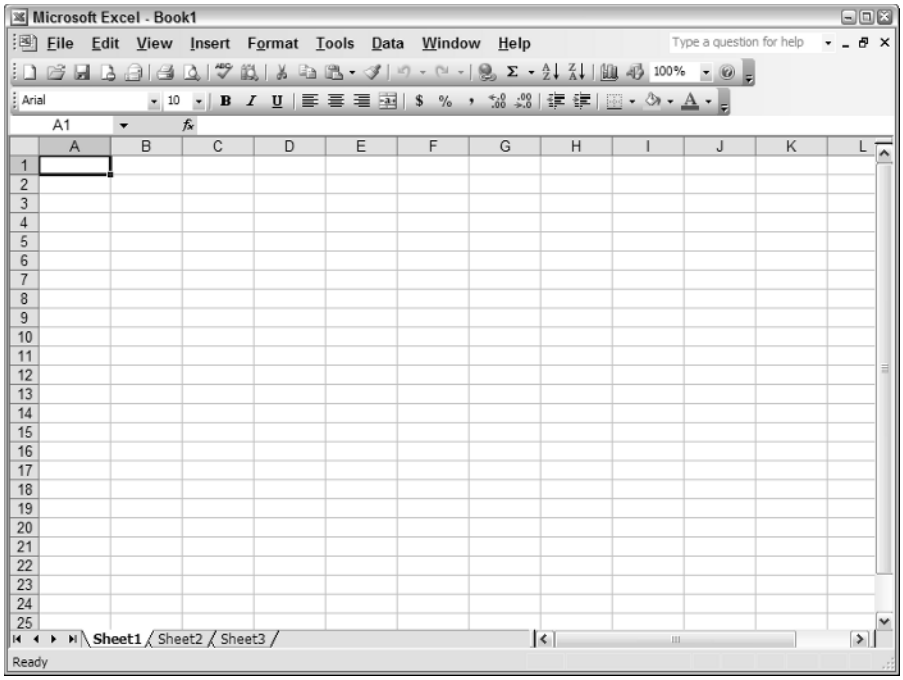

Figure 1-1: The old menu and toolbar interface looks cluttered.

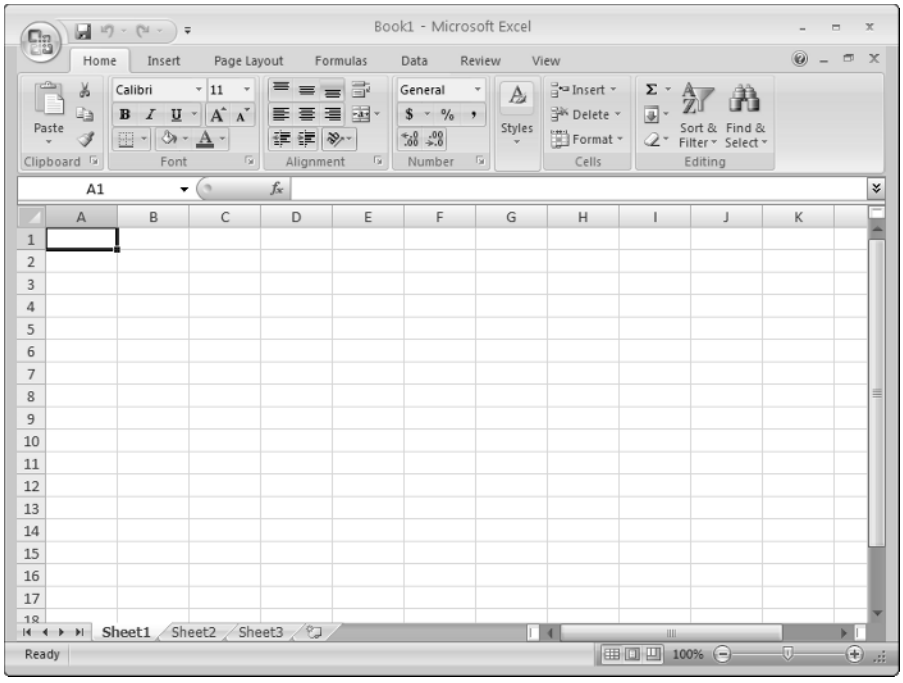

Figure 1-2: The new Ribbon interface groups like items together.

When you decide to update your application, you'll want to consider how you can also use these new features to your advantage. Chapter 2 discusses this topic in detail. However, for now, you'll want to think about the advantages that grouping and the new controls provide. Instead of placing certain features on a toolbar, you can place them on a tab. The tab need not be available at all times; you can make it visible only when the user's task context requires. In short, the developer also receives a number of benefits by using the Ribbon.

