Essential SharePoint

By Jeff Webb

.....
Publisher: O'Reilly
Pub Date: May 2005
ISBN: 0-596-00880-5
Pages: 336

Table of Contents | Index

Overview

Want to work more efficiently and effectively? Want to improve productivity? Microsoft is betting that you do. That's why it created Windows SharePoint Services--a set of collaboration tools that helps organizations increase individual and team productivity by enabling them to create web sites for information sharing and document collaboration.

Through these team-oriented web sites, users capture and share ideas, and work together on documents, tasks, contacts, etc.--either among themselves or with partners and customers. And if you have Windows 2003 Server, then you already have SharePoint, since it's built right in. But before you can enjoy the benefits of SharePoint, you need to know how to turn it on, set it up, and get your applications working with it.

Essential Sharepoint will help you do just that. It's not only the most complete guide for setting up and using these increasingly popular sites, but it also explains in detail the integration that makes SharePoint exciting. Everything you need to know about SharePoint is covered, including:

- hosting choices
- administration
- customization
- integration with Microsoft Office
- developing new SharePoint functionality
- when to use SharePoint portal server

Essential Sharepoint covers all the key topics for getting up and running with this powerful and popular set of collaboration tools. And it's not just for members of the IT staff. This comprehensive guide is for anyone in an organization who wants to explore Microsoft SharePoint in order to foster collaboration with other users.

Essential SharePoint

By Jeff Webb

.....
Publisher: O'Reilly
Pub Date: May 2005
ISBN: 0-596-00880-5
Pages: 336

Table of Contents | Index

PREV NEXT

Essential SharePoint

by Jeff Webb

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*safari.oreilly.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or corporate@oreilly.com.

Editor:	Simon St.Laurent
Production Editor:	Jamie Peppard
Cover Designer:	Emma Colby
Interior Designer:	David Futato
Printing History:	
May 2005:	First Edition.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. *Essential SharePoint*, the image of a wombat, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 0-596-00880-5

[M]

Dedication

For Sophiayou are talented, beautiful, smart, and fun to be with.

Preface

If you don't have SharePoint Services, you need it. Even if you work alone from your home office as I do, SharePoint is too useful to pass upand it's *free* (well, kind of). SharePoint Services is part of Windows 2003, so if you already have Windows Server 2003, you can download the installation from Microsoft and install it fairly easily.

If you don't have Windows Server 2003, you can sign up for a free 30-day trial through one of the SharePoint hosting providers, which operate much like other web-hosting services. With the free trial, you get an account and some server space. You can administer your site yourself through a web-based interface, and after the free trial period they'll bill you monthly.

Who Should Read This Book

SharePoint is a server product that is used by a wide range of people. This book was written to serve readers based on their varying roles in your company. The following table identifies these roles and recommends specific chapters that may be of interest.

Role	Responsibilities	Read these chapters
Administrator	Installs server software, configures web sites, manages security.	1, 2, 3
Designer	Designs pages on site, creates site templates.	1, 3, 8
Contributor	Adds documents, lists, tasks, announcements, and other content.	1, 4, 5, 6
Contributor with data entry	Completes InfoPath forms.	1, 4, 5, 6, 7
Reader	Views sites and documents but makes no changes.	1, 4, 6
Programmer	Creates custom web parts.	1, 3, 8, 9, 10

Because I cover the full range of things you can do with SharePoint in this book, certain sections may be too technical for some readers. For example, you won't want to read Chapter 9 if you're not a programmer! Still, I wanted to provide a book that gives readers room to grow and that does more than reorganize information that is already available for free in Help or on the Internet.

Navigating in SharePoint

Many SharePoint tasks involve navigating through several web pages. I provide the navigation path based on the links on each page. For example, "choose Site Settings → Go to Site Administration → Delete this site" means click Site Settings on the navigation bar, click Site Administration on the next page, and finally click Delete this site. Each of these links takes you to a new page, and sometimes you have to search a bit to find the next link. I use this abbreviated style because I think the alternatives are more wordy, but not much clearer.

If you get lost trying to follow the path, you can enter the address of the page directly in your browser's Address bar. The figures that accompany tasks show the addresses of most important pages. For example, this is the address of the page used to delete a site:

```
http://wombat1/newsite/_layouts/1033/deleteweb.aspx
```

In this case, you'd want to replace `//wombat1/newsite/` with the address of the site to delete. Using the address bar is also a handy shortcut for repeating a task on multiple sites: to delete multiple sites, just change the site name as described and repeat for each site you wish to delete.

Getting the Code Examples

The code examples are available on my web site at http://www.usingsharepoint.com/Samples. You may want to bookmark that site in your browser so you can see the examples in action while you're reading. In addition, there is a version of code samples that can be installed on your own server as a template at http://www.oreilly.com/catalog/essentialsp.

Although installing the samples yourself is a little complicated, it gives you complete access to the source and lets you make changes. You don't need to install the samples right away, though: get comfortable with SharePoint first.

You are free to use the samples in this book in your own work but not to claim them as your own. Ditto for sections of text in this book. It is always polite to credit the source of your quotes so that other can find (and possibly buy) the book if the find it useful. If you have questions about your use of samples, please email permissions@oreilly.com.

If You Need Help

In addition to the help that can be found through my web site and those of O'Reilly, Microsoft, and Google, here are some otherSharePoint resources:

- The newsgroups *microsoft.public.windows.sharepointservices* and *microsoft.public.windows.sharepointservices.development* are actively supported by the SharePoint community.
- The FAQ at http://wss.collutions.com answers a lot of specific questions.
- The site at http://www.wssdemo.com provides excellent examples of ways to handle content in SharePoint.
- A Resources section at the end of each chapter cites sources for help that is specific to the material coverered in that chapter.

Font Conventions

This book follows certain conventions for font usage. Understanding these conventions up front makes it easier to use this book.

Plain text

Indicates menu titles, menu options, menu buttons, and keyboard accelerators (such as Alt and Ctrl).

Italic

Indicates new terms, URLs, email addresses, filenames, program names, file extensions, pathnames, directories, and new terms where they are defined.

`Constant width`

Is used for all code listings, commands, options, variables, attributes, properties, parameters, values, XML tags, HTML tags, the contents of files, the output of commands, and anything that appears literally in a SharePoint page.

`Constant width italic`

Indicates text that should be replaced with user-supplied values.

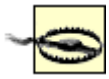
`Constant width bold`

Indicates additions in the code examples.

~~`Constant width strikethrough`~~

Shows deletions from the code examples.

This icon indicates a tip, suggestion, or general note.



This icon designates a caution or warning.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international or local)
(707) 829-0104 (fax)

We have a web page for this book, where we list errata, examples, or any additional information. You can access this page at:

http://www.oreilly.com/catalog/essentialsp

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our web site at:

http://www.oreilly.com/

Safari Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at http://safari.oreilly.com.

Acknowledgments

John Craig reviewed this book for me and his help is more than a pleasure. Years ago I joined John in writing two editions of his brainchild *Visual Basic Developer's Workshop*. I'm still proud to have worked on that book, and I hope we continue to work together for a long time.

Simon St.Laurent is my editor. He's right about everything, which saves me a great deal of time arguing and keeps me from crossing the line of good taste. This is my second book with Simon and, as with John, I hope one of many.

The only other team members who've been with me longer than John and Simon are my family: Trish, Dorian, and Sophia. They've been patient with my programming jokes, take me to the beach on weekends, and find me when I'm lost.

Finally, I must thank everyone in the SharePoint community. No individual can do as much as those working together, and the newsgroups and sites mentioned earlier help all of us. I encourage you to ask *and* answer questions whenever possible. Several key people facilitate these discussions tremendously, and I'd like to acknowledge Mike Walsh and Ian Morrish for all their work. I hope they are well-rewarded.

Chapter 1. Why Use SharePoint?

SharePoint is a component of Windows 2003 that lets you share Microsoft Office documents with others through web pages. Unlike most web sites, SharePoint sites are designed to be highly dynamic. Team members can easily upload documents, add public announcements, send alerts, track work items, and call meetings right from within Office products.

1.1. Solving Problems

SharePoint solves four problems:

- It's difficult to keep track of all the documents in even a small office.
- Email isn't a great way to share files.
- We work all over the place.
- It's hard to create and maintain web sites on your own.

Most offices have addressed these problems using a combination of tools or work procedures. For instance, the boss says, "Route your proposal to me, Ed, and Jane for approval," and you email the file to each of them, asking for comments with change-tracking enabled. You set a deadline, keep copies of each reviewer's response, and reconcile conflicting comments.

That approach works because your boss, Ed, and Jane are great coworkers, check their email often, and communicate well with each other, and because the proposal is well-suited for this approach. It's pretty easy to throw a wrench into that machine, however. Say, for instance, your proposal isn't a Word document, but rather a set of drawings, a spreadsheet of test results, and a list of links to related products. How do you route that? How do you collect comments?

Or say your project has multiple authors *and* multiple files. Each of these complications increases the vulnerability of the process, and improvised solutions start to break down: zipped files bounce back from mail servers, comments are lost or not archived, out-of-date drawings are included, deadlines are missed.

SharePoint helps solve all these problems using the Office system. Instead of routing the files by email, you set up a workspace for project documents on a SharePoint web site. Email alerts notify reviewers when files are available; reviewers can discuss changes online, read each other's comments, and assign tasks and deadlinesand all changes are recorded in version history.

SharePoint is a big improvement over improvised solutions, but the degree of improvement is affected by two conditions:

- SharePoint affects work processes, so you need to think about how to influence process effectively before it can help.
- SharePoint is closely tied to Office 2003; although you can use earlier Office versions or even other applications, the latest Microsoft suite provides the greatest benefit.

If you can live with those two caveats, then we can get started. Otherwise, you'd probably better put this book back on the shelf so someone else can buy it.

For a list of SharePoint features supported by Office 2000 and 2002 (XP), see Appendix B.

1.2. Storing and Sharing

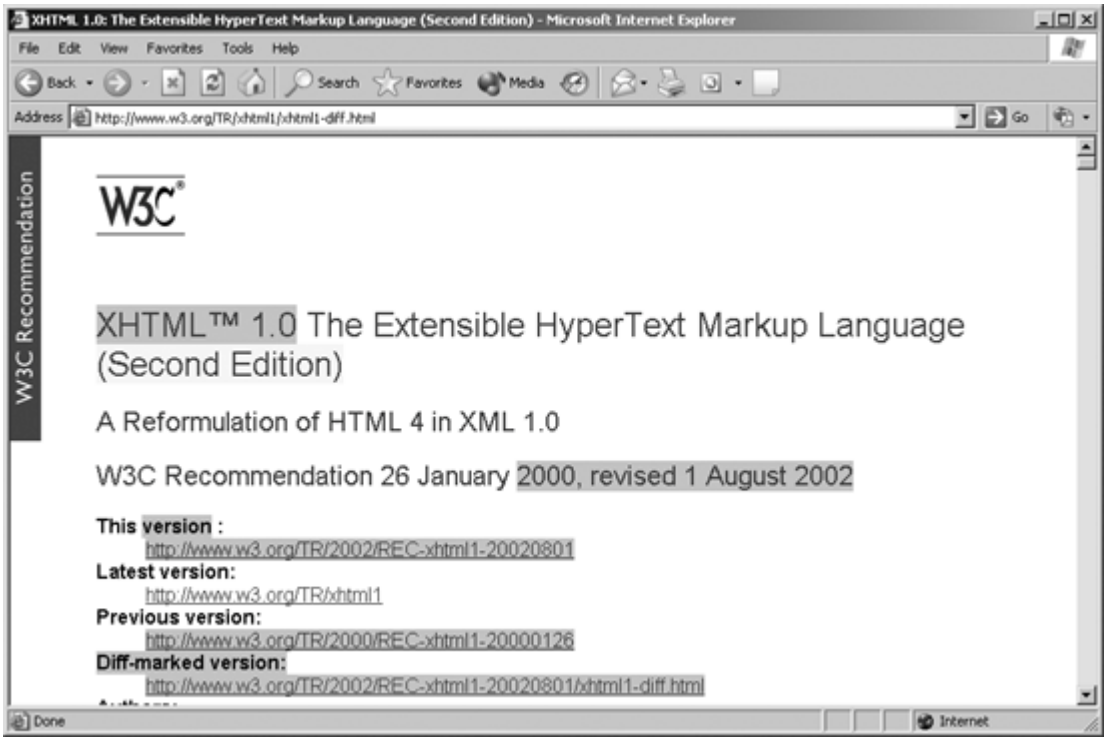
Documents store and share information, but there are two aspects that documents sometimes ignore: content modification and change tracking. To see this handled well in a web-based document library, visit the W3C web site, which indicates change using formatting as shown in Figure 1-1.

If you want to find the reason for a change, you can visit a discussion at the link to the HTML working group. You have to be a member of the group to see the discussion, but that restriction is appropriate since reading and participating in discussions are key benefits of joining the W3C.

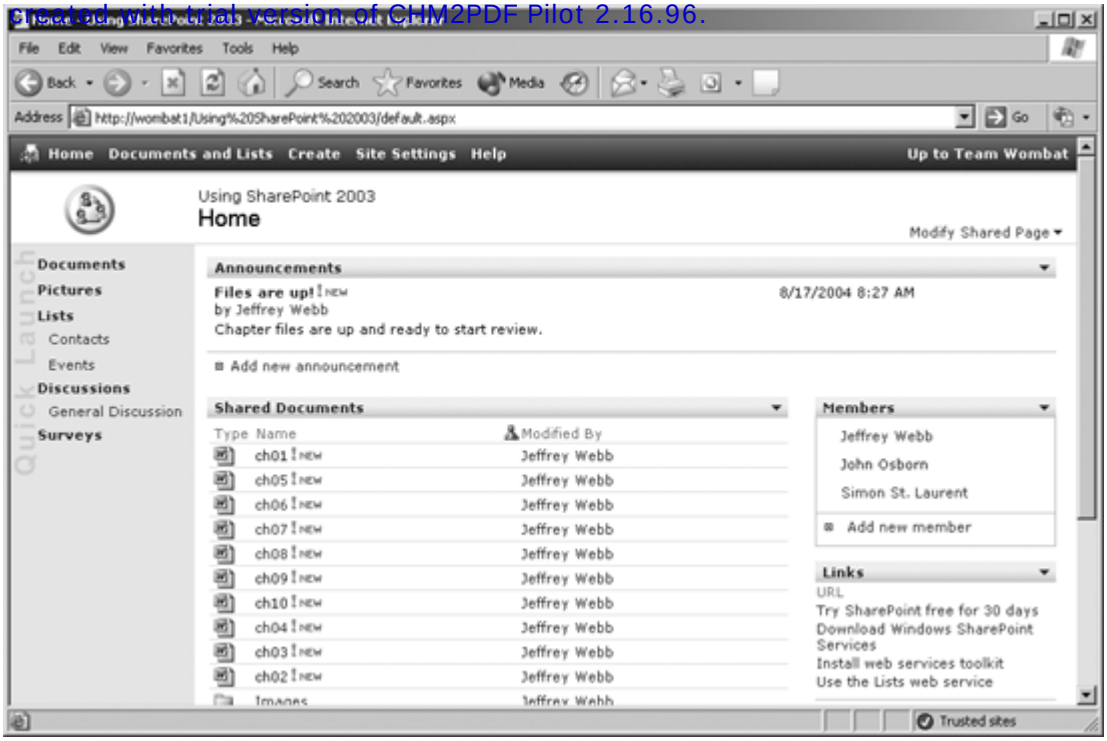
The W3C put a lot of thought and effort into designing and maintaining their site, which does an excellent job of storing and sharing knowledge about web standards. It's a library of current standards, the history of those standards, discussions, and lists of related work.

SharePoint lets you create the same type of document libraries, histories, discussions, and lists by gathering the work from around your office and making it available from a web page. Figure 1-2 shows the SharePoint workspace I used to create this book.

animal 1-1. W3C uses diff-marked web pages to share changes



animal 1-2. Sharing chapter files with my publisher via SharePoint



I didn't put as much thought or effort into my site as the W3C did, but the two sites have a lot in common. They both:

- Contain a library of documents.
- Record change history for those documents.
- Provide a forum for online discussions.
- List links to related information.
- Permit members different levels of access.
- Allow access from any location via the Internet.

From the workspace, my editors can read and comment on my chapters, track changes, try the samples, retrieve art files, and measure my progress against the schedule.

1.3. Improving Collaboration

The main difference between my site and the W3C site is that mine contains mostly Word documents, while their site is mostly HTML files. That means my users must have Word to view the files in Figure 1-2, but it also means that those users can open those files directly from the web site and make changes if they have permission.

In fact, my editors don't even need to open my site to make changes. They can open the Word files listed in Figure 1-2 and save them to their own computers as linked documents; then whenever they open their local copy, it is automatically refreshed with changes from the SharePoint site. Similarly, any changes they make to the files are sent to SharePoint when they close their local copy, as shown in Figure 1-3.

Figure 1-3 illustrates a distributed file system where O'Reilly (my publisher) can share access to the files I am working on, even though I live in Florida, and they are in Massachusetts and California (and sometimes in between).

The files are synchronized over the Internet whenever the file is opened or closed. If Simon (my editor) finds himself somewhere without a good network connection, however, he can cancel the updates but still work on the fileas long as he promises to synchronize it later!

For me, SharePoint replaced zipping files and emailing them as they were completed, reviewed, or changed, because very large zip files sometimes bounced back from the mail serverputting the "dead" in deadline.

animal 1-3. Local copies can be linked to SharePoint documents

SharePoint provides several types of collaboration tools, as shown in Figure 1-4:

Announcements

Use to keep teammates informed.

Alerts

Notify team members if a document changes.

Task lists

Assign work to team members.

Events

Use to schedule meetings, deadlines, and get-togethers.

Discussions

Allow online discourse between members.

Contact lists

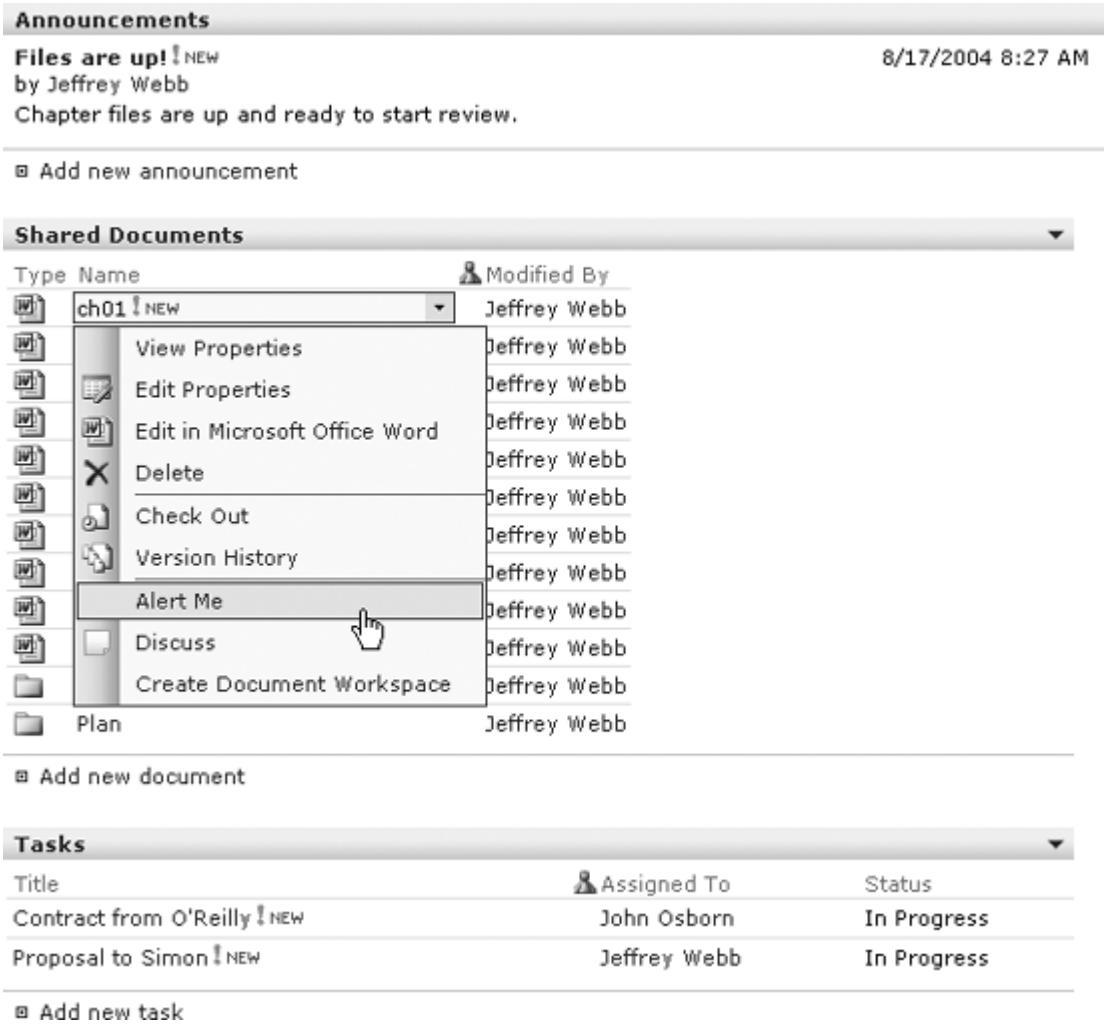
Share email addresses and phone numbers.

1.4. Going Public

SharePoint creates web sites. That means your documents can be made public over the Internet, shared within a private intranet, or both. My site is public, but access to certain areas is restricted so that no one steals my chapters.

I assigned different permissions to different members of my site. As site administrator, I have full control; Simon and John are contributors and can make changes; my technical reviewers can read files; and so on. Once the chapters are complete, I move

animal 1-4. SharePoint sites provide tools to communicate with team members



selected excerpts to a public area that allows everyone to read them. I maintain these levels of access from a web page within the SharePoint site as shown in Figure 1-5.

Membership is a key aspect of SharePoint. In most organizations, individuals may belong to many different teams. Some teams are organizational (company, division, department, etc.) while other teams span organization lines (project teams, task forces, and so on). SharePoint accommodates both structures well; there's a lot more on this in Chapter 2.

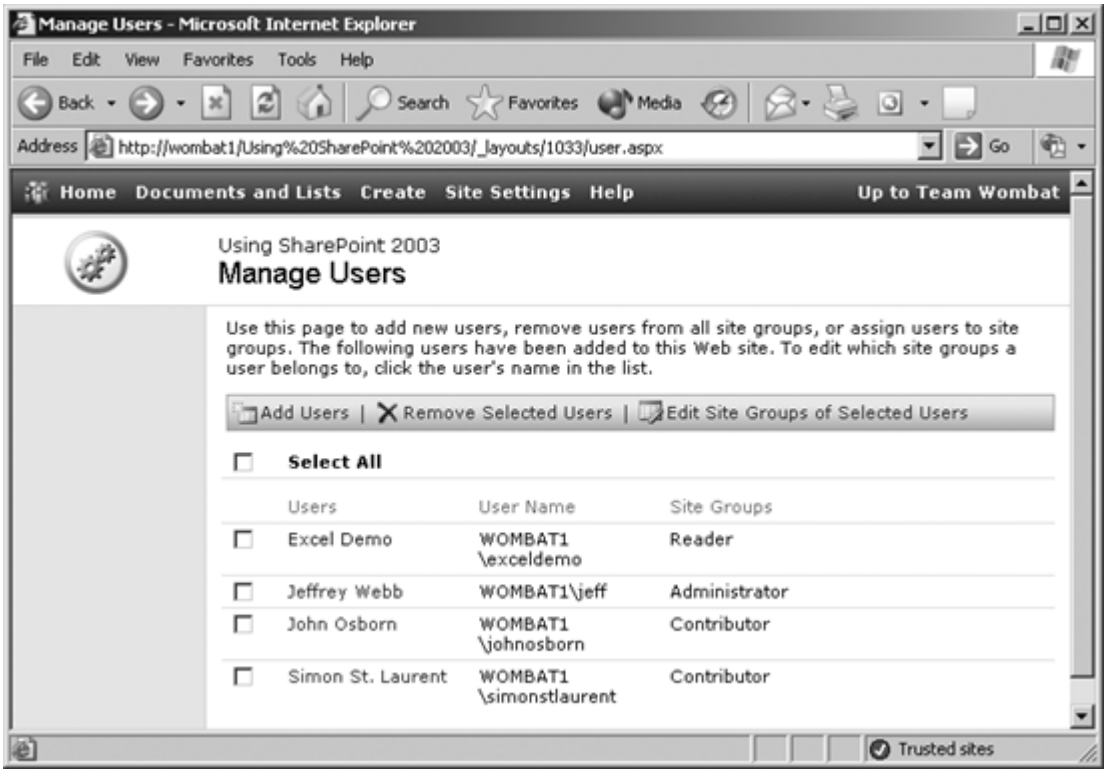
1.5. Organizing Sites

SharePoint sites are organized into folders, just like any other web site, but there are some terms SharePoint uses that should be explained up front:

Virtual server

The root location for one or more web sites. SharePoint can host multiple virtual servers on a single server. Each virtual server can have its own domain name. For example, www.usingsharepoint.com and www.mstrainingkits.com are hosted on the same SharePoint server.

animal 1-5. Adding members and setting permissions from a web page in SharePoint



Site collection

A group of top-level sites on one virtual server. Site collections allow separate administration of related sites.

Top-level site

The primary *web site* for a domain. Each virtual server can host many different top-level web sites, which in turn contain subsites. If SharePoint is set up to allow self-service site creation, other top-level sites may also exist in the *//domain/sites/* folder.

Subsite

A site within a top-level web site. Subsites are used to control access and organize the content within a site; members of the top-level site may or may not have access to a specific subsite.

Page

Sites contain web pages that present their content. Most SharePoint pages are web part pages (*.aspx*), but you can also add HTML pages or other content to SharePoint sites as document libraries.

List

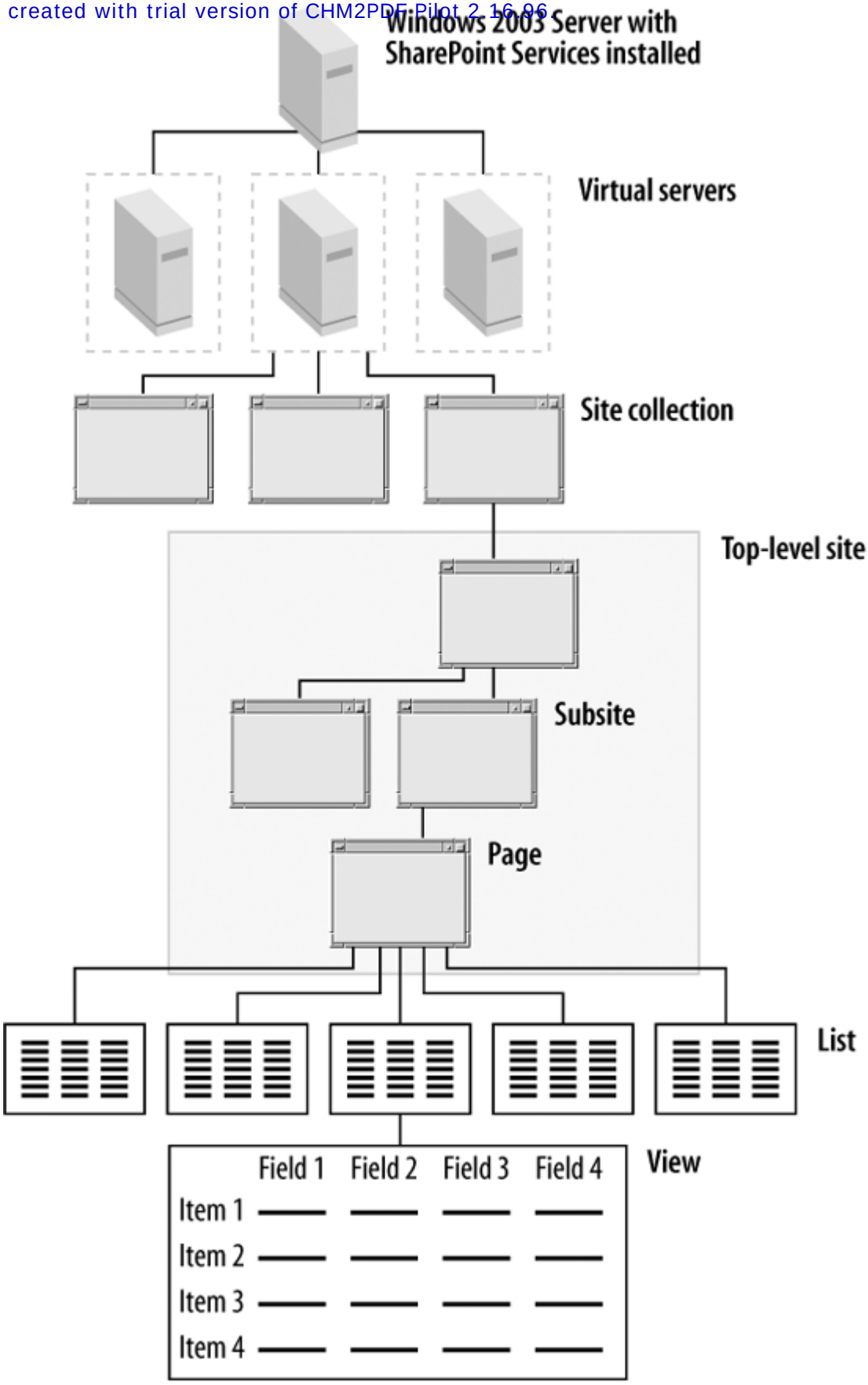
A table of related data. Lists are used throughout SharePoint: announcements, contacts, discussions, and other types of content are implemented through lists. Users can also create lists using Excel and link the contents of those shared lists to other documents in Word and Excel.