You might notice something else in Figure 1-2: The Ribbon seems to take up a lot of space. In fact, during beta testing, many people complained that the new interface requires too much space — so Microsoft makes it possible to hide the Ribbon when you're not using it. Simply right-click the menu bar and choose Minimize the Ribbon from the context menu. Consequently, any concerns that the Ribbon consumes more space than the old toolbar and menu interface are unfounded. In fact, the Ribbon can actually provide you with more screen real estate for editing documents.

However, the biggest plus of the new Ribbon is the ability to hide things in such a way that the user knows that they exist, but can easily ignore them. For example, many groups contain a special button in the lower-right corner that lets you display a dialog box with detailed settings, as shown in Figure 1-3. Simply click the button to display the associated dialog box.

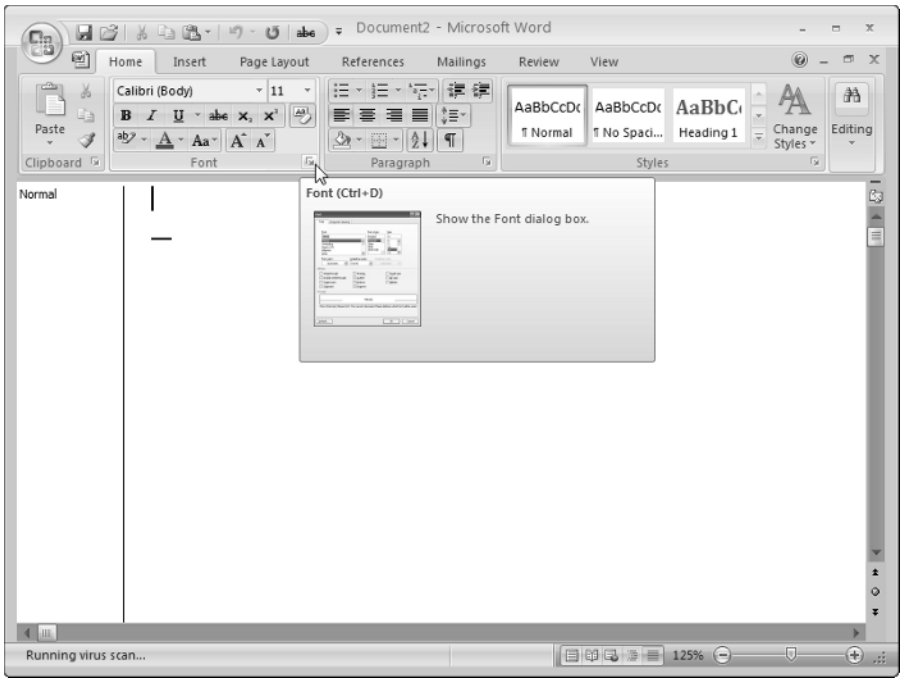

Figure 1-3: Use the Ribbon to hide special features in plain sight.

The user knows that the Font dialog box exists, but can choose to ignore it when the controls provided on the Ribbon suffice. Using this approach means you can monitor users of your application to determine which features they use most often, and then you can place those features on the Ribbon. This approach addresses the needs of normal users. Power users can display a dialog box that includes the full set of application features, so they won't have to give up flexibility to make the interface easier for novice users to use.

Understanding support for old toolbars and menus

You might be under the impression that the new Ribbon interface is an all-or-nothing proposition. In fact, there's a migration path of sorts; you can exploit that path in a number of ways. (For example, the "Performing Simple Interface Changes and Storing Them" section of Chapter 12 tells how you can make the transition easier by highlighting custom styles and adding some features to the user's Quick Access Toolbar.) The following sections describe the tools that Microsoft offers to make the transition easier. The Office Compatibility Pack lets you continue using Office 2003 even as you move to Office 2007 documents, while the Office Migration Planning Manager reduces the work required to plan an upgrade path.

Working with the Office Compatibility Pack

One capability that will undoubtedly make your life interesting is that you can provide backward compatibility for users of previous versions of Office as you move to Office 2007. The Office Compatibility Pack installs support for the new Office 2007 file formats for Office XP and Office 2003 users. Although users of these older versions of Office still can't use the new RibbonX applications, they can at least interact with the data found in the documents. In fact, you can create document templates in a way that provides support for *both* older and newer users, using parallel code. (The "Designing Parallel Version Solutions" section of Chapter 13 addresses this technique.)

Make certain you install all current updates for Office XP or Office 2003 before you install the Office Compatibility Pack. Otherwise, the installation will fail and you might find it difficult at best to obtain the desired results. Use the Office Update link at

```
http://office.microsoft.com/en-us/default.aspx
```

to locate and install the required updates.