View

The way a list is displayed. The default view of a list is to display the data in columns that can be sorted or filtered, but you can also display lists in calendar form or as a spreadsheet-like data grid. Views can include criteria to display only certain columns or rows and may group items to create expandable tree-views.

Virtual servers, top-level sites, subsites, and lists create a hierarchy that is illustrated in Figure 1-6.

animal 1-6. SharePoint's terms for organizing content



1.6. Types of Sites

Authorized team members can create new sites in SharePoint very easilyin fact, just a few clicks in Word creates a new document workspace site almost instantly. To help impose a unified look, SharePoint includes *site templates* that influence the type of site created. SharePoint comes with eight different templates, but there are only three types you need to worry about for now, as shown in Table 1-1.

Table 1-1. Main types of SharePoint sites

Template	Use to	Contains these lists
Team site	Create, organize, and share information among team members. This is usually the root site for a department or project team.	Document library, Announcements, Events, Contacts, Quick links
Document workspace	Collaborate on one or more documents. This is the template used when a shared workspace is created in Word, Excel, or PowerPoint.	Document library, Tasks, Links
Meeting workspace	Schedule and track an in-person meeting. This is the template used when Outlook creates a meeting workspace.	Objectives, Attendees, Agenda, Document library

Most groups or departments will have a team site as their main top-level site, then use document workspaces and meeting workspaces to organize projects and meetings within the group. Figure 1-7 shows the default sites created by the three main site templates.

SharePoint sites have three key navigation areas, as illustrated in Figure 1-8:

Quick launch area

Located on the left of page; links take members to content within the site. When you create a new list or library in a site, SharePoint asks whether to include it in Quick launch.

Links list

Located on the right of page; navigates down to document and meeting workspace sites within a team site. You can also add links to external sites or other locations in the Links list.

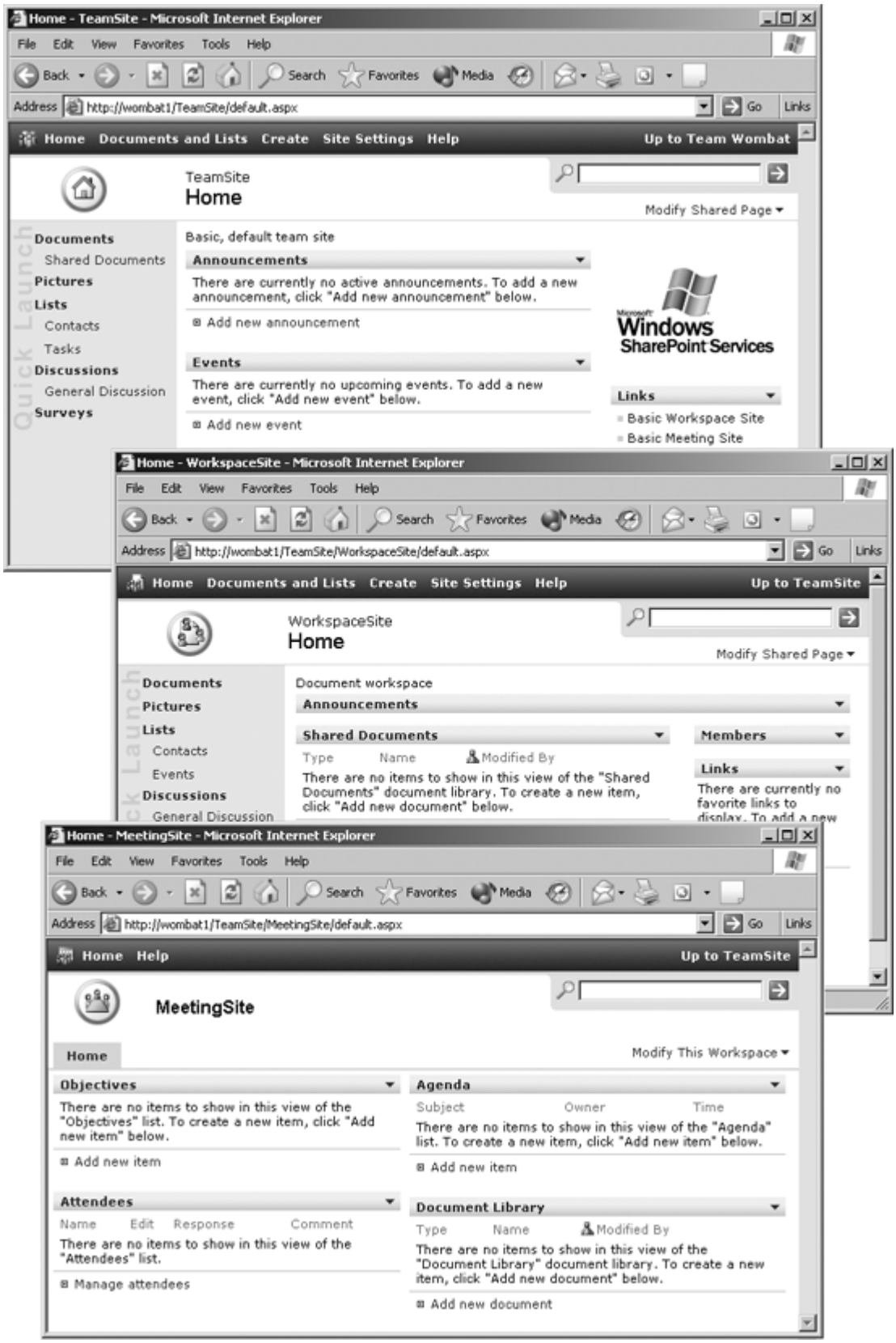
Navigation bar

Located at the top of page; navigates to the site's home page or to the team site from a document or meeting workspace. The navigation bar also includes Help and administrative links.

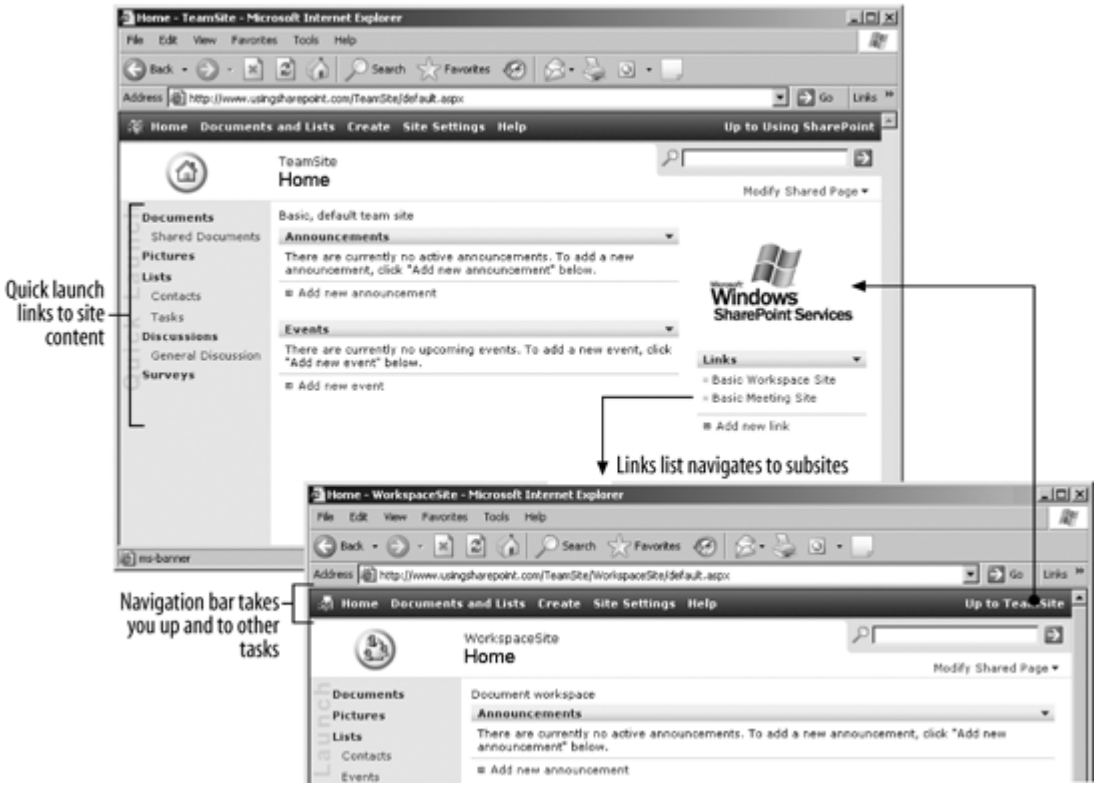
1.7. Parts of a Page

Each SharePoint site has a home page named *default.aspx*. Different site templates create different home pages, as shown in Figure 1-7. These pages are made up of web partsa *web part* is a type of custom control that contains a title bar, a frame, and content. Figure 1-9 indicates the web parts included in a document workspace.

animal 1-7. Team, workspace, and meeting sites



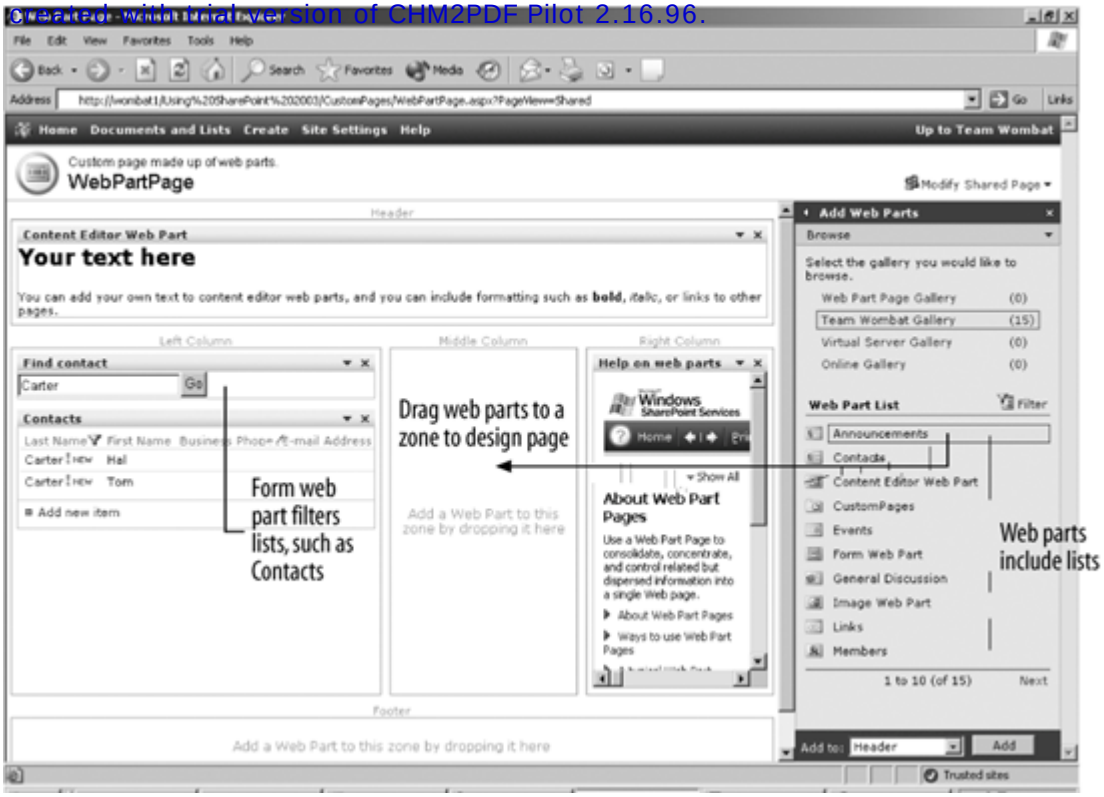
animal 1-8. Navigating between team and workspace sites



animal 1-9. Web parts on a home page

Web parts appear on a *web part page*, which is a special type of page that contains zones where you can drag and drop web parts. The home page of each site is a web part page. Web parts often present data from lists within the site. In fact, all of the lists in a site show up as web parts when you create a new web part page, as shown in Figure 1-10.

animal 1-10. Designing a web part page



I said that lists are tables of related data, but that doesn't really convey the importance of lists within SharePoint. Lists are used to organize and store Office documents, links, members, discussions, images, and all other types of content in SharePoint. For the most common tables, SharePoint includes a set of *list templates*, which are categorized and described by task in Table 1-2.

Table 1-2. Types of lists

Category	List template	Use to
Team projects	Announcements	Post messages on the home page of your site.
	Contacts	Share information about people your team works with.
	Document library	Share Word, Excel, and PowerPoint documents with the team.
	Events	Keep informed of upcoming meetings, deadlines, and other important events.
	Form library	Share InfoPath forms with the team.
	General discussion	Get answers and share opinions within the team.
	Links	Share links to web pages relevant to team members.
	Survey	Get quantifiable feedback from the team.
	Tasks	Track work that you or your team needs to complete.
	Picture library	Share images.
	Problem report	Track customer complaints and feature request. Assign items and record resolution.
Meetings	Agenda	Outline meeting topics, who will cover them, and how much time each presenter is allotted.
	Attendees	Invite members to the meeting.
	Decisions	Enter, review, and track decisions made at the meeting.
	Issues	Manage a prioritized set of problems meeting will address.
	Meeting series	Handle recurring meetings, such as monthly status reviews.
	Things to bring	List things that attendees should bring to be prepared for the meeting, such as notebooks, handouts, or lunch.

	Objectives	List goals for the meeting.
	Workspace pages	Link to other pages in a multipage meeting workspace.
Site administration	List template gallery	Add or remove custom list templates.
	Site template gallery	Add or remove custom site templates.
	Web part gallery	Add or remove custom web parts.

In addition to displaying lists, there are also web parts that perform specific tasks, as shown in Table 1-3.

Table 1-3. Other web parts included with SharePoint

Web part	Use to
Content editor	Display HTML text that can be edited within a frame.
Image	Include an image on a page.
Members	Display a list of site members and status.
Page viewer	Include a view of another web page in a frame.
Form	Filter a list displayed in another web part.
XML	Transform an XML data source using XSL and display the result in a frame on the page.

I explain how to construct new pages and modify existing ones in more detail in later chapters.

PREV NEXT

1.8. Where Are the Files?

If you install SharePoint, create a few sites, then search your server's file system, you might be surprised that you can't find the web pages you just created. That's because SharePoint stores content in a SQL Server database, not in conventional files and folders.

Using SQL Server gives SharePoint advantages in performance, security, and indexing (for site searches). Remember what I said about lists being tables? Tables...SQL...does it start to make sense?

1.9. Putting SharePoint to Work

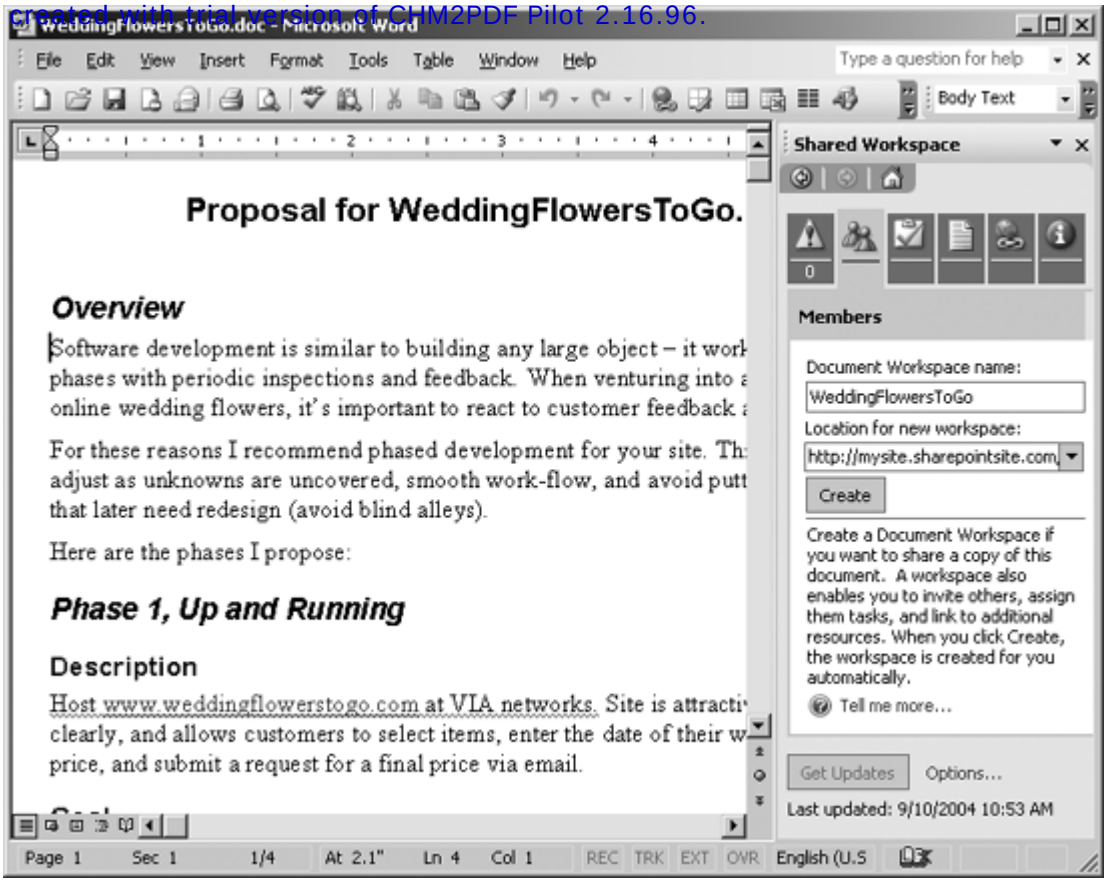
Once SharePoint is installed and your group or department has set up a team site, authorized members of the team can create document workspace sites from their desktops using Word, Excel, or PowerPoint. Members create document workspaces when they want to get input from others. Workspaces are intended for works-in-progress rather than for completed documents.

To create workspaces over the Internet from Office, you must first add the SharePoint site address to the Trusted Sites list in Internet Explorer. Choose Tools → Internet Options → Security → Trusted Sites → Sites. Be sure to clear the Require server verification (https:) checkbox before entering the address.

The 2003 versions of Word, Excel, and PowerPoint all include a Shared Workspace task pane, as shown in Figure 1-11, that you use to create and maintain workspaces. The following example illustrates how to use a workspace to develop a project:

1. Start Word 2003 and write a draft of the project plan.
2. Once you are happy with your first draft, save the document on your PC, then choose View → Task Pane → Shared Workspace.
3. On the Shared Workspace task pane, name the workspace, elect the location of the team site, and choose Create. SharePoint creates a new workspace site, adds the project plan to it, and links your local copy of the document to the copy on the SharePoint server.
4. Use the Shared Workspace task pane to add the names of the teammates you want input from; then choose Send email to all members to let them know a draft is ready for review.
5. Reviewers can open the draft from the SharePoint site, make comments or changes in the document, and then save those changes to share them with other members.
6. As the project moves forward, members can open the project plan document, then add other documents to the workspace using the Shared Workspace task pane as shown in Figure 1-12.

animal 1-11. Creating a new document workspace from Word



The Shared Workspace task pane can perform almost all of the tasks that can be done while viewing the SharePoint site in a browser, but you can always choose Open site in browser to view the shared workspace site as shown in Figure 1-13.

Sharing a document through a workspace allows multiple users to edit the document at the same time. Changes are resolved through the Shared Workspace task pane by merging or by comparing versions and accepting or rejecting selected changes.

1.10. Holding Meetings

Similarly, you can create meeting workspaces from within Outlook by following these steps:

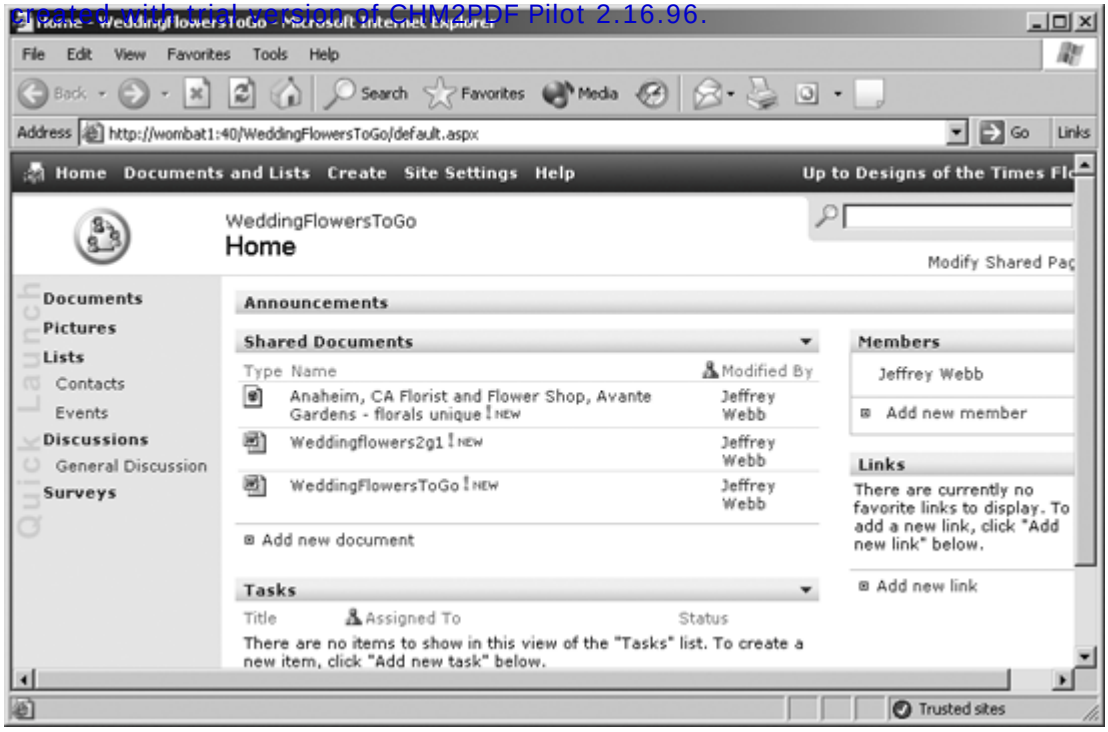
1. In the Outlook Calendar, select a date and time for the meeting and choose Actions → New Meeting Request or Actions → New Recurring Meeting. Outlook displays the Meeting dialog box.

2. Choose Meeting Workspace. Outlook displays the Meeting Workspace task pane as shown in Figure 1-14.

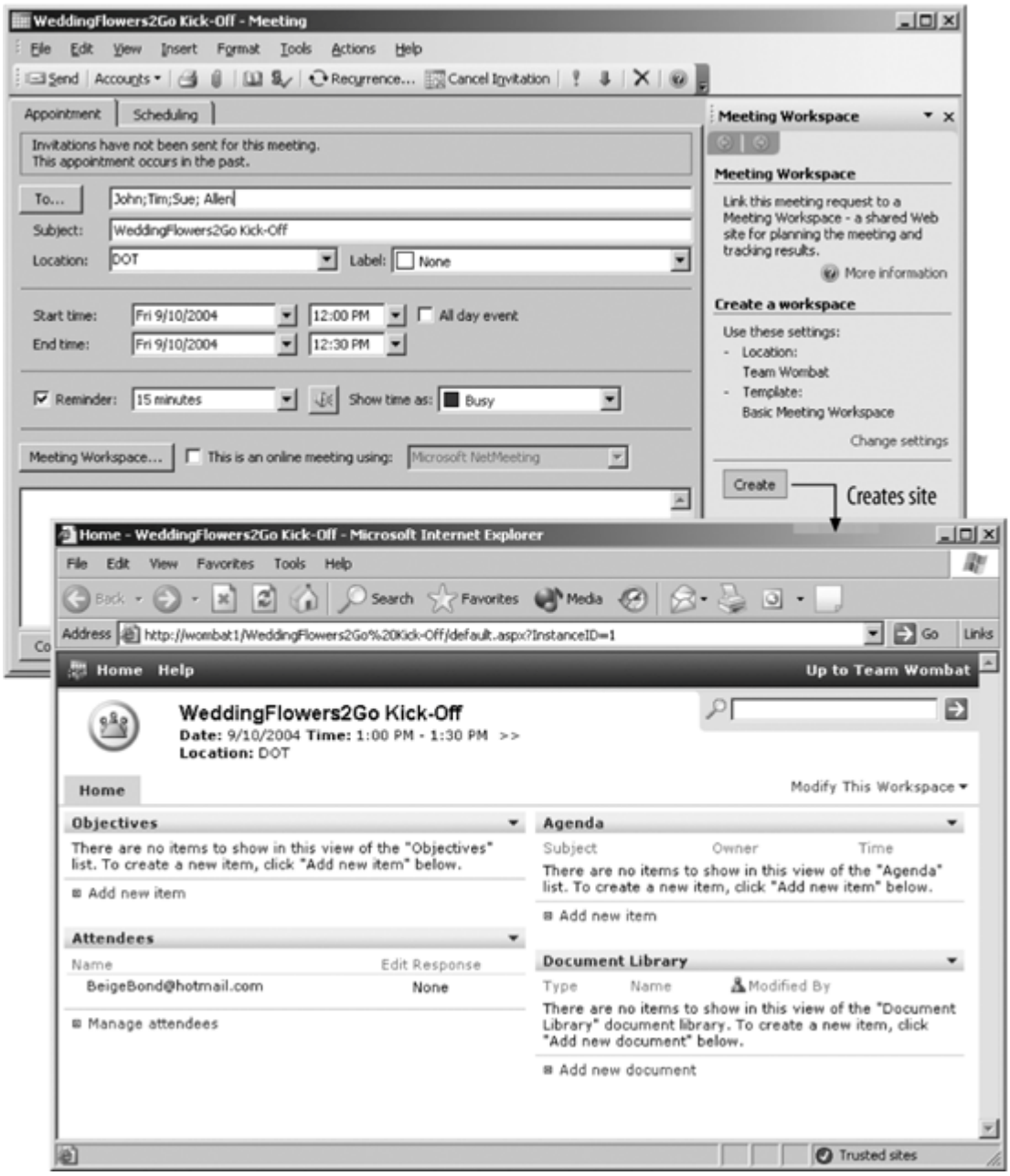
3. Choose Create in the task pane and Outlook creates a meeting workspace site in SharePoint.

animal 1-12. Adding documents to the workspace from Word

animal 1-13. Performing workspace tasks through the browser



animal 1-14. Creating meeting workspaces from within Outlook



Meeting workspaces aren't meant to host meetings online, but rather to provide a way to track objectives, agendas, and decisions of meetings held in person. For example, someone might take notes at the meeting, then enter those notes later in the meeting workspace to communicate the results. You could even do so using a laptop during the meetingif you're good at typing and talking at the same time.

PREV NEXT

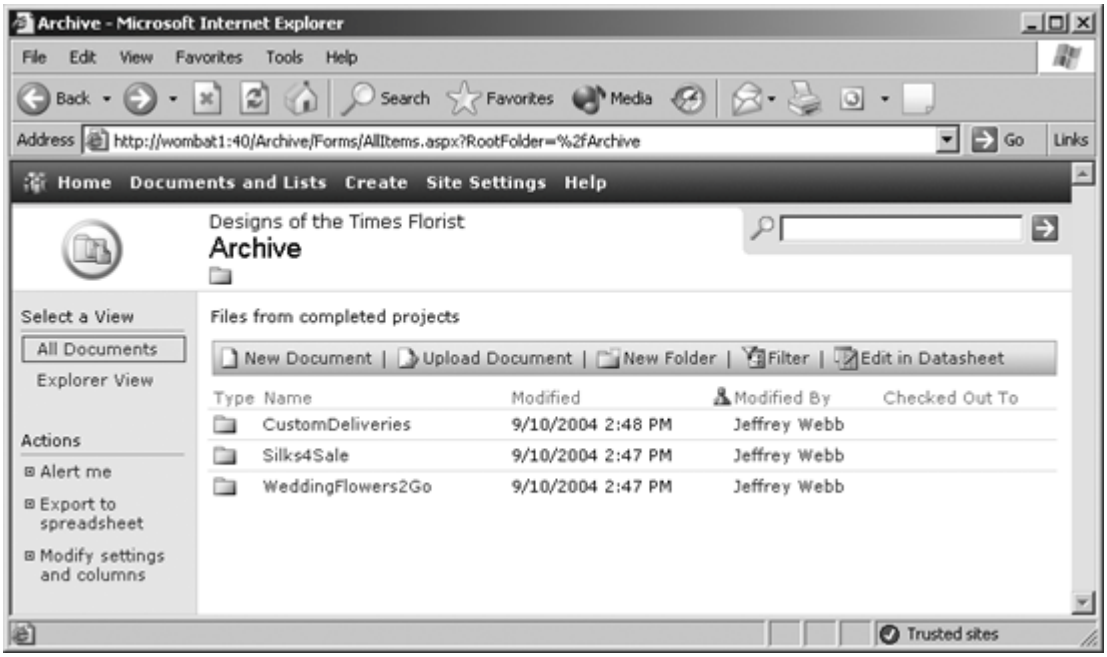
1.11. Building Libraries

Because the names are similar, it's easy to confuse document workspaces and document libraries, but they are very different.

A *document workspace* is a special type of subsite that contains works-in-progress. Workspaces often contain only one document (perhaps a large report or spreadsheet) that a team is working on. Document workspaces have special features that are covered more in Chapter 5.

A *document library* is used to store multiple documents within a site. Completed documents should be stored in document libraries that are part of the team site. Document libraries are a special type of list (not a type of site), and are covered extensively in Chapter 6. Libraries collect related documents and share them with all members of the site. Documents can be organized into folders within the library, allowing you to categorize documents as shown in Figure 1-15.

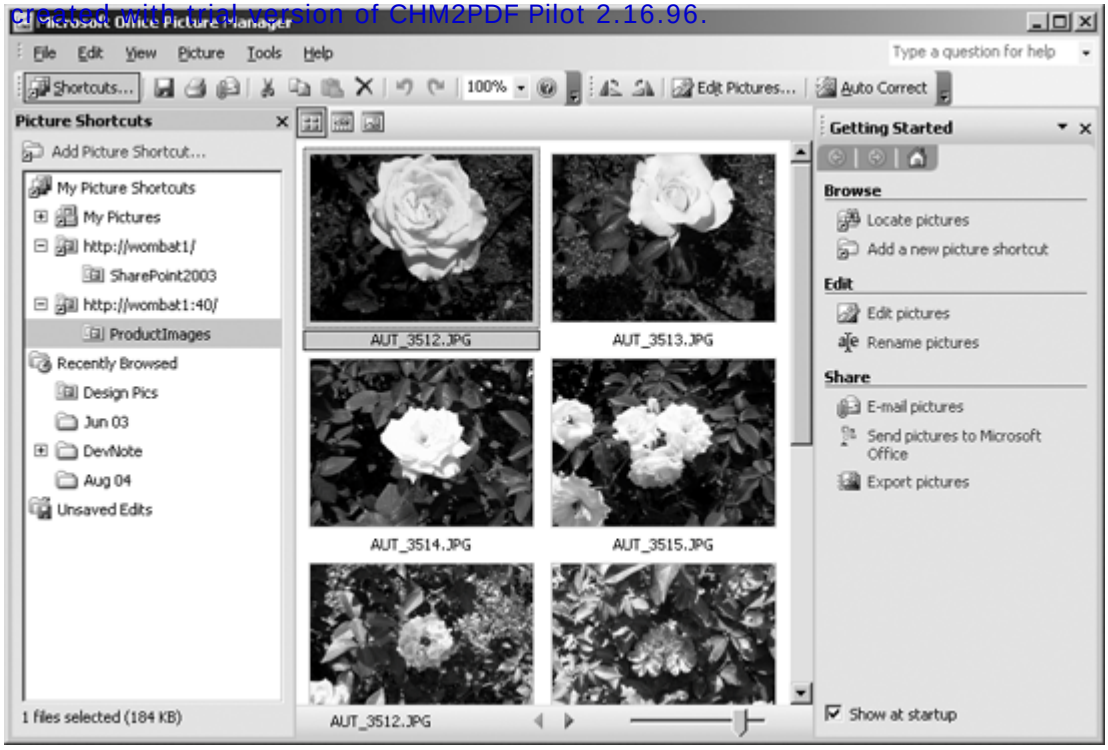
animal 1-15. Using document libraries to organize completed documents



Every document workspace site actually includes a document library for the work-in-progress, but because document workspaces are subsites of the team site, not everyone has access to those files. Storing completed documents in a library at the team site makes those documents more widely available.

You can't create document libraries from the Shared Workspace task paneyou do it by viewing the SharePoint site in a browser and choosing Create → Document Library and completing the New Document Library form. Of course, there's always an exception: you can create picture libraries using the task pane in Office Picture Manager, as shown in Figure 1-16.

animal 1-16. Creating and modifying picture libraries from Office Picture Manager



PREV NEXT

1.12. Creating Internet Sites

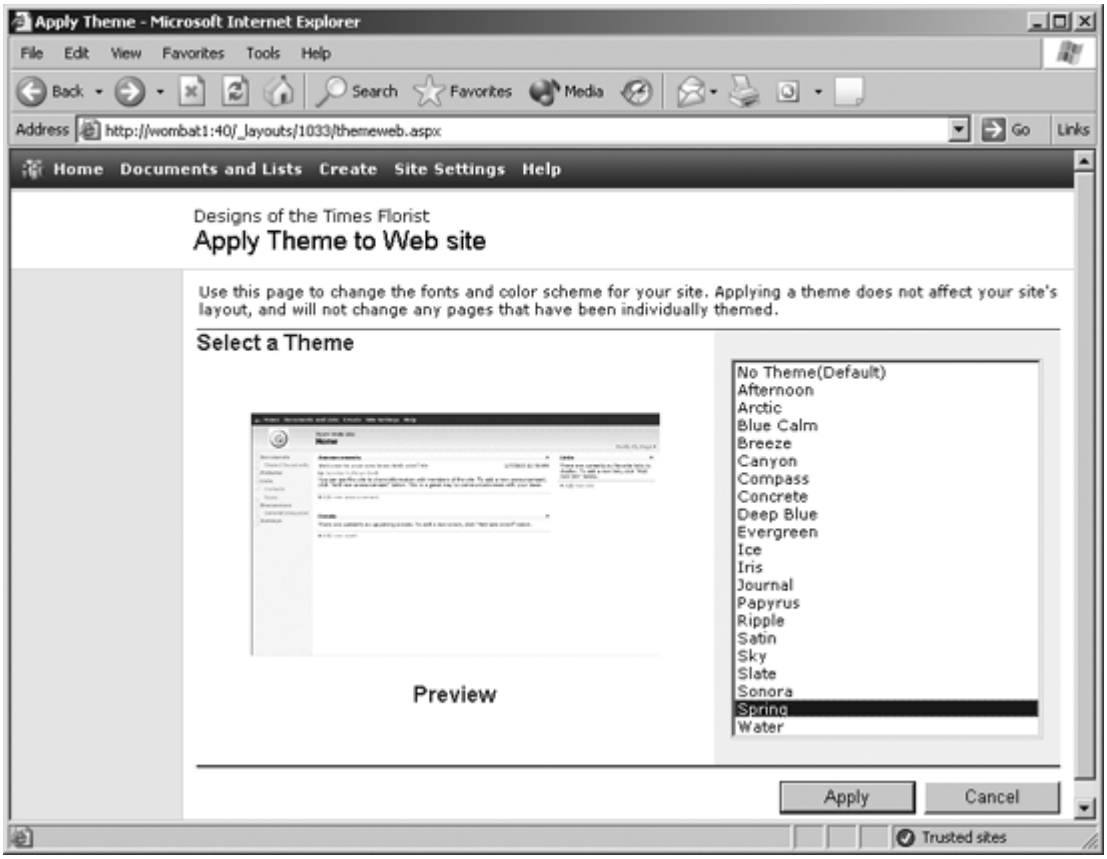
SharePoint's default site template is kind of boring-looking in my opinion. If you are creating a site for public view, you'll probably want to customize the look by applying a theme, modifying the home page, and/or adding your own pages to the site.

Themes control the color scheme of a site. To choose a theme for a SharePoint site, choose Site Settings → Apply a Theme. SharePoint lets you choose from a list of themes and preview the result, as shown in Figure 1-17.

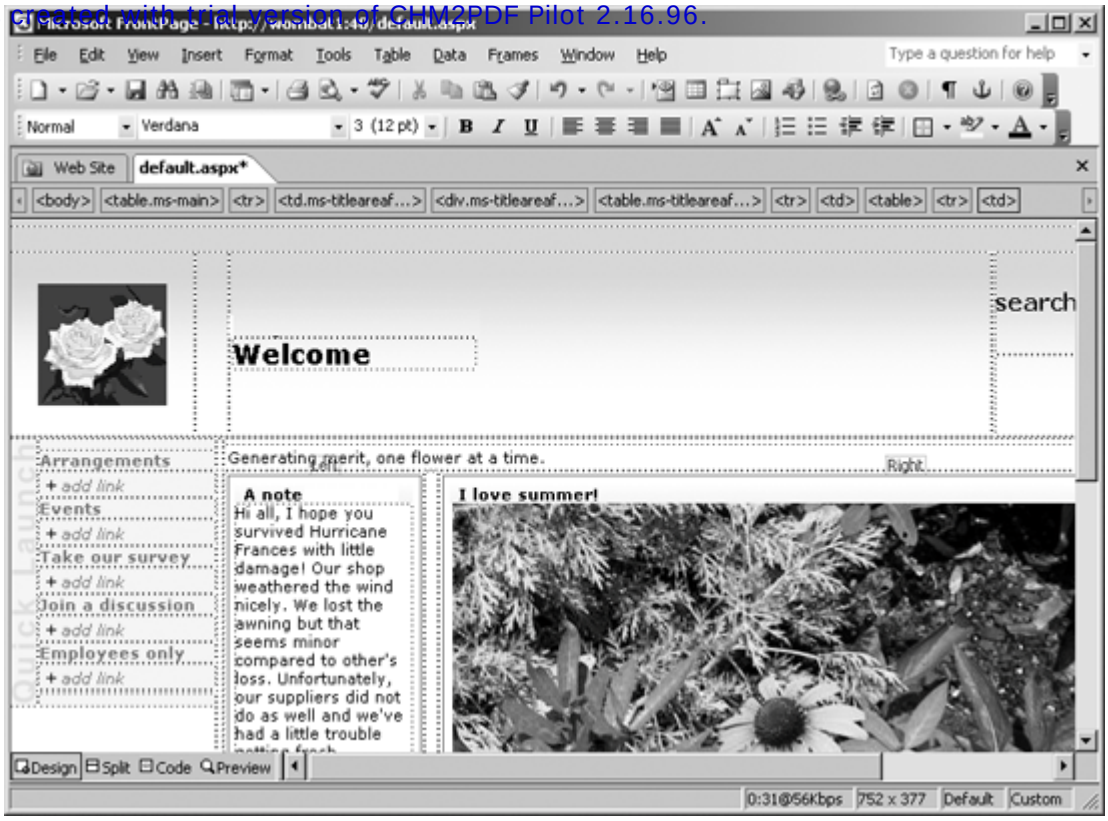
You can modify any SharePoint page by choosing Modify Shared Page → Design This Page, but you have more control over changes if you open the site in FrontPage 2003. FrontPage can open SharePoint sites right from their server location, make changes, and save the new pages right back to SharePointyou will never even notice you're writing to a database (see Figure 1-18).

Finally, you can upload pages from existing web sites using FrontPage or by creating a document library in SharePoint. Document libraries can contain any type of content, including HTML and ASPX pages.

animal 1-17. Use themes to improve the look of a site



animal 1-18. Using FrontPage to edit SharePoint pages



PREV NEXT

1.13. What Software Do You Need?

Because SharePoint is designed to be used by different members with different responsibilities, the software needs for each team member vary depending on their role, as shown in Table 1-4.

Table 1-4. Software needs for different team members

Role	Responsibilities	Software needed
Administrator	Creates team sites, adds members.	Web browser, Remote Desktop Connection utility included with Windows XP Professional (recommended)
Designer	Designs pages on site, creates site templates.	FrontPage 2003
Contributor	Adds documents, lists, tasks, announcements, etc.	Office 2003
Contributor with data entry	Completes InfoPath forms.	Office 2003, InfoPath 2003
Reader	Views sites and documents, but makes no changes.	Office Reader
Programmer	Creates custom web parts.	Visual Studio .NET, FrontPage 2003, InfoPath (optional)

Where's the server software? You don't need actual *physical* access to a SharePoint server to create or manage SharePoint sites, because such tasks can be done remotely from your own PC by viewing one of the administrative web pages in the browser. However, you do need access to a SharePoint server, and here, listed from least to most expensive, are your access options:

- You can purchase SharePoint services on a monthly basis from an Internet Service Provider (ISP). This is a simple and inexpensive way to try SharePointin fact, Apptix offers a free 30-day trial at http://www.sharepointtrial.com/welcome.aspx.
- You can purchase a Windows 2003 server. SharePoint is part of Windows 2003, so you can configure the machine as a SharePoint server using the Windows setup procedure.
- You can purchase multiple servers and buy a SharePoint Portal Server license from Microsoft if you plan to create an enterprise-wide web portal based on SharePoint.

Table 1-5 lists the relative advantages of each approach.

Table 1-5. Different ways to purchase SharePoint

Buy SharePoint as	Advantage	Disadvantage
Hosted site	Inexpensive, simple, quick.	Access is over Internet; user accounts must be added manually.
Windows 2003 service	One-time cost, integrates with network and user accounts.	Requires some installation and maintenance.
SharePoint Portal Services	Can provide a single interface to all your corporate resources.	More expensive than other options; configuring multiple servers is complicated.

SharePoint Portal Server lets you integrate multiple sites across physical servers and provides single-sign-on access through the web portal to corporate databases and other secured resources. However, the price of Portal Server ranges from about $4,000 to $30,000 per physical server, depending on the number of users.

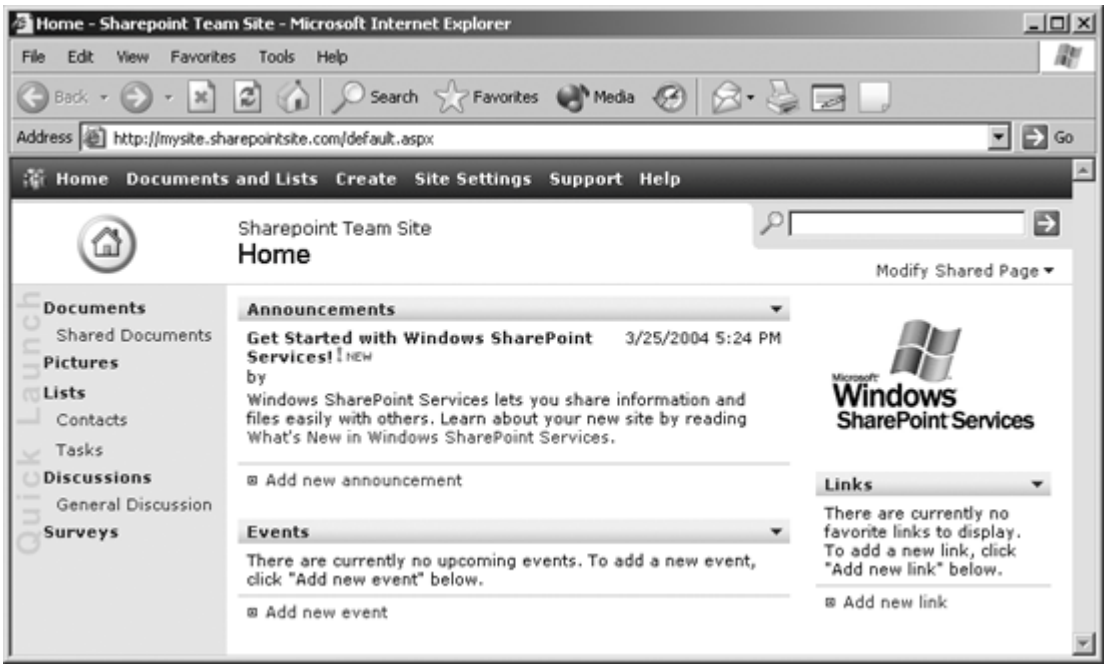
In contrast, SharePoint Services is part of Windows 2003, and so is included under your server license. SharePoint Services is a subset of Portal Server, so I'll focus most of my attention on using SharePoint Services.

1.14. Trying Out SharePoint

I recommend using the 30-day free trial of SharePoint before installing it on Windows 2003. There are three good reasons for this: it's free, it's painless, and it's quick. Then, if you like it go ahead and put it on your server.

To get a free SharePoint site, go to http://www.sharepointtrial.com/default.aspx. Follow the signup instructions and wait for email confirming your site has been created. If everything goes correctly, you'll get a new SharePoint site that looks something like Figure 1-19.

animal 1-19. Creating a free trial site before installing SharePoint



You can then experiment with sharing workspaces and lists, or you can just mess around with your new SharePoint site for a while, changing the home page, adding users (choose Site Settings → Manage Users), adding a welcome announcement (choose Add new announcement), or posting your vacation pictures (choose Pictures → Create Picture Library). Get a feel for what it can do.

SharePoint is a foundation for building sites that solve specific problems. The general site types Microsoft includes (team sites and workspaces) are a starting point, not an end point. If you have more specific needs, look for SharePoint add-ons (sometimes called *accelerators*). Some accelerators, such as workflow management systems, are already available from multiple vendors. See the "Resources" section for a link to a set of Microsoft accelerators you can try for free.

1.15. What's SharePoint Not Good For?

SharePoint is an excellent way to create data-driven web sites, in my opinion, but there are other times where I don't think it's the best choice. For example, SharePoint isn't a substitute for a code management library, such as SourceSafe. (Microsoft is working on this for Visual Studio 2005, however, so watch out!) Also, SharePoint's integration with Microsoft Office, Windows, and .NET means that users of other operating platforms (Mac, Linux) or non-Microsoft browsers may have problems using the sites.

In particular, SharePoint authentication does not seem to work with Internet Explorer for the Macintosh or Mozilla prior to Version 1.7.2. In addition, SharePoint pages look different in FireFox, Mozilla, and Netscape Navigator than they look in Internet Explorer. You can compare these differences by using different browsers to view public SharePoint sites such as the last two listed in "Resources." (Users of these other platforms can still get to the files, but they can't really take advantage of SharePoint's integration and management features.)

1.16. Resources

To get	Look here
Free trial versions of SharePoint	http://www.microsoft.com/sharepoint/
SharePoint Portal Server pricing	http://www.microsoft.com/office/sharepoint/howtobuy/default.mspx
Online SharePoint tutorial	http://www.usingsharepoint.com/SPTraining/
A great SharePoint FAQ	http://wss.collutions.com/
A great SharePoint demo site	http://www.wssdemo.com/
SharePoint accelerators	http://www.microsoft.com/office/solutions/accelerators/

Chapter 2. Getting Started

There are several ways to get SharePoint Services. The biggest choice is whether to host SharePoint on your own server or whether to buy the services from a hosting provider. The primary advantage of hosted services is that you don't need to wrangle with installing and maintaining SharePoint yourself. The main disadvantage is that you lose some of the flexibility and control you get from using your own server.

In this chapter, I show you how to get started using a hosted site; then I explain how to install SharePoint on your own server. These two tasks build on each other: if you create a hosted site first, you'll be better prepared when you install SharePoint. Even if you don't intend to host your own site, you can still develop some advanced skills by completing the chapter.

2.1. Before You Begin

It's easy to create new web sites using SharePointin fact, perhaps it's *too* easy. Before you begin, it's important to understand how SharePoint structures sites and how those structures affect what you can do later.

The simplest site structure is a single top-level site located on a single server. In that scenario, members of the site have one set of permissions, and those permissions determine which lists and libraries they can see, as well as what actions they can take on those lists and libraries.

Lists and libraries in this simple site are stored in subfolders (for example, http://www.mysite.com/Lists/Announcements), but those subfolders exist within the site's boundaries. The concept of *site boundaries* is important for the following reasons:

- Member permissions may or may not be inherited across site boundaries.
- Lists and libraries cannot be easily shared across site boundaries.
- Searches do not cross site boundaries (unless you purchase SharePoint Portal Server, which provides cross-site searching).

Notice that I haven't mentioned workspaces. That's because workspaces are a type of site and so establish new boundaries.

Let's extend the simple scenario above: you open a document in Word and choose Tools → Shared Workspace to share the document on http://www.mysite.com. SharePoint creates a new document workspace site and copies that document up to the workspace.

The new workspace site is named after the document (http://www.mysite.com/ProjectPlan, for example) and has only one member: you. That's because new workspaces use unique permissions, and by default you have to add members manually. To change the default permissions so that all members from the parent site automatically have access:

1. Open the workspace in the browser.
2. Choose Site Settings → Go to Site Administration → Manage Permission Inheritance, select Use the same permissions as the parent site, and click OK.
3. Close the Word document; open it again in Word; the Shared Workspace task pane will include all the members inherited from the parent site.

This is still a very simple structure: a root site with one subsite (the workspace) and members have access to both sites. However, you can quickly go wild, adding subsites and workspaces to create very deep structures (for example: http://www.mysite.com/ProjectPlan/NewSite/Subsite1/Subsite2/etc).

Don't do that! Creating deep, nested structures makes Search almost useless, and it prevents you from sharing content effectively. Instead, try to follow these guidelines:

- Use sites to manage membership, not content.
- Create broad, rather than deep, site structures.
- Use document libraries or lists to store different categories of content.
- Limit workspaces to one level deep (never nest workspaces).

The wisdom of these recommendations will be revealed in the following chapters. For now, please take them on faith, but also remember that they are recommendations, not commandments.

2.2. Creating Hosted Sites

Hosted solutions are a great way to try out SharePoint without committing a lot of resources up front. Table 2-1 lists some of the hosting providers currently available; several offer a free trial period.

Table 2-1. Some SharePoint hosting providers

Company	Site	Provides
Apptix	http://www.apptix.net/default.aspx	Free 30-day trial
ASP-One, Inc.	http://www.asp-one.com/default.asp	Combined Microsoft Exchange/SharePoint Server hosting
Enlightened Technology Group, Inc.	http://www.etgroup.net/pages/sharepoint_hosting.aspx	SharePoint Server hosting at different service levels
FrontPages Web Hosting Network	http://www.frontpages-web-hosting.net/sharepoint.hosting.htm	SharePoint and other types of hosting
724 Hosting	http://www.724hosting.com/sharepoint/	Free 30-day trial

Setting up a hosted account is as easy as navigating to one of the sites in Table 2-1 and completing the registration process. After you've done that, you'll want to:

- Add members to the site.
- Customize the site's home page.
- Add content to the site.

The following sections demonstrate those tasks using a hosted site; they provide a quick tutorial, so you may want to follow along using your own trial site to get a better understanding of SharePoint.

2.3. Adding Members

To add members to a SharePoint site:

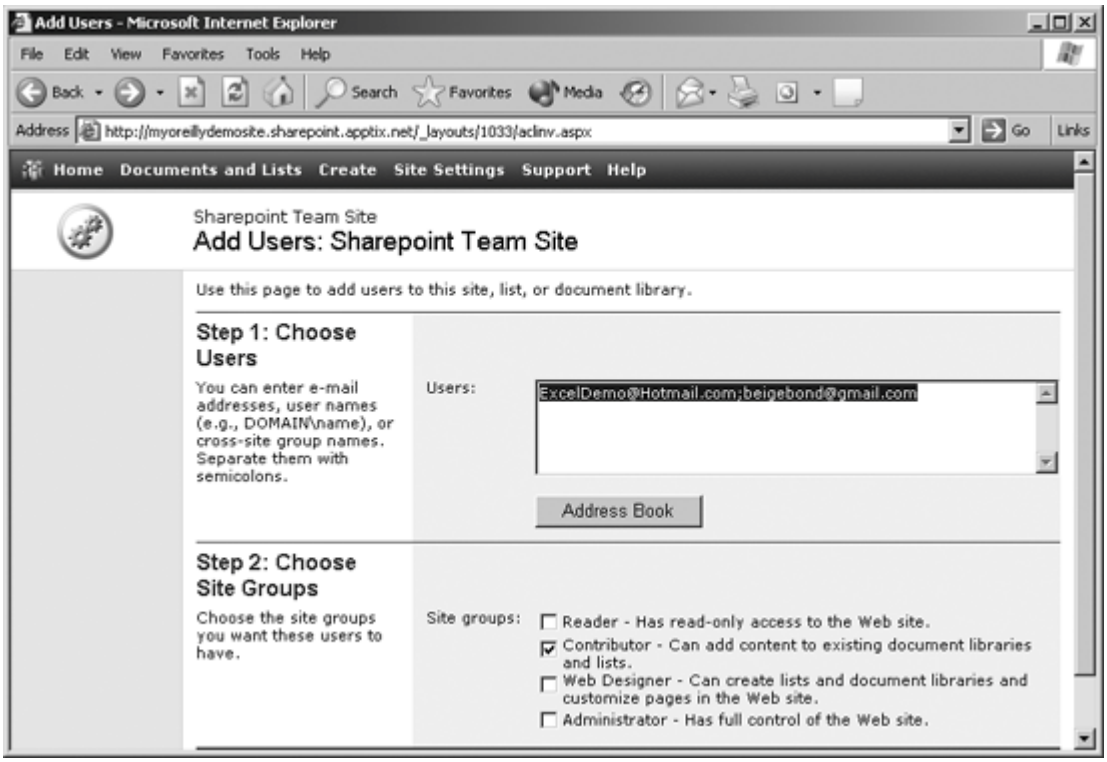
1. From the home page navigation bar, click Site Settings → Manage users → Add users. The results are displayed in Figure 2-1.

2. Enter the usernames to add and select a group for the new members. Groups determine what privileges members have on the site as listed in Table 2-2. When you click Next, SharePoint displays Figure 2-2.

3. Type a greeting to send the new SharePoint members and click Finish to complete the task. New members will receive email including their user name and an automatically generated password for the site.

Adding members by site group lets you add new members in batches, rather than one at a time. Table 2-2 lists SharePoint's built-in groups in ascending order of access privileges.

animal 2-1. Adding names for members and selecting their group



animal 2-2. Sending email to alert new members

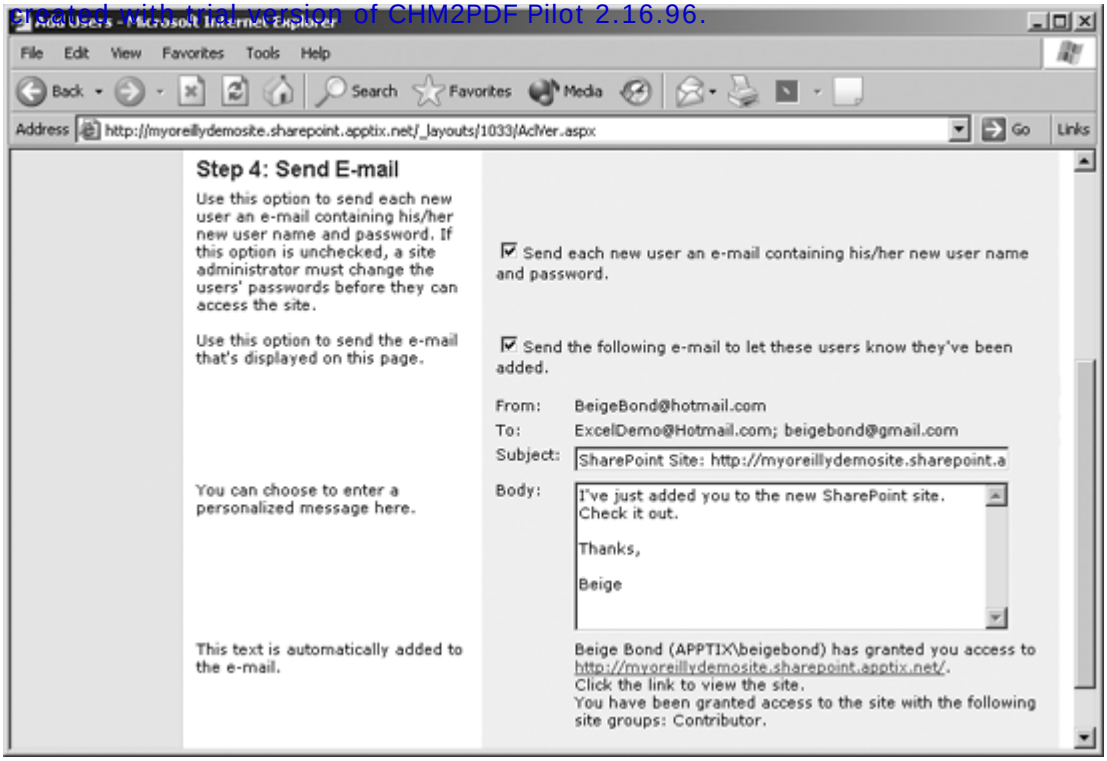


Table 2-2. Built-in SharePoint member groups

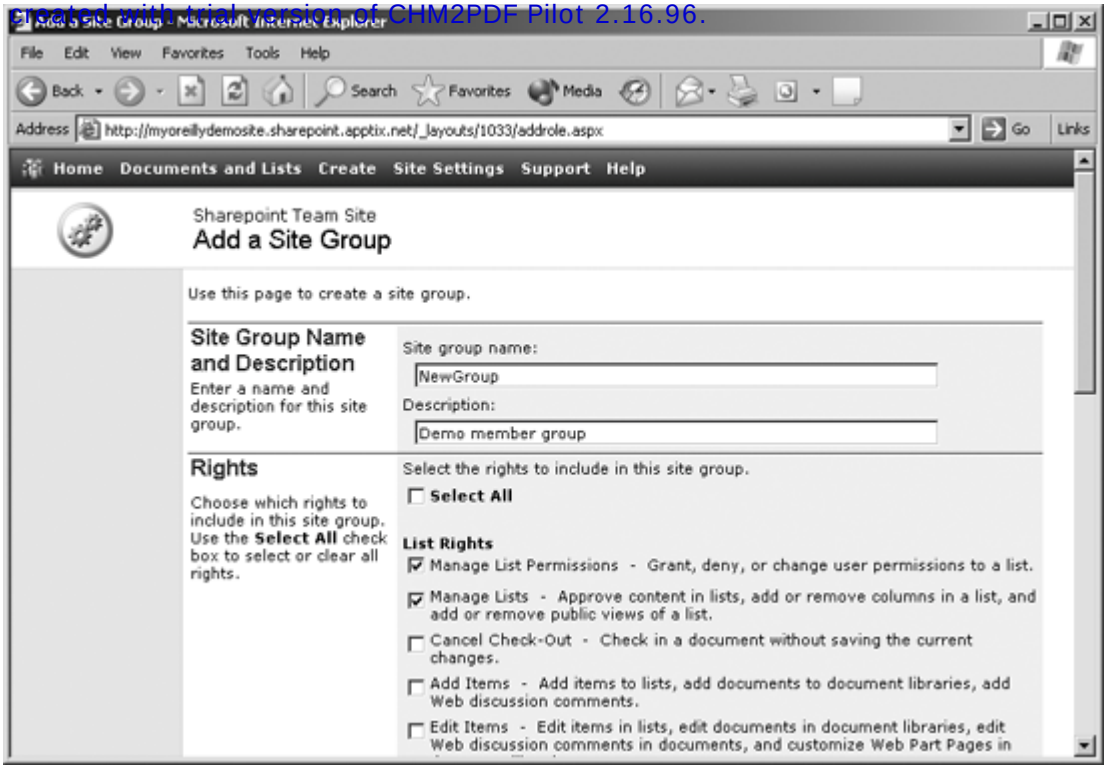
Group	Allows
Guest	No access by default, but can be granted access to specific items.
Reader	Read-only access to the site.
Contributor	Add content to existing document libraries and lists.
Web designer	Create lists and document libraries and customize pages in the site.
Administrator	Full control of the site.