After you download and install all the required Office updates, you can download the Office Compatibility Pack from

`http://www.microsoft.com/downloads/details.aspx?familyid=941b3470-3ae9-4aee-8f43-c6bb74cd1466`

Install the Office Compatibility Pack as you would any other program. Figure 1-4 shows the changes to Word. Notice that you can now open Office 2007 documents.

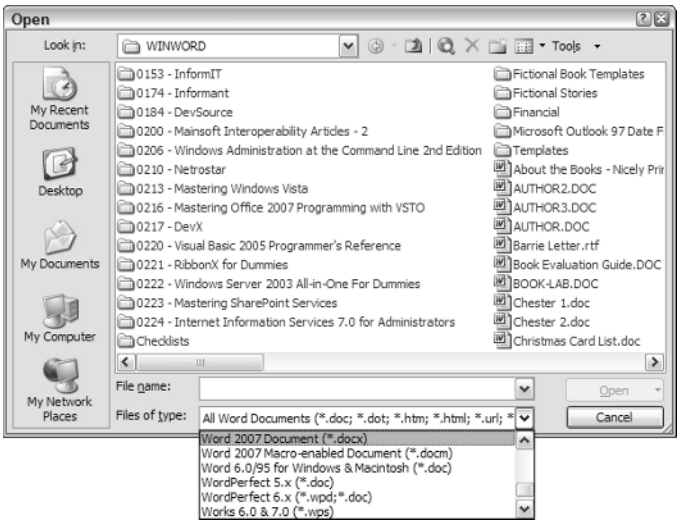

Figure 1-4: Open Office 2007 documents using the familiar Office XP/2003 interfaces.

It's important to remember that Office 2007 makes a distinction between standard documents and those that contain macros. For example, when working with Word 2003, you'd normally use the `DOC` file extension. However, when working with Word 2007, you'll always save documents with macros using the `DOCM` file extension. Likewise, if the file contains no macros, save it using the `DOCX` file extension.

Working with the Office Migration Planning Manager

It's important to plan the migration to Office 2007. No one is going to move a complex set of applications to Office 2007 without performing some type of review process and understanding precisely what the move requires. The

Office Migration Planning Manager (OMPM) performs an audit of your system and helps you plan the migration to Office 2007 with greater ease. You can download this tool from

```
http://www.microsoft.com/downloads/details.aspx?FamilyID=
13580cd7-a8bc-40ef-8281-dd2c325a5a81
```

The product actually consists of two downloads. The first download contains release information that tells you about the changes in the OMPM, which is in version 2 as of this writing. For example, if you don't read the release notes, you won't know that the OMPM doesn't work on 64-bit systems. The second download is the actual Office Migration Planning Manager utility.

You should also review the documentation at

```
http://technet2.microsoft.com/Office/en-us/library/
d8f318d4-84ea-4d3e-8918-ea8dacd14f7e1033.mspx
```

This documentation explains the inner workings of the Office Migration Planning Manager and tells how you can best use it to meet specific migration needs. You begin by extracting the files to a specific location on your hard drive, such as `\OMPM`. The rest of this discussion assumes you extracted the files to the `C:\OMPM` folder of your hard drive, but you can use any other folder you choose. (Choosing the `C:\OMPM` folder has the advantage of reducing the number of configuration choices you have to make.)

If you don't really want to work with OMPM now, you can probably skip the rest of this section. The OMPM tool uses a command line interface. You'll begin by configuring a special file to tell the program how to run. The listing of `Offscan.ini` entries appears at

```
http://technet2.microsoft.com/Office/en-us/library/
1850987f-87bb-47e9-b370-f4b8af3c39d71033.mspx
```

You'll find this file in the `C:\OMPM\scan` folder. Of all of the setting changes you can make, the most important configuration change (the one you must make) is the `[FoldersToScan]` entry. Beneath this heading, you include the `Folder=` entries that define the locations to scan. You must also provide a unique `RunID=` entry for each scan you perform. After you complete the `Offscan.ini` changes, you can run the Offscan utility. When working in Vista, you must open the command prompt by right-clicking the Command Prompt entry in the Start menu and choosing Run as Administrator from the context menu. Figure 1-5 shows typical output from the command (notice that the title bar begins with the word Administrator to show that this command prompt is in administrator mode).

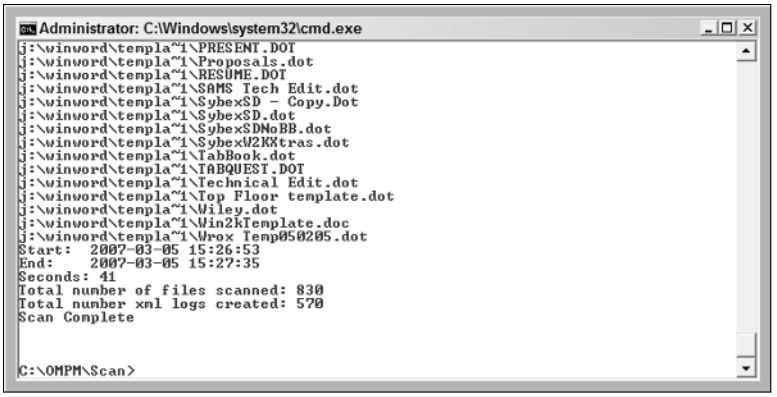

Figure 1-5: Scan your hard drive for potential problem documents and conversion issues.

Microsoft suggests that you distribute the scanning tools to everyone on the network to ensure you get all of the client drives, as well as the data, on the server. Whether you need to perform this step depends on how centralized you keep your data. Some organizations need only an administrator scan of the server's hard drive to locate the required updates.

Eventually you'll end up with a host of `CAB` files that are useless by themselves; you need to import the data into a database to make it useful. This tool requires a copy of SQL Server to run, and, because of compatibility problems, you need SQL Server 2005 SP2 or better when working in Vista. Fortunately, you can obtain a free copy of SQL Server 2005 Express SP2 at

`http://www.microsoft.com/sql/editions/express/default.mspx`

that performs well for this task. You can download the basic version of SQL Server Express (a 36.5MB download), but I recommend getting SQL Server 2005 Express Edition with Advanced Services (a 234MB download) to ensure you have everything you need. If you haven't worked with SQL Server before, check out *SQL Server 2005 For Dummies,* by Andrew Watt, or *SQL Server 2005 Express Edition For Dummies,* by Robert Schneider (both from Wiley Publishing, Inc.).

SQL Server requires a second installation to work with XML data. Be sure you download and install the SqlXml 3.0 Service Pack 3 (SP3) add-in found at

`http://www.microsoft.com/downloads/details.aspx?FamilyID=51d4a154-8e23-47d2-a033-764259cfb53b`

After you install SQL Server or SQL Server Express, use the CreateDB batch file found in the `C:\OMPM\Database` folder to create a database to hold the data you've collected. When you run CreateDB, you must supply the name of the server, the SQL Server instance, and the name of the database you use. For example, if you're using SQL Server Express and the name of your server is `MyServer`, you'll likely type something like

```
CreateDB MyServer\SQLExpress OMPM001
```

at the command line and then press Enter. You can always delete the database if you make a mistake by using the DeleteDB batch file.

You may get an error message from the `CreateDB` batch file, complaining that it can't create the required database. Don't worry; you can normally fix this problem with a simple change. The following steps tell you how to perform this task.

1. **Select Start➪Programs➪Microsoft SQL Server 2005➪Configuration Tools➪SQL Server Configuration Manager.**

   You'll see the SQL Server Configuration Manager window.

2. **Open the SQL Server Configuration Manager (Local)\SQL Server 2005 Network Configuration\Protocols for SQLEXPRESS (or other server instance) folder.**

   You'll see a list of protocols for the selected SQL Server instance, as shown in Figure 1-6.

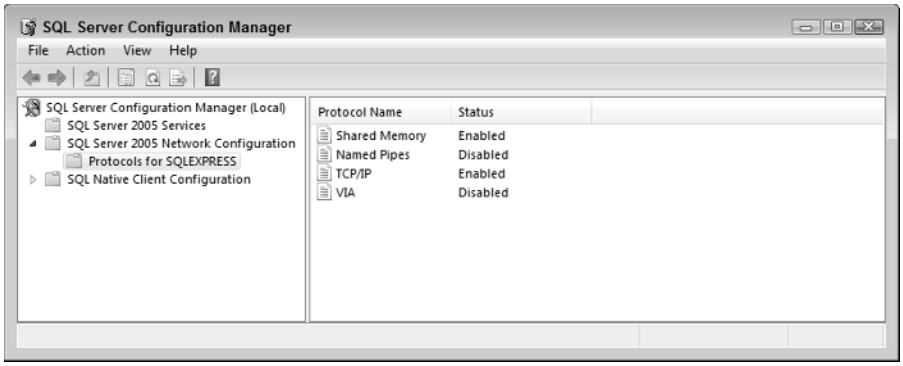

Figure 1-6: Enable the TCP/IP protocol so the system can communicate with SQL Server.

3. **Right-click TCP/IP and choose Enable from the context menu.**

   You'll see a warning message stating the change won't take effect until you restart the service.