As the site's creator, you are a member of the Administrator group. The Guest group doesn't show up in Figure 2-1 because it is only used when granting access to specific lists or document libraries in the site. You can also create new custom groups or modify the built-in groups for your site. To create a new group:

1. From the home page navigation bar, click Site Settings → Go to site administration → Manage site groups → Add a site group. SharePoint displays Figure 2-3.

2. Enter a name and description for the group, select the rights to grant members of the group, and click Create Site Group to complete the task.

animal 2-3. Creating new member groups



Associating members with groups containing specific rights is similar to how Windows security uses groups to grant privileges to users. This technique is sometimes called *role-based security*. Table 2-3 lists the rights assigned to each of the built-in groups by default.

Table 2-3. Rights for built-in groups

Group	Right
Guest	None (set at list level)
Reader	Use Self-Service Site Creation View items View pages
Contributor	All Reader rights, plus: Add items Add/remove private web parts Browse directories Create cross-site groups Delete items Edit items Manage personal views Update personal web parts
Web designer	All Contributor rights, plus: Manage lists Add and customize pages

	Apply style sheets Apply themes and borders Cancel check-out
Administrator	All Web designer rights, plus: Create subsites Manage list permissions Manage site groups Manage web site View usage data

When a new member visits the SharePoint site, he is prompted for his user name and password as shown in Figure 2-4. If he tries to perform a task beyond his rights, SharePoint displays the dialog again so he can sign on as a different member with more rights.

animal 2-4. Members are prompted for user name and password

2.4. Changing Pages

Administrators and web designers can change the appearance of pages on the SharePoint site by applying themes, designing an existing page, or by using FrontPage. Table 2-4 summarizes those different techniques.

Table 2-4. Ways to change SharePoint pages

Use	To
Themes	Change the fonts and color scheme for all pages on a site.
Design a page	Add, move, or delete web parts on a page through the browser.
FrontPage 2003	Significantly change the content or appearance of a page, or create new pages on the site.

To apply a theme, click Site Settings → Apply Theme to site, and choose a theme. Themes affect only the current site: subsites and workspaces subordinate to the site are unaffected.

To design a page, click Modify Shared Page → Design this page in the upper right-hand corner (Figure 2-5). SharePoint displays the page in edit mode, showing the web part zones. You can drag web parts from one zone to another, or you can select Modify shared web parts to change a web part Changing a page in this way is handy for minor changes such as changing the text in a title bar of a web part.

animal 2-5. Changing a page through the browser

For example, to change the "Announcements" titlebar shown in Figure 2-5 to "Quote of the day":

1. Click Modify Shared Page → Modify Shared Web Parts → Announcements. SharePoint displays the Announcements web part in edit mode.

2. In the task pane, click Appearance and change the title to "Quote of the day"; then click OK. SharePoint makes the change.

Some elements on the home page can't be changed through the browser. The navigation bar, Quick Launch, page title, and search box aren static web parts. The easiest way to change those items is by editing the page

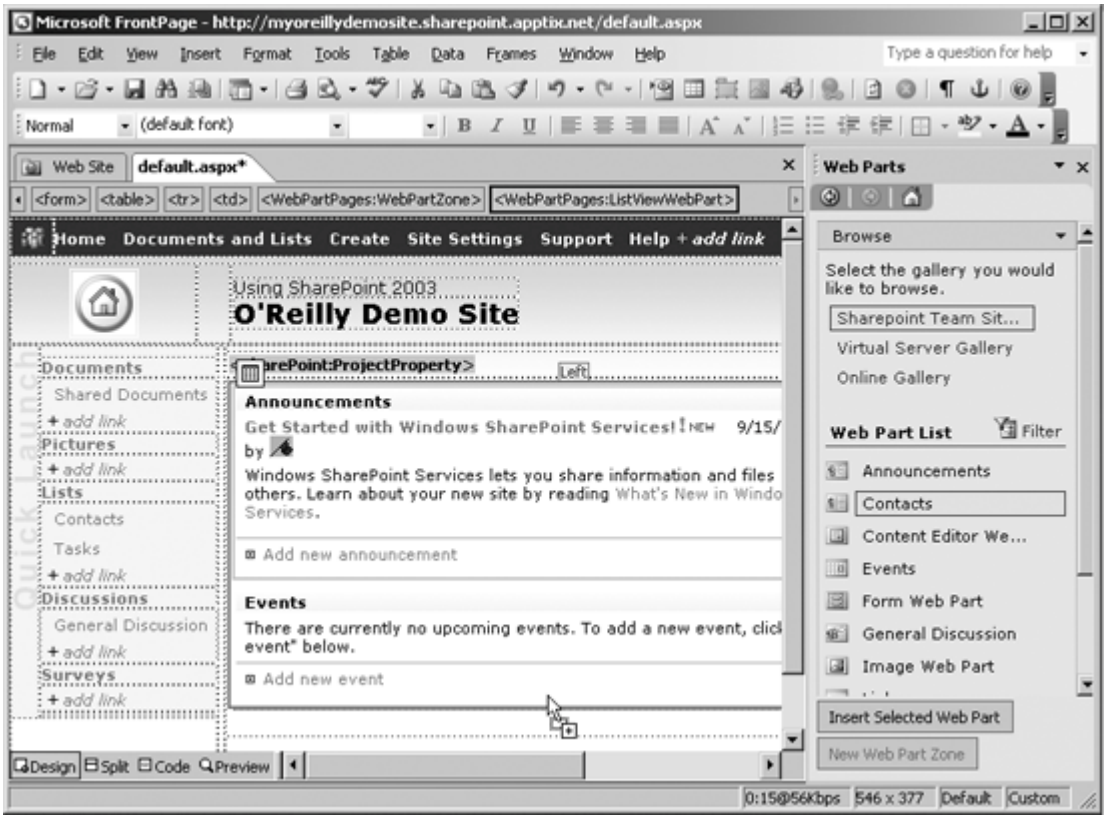
in FrontPage.

To open a SharePoint web site from FrontPage, choose File → Open Site and enter the address of the site to open. For example, http://myoreillydemosite.sharepoint.apptix.net/.

With FrontPage, you can directly edit text on the SharePoint home page, add or remove links, and edit web parts, although the web part tasks are a little hard to locate (they are found on the Data menu). For example, to add a web part to the SharePoint page in FrontPage:

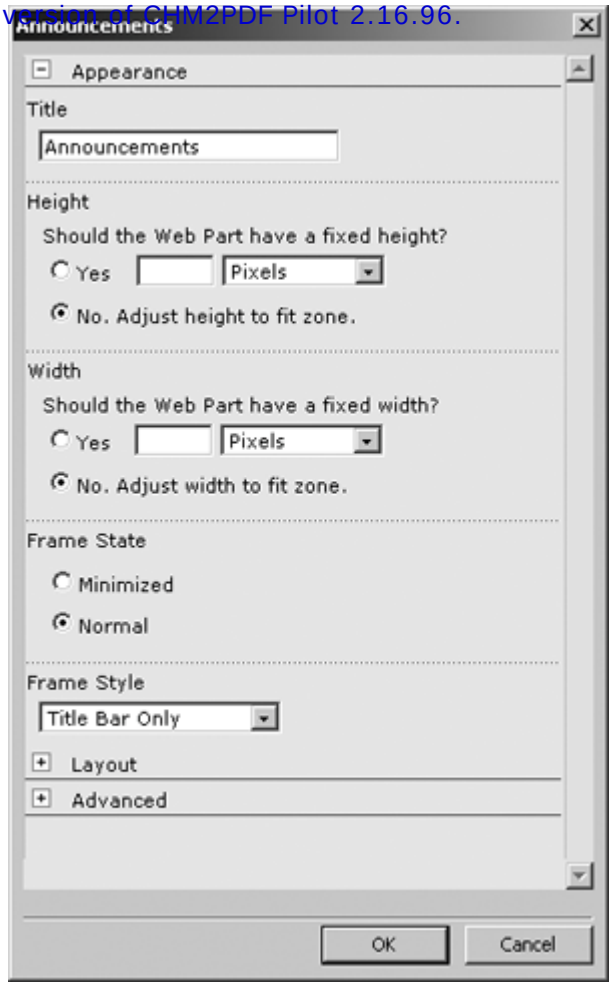
1. Choose Data → Add Web Part. FrontPage displays the Web Parts task pane.
2. Drag web parts from the task pane to web part zones on the SharePoint page as shown in Figure 2-6.

animal 2-6. Use the Data menu to get at web part tasks in FrontPage



To edit a web part in FrontPage, right-click the web part and select Web Part Properties. FrontPage displays the part's properties in a new window as shown in Figure 2-7.

animal 2-7. Editing a web part in FrontPage



2.5. Adding Content

There are many ways to add content to a SharePoint site depending on the type of content you want to add. The following sections describe different ways to add lists, libraries, pages, and workspaces to a SharePoint site and explain when to use each approach.

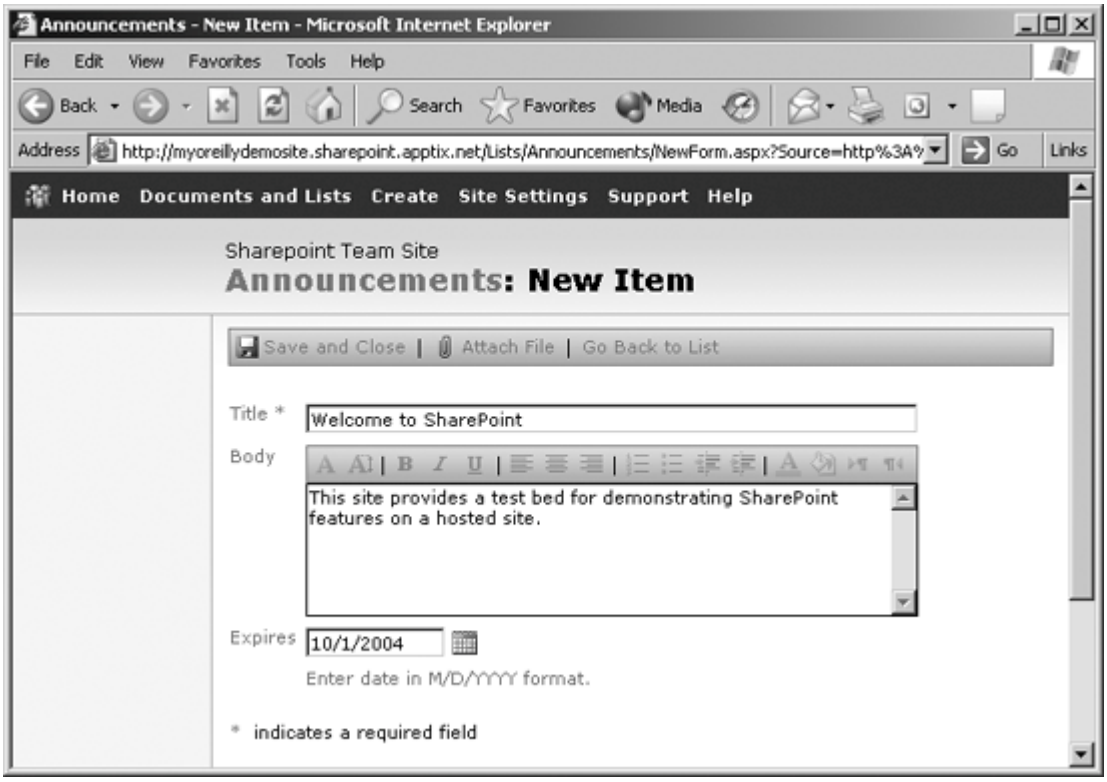
2.5.1. Adding Lists

At the simplest level, you can add content to the home page by clicking Add new announcement, Add new event, or Add new link. Clicking on any of these displays a form view for adding items to the SharePoint list, as shown in Figure 2-8. When you click Save and Close, SharePoint adds the announcement to the list which appears on the home page in the Announcements web part.

To create your own lists from the browser:

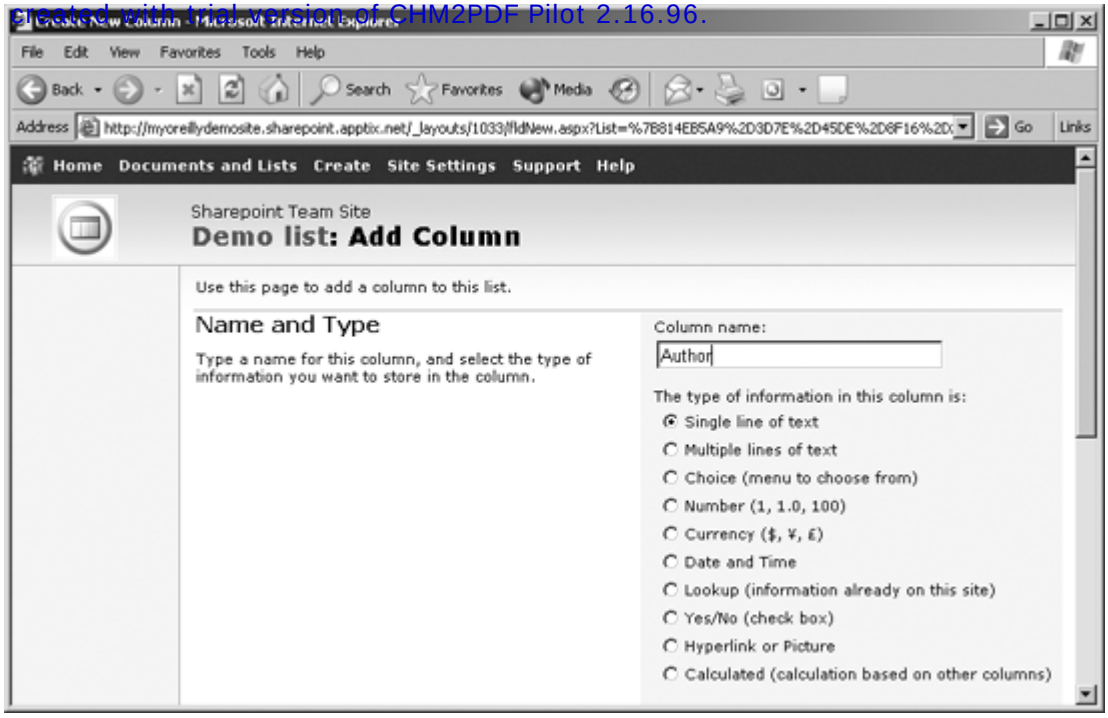
1. Click Documents and Lists → Create → Custom List. SharePoint displays the New List page.
2. Enter the list information and click Create. SharePoint displays the new list in its default view.

animal 2-8. Adding items to the Announcements list



3. Click Modify settings and columns → Add new column to add columns to the list. SharePoint displays Figure 2-9.
4. Add column information and click OK.
5. Repeat steps 3 and 4 for each column in the list.

animal 2-9. Adding columns to a new list



To display the new list on the home page:

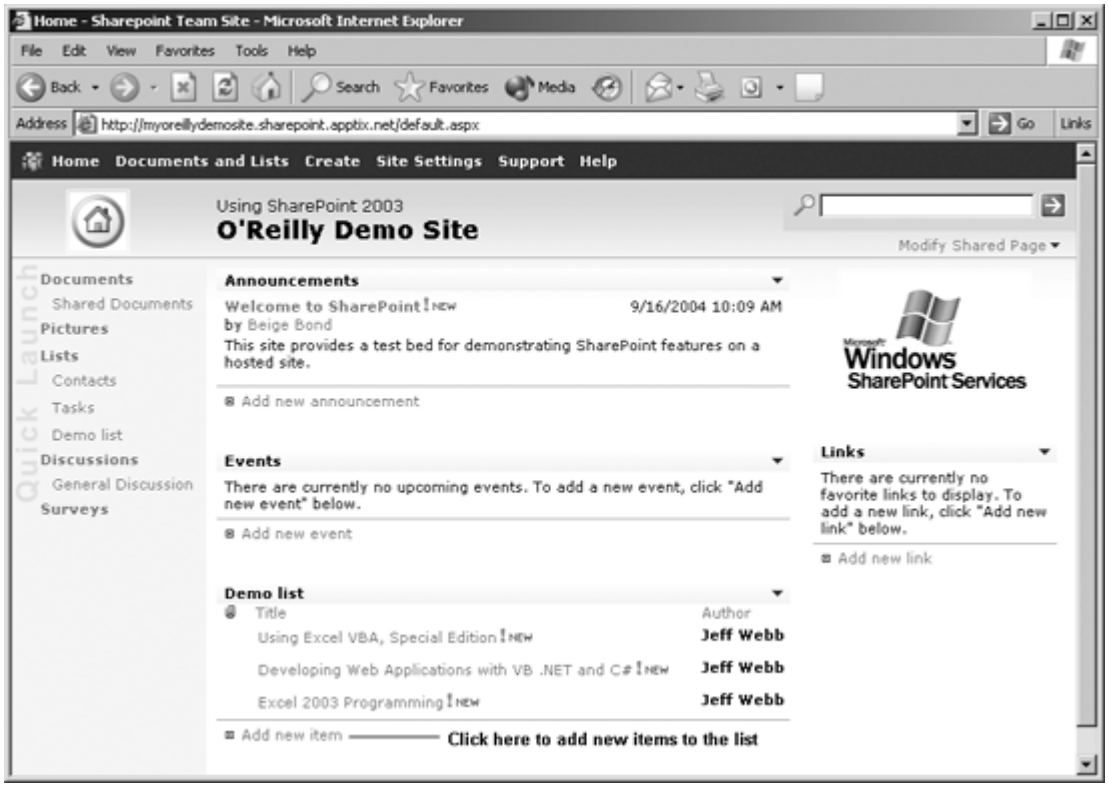
1. From the home page, click Modify Shared Page ⟶ Add Web Parts ⟶ Browse. SharePoint displays the page in edit mode and shows the new list as a web part in the task pane.

2. Drag the new list from the task pane to a web part zone on the home page.

3. Click Modify Shared Page ⟶ Modify Shared Web Parts ⟶ list name to change the properties of the displayed list. For example, select Toolbar Type ⟶ Summary toolbar and click OK.

4. You can now add items to the list by clicking Add new item on the list's web part as shown in Figure 2-10.

animal 2-10. Adding a new list to the home page as a web part



To close a web part on the home page:

1. Choose Modify Shared Page → Modify Shared Web Parts → web part name. SharePoint selects the list and places a dashed border around it (Figure 2-11).

2. Click the X in the upper right-hand corner to close the web part.

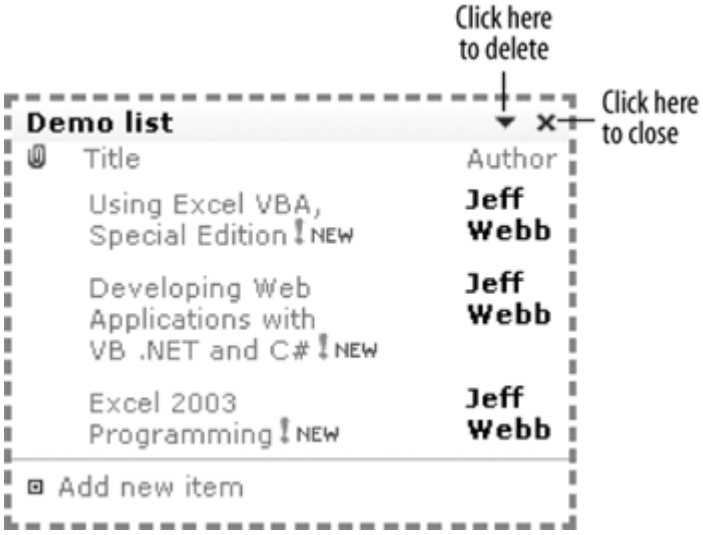
Closing hides the web part, but does not remove it from the page. Instead, the web part is moved to the Web Part Page Gallery in the web parts task pane. You can drag web parts from the Web Part List back onto the page to restore them.

To delete a web part from a page:

1. Choose Modify Shared Page → Modify Shared Web Parts → web part name or choose Modify Shared Page → Add Web Parts → Browse.

2. Click the down-arrow in the upper right-hand corner of the web part and choose Delete.

animal 2-11. Closing or deleting a web part



2.5.2. Building Libraries

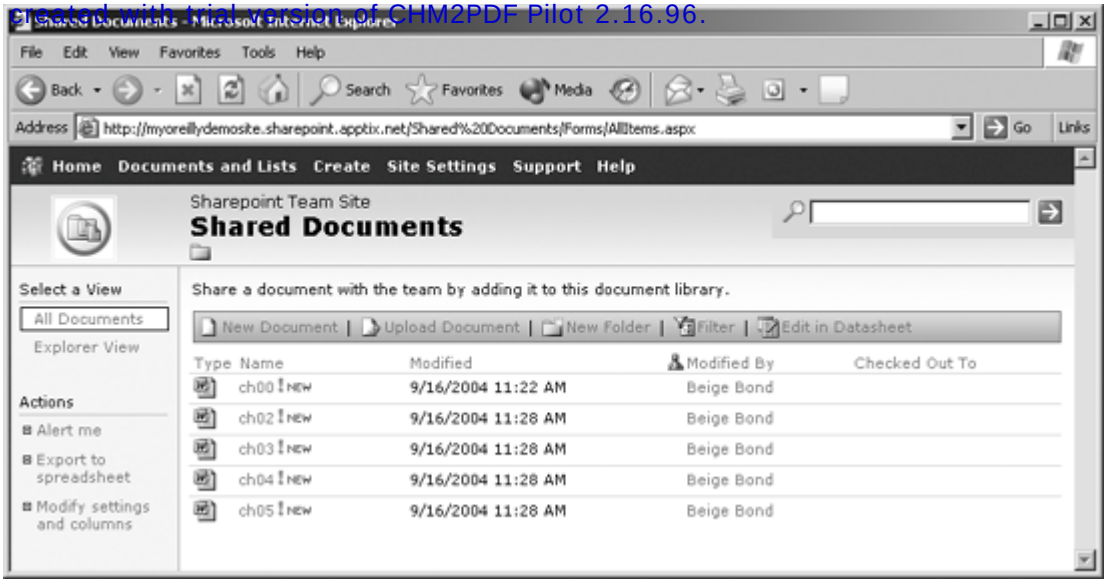
A *library* is a type of list that includes documents and version histories. Team sites include one built-in library named Shared Documents. To add documents to that library:

1. From the home page, click Shared Documents → Upload document.

2. Click Browse to upload documents one at a time or Upload multiple files to upload a group of files from a single folder on your computer.

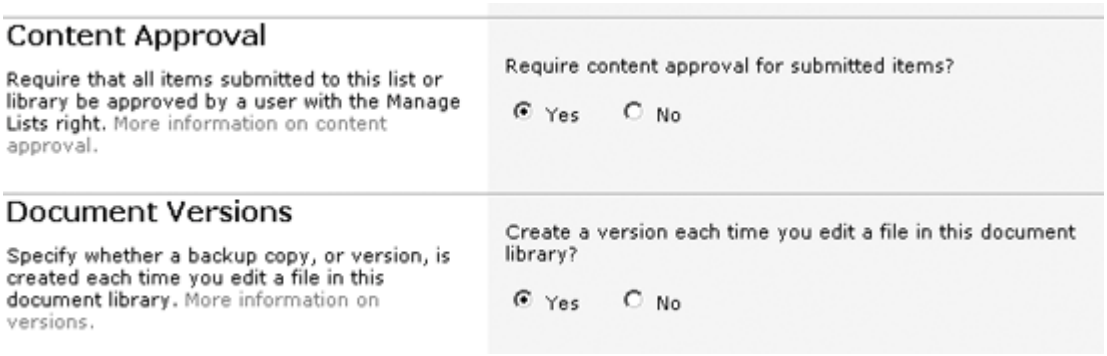
3. Click Save and Close. SharePoint uploads the documents and adds them to the library as shown in Figure 2-12.

animal 2-12. Adding documents to a library



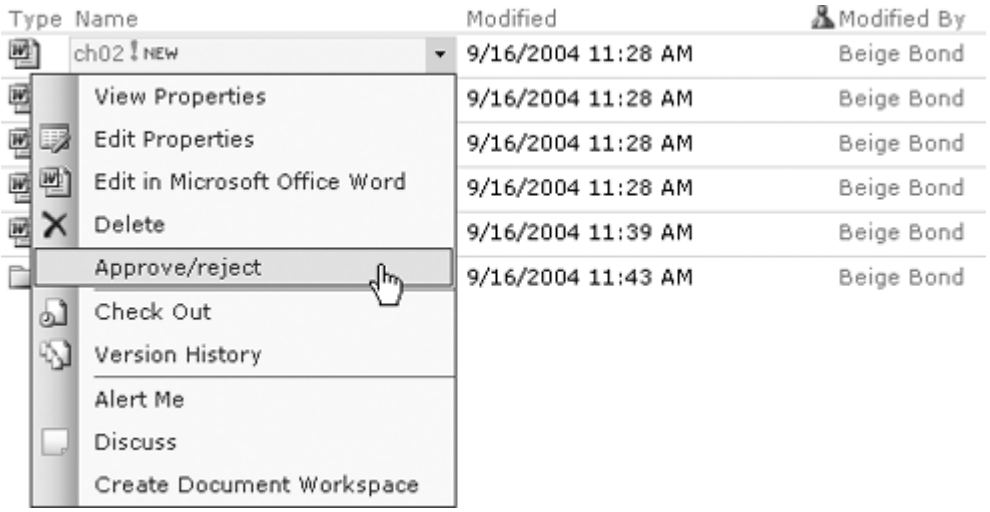
The Shared Documents library doesn't keep track of version history by default. You can change that by clicking Modify settings and columns → Change general settings → Create a version each time you edit a file in this document library? → Yes. You can also require approval before documents are accepted to the list, as shown in Figure 2-13.

animal 2-13. Changing the library's general settings to require approval and keep version history



Once documents are uploaded to a library, you can perform a variety of tasks by clicking the down arrow next to the document name, as shown in Figure 2-14.

animal 2-14. Using the context menu to perform document tasks



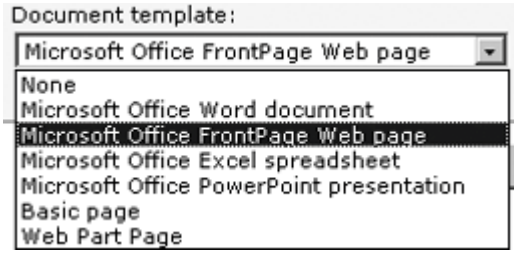
To create a new library in a SharePoint site:

1. Click Create → Document Library. SharePoint displays the New Library Page.

2. Enter the library name, description, navigation, versioning, and template information and click Create.

The template list in step 2 includes the options shown in Figure 2-15. In other words, libraries provide a way to add web pages to a site.

animal 2-15. Libraries can include Office documents or new web pages



Use views, not folders, to organize content in a document library. Folders are best used for supporting files, such as images. If you want to categorize documents, add Category or Product columns to the library, then create a custom view that filters or groups the library based on those columns. See Chapter 7 for details on adding fields and filtering/grouping items.

2.5.3. Constructing New Pages

As you just saw, you can create new web pages on a site using document libraries. It's pretty easy to create a library for web pages, manage folders, and add other content to create a web site, as shown in Figure 2-16.

animal 2-16. Creating a web page library in Explorer View

Libraries are stored as folders within the site. To navigate to a particular web page, use the address where that page is stored. For example, enter http://myoreillydemosite.sharepoint.apptix.net/Demo%20library/SampleSite1/default.aspx into the address bar. (You can even omit *default.aspx* since IIS automatically looks for a default page if you omit the file name.)

There are a couple of things you can't include in a library, however:

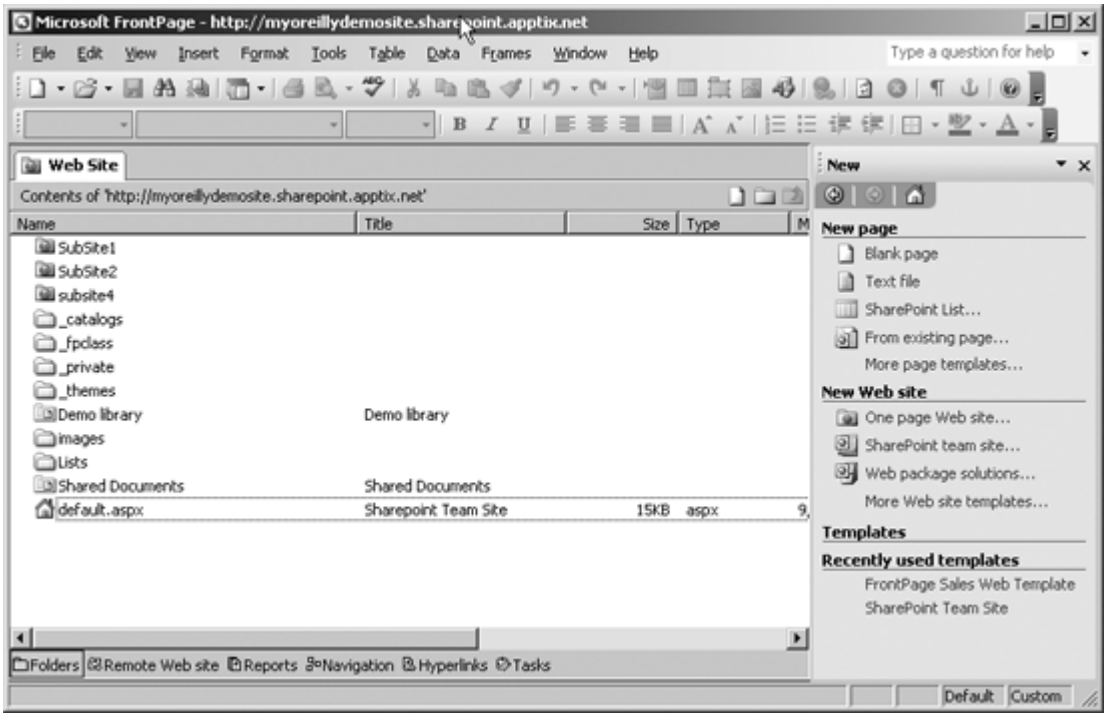
- Executable components (*.exe*, *.dll*, *.com*, etc.) are rejected as a security precaution.