4. **Click OK.**
5. **Open the Server Configuration Manager (Local)\SQL Server 2005 Services folder.**

   You'll see a list of SQL Server-related services installed on your system, as shown in Figure 1-7.

Figure 1-7: Restart the SQL Server service to make the changes permanent.

6. **Right-click the SQL Server (SQLEXPRESS) (or other instance) entry and choose Restart from the context menu.**

   The `CreateDB` batch file should execute properly when you try it again.

Now that you have a handy database to use for storage, you can import the data into it. The `ImportScans` batch file in the `C:\OMPM\Database` folder performs this task. As with the CreateDB batch file, you supply the name of the server, the SQL Server instance, and the database. In addition, you must supply the location of the scan files, which is `C:\OMPM\SCANDATA` if you use the default settings. Given the same setup as before, you might type

```
ImportScans MyServer\SQLExpress OMPM001 C:\OMPM\SCANDATA
```

at the command line and press Enter to complete this part of the task.

If you find that the ImportScans batch file fails because it can't find the OSQL utility, you can add the utility to the command prompt path. Normally you'll find this utility in the following folder on your system:

```
C:\Program Files\Microsoft SQL Server\90\Tools\Binn
```

To get there, type

```
Path = C:\Program Files\Microsoft SQL Server\90\Tools\
Binn;%PATH%
```

With this new path added to your command prompt, try the `ImportScans` batch file again.

After all of this work, you're probably wondering about the payoff. Importing the data into the database lets you begin analysis. The OMPM helps you create reports that you'll use to update your Office environment later. To begin using reports, double-click the `OMPM.ACCDR` icon in the `C:\OMPM\Report` folder. If you're using Vista, you'll very likely see a number of security messages that you'll use to elevate your privileges to the required level. After you supply the name of the database used to hold your data (OMPM001 in the example), you can begin working with reports. Figure 1-8 shows a typical example. In this case, the report tells you about the Word compatibility concerns for a set of existing documents.

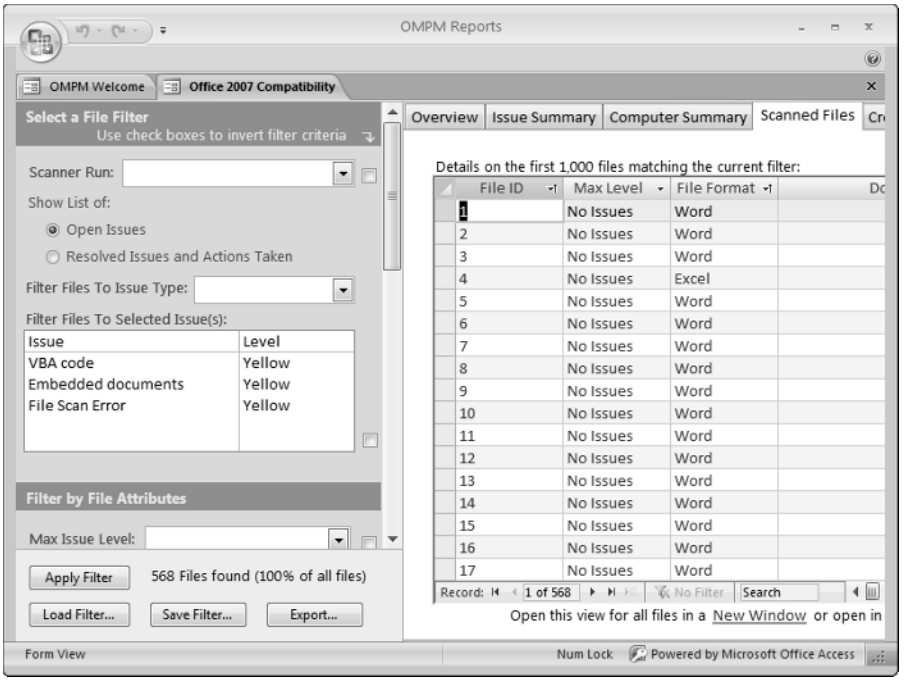

Figure 1-8: Create a strategy for updating to Office 2007 using these reports as a basis.

Getting quick information about XML

Many of the new features of Windows and Office rely on XML for configuration. The reasons that Microsoft uses XML are that it's easy to understand, very flexible, standardized in format, and text based. Of course, you may not have used XML and might not understand how it works. Fortunately, the XML used with Office for RibbonX is straightforward; you don't need to delve into the depths of XML to understand it. Even so, if you haven't used XML before, you might want to visit the XML tutorial at `http://www.w3schools.com/xml/`.

RibbonX also relies on at least one namespace to get the job done. In this case, the URL defines a location that specifies how the RibbonX entries work. Generally, you don't need to know too much about the namespace — except for how to include it (as described in the "Understanding tabs" section of the chapter). If you do want to know more about namespaces, check out the tutorial at `http://www.zvon.org/index.php?nav_id=172&ns=34`.

At some point, you might find that you want to go deeper into XML. You'll find great references online — including one at `http://www.xml.com/axml/axml.html`. Don't forget to review Microsoft's offerings at `http://msdn.microsoft.com/xml/`.

It's important to consider all the compatibility concerns for a set of documents before you begin updating your application. Only when the documents provide a reasonable level of compatibility should you consider updating your applications. If you have hundreds of documents that require considerable conversion, it may be better to stick with the old version of Office for those documents and start fresh with a new application for new documents. Obviously, the decision to update depends on a number of factors, including the value of the data to your organization. Sometimes you have to update the documents, no matter how much it costs, because of the data's value. You can find a complete listing of migration considerations at

`http://technet2.microsoft.com/Office/en-us/library/1db55715-df10-428d-ad42-4ce3c58a8edf1033.mspx`

Defining the RibbonX Elements

Every physical element in the Ribbon has a corresponding element in the RibbonX programming interface. The user sees the results of changes you make with code. Unlike previous versions of Office, however, RibbonX doesn't

rely on a hierarchical set of objects to control the interface. Instead, the interface relies on an XML file that describes the various elements. This file follows the following hierarchy of XML elements:

- Tabs
- Groups
- Controls

Unlike many other Microsoft offerings, the hierarchy for RibbonX is relatively absolute. A tab can't contain other tabs, and it only holds groups. Likewise, you place controls in groups; you don't place them directly in tabs. Chapters 2 and 3 provide detailed descriptions of how to build a good Ribbon interface. The following sections provide an overview of the various elements.

Understanding tabs

Tabs are the uppermost element in the Ribbon hierarchy. You can see an example of a custom tab in Figure 1-9. In this case, you're looking at a custom tab called My Tab that contains a single group. The group, My Group, contains a single button named My Button. Obviously, you wouldn't create a real-world tab like this, but it's good for explanation purposes.

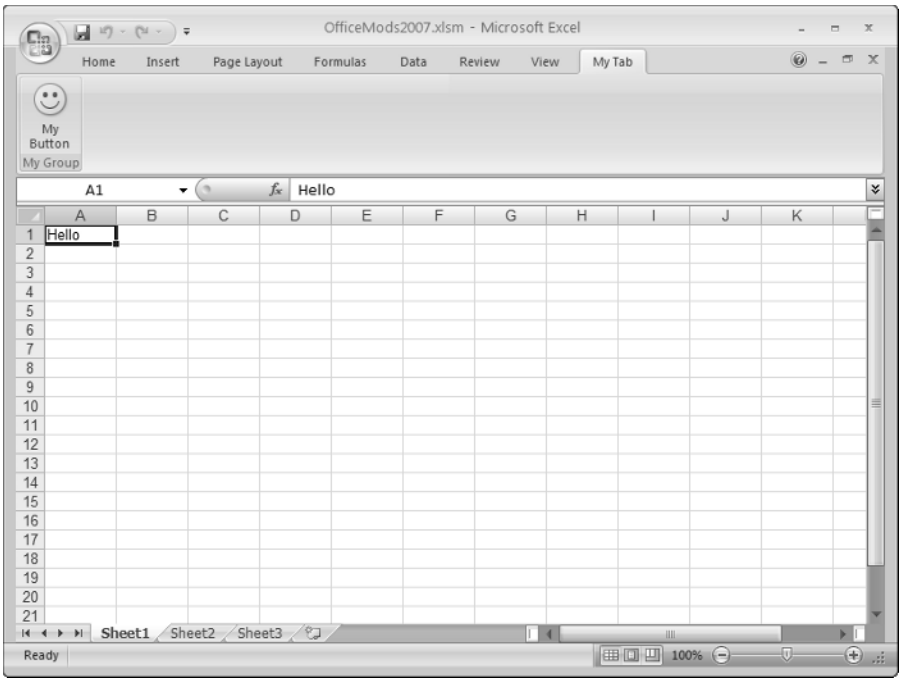

Figure 1-9: Each physical element of the Ribbon has a corresponding RibbonX element.

Generally, you'll use tabs to focus user attention on a particular task or requirement. For example, when you look at the Word Mailings tab, you see everything required to mail a document to someone. The tab focuses the user's attention on mailing the document; not on another task such as formatting it. You can also create custom tabs that focus user attention on specific tasks for your company.