- ASP.NET pages containing server-side scripts. These are allowed in a library, butagain as a security precautionthe scripts aren't allowed to run.

Libraries provide a browser-based interface for adding and editing web pages. In some cases, you may want to add the pages directly to the site rather than going through a library. To add web pages directly to the site:

1. Open the SharePoint site in FrontPage.
2. Click File → New. FrontPage displays the New page task pane, as shown in Figure 2-17.
3. Select one of the page types or click More page templates to see a list of templates that include web part pages.

animal 2-17. Adding new pages directly to the site using FrontPage



FrontPage is the most flexible way to add or edit pages in the site; however, pages won't have a revision history unless they are part of a document library.

2.5.4. Creating Workspaces

A *workspace* is a type of subsite within a team site. You create a workspace to give multiple members input into one or more related documents. You can create a workspace from within a document library, an Office application, or the home page of a site. To create a workspace from Word, Excel, or PowerPoint:

1. Open the document.
2. Choose Tools → Shared Workspace. The application displays the Shared Workspace task pane.
3. Enter the name and location of the workspace and click Create. The application connects to the SharePoint site and creates a workspace for the open document.

Creating a workspace in this way links the local copy of your document to the copy stored in the SharePoint workspace. The next time you open the local copy, the application checks for updates from the SharePoint site. Similarly, changes you make are automatically sent to the SharePoint site. To create a workspace from a document stored in a SharePoint library:

1. Select the document and choose Create Document Workspace from the context menu shown in Figure 2-14.
2. SharePoint displays a confirmation page. Click OK to create the workspace.

Creating a workspace from a library allows members to publish the document back to the library when they have completed their changes. To create a workspace from the home page:

1. Click Create → Sites and Workspaces. SharePoint displays the New SharePoint Site page.
2. Enter the workspace name and description and click Create. SharePoint displays the Template Selection page.
3. Choose Document Workspace and click OK. SharePoint creates an empty workspace and displays its home page.

Creating workspaces from a home page is handy if your site restricts who can create new workspaces. In that case, members may need to ask an Administrator or Web Designer to set up a workspace for them.

If you open a document from a workspace and save it to your local computer, you will be asked if you want to be able to update the workspace copy. Choose Yes if you want to link your local copy to the workspace copy.

It might seem strange that workspaces created from an Office application or from a library contain only a single document. Why create a whole web site for a single Word or Excel file? I'm not sure I know the answer to that, but you can add documents to the workspace from an Office application fairly easily by doing the following:

1. Open a document that is part of the workspace.
2. From the Shared Workspace task pane, click the Documents icon and click Add new document.

To add documents from the workspace home page, just click Add new document.

2.5.5. Workspaces Versus Libraries

Document workspaces and libraries may seem similar. That's because a document workspace actually contains a Shared Documents list, which is a library. Since workspaces are a type of subsite, however, they let you restrict who has access. Table 2-5 may help you understand the differences.

Table 2-5. Workspace features compared with Library features

Feature	Workspaces	Libraries
Boundaries	Establish new site boundaries.	Do not establish new site boundaries.
Membership	May have a unique set of members, or they may inherit members from the parent site.	Do not have their own sets of members, although a library may be restricted so that only some members can view or change it.
Searching	Not included in searches on the parent site.	Included in searches on the parent site.
Best used for	Documents under development or with a restricted audience.	Documents available to everyone.

Workspaces also enable certain features not available from libraries in team sites. For example, workspace documents can be saved as local files linked to the SharePoint workspace.

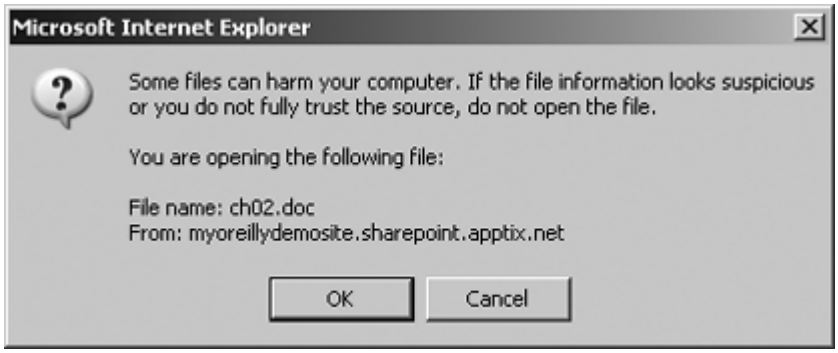
2.6. Setting Client Security

You'll encounter two security issues when first using a SharePoint site over the Internet:

- Opening files from a library or workspace displays a security warning, as shown in Figure 2-18.
- You can't create workspaces from Office applications.

You can't get rid of the security warning: Internet Explorer always warns you when you open files, such as Word or Excel files, that might contain viruses. However, if you open the file using the Office application or open a local file linked to the SharePoint site, you won't see the warning in Figure 2-18.

animal 2-18. Security warning when opening Office files from the Internet

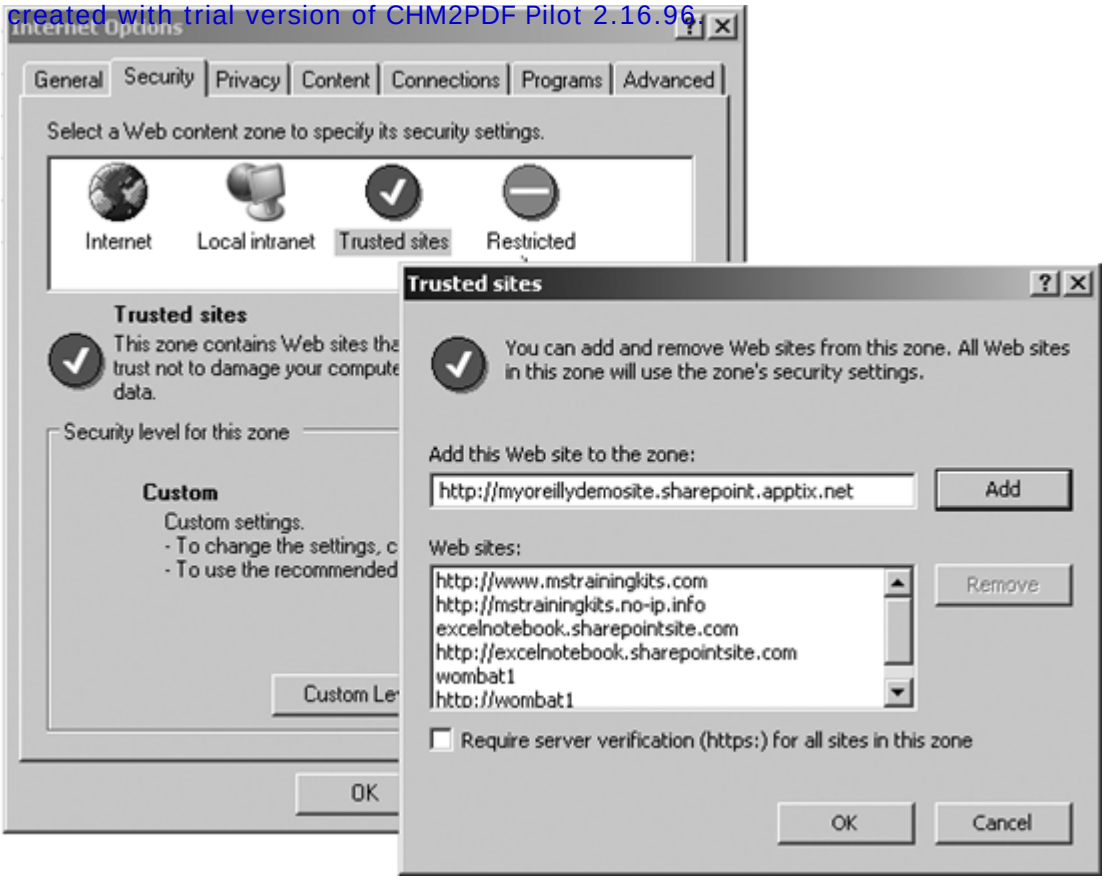


The virus risk comes from macros the file might contain, and in any case, the Office macro security settings determine whether or not macros are run. To check those settings in your Office application, choose Tools → Macro → Security.

You can solve the second issue by adding your SharePoint site to the list of trusted sites in Internet Explorer:

1. Choose Tools → Internet Options → Security → Trusted Sites → Sites. Internet Explorer displays a list of the trusted sites for your local computer (Figure 2-19).
2. Type the address of your site and click Add → OK → OK to close the dialog.

animal 2-19. Adding a SharePoint site to Internet Explorer's list of trusted sites



Once the SharePoint site is trusted, Office applications can create workspaces on the site from the Shared Workspace task pane. Neither of these security issues occurs for SharePoint sites on your local area network: those sites are included in the Local intranet zone, which has a higher level of trust.

PREV NEXT

2.7. Creating Self-Hosted Sites

All of the tasks in the preceding sections also apply to sites you create on your own server. The major differences between hosted and self-hosted sites are that:

- You usually access self-hosted sites over an intranet rather than the Internet, so those sites are automatically trusted, and you have fewer client security issues to worry about.
- Since self-hosted sites are usually part of your network, you can quickly grant all network users access to the site (see the section "Adding Members Quickly" later in this chapter).
- You must install and maintain SharePoint yourself, which requires more knowledge about tools like IIS, SQL, and Windows security than you need when using hosted sites.
- You have direct access to the server, so you have both more control and more responsibility (for backups, etc.) than in a hosted environment.

The rest of this chapter explores installing SharePoint and creating SharePoint sites on your own server. If you've already decided to stick with a hosted site, you can skip ahead to the section "Allowing Anonymous Access," because the next sections don't really apply to you.

2.8. Installing SharePoint Services

SharePoint Services is a component in Windows Server 2003. To install those services on an existing server you'll need the following:

- An account with Administrative privileges for the server
- Physical access to the server or access through the Windows XP Professional Remote Desktop utility
- Access to a dedicated SQL Server or SQL Server installed on the target server (recommended)

There are three configuration options for installing SharePoint Services, and you should choose your target configuration before proceeding. The main difference among configurations is where data is stored, as described in Table 2-6.

Table 2-6. Possible SharePoint Services database configurations

Description	Database used	Advantages	Disadvantages
Default	WMSDE	Least expensive, simplest configuration	Capacity limited to 2 GB; no full-text search; database maintenance more difficult
Single server with SQL Server	Microsoft SQL Server 2000 SP3	Supports full-text search; backups are easy to manage; capacity is better than WMSDE	Additional expense; web server and database share single processor; limiting performance
Server farm	Microsoft SQL Server 2000 SP3 installed on dedicated server	All of the SQL Server advantages plus better performance and scaling	Additional expense; more complicated to set up

For anything more than a personal or small-office site, I recommend using one of the SQL Server configurations because otherwise you just won't get the capacity you may need. Once you've met the prerequisites and chosen a configuration, installing SharePoint follows these major steps:

1. Answer key questions (listed in Table 2-7).
2. Check your SQL Server installation and gather connection information. (Skip this step if you are using WMSDE.)
3. Prepare your server.
4. Install SharePoint Services.
5. Create a virtual server for new SharePoint sites.
6. Integrate web sites that the server was hosting before install.

The following sections examine these steps in more detail.

2.8.1. Questions to Ask Before You Install

Before installing SharePoint Services, you should have answers to the questions in Table 2-7, because the answers affect how you install and set up your SharePoint site. Note that these questions may require discussion with managers or team members.

Table 2-7. Installation checklist

Question	Choices	Comments
What database server will you use?	WMSDE SQL Server (local) SQL Server (separate dedicated server)	See Table 2-6.
Do you want to enable full-text searches?	Yes No	If yes, requires SQL Server.
Should all network users have read access?	Yes No	If yes, see "Adding Members Quickly."
Do you want members to be able to create their own personal sites?	Yes No	If yes, see "Enabling Self-Service Site Creation."
Will you allow anonymous access from the Internet?	Yes No	If yes, see "Allowing Anonymous Access."
Will you allow users to automatically create accounts on your network for use with SharePoint?	Yes No	If yes, requires Active Directory. This is an advanced setting, usually used by ISPs, that can't be changed without reinstalling.
Does the server host other web applications?	Yes No	If yes, see "Re-Enabling Existing Sites."

2.8.2. Checking SQL Server

If SharePoint is to use SQL Server, before you begin your installation, the full-text search component and Service Pack 3 (SP3) or later should be installed.

The full-text search component must be installed *before* the service pack to enable SharePoint's full-text search feature. Full-text search increases the amount of space required to store a database, and you may choose not to use it, but if you don't install it first you won't have that option later in SharePoint.

To install full-text search in SQL Server:

1. Run setup from your SQL Server 2000 installation disk.
2. Choose SQL Server 2000 Components → Install Database Server → Next and choose the server to modify.
3. Choose Next → Upgrade, remove, or add components to an existing instance of SQL server → Next → Next → Add components to your existing installation → Next. Setup displays the Select Components dialog box.
4. Select Full-Text Search from the Sub-Components list and choose Next. Setup installs the Microsoft Search component.

After installing full-text search, you need to install SP3 or later to enable SharePoint to use it. It doesn't matter if you installed the service pack prior to installing full-text searchyou have to do it again. To install the SP3:

1. Download SP3 from Microsoft. Search http://www.microsoft.com/downloads for "SQL SP3" and download *sql2ksp3.exe*.

2. Run *sql2ksp3.exe* to unpack the files.

3. Run *setup.bat* from the unpacked files. Setup starts the installation wizard to walk you through the rest of the process.

The version number of SQL Server 2000 SP3 is 8.0.760 (SP3). To verify that a SQL Server is running SP3:

1. Start SQL Server Enterprise Manager.

2. Select the server and choose Action → Properties → General. The version information appears on the dialog box.

If SQL Server is running on a dedicated server, get the connection settings SharePoint needs during the installation process:

- Server name.
- Authentication type (Windows integrated or SQL).
- If using Windows integrated authentication, record the account the SharePoint server will use to connect to the SQL server.
- If using SQL authentication, record the SQL user name and password SharePoint will use to connect.

2.8.3. Preparing Your Host Server

If you are installing SharePoint on a server that is already hosting other web sites, be aware that installing SharePoint makes those sites unavailable at times during the installation. Before installing you should:

- Check for integration, upgrade, and compatibility issues with existing sites.
- Choose a time when site use is at a minimum.
- Notify users that sites will be unavailable during installation.
- Consider moving existing sites to another server during installation.
- Create a full backup of the server (of course).

The following sections outline some of the integration, upgrade, and compatibility issues you may encounter.

2.8.3.1 Integrating existing sites

If you have existing web sites on the server, you need to figure out how your new SharePoint sites fit in the organization of your sites. You may make the SharePoint site the new top-level site for your server, make it subordinate to your existing top-level site, or set up separate host headers or port numbers for your existing site and new SharePoint site.

Changing the structure of an existing site affects current users and breaks links they have established to your site, but it may be the best option if you are rolling out new capabilities. In general, it is best to roll out features gradually, so you may want to do some of both. See the section "Re-Enabling Existing Sites" later in this chapter for more information.

2.8.3.2 Upgrading from SharePoint Team Services

If you're upgrading a previous installation of SharePoint Team Services V1.0, Microsoft provides a migration tool (*smigrate.exe*). Extensive instructions can be found in the Administrator's Guide

(*WindowsSharePointServicesAdmin.chm*), in the section "Migrating and Upgrading Web Sites."

2.8.3.3 Coexisting with Exchange Server

If your host server is also hosting Microsoft Exchange Server, you'll need to change the authentication method used by SharePoint in IIS after SharePoint is installed (see http://support.microsoft.com/default.aspx?scid=kb;en-us;823265). Note that you can't host Portal Server and Microsoft Exchange on the same machine.

2.8.4. Installing SharePoint

I hope the preceding sections didn't scare you off. Installing SharePoint Services is actually very simple, but I wanted you to be prepared because it's critical to choose your data source before you begin. Changing a live web server is serious business, and you don't want your support line to ring off the hook. Life is a lot easier if you are starting with a new, clean server.

To start installing SharePoint:

1. Download the SharePoint Services installation (*STSV2.EXE*). Search http://www.microsoft.com/downloads for "SharePoint Services."

2. Run *STSV2.EXE*. The download extracts the installation files and starts the SharePoint setup program. The first step is to select your database configuration, as shown in Figure 2-20.

In some cases, Setup may require you to restart Windows 2003 after installing some components. After you restart, you'll have to run Setup again to resume installation.

If you choose the Typical Installation option, Setup installs SharePoint, configures it to use WMSDE, and creates an SQL instance named *machinename*`/SHAREPOINT` which is used to contain the SharePoint configuration and content databases.

To use a local installation of SQL Server, rather than WMSDE, choose Typical Installation, then upgrade the WMSDE database to SQL Server after installation is complete.

To use a SQL Server instance on a dedicated server, choose the Server Farm option. As mentioned earlier, setting up a dedicated SQL Server is more complex than using

animal 2-20. Choosing the database configuration

the other two database configurations, but it provides much better performance and room for growth in the future.

2.8.5. Upgrading WMSDE to SQL Server

To upgrade the default WMSDE database created by the Typical Installation option to use SQL Server:

1. Install SQL Server 2000 on the web server. Be sure to include the Full-Text Search component.

2. Install SP3 (see the section "Checking SQL Server" earlier in the chapter). Installing SQL Server should upgrade the instance of the WMSDE database to use SQL Server automatically.

3. Start SQL Server Enterprise manager and register the *machinename*`\SHAREPOINT` server instance.

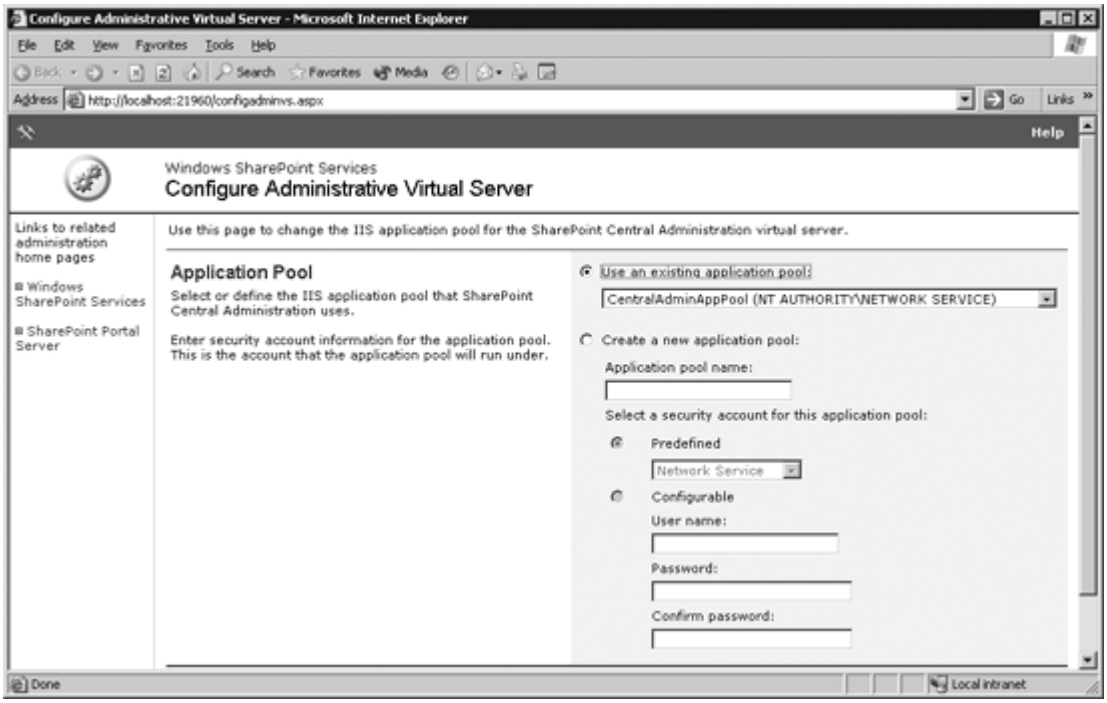
If SQL Server SP3 was present when you installed SharePoint with the Typical Installation option, Setup still configures SharePoint to use WMSDE. To upgrade that database to use SQL Server without reinstalling SQL, use SQL Enterprise Manager to copy the content and configuration databases from the *machinename*`\SHAREPOINT` server instance to the installed SQL Server 2000 instance; then change the SharePoint database settings using the SharePoint Central Administration site. See Appendix A for details.

2.8.6. Setting Up a Web Farm

To install SharePoint using the Web Farm option, use the SharePoint Central Administration site to configure SharePoint's IIS settings and database connection:

1. Choose the IIS application pool to use for the Central Administration site shown in Figure 2-21. IIS can run each web site in a separate process, and those processes run under a Windows identity that determines their permissions.

animal 2-21. Configuring IIS settings



2. If you change the application pool, you must run *iisreset.exe* to restart IIS before continuing.

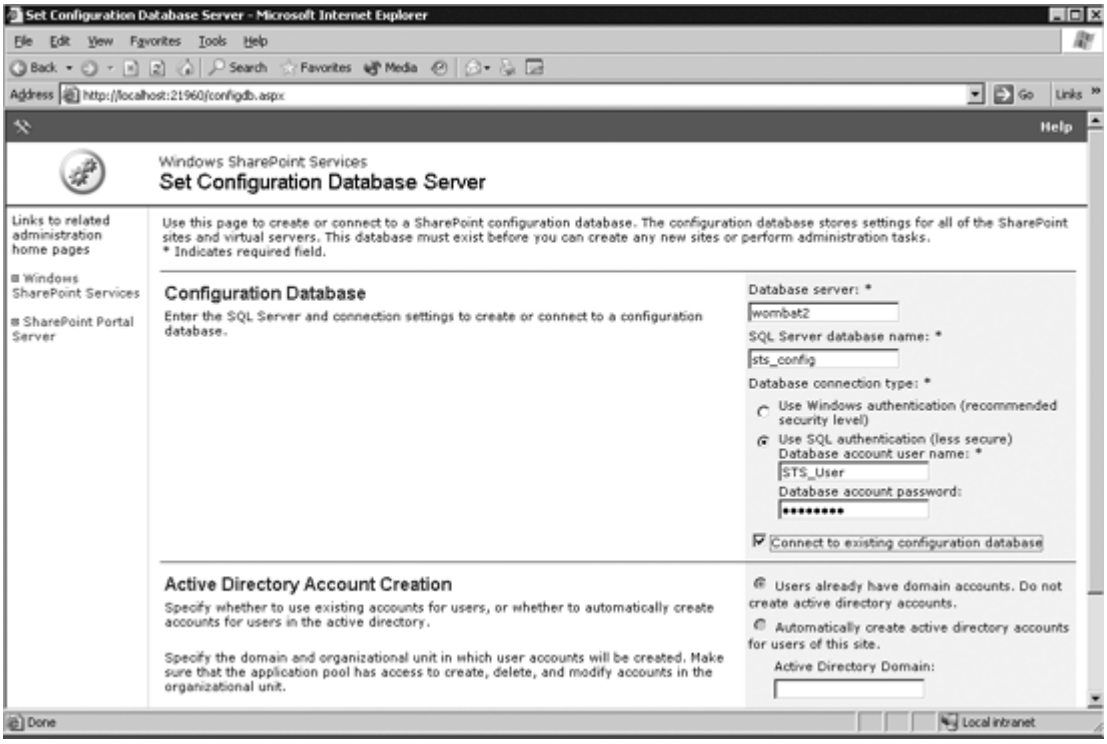
3. Enter the database connection information for the SharePoint configuration database shown in Figure 2-22. The configuration database stores information about the SharePoint server and all the virtual servers it contains.

In Figure 2-22 I specified SQL server authentication rather than Windows-integrated authentication because my network is based on workgroups rather than domains. Most home networks are workgroup-based and most business networks are domain-based. You should use Windows authentication in domain-based networks since it makes administering security easier and passwords more secure.

The Active Directory Account Creation section in Figure 2-22 is used mainly by ISPs, such as those providing

SharePoint hosting services. It allows SharePoint to automatically create user accounts and generate passwords, as mentioned in "Adding Members" earlier. If you're not an ISP, you'll want to keep the default setting.

animal 2-22. Configuring the database connection



2.8.7. Creating Virtual Servers

SharePoint uses separate databases for the server configuration and for virtual server content. Each *virtual server* in SharePoint represents a top-level web site in IIS. The content for each virtual server is contained in a database, as shown in Figure 2-23.

You can't see this association if you used WMSDE as your database because WMSDE doesn't include SQL Server Enterprise Manager. However, WMSDE creates a database instance called SHAREPOINT and then creates databases within that instance. You can find those databases on the server in *C:\Program Files\Microsoft SQL Server\MSSQL$SHAREPOINT\Data*.

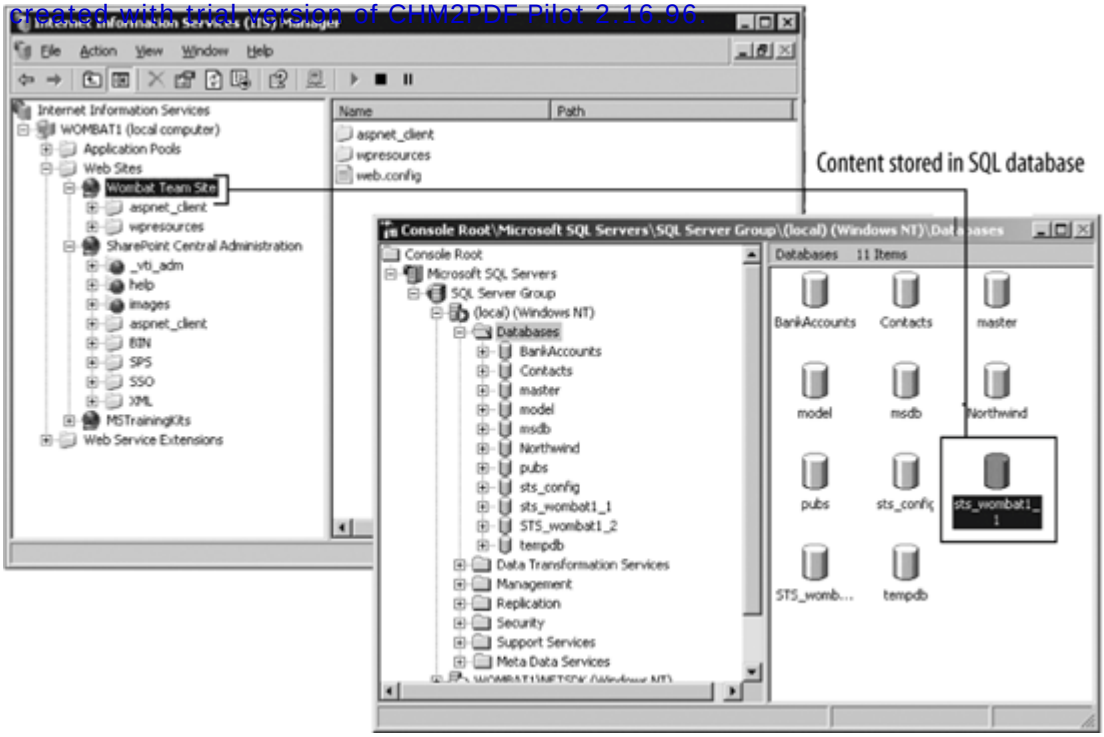
To create a new virtual server in SharePoint, create a new, top-level site in IIS. Then extend that site using SharePoint Central Administration. The following sections describe these steps in greater detail.