To create My Tab, I wrote the XML file shown in Listing 1-1. As you can see, this is a standard XML file with all of the usual features including a processing instruction, root element (`<customUI>`), child elements such as `<ribbon>`, and properties. (You can find the source code for this example on the Dummies.com site at `http://www.dummies.com/go/ribbonxfd`.)

Listing 1-1: Creating New Ribbon Elements Using XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<customUI
xmlns=
"http://schemas.microsoft.com/office/2006/01/customui">

  <ribbon>
    <tabs>
      <tab id="myTab" label="My Tab">
        <group id="myGroup" label="My Group">
          <button id="myButton"
                  label="My Button"
                  imageMso="HappyFace"
                  size="large"
                  onAction="myButton_ClickHandler"/>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

You must define a namespace for the custom interface. In this case, the namespace resides at `http://schemas.microsoft.com/office/2006/01/customui`, which is the location you'll use for every custom interface element you create.

Office does a lot of the work for you when it comes to creating a new tab. Although this implementation is minimal, it gives you an idea of what you can do with very little XML. All you need to create a new tab is the `<tab>` element with `id` and `label` properties. You use the `id` attribute to access the tab from your application. The user sees the text you provide as part of the `label` attribute. As you can see, the physical presentation on the Ribbon always corresponds to the XML you provide as part of the RibbonX interface.

Understanding groups

Groups gather like controls together so the user doesn't spend as much time looking for the right control. Using groups reduces user confusion and makes it easier to show users how to perform a particular task. For example, when working with the Word Mailings tab, you'll see a Create group. The Create group contains controls for creating both envelopes and labels. The user doesn't have to look around for either item; they both appear in the same place.

In the example shown in Figure 1-9, you see My Group, which contains a single control. You create a group in XML by using the `<group>` element. As with a tab, you must include an `id` attribute, which identifies the group in your code, and a `label` attribute, which provides text that the user can use to identify the group on-screen.

Understanding controls

A control performs a specific task. You don't want to pile multiple tasks onto one control because, in many cases, doing so confuses the user. Each control should perform a specific task; you should always choose a control that performs the task well. For example, you can use a pushbutton control to help the user execute a task. On the other hand, you might use a check box to let the user make a choice. A drop-down list box lets the user choose between multiple choices instead of a simple yes/no choice.

Some controls come in multiple sizes. Look at the Word Review tab and you'll notice that the Proofing group contains multiple pushbutton sizes. The large pushbuttons draw the user's attention to major tasks, such as checking spelling and grammar. The small pushbuttons help the user perform less common tasks, such as setting the document language or performing a word count.

Controls always require more code than any other Ribbon element because they aren't static; they perform some task. Look again at Listing 1-1 — you'll notice that the simple `<button>` element requires multiple arguments. As with the other elements discussed so far, you must provide `id` and `label` attributes. When your button includes an image, you must provide the name of the image as part of the `imageMso` attribute. The image must also appear within the file that references it; you'll see how the referencing works in Chapters 3, 4, and 5. The `size` attribute defines the size of the pushbutton. Most pushbuttons also include an `onAction` attribute that connects the pushbutton to code you create.

The `onAction` attribute is special because it reflects an event instead of a property. When the user clicks the pushbutton, the `onAction` event occurs. Most controls provide access to more than one event (as you'll see in the "Defining the RibbonX Controls " section of Chapter 3).

Considering the Ribbon in Office 2007

Before you embark on the journey of creating your own Ribbon elements, it helps to understand what Microsoft provides within Office. You may find that you can make small changes to the existing setup and still obtain a usable interface. For example, you may decide to place your custom styles in the Styles group of the Home tab, rather than create a custom tab for the purpose. The following sections provide you with an overview of the existing Ribbon elements found in Office 2007 so you can choose which application features to update and which to make part of existing Ribbon elements.

Understanding the common Ribbon elements

Every Office application except Outlook has a Home tab included with it. The Home tab contains the elements that the user needs most often. For example, the Home tab for Word contains common formatting groups such as Font, Paragraph, and Styles. It also has common editing tools and provides access to the Clipboard. The Home tab for PowerPoint includes many of these elements, along with a Drawing group that contains common drawing tools. Access doesn't really feature much in the way of data formatting; its Home tab focuses more on data sorting and manipulation. Excel includes some formatting tools on its Home tab, but you'll also find Number, Alignment, and Cells groups that contain tools for working with worksheets.

The Office applications also include common task-based tabs. The two most common tabs are Insert and Review. The Insert tab contains groups that help the user insert data. When working with Word, you'll find groups for headers, footers, text, symbols, links, tables, illustrations, and pages. Excel has special features on its Insert tab to add charts to a worksheet; PowerPoint provides numerous graphics features on its Insert tab. The Insert tab is an example of a task-based tab that focuses on a particular task as it occurs in that application.

Contrast the Insert tab with the Review tab. The Review tab varies little between applications. For example, all of the applications that support it include Proofing and Comments groups since these are common review task requirements. The Review tab also includes document protection features, even though these features don't always appear as part of a Protect group. The point is that this task is common among applications and you should strive to maintain that commonality as much as possible when creating a custom application.

Because Word and Excel deal with larger documents, they both include a Page Layout tab as well. In both cases, you find Themes and Page Setup groups that contain controls for managing the pages as a whole. Word includes a Paragraph group because it works specifically with text in paragraphs. Excel, on the other hand, includes a Sheet Options group with controls that help you manage the appearance of a single large worksheet. The Page Layout tab is an example of a tab that includes both common and application-specific features.

Looking at the Ribbon in Word

Word is one of the most complex implementations of the Ribbon for good reason: Manipulating text so it looks perfect when printed is a difficult task, and even with the right tools you can make mistakes. The References tab in Word is an example of a tab that performs a specialized task. You use the groups on the References tab to add citations, footnotes, a table of contents, an index, and other document-specific features to a file. Not everyone will use this particular tab, so this is a tab you may choose to remove from view when creating a custom application.

The specialized toolbars and menus that developers add to Word also indicate the complexity of working with the written word. Unfortunately, Word has the dubious distinction of providing the applications least likely to move from previous versions of Office to Office 2007. The reason that people have created so many applications for Word is also the reason you'll probably end up moving the application to Office 2007 or starting over from scratch.

Looking at the Ribbon in Excel

Excel users require access to formulas to compute entries in a worksheet. The Formulas tab contains features that help the user work with formulas, including the Function Library group, which contains common formulas. You'll also

find groups for managing named ranges, auditing existing formulas, and performing calculations. Generally, if you have to add a new formula to Excel, this is the tab to hold it. Rather than create a custom tab for the task, you can remove the features you aren't using and add the custom features your application requires.

Worksheets often require external access to data stored in another location. The Data tab provides basic data-access functionality, including the Get External Data button. Unfortunately, these features are generic; and they may not meet your specific needs. If you have complex data requirements for your application, you'll very likely add those features to this tab. Unlike many other tabs, this one isn't loaded with a lot of extra features you won't use, so you should be able to maintain the default setup and simply add the new controls you need.

Looking at the Ribbon in Access

The first thing that will surprise you when you look at Access is the small number of tabs (as compared to other Office products). The two main tabs are Create and External Data. You use the Create tab to define new database objects, and the External Data tab to access data outside of Access. The Database Tools folder contains esoteric items for creating macros, defining relationships, analyzing and moving data, manipulating database content, and administering the database. Generally speaking, you probably won't want to modify any of these tabs, but you may want to hide them from users. Placing any custom features you want to provide on a special tab makes sense with Access, especially considering that you aren't fighting with the application for space.

Looking at the Ribbon in Outlook

When you initially start Outlook, you might wonder whether it even uses the Ribbon. The Ribbon doesn't appear as part of the main application, but it does appear as part of special features, such as creating a message, adding an appointment, and defining a new task. Consequently, any work you perform with the Outlook Ribbon will be task-specific.

Outlook developers are also going to have to maintain their current skills, because Outlook uses the older toolbar and menu interface for performing general tasks. For example, if you want to add a special feature for signing up

for Internet access, you'll need to add it to a toolbar or menu, not to the Ribbon. Only when you want to add a task-specific feature — such as special headings for an e-mail message — do you need to worry about the Ribbon in Access.

Looking at the Ribbon in PowerPoint

The special Ribbon features in PowerPoint all deal with the slides you create. These tabs include Design, Animations, and Slideshow.

All three tabs control the slides in some way. The Design tab includes features for changing the slide appearance. For example, you can use the features in the Themes group to control the overall theme for your presentation. The Animations tab controls how you move from one slide to another. For example, you can add a special transition or sound between slides. The Slideshow tab contains controls for changing the slide presentation. For example, the Set Up group contains controls that let you choose how the slideshow begins and ends. You can also use these features to rehearse the slideshow timing. When you want to add custom features to PowerPoint that affect the slides in any of these ways, you'll want to augment one of the existing tabs.

Of course, slideshows don't deal just with slides. You might want to add a special tab for accessing external data. In some cases, you won't find a standard tab to hold the special features you want to add to PowerPoint, in which case, you'll want to add a new tab, rather than change the focus of an existing tab.