2.8.7.1 Creating a top-level site in IIS

To create a new top-level site in IIS:

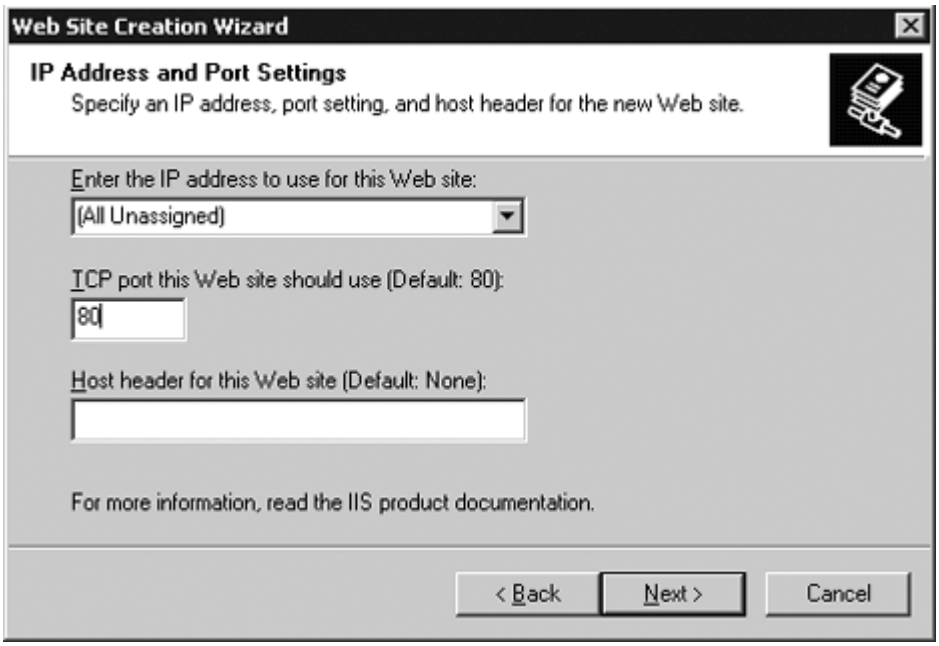
1. On the SharePoint server, choose Start → Administrative Tools → Internet Information Service (IIS) Manager.
2. In the IIS Manager, select the Web Sites folder then choose Action → New → Web Site. IIS starts the Web Site Creation wizard.
3. Follow the steps in the wizard to create the new site.

animal 2-23. A SharePoint virtual server in IIS and SQL



When creating a top-level site in IIS, you can assign the site a specific IP address or port number on the server, as shown in Figure 2-24.

animal 2-24. Assigning top-level sites different IP addresses or port numbers



You can set different port numbers to handle different top-level sites on your server. In fact, that's how SharePoint configures the Central Administration siteby assigning it to an available port number. You can view that site by including the port number along with the domain, for instance http://wombat1:21960/ displays the Central Administration site on my network.

Each top-level site must have a unique IP address or port number so IIS knows where to route requests. Port 80 is the default port for HTTP requests, so whichever site you assign to port 80 is the *default site* for that IP address.

Networks can assign multiple IP addresses to a single server, but SharePoint only supports the All Unassigned setting: you can't host multiple IP addresses on a SharePoint server.

You can change the port number assigned to a top-level site from the Properties dialog box in IIS after the site is created.

2.8.7.2 Extending the site with SharePoint

To extend a top-level site with SharePoint Central Administration:

1. Choose Start → Administrative Tools → SharePoint Central Administration to display the Central Administration site.

2. Select Create a top-level Web site → Complete list and select the name of the site you just created in IIS. SharePoint displays the Extend Virtual Server page.

3. Select Extend and create a content database. SharePoint displays the Extend and Create Content Database page.

4. Enter the IIS application pool for the site, select the database settings, and enter the contact information for the site administrator; then choose OK.

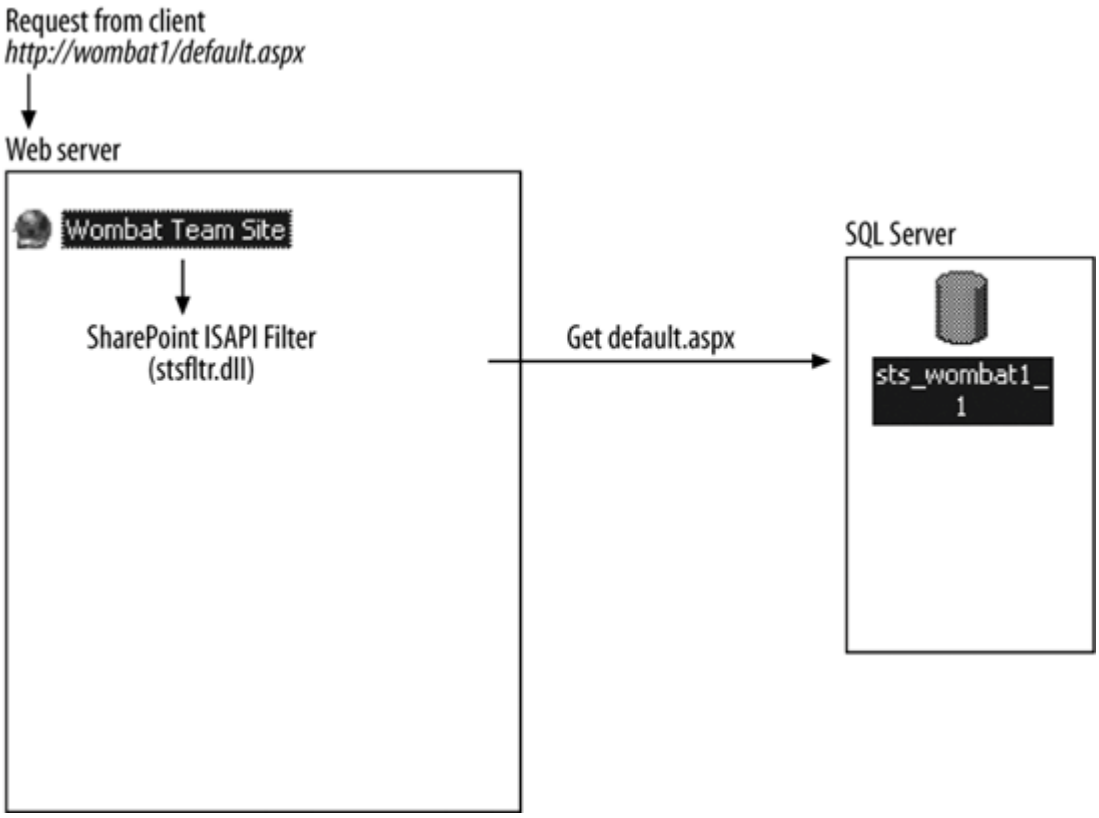
Extending a site with SharePoint configures IIS to send requests through the SharePoint ISAPI filter (*stsfltr.dll*), which then responds to those requests with data from the virtual server's content database as shown in Figure 2-25.

2.8.8. Re-Enabling Existing Sites

To re-enable non-SharePoint sites on your server after you install SharePoint services, add a unique host header for the site; host the site under a different port number; or host the site as a subsite.

Adding a host header for the site creates a unique domain for each virtual server hosted at a single IP address. For example, both http://www.mstrainingkits.com and

animal 2-25. Extending a site lets SharePoint handle requests



http://www.usingsharepoint.com refer to the IP address of my Windows 2003 server. IIS determines which site

to display based on the header received with the request. To add a host header in IIS:

1. Select the site in IIS and choose Action → Properties → Web Site and click Advanced. IIS displays the Advanced Web Site Identification dialog.

2. Click Add and enter the following settings:

Setting	Value
IP Address	All unassigned
TC Port	80
Host Header Value	Your domain name (for example, www.mstrainingkits.com)

3. Click OK twice to close the dialogs and make the changes.

4. Make sure the domain you specified in step 2 is registered for the IP address. On the Internet, that is done with a domain registrar. Within an intranet, that is done on the DNS server or through the client's *hosts* file (search your system folder for "hosts").

Hosting the site under a different port number means the domain part of the address includes a specific port number rather than using the default HTTP port (80) (for example, http://www.usingsharepoint.com:8080/). To change a site's TCP port, select the site in IIS and choose Action → Properties → Web Site and change the TCP port settings.

The disadvantage of this approach is that the web address now includes an ugly port number (8080)-- but it is the simplest way to get things working quickly.

Hosting an existing site as a SharePoint subsite is more complicated, but establishes a more hierarchical site address. For example: http://www.mstrainingkits.com/ExistingSite. To host an existing web site as part of the top-level SharePoint site:

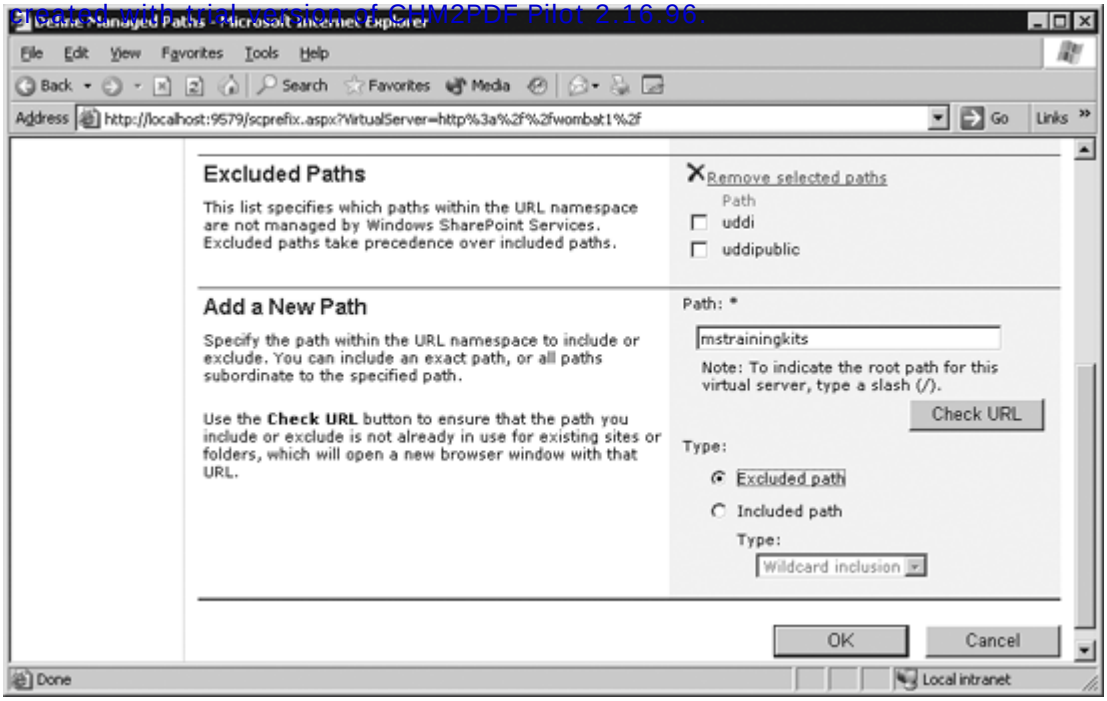
1. Create a new virtual folder for the existing web site within the SharePoint web site. Select the SharePoint site in IIS and choose Action → New → Virtual Directory and complete the Directory Creation wizard.

2. Exclude the virtual folder from SharePoint management. In SharePoint Central Administration Virtual Server Settings, choose Define Managed Paths and add the virtual folder to the list of excluded paths as shown in Figure 2-26.

3. Change the SharePoint site's security and session state settings in *Web.config* to enable the subsite to execute.

4. Test your site to verify that it works from the new address.

animal 2-26. Excluding a folder from SharePoint management



Excluding a path tells SharePoint not to apply its ISAPI filter to requests for resources in that subsite or sites beneath it. In this case, the subsite mstrainingkits maps to a separate site, which is a conventional ASP.NET application.

If your existing site is an ASP.NET application, you'll need to change the security settings SharePoint uses. By default, SharePoint runs under limited permissions, which prevents hosting ASP.NET applications. To increase these permissions, open the SharePoint site's *Web.config* file and change the `trust` element's `level` attribute to `WSS_Medium` as shown here:

```
<trust level="WSS_Medium" originUrl="" />
```

SharePoint omits the ASP.NET Session module by default, so you need to make the change (indicated in bold) to *Web.config* if your ASP.NET application uses session state:

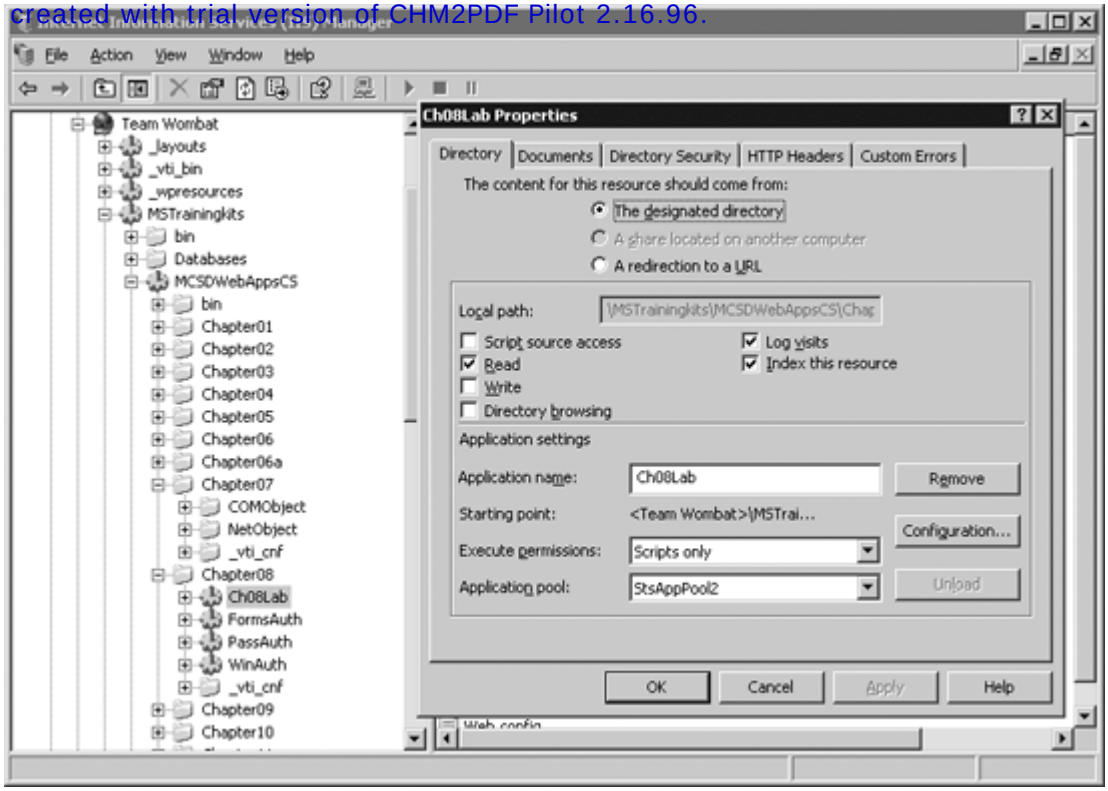
```
<httpModules>
  <clear />
  <add name="OutputCache" type="System.Web.Caching.OutputCacheModule" />
  <add name="WindowsAuthentication"
  type="System.Web.Security.WindowsAuthenticationModule" />
  <add name="Session" type="System.Web.SessionState.SessionStateModule"/>
</httpModules>
```

After you complete these steps, verify that your existing application works from the subsite address. There are a couple things you should check for:

- Broken links or missing resources. If your site used absolute addresses rather than relative ones, the change to your site structure may break those links.
- Missing application starting points. If your existing site included subsites with their own executables, you'll need to recreate those application starting points in IIS.

To create application starting points in IIS, select the folder containing the application and choose Action → Properties. Then, on the properties dialog, click Create, select the application pool to run under, and click OK. Figure 2-27 illustrates adding an application starting point for a subsite within *MStrainingkits*.

animal 2-27. Adding application starting points in IIS



PREV NEXT

2.9. Adding Members Quickly

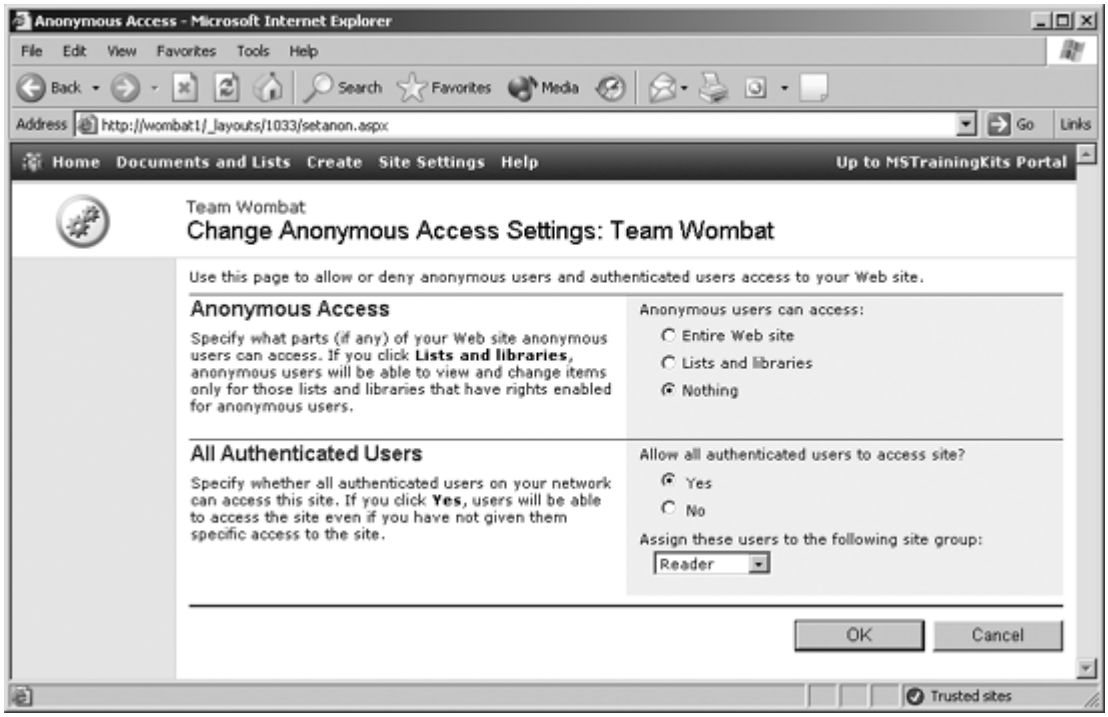
By default, SharePoint uses network domain accounts to authenticate users. Once authenticated, SharePoint checks the user's identity against the list of members for the site. If the user is a member, he or she is granted permissions based on the group that he or she belongs to (Guest, Reader, Contributor, Web Designer, or Administrator).

You can add individual members to your SharePoint site by following the procedure in "Adding Members" earlier in this chapter, but that can be a lot of work if you have a large organization with a lot of users. To add everyone that has a network account quick access:

1. From your SharePoint site, choose Site Settings → Go to Site Administration → Manage anonymous access. SharePoint displays Figure 2-28.

2. Select Yes under Allow all authenticated users to access site, choose the Reader or Contributor group for those users and click OK.

animal 2-28. Adding all network users to the members list quickly



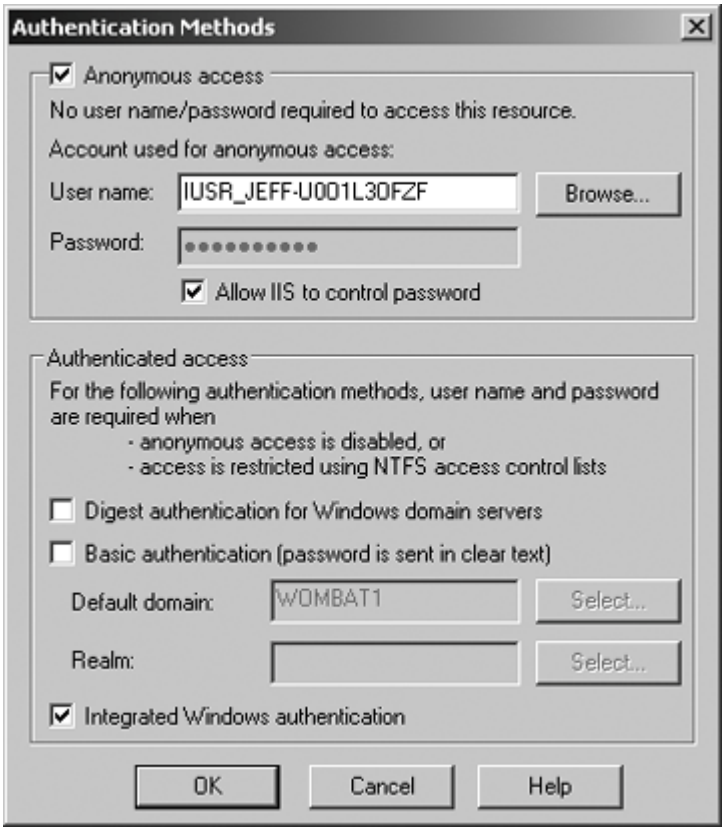
2.10. Allowing Anonymous Access

If your SharePoint site is used over the Internet, you may want to allow users who don't have network accounts access to parts of your site. This allows the general public to view pages, participate in discussions, and complete surveys. SharePoint disables anonymous access when it is installed, so you must re-enable it in IIS and then authorize anonymous access within the SharePoint site.

To allow anonymous access to a SharePoint site:

1. From your SharePoint server, start IIS and select the SharePoint site.
2. Choose Action → Properties → Directory Security, then click Edit. IIS displays the page shown in Figure 2-29.
3. Select Enable Anonymous Access. IIS maintains the anonymous user password, so don't worry about that. Click OK to close each of the open dialogs.
4. From your SharePoint site, choose Site Settings → Go to Site Administration → Manage anonymous access. SharePoint displays Figure 2-28.
5. Select the level of access to provide anonymous users and click OK.

animal 2-29. Enabling anonymous access in IIS



Enabling anonymous access for the entire web site allows unauthenticated users to view all of the folders in your site. Allowing access to lists and libraries restricts anonymous users to folders that have anonymous access specifically enabled. This is the most practical setting for most SharePoint sites since your site usually contains a mix of public and not-so-public information.

To enable anonymous access to a specific list or workspace:

1. Follow the previous procedure and select Lists and libraries (Figure 2-28).

2. Navigate to the workspace or list on the SharePoint site and select Modify settings and columns → Change permissions for this document library → Change Anonymous access. The Anonymous Access page displays.

3. Select whether anonymous users can add, edit, or just view items and click OK.

Allowing anonymous access allows users to view SharePoint site pages without being prompted for a user name and password. When a user tries to open an Office document from the site, SharePoint displays the Windows authentication dialog. If the user cancels that dialog, the document opens in read-only mode.

PREV NEXT

2.11. Maintaining Server Security

Access to SharePoint sites is controlled through the authentication settings in IIS. The default setting is to use Windows integrated authentication, but sites can also use digest or basic authentication.

Digest authentication is used when SharePoint is installed in Active Directory mode (as when configured for use by an ISP). *Basic authentication* sends user name and password information as text, which provides less protection for that information but allows it to pass though a network firewall.

In addition, the security settings in the site's *web.config* file can control which users are allowed or denied permission to access the site. For example, the following settings only allow access to users with Administrative privileges on the server:

```
<authentication mode="Windows" />
<authorization>
      <allow roles="Administrators" />
      <deny users="*" />
</authorization>
<identity impersonate="true" />
```

The `roles` attribute above refers to the Windows account group, not the SharePoint group. You can use `allow` and `deny` element to add or remove specific roles or users. For example the following element blocks the BeigeBond from access the site:

```
<deny users="WOMBAT1\BeigeBond" />
```

The `impersonate` attribute determines the identity used to run applications within the SharePoint site. In this case, SharePoint *.aspx* pages and web parts execute using the permissions granted to the user's account.

Once a user is authenticated, SharePoint uses the members list stored in the site's content database to determine what the user can see and do. This two-tier system allows a lot of flexibility. For example, it is very easy to grant all network users access to the SharePoint site (see "Adding Members Quickly" earlier in the chapter), and then add a few specific members to a particular workspace.

SharePoint automatically blocks executable file types from being uploaded and includes a virus scanner for uploaded files. To configure these settings from SharePoint Central Administration, choose Manage blocked file types or Configure antivirus settings.

The default settings do not enable virus-checking, so it's a good idea to change that setting if your site allows access through the Internet.

2.12. Enabling Self-Service Site Creation

One of the advantages of hosting SharePoint yourself is that you can let members of your network create and maintain their own SharePoint sites without much intervention. To enable self-service site creation:

1. On the server, choose Start → Administrative Tools → SharePoint Central Administration to display the Central Administration site.

2. Select Configure virtual server settings, and then select the server to configure.

3. Select Configure Self-Service Site Creation.

4. Turn Self-Service Site Creation on and click OK. SharePoint activates site creation and adds an announcement to the site featuring a link to the site creation page (*scsignup.aspx*).

SharePoint grants site creation rights to the Reader members group. You may want to remove that permission so that only Contributor or higher-level members can create new sites. To do that:

1. Display the SharePoint top-level site and choose Site Settings → Go to Site Administration → Manage site groups.

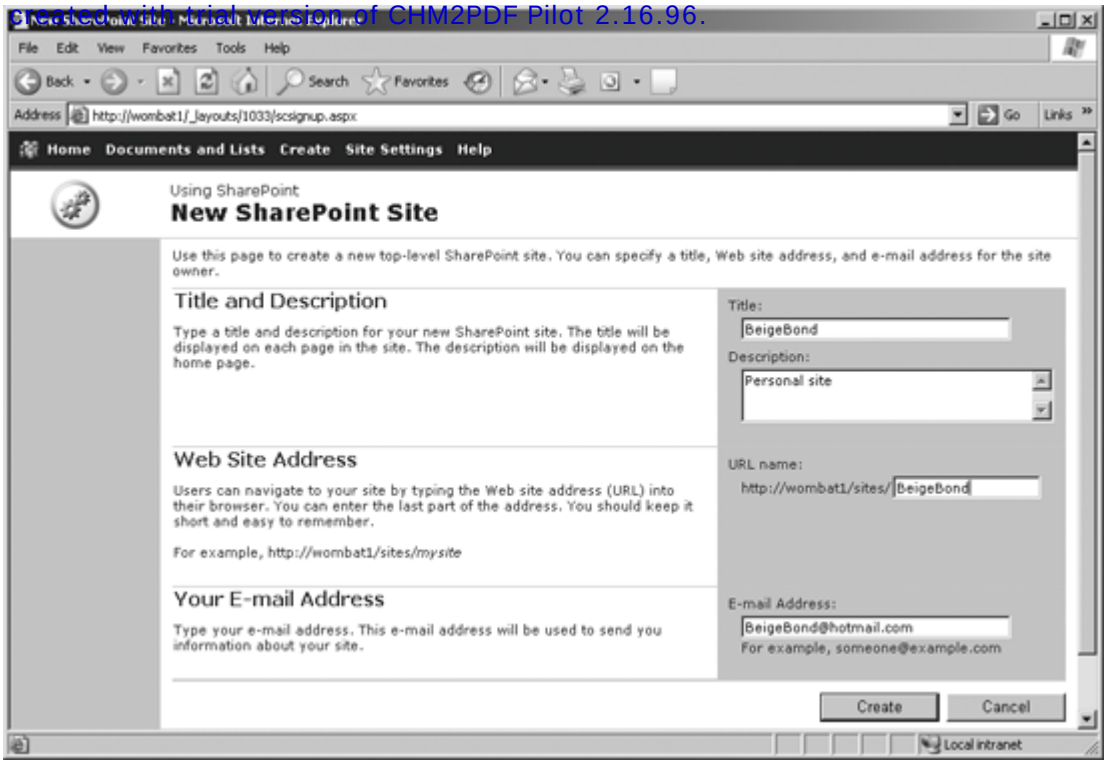
2. Click on the Reader site group and then click Edit Site Group Permissions. SharePoint displays all the permissions for the Reader group.

3. Select the Use Self-Service Site Creation permission to clear the check box next to it and choose Submit to make the change.

If you allow anonymous access over the Internet, those users can't create sites since members must sign on to use *scsignup.aspx*. Only authenticated users can create sites.

The site creation page (*scsignup.aspx*) allows members to create new sites in the */sites* folder of the parent site as shown in Figure 2-30.

animal 2-30. Using self-service site creation



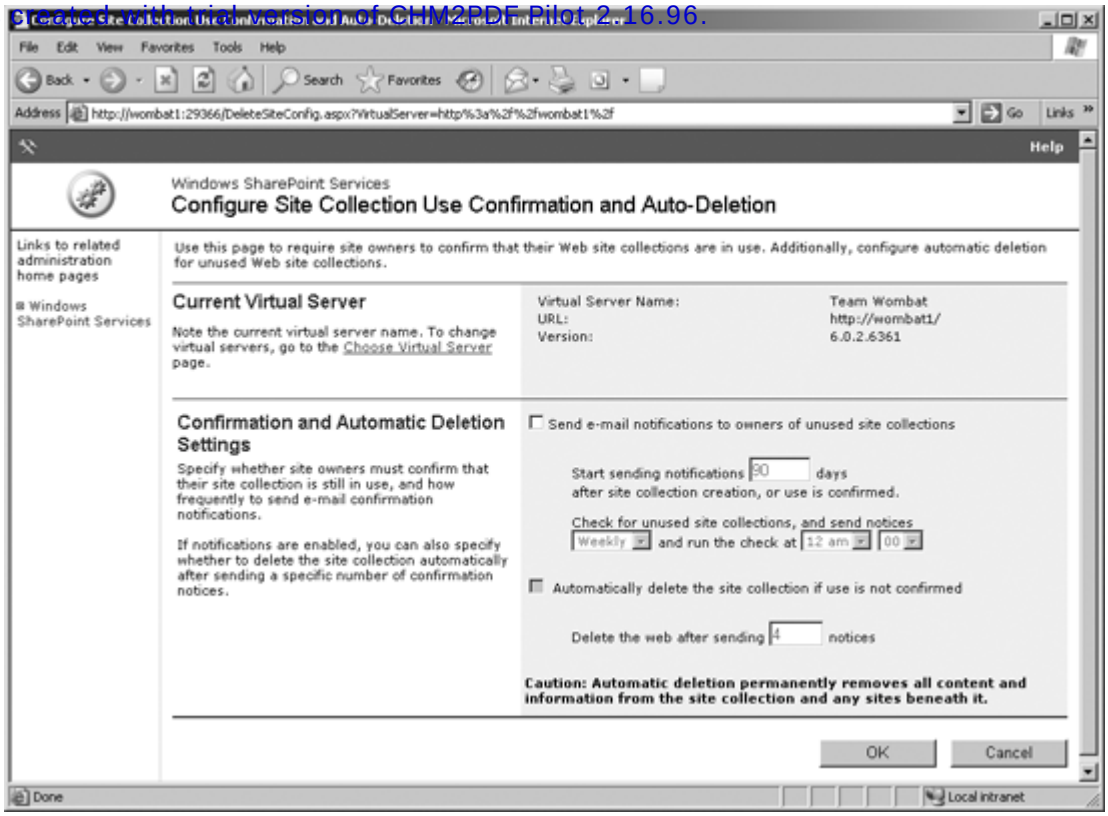
The creator of a new site owns the sites under the new site. For example, in Figure 2-30 Beige Bond owns all of the sites under http://wombat1/sites/BeigeBond. He can create and delete sites, add content, etc. If you didn't restrict site creation to Contributors, that means Readers can create new sites and add content on your server. But that's not as quite as risky as it sounds, because SharePoint allows you to:

- Set quotas to restrict the amount of space used.
- Alert owners of unused sites and automatically delete the site if there is no response.
- Monitor, lock, and delete sites as the collection administrator.

These tasks are performed from the SharePoint Central Administration site. For example, to configure SharePoint to automatically delete unused sites after a certain period:

1. On the server, choose Start → Administrative Tools → SharePoint Central Administration to display the Central Administration site.
2. Select Configure virtual server settings, and then select the server to configure.SharePoint displays the Virtual Server Settings page for the site.
3. Select Configure site collection use confirmation and auto-deletion. SharePoint displays Figure 2-31.
4. Choose the confirmation options and click OK.

animal 2-31. Using confirmation/auto-deletion to help with housekeeping



The SharePoint Services Administrator's Guide contains good information on other related management topics. See the "Configuration" and "Maintenance" sections of the guide's Table of Contents for that help file.

PREV NEXT

2.13. Resources

To get	Look here
Popular SharePoint downloads	http://www.microsoft.com/technet/downloads/sharepnt.mspx
SharePoint Services	Search http://www.microsoft.com/downloads for "SharePoint Services."
SharePoint Services Administrator's Guide	http://www.microsoft.com/resources/documentation/wss/2/all/adminguide/en-us/default.mspx
SharePoint Services Administrator's Guide as a Help file	http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technologies/sharepoint/wssagabs.mspx
Portal Server Administrator's Guide as a Help file	Search http://www.microsoft.com/downloads for "SharePoint Administrator."
Help on coexisting with other web applications	http://support.microsoft.com/?id=823265
Information about using the *Web.config* `authorization` element	http://msdn.microsoft.com/library/en-us/vsent7/html/vxconASPNETAuthentication.asp
A list of built-in group rights	http://www.microsoft.com/resources/documentation/wss/2/all/adminguide/en-us/stsk04.mspx. Or search SharePoint Services Administrator's Guide Help file for "User Rights and Site Groups."
Help on modifying *Web.config* to support ASP.NET applications	http://msdn.microsoft.com/library/en-us/spptsdk/html/tsptWebConfigAppCoexist_SV01134837.asp
Help setting quotas and locks for self-service site creation	"Configuring Site Quotas and Locks" in *WindowsSharePointServicesAdmin.chm*
Instructions on monitoring site usage	"Analyzing Web Site Usage" in *WindowsSharePointServicesAdmin.chm*

Chapter 3. Applying Templates, Themes, and Styles

SharePoint provides several tools that give you a leg up when creating new sites and customizing their appearance:

- *Site templates* provide solutions to the most commonly performed tasks.
- *List templates* define fields and views for common data tables used by sites.
- *Themes* control the colors and backgrounds used by pages within the site.
- *Cascading style sheets* manage the fonts used on pages.

As you use SharePoint and integrate it into your workplace, you'll want to add your own look and feel to these components. For example, you might want to add a Picture Library list to every new workspace or a thumbnail image column to the Contacts list. In fact, you might want to create entirely new templates to solve common tasks unique to your workplace.

This chapter tells you how to create your own site and list templates. First, I provide an overview of what comes with SharePoint Services so you don't reinvent the wheel; then I cover modifying those templates and incorporating the changes into SharePoint. Finally, I explain how to customize the themes and style sheets that control the appearance of sites.

3.1. Understanding Templates

SharePoint includes a set of predefined site and list templates. Each site template includes a set of predefined lists and features a home page that displays or links to the content of those lists. Table 3-1 describes the use and content of each of the predefined site templates.

Table 3-1. Predefined site templates

Template	Use to	Contains these lists
Team site	Create, organize, and share information among team members. This is usually the root site for a department or project team.	Document Library, Announcements, Events, Contacts
Document workspace	Work together on one or more documents. This is the template used when Word, Excel, or PowerPoint create a shared workspace.	Document Library, Tasks, Links
Basic meeting workspace	Schedule and track an in-person meeting. This is the template used when Outlook creates a meeting workspace.	Objectives, Attendees, Agenda, Document Library
Decision meeting workspace	Review relevant documents and record decisions.	Objectives, Attendees, Agenda, Document Library, Tasks, and Decisions
Multipage meeting workspace	Schedule and track an in-person meeting. Includes two blank pages for you to customize.	Objectives, Attendees, and Agenda
Social meeting workspace	Plan social occasions. Features a discussion board and a picture library to post pictures of the event.	Attendees, Directions, Things to Bring, Discussions, Picture Library
Blank site	A blank site for you to customize as a team or workspace site. Includes a navigation bar on the home page.	None
Blank meeting workspace	A blank site for you to customize for a meeting. Includes a minimal navigation bar on the home page.	None

The lists in the predefined site templates are, in turn, based on list templates.

Most SharePoint lists include a common set of fields that help with approval and versioning, but which aren't listed in Table 3-2 because they are common to all the built-in lists. Those common fields are: Approval Status, Approver Comments, Created, Create By, ID, Modified, and Modified By.

Table 3-2. Predefined list templates

Template	Use to	Key fields
Agenda	Outline meeting topics, who will cover them, and how much time each presenter is allotted.	Attachments, Notes, Order, Owner, Status, Subject, Time
Announcements	Post messages on the home page of your site.	Attachments, Body, Expires, Order, Title
Attendees	Invite members to the meeting.	Attachments, Attendance, Comment, Name, Order, Response, Title, User
Contacts	Record and share information about people that your team	Address, Attachments, Business Phone, City, Company, Country, E-mail Address, Fax Number, First Name, Full Name,

	works with.	Home Phone, Job Title, Last Name, Last Name Phonetic, Mobile Phone, Notes, Order, Postal Code, State, Web Page
Decisions	Track and communicate decisions that were made in a meeting.	Attachments, Contact, Decision, Order, Status, Title
Directions	Give driving directions to a social meeting.	Title, Description
Document Library	Store a collection of documents or other files that you want to share. Document libraries support features such as subfolders, file versioning, and check-in/check-out.	Checked Out To, Document Created By, Document Modified By, File Size, File System Object Type, File Type, HTML File Type, ID of the User who has the item Checked Out, Last Modified, Modified, Modified By, Name, Server Relative URL, Shared File Index, Source Url, Title, Type, URL Dir Name, URL Path, Virus Status
Events	Keep informed of upcoming meetings, deadlines, and other important events.	Attachments, Begin, Description, Duration, End, Event Canceled, Event Type, Location, Order, Recurrence, Title, Workspace, Workspace Url
Form Library	Store InfoPath forms, such as status reports or purchase orders.	Checked Out To, Document Created By, Document Modified By, Encoded Absolute URL, File Size, File System Object Type, File Type, HTML File Type, ID of the User who has the item Checked Out, Last Modified, Merge, Name, Order, Relink, Server Relative URL, Shared File Index, Source Url, Template Link, Title, Type, URL Dir Name, URL Path, Virus Status
General Discussion	Provide a place for newsgroup-style discussions. Discussion boards provide features for managing discussion threads and ensuring that only approved posts appear.	Attachments, Order, Ordering, Posted At, Posted By, Reply, Subject, Text, Thread ID
Issues	List outstanding issues that need to be resolved during or after a meeting.	Add Related Issue, Assigned To, Attachments, Category, Comment, Current, Due Date, Issue ID, Order, Priority, Related ID, Remove Related ID, Status, Title
Links	Share links to web pages that your team members will find interesting or useful.	Attachments, Notes, Order, Title, URL
List Template Gallery	Add custom lists to a site. Built-in list templates aren't included in the list template gallery list.	Base Type, Checked Out To, Description, Document Created By, Document Modified By, Encoded Absolute URL, File Size, File System Object Type, File Type, Hidden, HTML File Type, ID of the User who has the item Checked Out, Language, Last Modified, Name, Order, Server Relative URL, Shared File Index, Site Definition ID, Source Url, Sub-type, Template Type, Title, Type, URL Dir Name, URL Path, Virus Status
Meeting Series	Track a series of recurring meetings.	Attachments, Duration, End Date, Event Date, Event Type, Event Url, Exclusion Rule, Has Recurrence, Location, Order, Organizer, Recurrence ID, Recurrence Rule, Recurrence Data, Select, Sequence, Suppress Until, Title
Objectives	List goals for a meeting.	Attachments, Objective, Order, Select, Title
Picture Library	Share pictures used by team members. Picture libraries provide special features for managing and displaying pictures, such as thumbnails, download options, and a slide show.	Checked Out To, Created Date, Date Picture Taken, Description, Document Created By, Document Modified By, Encoded Absolute URL, File Size, File System Object Type, File Type, HTML File Type, ID of the User who has the item Checked Out, Keywords, Last Modified, Name, Order, Picture Height, Picture Size, Picture Width, Selection Checkbox, Server Relative URL, Shared File Index, Source Url, Thumbnail, Thumbnail URL, Title, Type, URL Dir Name, URL Path, Virus Status, Web Image URL, Web Preview
Site Template Gallery	Add custom site templates to a site. Built-in site	Checked Out To, Description, Document Created By, Document Modified By, Encoded Absolute URL, File Size, File System

	templates aren't included in the site template gallery list.	Object Type, File Type, Hidden, HTML File Type, ID of the User who has the item Checked Out, Instance ID, Language, Language LCID, Name, Order, Server Relative URL, Shared File Index, Site Definition ID, Source Url, Template Type, Title, Type, URL Dir Name, URL Path, Virus Status
Survey	Poll members to get opinions. Surveys allow you to quickly create questions and multiple choice answers.	Order, Title, View Response
Tasks	Track work that you or your team needs to complete.	% Complete, Assigned To, Attachments, Description, Due Date, Order, Priority, Start Date, Status, Title
Things to Bring	Display a list of items meeting attendees should bring.	Attachments, Comment, Item, Order, Owner, Title
Web Part Gallery	Add custom web parts to a site. Built-in web parts aren't included in the web part gallery list.	Checked Out To, Description, Document Created By, Document Modified By, Encoded Absolute URL, File Size, File System Object Type, File Type, Group, HTML File Type, Name, Order, Select, Server Relative URL, Shared File Index, Source Url, Title, Type, URL Dir Name, URL Path, Virus Status, Web Part
Workspace Pages	Add custom web part pages or HTML pages to a site.	Checked Out To, Description, Document Created By, Document Modified By, Encoded Absolute URL, File Size, File Size, File System Object Type, File Type, HTML File Type, Name, Order, Server Relative URL, Shared File Index, Source Url, Title, Type, URL Dir Name, URL Path, Virus Status

You can combine Tables 3-1 and 3-2 to create a graphical overview of the fields in a SharePoint site, as shown in Figure 3-1.

animal 3-1. Lists and fields in the Team Site template

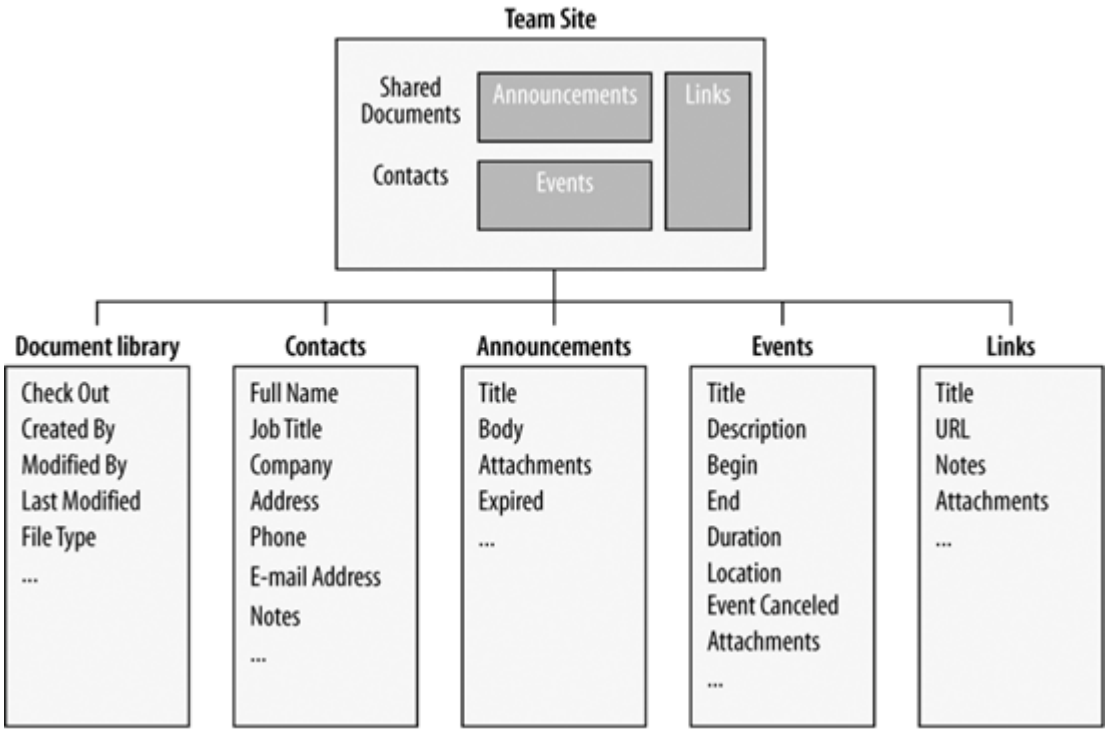


Figure 3-1 shows the default content of a new site based on the Team Site template. Once the site is created, users can add new lists or modify existing lists to add or delete fields; templates are simply a starting point for organizing the content of the site.

The predefined site and list templates are stored as *site definitions* and *list definitions*. Those definitions are made up of files and folders on the serverthey are not stored in the configuration or content databases. When you create a custom template, you base the template on one of these definitions. SharePoint stores custom

templates in the configuration database.

The difference between predefined templates and custom templates will become clearer as you read this chapter. For now, you need to know that both types of templates appear when you create a new site or list, but only custom templates appear in the template galleries.

3.2. Creating Custom Site Templates

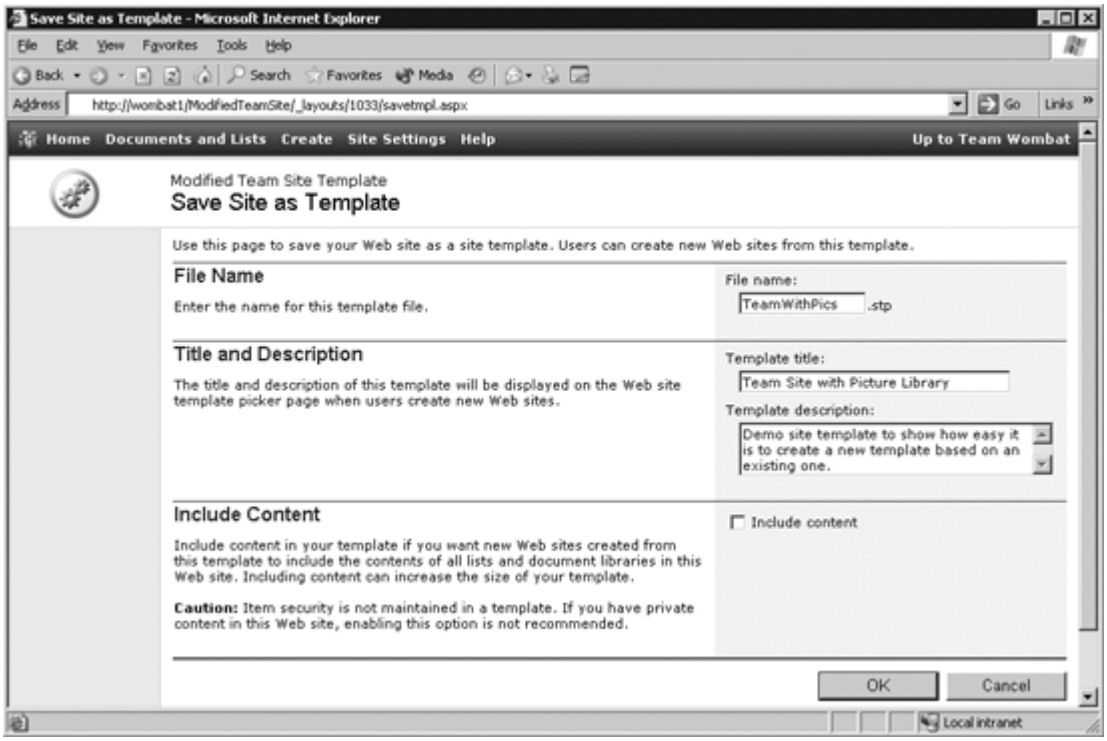
To create a custom site template:

1. Create a new site based on one of the predefined site templates.
2. Modify the site.
3. Save the site as a template.

For example, to extend the standard Team Site template to include a Picture Library:

1. Create a new site based on the Team Site template. Choose Create → Web Pages → Sites and Workspaces, enter a title and address, and then click Create.
2. Add a Picture Library to the site. Choose Pictures → Create Picture Library → Picture Library, enter a library name, and then click Create.
3. Save the site as a custom template. Choose Site Settings → Go to Site Administration → Save site as template. Figure 3-2 illustrates the results.
4. Complete the Save Site as Template page and choose OK to create the template.
5. SharePoint saves the site as a template file and adds it to the Site Template Gallery list.

animal 3-2. Saving a modified site as a new, custom template



Custom templates added to the Site Template Gallery appear in the templates list when you create a new site, as shown in Figure 3-3.

3.2.1. Viewing, Editing, and Deleting Template Files

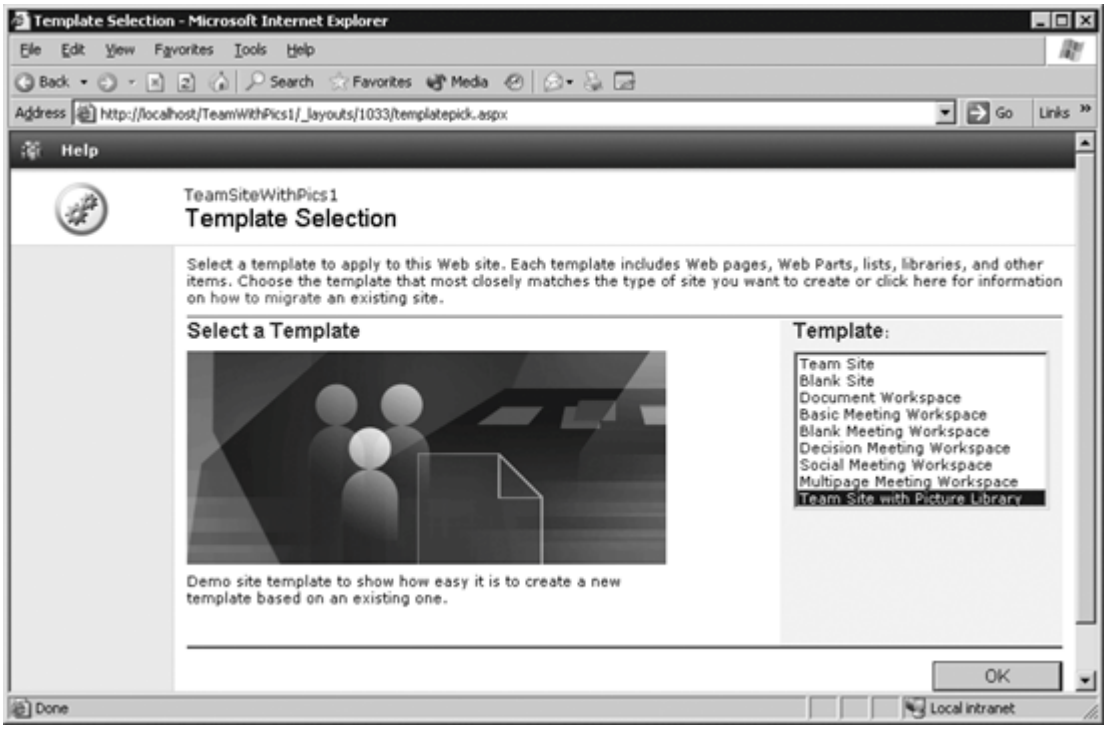
The custom template file (*.stp*) in Figure 3-2 is actually a compressed cabinet file containing an XML template description. To see the contents of the template file:

1. Choose Site Settings → Go to Site Administration → Manage site template gallery. SharePoint displays a list of the custom templates.

2. Choose the template name. SharePoint asks if you want to open or save the file.

3. Save the file to disk on your client computer.

animal 3-3. The custom template automatically appears when creating new sites



4. Rename the downloaded file to have the *.cab* file extension.

5. Open the file using Extract, WinZip, or another file compression utility and extract the file *manifest.xml*.

6. Open *manifest.xml* using FrontPage, WordPad or an XML editor (not Notepad; *manifest.xml* doesn't include whitespace, so you won't be able to decipher it).

The elements in *manifest.xml*, like all XML elements, are case-sensitive. Errors in either the element names or nesting will prevent the template from working.

The following XML fragment shows the details portion of the *manifest.xml* file for the custom template created in Figure 3-2:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Web Url="http://localhost/ModifiedTeamSite">
    <MetaInfo>
    ...
    </MetaInfo>
    <Details>
        <TemplateDescription>Demo site template to show how easy it is to create a
          new template based on an existing one.
          </TemplateDescription>
        <TemplateTitle>Team Site with Picture Library</TemplateTitle>
        <Language>1033</Language>
        <TemplateID>1</TemplateID>
        <Configuration>0</Configuration>
        <Title>Modified Team Site Template</Title>
        <Description></Description>
        <CalendarType>1</CalendarType>
        <AlternateCSS></AlternateCSS>
```

```
            <CustomMasterUrl></CustomMasterUrl>
            <AlternateHeader></AlternateHeader>
            <Subweb>1</Subweb>
            <Locale>1033</Locale>
            <Collation>25</Collation>
            <TimeZone>13</TimeZone>
        </Details>
        ...
    </Web>
```

To make changes to a custom template directly:

1. Edit *manifest.xml*. You can find reference information about the elements in the SharePoint SDK Help file (*spptsdk.chm*).

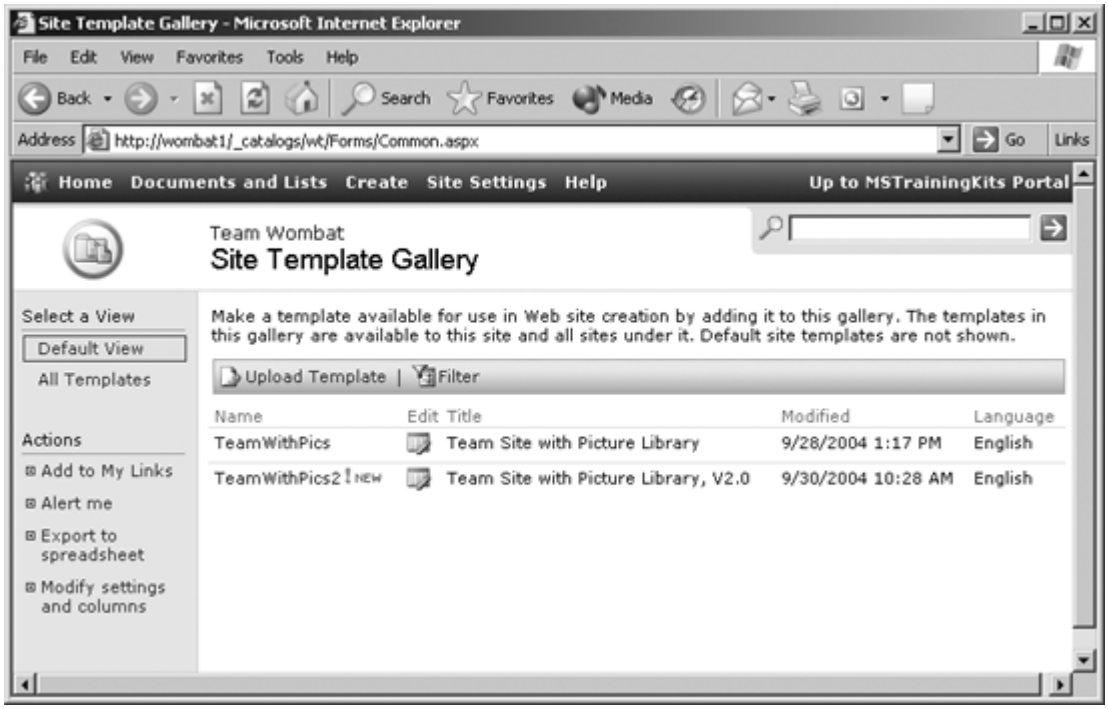
2. Package the changed file using *CabArc.exe*. For example, the following command line packages changes as *TeamWithPics2.stp*.

```
CabArc n TeamWithPics2.stp manifest.xml
```

3. Upload the packaged file to SharePoint from the Template Gallery. SharePoint adds the new template and displays its description in the site template gallery, as shown in Figure 3-4.

Go to http://msdn.microsoft.com/library/en-us/dncabsdk/html/cabdl.asp to find *CabArc*.

animal 3-4. Uploading a new site template



To delete a template from the gallery, click the Edit icon next to the template name and then choose Delete from the template's description page. Deleting a custom site template doesn't affect existing sites based on that template.

3.2.2. Replacing a Predefined Template

It's very difficult to modify the predefined templates, and in fact Microsoft recommends that you avoid doing so, since upgrading to a new release of SharePoint would overwrite your changes and possibly break sites based on

those templates.

It's simpler and safer to create a new, custom template with the same name, and then hide the predefined templatein effect replacing the predefined template with your own. To replace a predefined template with a custom template:

1. Create a new, custom template with the same title as the template you want to replace.
2. On the SharePoint server, edit the *WebTemp.xml* configuration file to hide the predefined template.
3. Stop and restart IIS to refresh SharePoint's template list. To stop and restart IIS, run *iisreset.exe* on the SharePoint server.

The *WebTemp.xml* file is found in the *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\1033\XML* folder of the SharePoint server. To hide a template, change the `Hidden` attribute to `trUE`, as shown here in bold:

```
<Templates xmlns:ows="Microsoft SharePoint">
 <Template Name="STS"     ID="1">
    <Configuration ID="0" Title="Team Site" Hidden="TRUE" ImageUrl="/_layouts/images/
stsprev.png" Description="This template creates a site for teams to create, organize,
and share information quickly and easily. It includes a Document Library, and basic
lists such as Announcements, Events, Contacts, and Quick Links."></Configuration>
...
</Template>
...
</Templates>
```

After you restart IIS, your custom template appears in the template list, as shown in Figure 3-3, and the built-in template doesn't. SharePoint always displays the built-in templates before the custom templates, so the order of the list is changed.

3.2.3. Adding Pages

When you create a new site template based on an existing site, SharePoint includes in the new site template any pages you added in these circumstances:

- You created the pages with FrontPage. Pages added to a site through FrontPage are automatically included in the custom template.
- You created the pages through SharePoint and you selected the Include content option when you created the template (Figure 3-2). Pages created through SharePoint are added to a library in the site and are treated as content.

Including content with the template copies the contents of all the site's lists into the new template; so if you've stored custom pages in the Shared Documents list, they will be included in any new site based on the custom template.

You can't selectively include some lists and exclude others. When including content in a template, create the template from a clean "base" site that contains only the content you want to copy into each new site.

You usually won't want to include lists that represent live data in a template. For example, including Contacts data in a template creates a new copy of Contacts each time a site is created from the template. If all of your sites share a common set of Contacts, you'll want those sites to link to the same data source rather than having to maintain multiple copies of the data.

3.2.4. Changing the Built-in Pages

Changes made to built-in pages, such as the home page (*default.aspx*), are copied to the new template when you create a new template based on an existing site. If you download the template file and open it with a cab-file viewer, such as WinZip, you can see the changes that the template includes, as shown in Figure 3-5.

The generated file names in the template file are mapped to the web-site file names through *manifest.xml*. For example, the *d1000000.000* file displayed in Figure 3-5 maps to *default.aspx*, as shown by this snippet from *manifest.xml*:

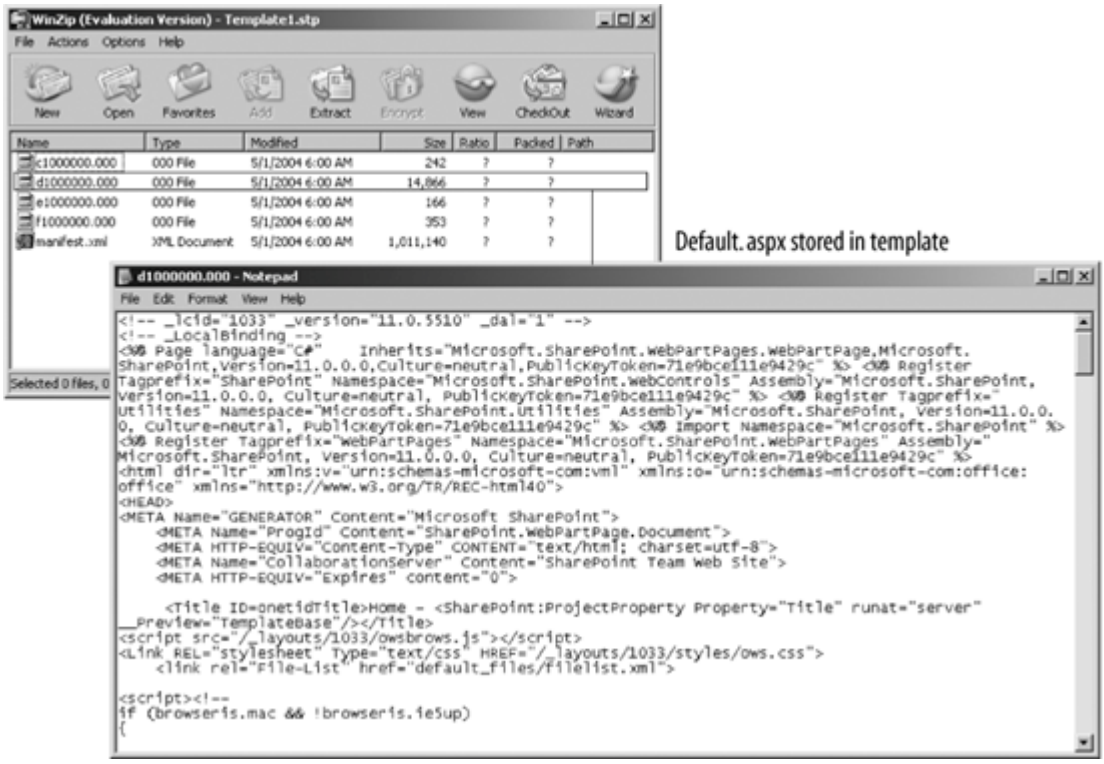
```
<File Name="default.aspx" Src="d1000000.000">
<MetaInfo>
...
</MetaInfo>
</File>
```

PREV NEXT

3.3. Creating Site Definitions

It is interesting to note that if you don't change a built-in page, SharePoint doesn't include it in the custom template. In that case, the built-in page comes from the SharePoint site definition.

animal 3-5. Viewing changed pages in a template file



Site definitions are how SharePoint provides predefined templates. These definitions are made up of a number of files that reside on the SharePoint server's file system. You can view the site definitions by browsing the SharePoint server's *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE* folder.

SharePoint groups the site definitions into two subfolders: *.\1033\STS* contains the site definition for the Team, Document Workspace, and Blank site templates; *.\1033\MPS* contains the site definition for the Meeting workspace site templates. Table 3-3 lists the files SharePoint uses to locate, define, and set the content for its site definitions.

Table 3-3. SharePoint site-definition files

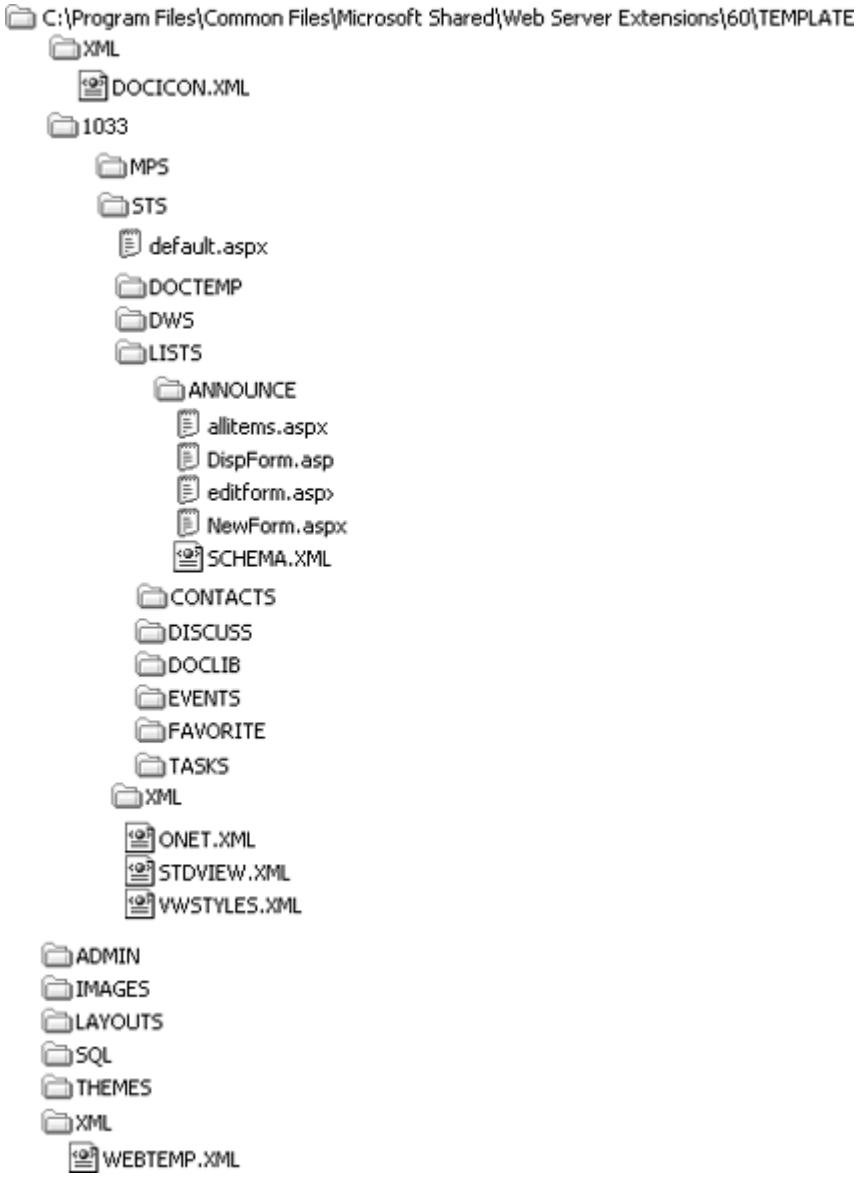
File(s)	Use to	Location
DocIcon.xml	Add file types and control how those files are opened from all sites.	*.\XML*
WebTemp.xml	Add or hide site definitions.	*.\1033\XML*
Default.aspx	Define scripts and layout of the home page for the site.	*.\1033\STS* or *.\1033\MPS*
ONet.xml	Define the navigation areas, list definitions, document templates, default lists, configurations, and modules for the site.	*.\1033\STS\XML* or *.\1033\MPS\XML*
Schema.xml	Describe lists included in the site.	*.\1033\STS\LISTS\listname* or *.\1033\MPS\LISTS\listname*
AllItems.aspx, *DispForm.aspx*,	Define scripts and layout for the various views	*.\1033\STS\LISTS\listname*

EditForm.aspx, *NewForm.aspx*	of a list.	or *.\1033\MPS\LISTS\listname*

`1033` is the locale ID for the United States. If you are working with other locales, the folder name matches your locale IDfor example, `2057` for the United Kingdom or `1081` for India.

Figure 3-6 shows an expanded view of the folders and files where the Team Site, Document Workspace, and Blank site templates are defined.

animal 3-6. Locating site definition files



3.3.1. Customizing Site Definitions

Creating custom site definitions is much more difficult than creating custom templates, but it provides complete control over all aspects of new sites and can provide better performance than custom templates. Since site definitions are complex and made up of numerous dependent files, it's easiest to create new definitions by copying and modifying an existing one. To create a custom site definition:

1. Make a backup copy of the entire Template folder so you can restore the original definitions if you break anything.

2. Copy and rename the STS folder. This copied folder will be the basis for your new site definition.

3. Modify the site definition in the copied folder. The key file to edit is *ONet.xml*.

4. Create a new *WebTemp.xml* file to reference the new site definition.

5. Restart SharePoint to load the new site definition.

6. Create a new site based on the site definition to make sure it works.

For example, I copied the STS folder and renamed it DotSites; then I edited *ONet.xml* to alter the top-level navigation bar and add a custom help page as shown by the following changes in bold:

```
<?xml version="1.0" encoding="utf-8" ?>
<Project Title="Team Web Site" ListDir="Lists" xmlns:ows="Microsoft SharePoint">
    <NavBars>
        <NavBar Name="SharePoint Top Navbar"
Separator="&amp;nbsp;&amp;nbsp;&amp;nbsp;" Body="&lt;a ID='onettopnavbar#LABEL_ID#'
href='#URL#' accesskey='J'&gt;#LABEL#&lt;/a&gt;" ID="1002">
            <NavBarLink Name="Our Collection"
Url="_layouts/[%=System.Threading.Thread.CurrentThread.CurrentUICulture.LCID%]/
viewlsts.aspx">  </NavBarLink>
          <!-- Deleted Create and Site Settings navigation links -->
          <NavBarLink Name="Help" Url='dothelp.aspx'>  </NavBarLink>
        </NavBar>
        <NavBar> ... </NavBar>
   </NavBars>
   <ListTemplates> ... </ListTemplates>
   <DocumentTemplates> ... </DocumentTemplates>
   <BaseTypes> ... </BaseTypes>
    <Configurations>
        <Configuration ID="0" Name="Default">
            <Lists>
                <List Title="Our Collection"
                    Url="Shared Documents"
                    QuickLaunchUrl="Shared Documents/Forms/AllItems.aspx"
                    Type="101" />
                ...
            </Lists>
            <Modules>
                <Module Name="Default"/>
                <Module Name="WebPartPopulation"/>
                <Module Name="DotHelp" />
            </Modules>
        </Configuration>
    </Configurations>
    <Modules>
        ...

        <Module Name="DotHelp" Url="" Path="">
            <File Url="DotHelp.aspx" Type="Ghostable" />
    </Module>
   </Modules>
</Project>
```

The ellipses in the example summarize some of the other elements *ONet.xml* containsI won't spell those out just yet, but they're covered in the SharePoint SDK if you're curious. Figure 3-7 illustrates the changes to the `NavBarLink`, `List`, and `Module` elements highlighted above.

Next, I created a new file based on *WebTemp.xml* and renamed *WebTempDotSites.xml* to tell SharePoint how to load the new site definition as shown here:

```
<?xml version="1.0" encoding="utf-8" ?>
<Templates xmlns:ows="Microsoft SharePoint">
 <Template Name="DotSites"     ID="3">
    <Configuration ID="0" Title="Dot Site" Hidden="FALSE"
```

```
    ImageUrl="/_layouts/images/stsprev.png" Description="Template for Designs of the
    Times.">
        </Configuration>
     </Template>
    </Templates>
```

The `Template` element's `Name` attribute matches the folder name containing the site definition (`DotSite`). The `Configuration` element determines the title, description, and image that appear when creating a new site based on a particular configuration from the site definition.

Each `Configuration` corresponds to a single template in SharePoint. A site definition can provide multiple templatesfor example, the STS site definition provides configurations for three templates: Team Site, Document Workspace, and Blank Site.

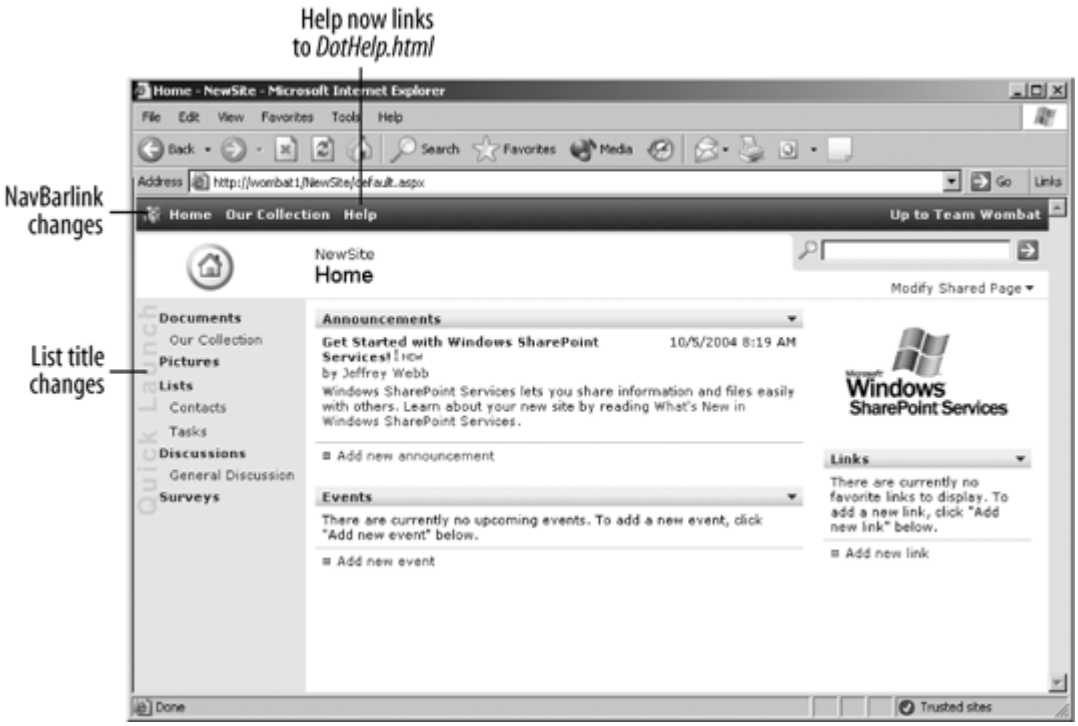
Finally, I ran *iisreset.exe* on the SharePoint server. When SharePoint restarts, it automatically loads site definitions described in all the files named *WebTemp*.xml*. Now, when I create a new site based on the Dot Site template, SharePoint creates a site that looks like Figure 3-7.

3.3.2. Adding and Changing Pages in Site Definitions

In the preceding section, I added a help page and changed the Help link on the navigation bar to point to that page by modifying its `Url` attribute. The change to the `NavBarLink` element was very simple:

```
<NavBarLink Name="Help" Url='dothelp.aspx'>  </NavBarLink>
```

animal 3-7. Seeing the effect of ONet.xml changes



The change to include the new page in the site definition was a little more complex. To add a new page to a site definition, add the `Module` element in two places as shown here:

```
<Configurations>
    <Configuration ID="0" Name="Default">
        <Modules>
            <Module Name="DotHelp" />
        </Modules>
    </Configuration>
</Configurations>
<Modules>
```

```
        <Module Name="DotHelp" Url="" Path="">
            <File Url="DotHelp.aspx" Type="Ghostable" />
    </Module>
    </Modules>
```

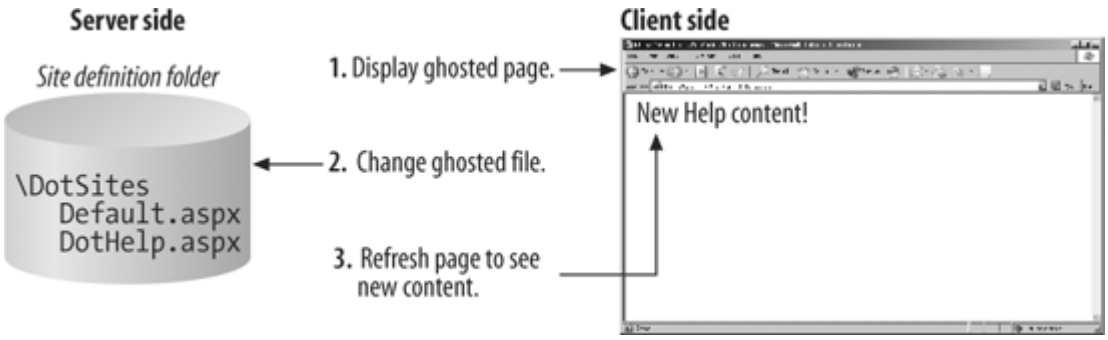
The `Configuration` element defines different template variations that a single site definition provides. The `Modules` element within `Configuration` tells SharePoint which modules to include in each configuration. The second `Modules` element lists the location where SharePoint can find the module and whether or not the module is ghosted.

Ghosted modules are not copied to the site's content. Instead, they are read from the site definition and cached in server memory. *Default.aspx* is a good example of a ghosted module. For the preceding example, I created a new file named *DotHelp.aspx* in the same server-side folder as *default.aspx*:

```
>dir /w
Directory of C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions
\60\TEMPLATE\1033\DotSites

[.]            [..]           default.aspx   [DOCTEMP]
DotHelp.aspx   [DWS]          [LISTS]        [XML]
```

Since ghosted files aren't copied into site content, you can change those files in the site definition, and sites based on that definition automatically change. Figure 3-8 shows how this works.

animal 3-8. Changing ghosted files automatically changes sites based on the definition



I tried using ghosting with an HTML page, but it didn't work because HTML pages are cached differently than ASP.NET (*.aspx*) pages.

You can easily try this on any of your existing sites. Go to the site definition you based your site on and add some text to the site's *Default.aspx* page. For example, add this text to the end of the page:

```
<h1>This is a change</h1>
</body>
</html>
```

Next, display the site's home pageyou'll see the new heading text at the bottom of the page. Notice that you didn't need to restart SharePoint to see the change to the file.

Changing *Default.aspx*, or any other ghosted file, in the site definition affects all sites based on that definition. However, if you open a site in FrontPage and edit one of the ghosted files, that file becomes unghosted and is instead stored as site content. Even minor edits remove ghostingwhich has two unintended consequences: it increases disk usage since the page is then stored as site content, and it prevents changes in the site definition from appearing automatically.

3.3.3. Debugging Site Definitions

You must restart SharePoint for changes to the site definition's XML files to take effect. After restarting SharePoint, you might get an error the first time you request a page from the SharePoint site in your browser. Repeat the request and it should work.

Any errors or inconsistencies in *ONet.xml* prevent the site definition from working. Even a simple typo, such as incorrectly capitalizing the `NavBar` element, will display a generic error page if you try to create a new site based on that definition. The error page doesn't give you a clue where the error occurred, so it is best to:

- Use a real XML editor, such as FrontPage, XML Spy, or Visual Studio, that can check the form of XML documents. Checking the form before saving can avoid XML syntax errors such as misspelled element names or incorrect capitalization.

- Keep a backup of *ONet.xml*, make one change at a time, restart SharePoint, and see if the change broke anything. This is tedious and slow-going, but it helps a lot when you are first learning to create site definitions.

If you don't have an XML editor you can install the Internet Explorer Tools for Validating XML (*iexmltls.exe*) from http://www.microsoft.com/downloads. Then simply open the XML file in Internet Explorer to check the form.

3.4. Distributing Site Templates

Custom templates that are based on built-in templates can be saved as *.stp* files and uploaded to other servers running SharePoint services as described in "Creating Custom Site Templates."

New, predefined templates created through site definitions can be copied to the Templates folder of other SharePoint servers as described in the section "Customizing Site Definitions" earlier in this chapter. Be sure to copy both the folder containing the site definition (for example, *.\1033\DotSite*) and the *WebTemp*.xml* file in the *.\1033\XML* folder that loads the definition in SharePoint. Finally, remember to restart SharePoint after the definitions are installed.

Custom templates can also be based on templates from custom site definitions. In this situation, you need to install both the site definition and the custom template on the SharePoint server.

3.5. Creating List Templates

SharePoint comes with predefined list templates for common types of lists, such as Announcements, Contacts, and Document Libraries. *List templates* define the fields, views, and (optionally) data included in new lists created from the template.

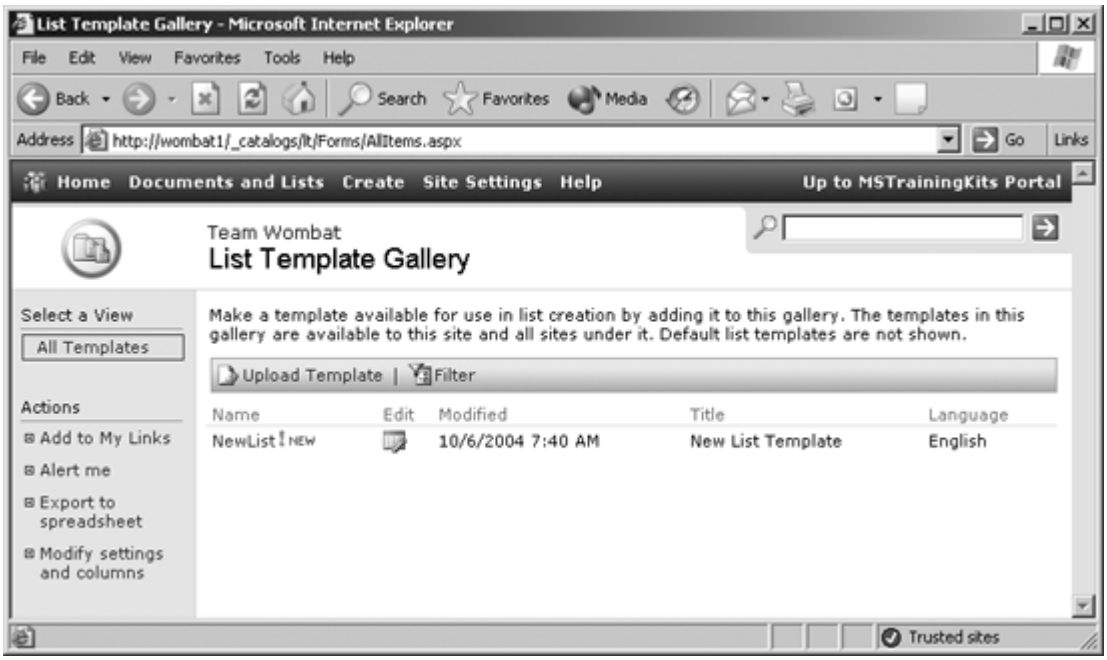
Creating a custom list template is very similar to creating a custom site template:

1. Create a list that you want to base your template on.

2. With the list displayed in the browser, choose Modify settings and columns → Save list as template. SharePoint displays the Save as Template page.

3. Enter a file name for the template, title, description, and choose OK to create the template. SharePoint displays a success page with a link to the list template gallery, as shown in Figure 3-9.

animal 3-9. Viewing list templates in the gallery



The list template gallery is similar to the site template gallery shown earlier in Figure 3-4. You can click on a template name to download the file (*.stp*), rename the file with the *.cab* extension, and view the contents by opening it with WinZip or Extract. List template files contain an XML description of the template (*manifest.xml*) as summarized here:

```
<?xml version="1.0" encoding="UTF-8"?>
<ListTemplate WebUrl="http://wombat1">
    <Details>
        <TemplateDescription>Demo list template.</TemplateDescription>
        <TemplateTitle>NewListTemplat</TemplateTitle>
        <Language>1033</Language>
        <TemplateID>1</TemplateID>
        <Configuration>0</Configuration>
        <TemplateType>105</TemplateType>
        <BaseType>0</BaseType>
    </Details>
    <Files>
        <File Name="AllItems.aspx" Src="00000000.000"> ... </File>
        <Folder Name="Attachments"> ... </Folder>
        <File Name="DispForm.aspx" Src="10000000.000"> ... </File>
        <File Name="EditForm.aspx" Src="20000000.000"> ... </File>
```

```
        <File Name="NewForm.aspx" Src="30000000.000"> ... </File>
      </Files>
      <UserLists>
          <List Name="{27E1B3BE-32C0-4CB3-A270-21E45D7429C8}" Title="NewListTemplate"
 Description="Demo list template." Direction="0" BaseType="0" ServerTemplate="105"
Url="Lists/Contacts" Version="0"> ... </List>
      </UserLists>
      <WebParts> ... </WebParts>
    </ListTemplate>
```

To make changes to a custom list template directly:

1. Edit *manifest.xml*. You can find reference information about the elements in the SharePoint SDK Help file (*spptsdk.chm*).

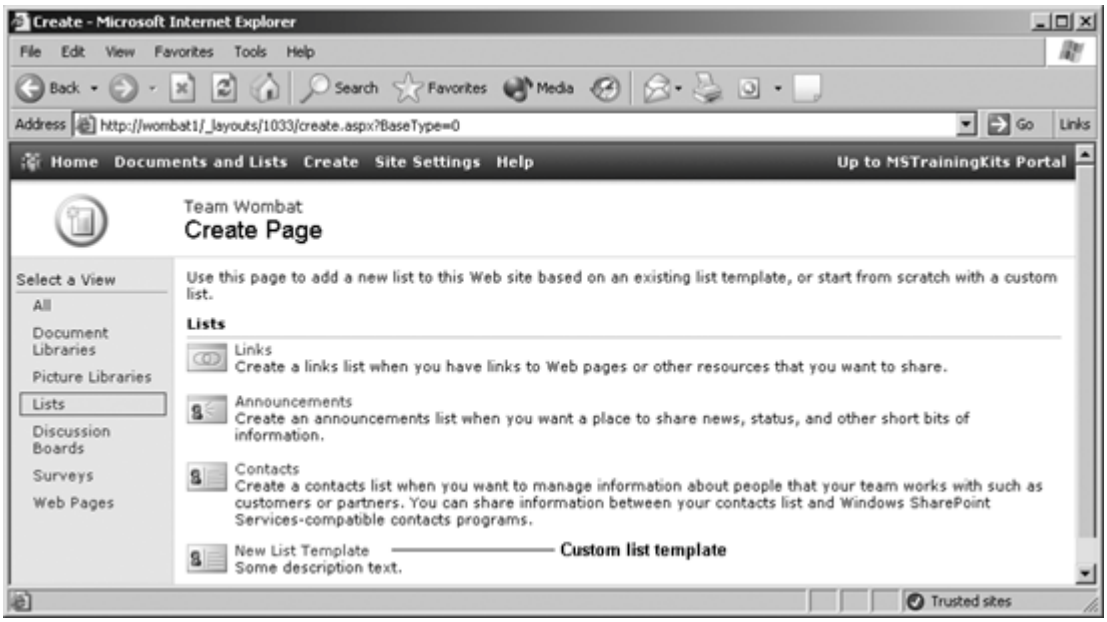
2. Package the changed file using *CabArc.exe*. For example, the following command line packages changes as *TeamWithPics2.stp*.

```
CabArc n NewListTemplate2.stp manifest.xml
```

3. Upload the packaged file to SharePoint from the list template gallery (Figure 3-9).

Custom list templates added to the list template gallery appear on the Create page in SharePoint, as shown in Figure 3-10.

animal 3-10. Creating a list from a new custom template



As with custom site templates, list templates are stored in the configuration database. The configuration database provides top-level resources that are available to all SharePoint sites on the server, so custom list templates are automatically available to all sites. You can't selectively add or hide custom list templates for specific sites.

3.6. Adding List Views

Views are web part pages used to view or edit lists. Most lists have the built-in views listed in Table 3-4.

Table 3-4. Built-in list views

View	Displays
AllItems.aspx	All items in the list. This is the default view for lists.
DispForm.aspx	A single item in the list (read-only).
EditForm.aspx	Data entry form for changing an existing item in the list.
NewForm.aspx	Data entry form for adding a new item to the list.

You can add new views to a list from the browser by displaying the list and following these steps:

1. Choose Modify settings and columns → Create a new view. SharePoint displays the Create View page.
2. Choose a type of view to create. SharePoint displays a Create View page for the selected type of view.
3. Name the view, select the fields to display and how to display them; then click OK to create the view.

There are three basic types of views:

`Standard view`

Displays the list in a very readable form.

`Datasheet view`

Displays list items in a spreadsheet-like grid that can be edited directly.

`Calendar view`

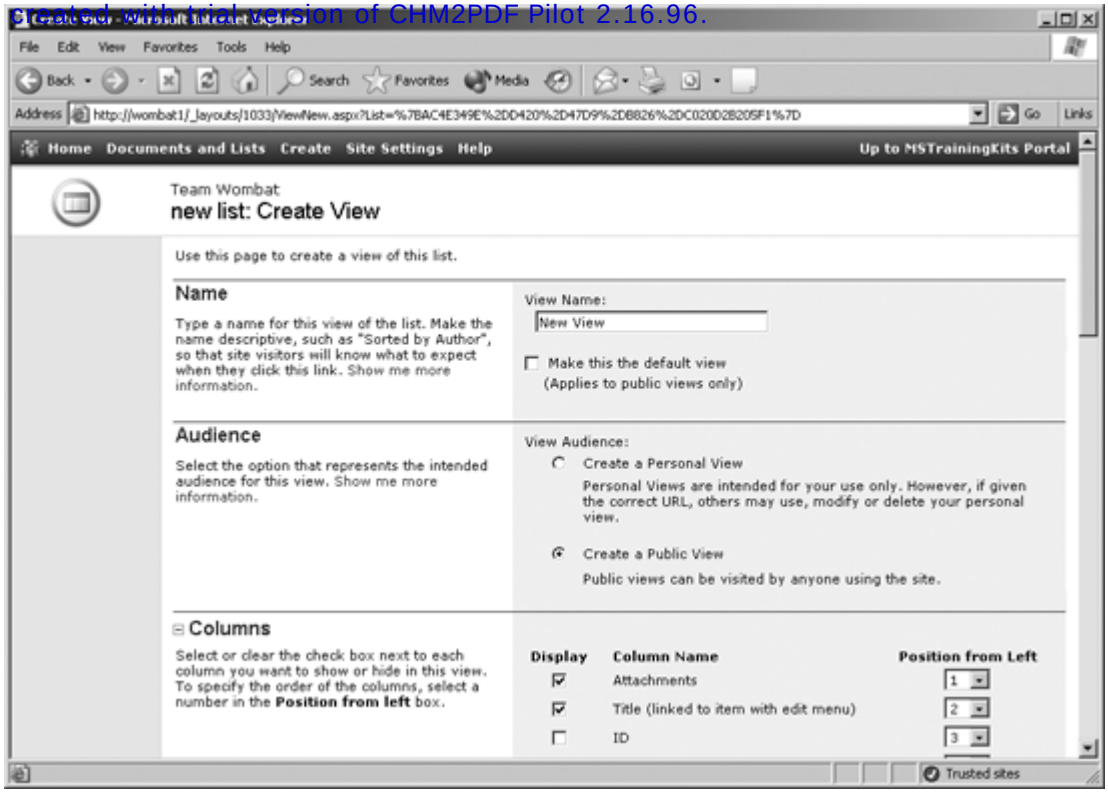
Displays items by month, week, or day. This is a handy view for time-critical lists such as appointments.

Views can also be public or private. Public views can be seen by all members. Private views can only be seen by the member that created the view. Figure 3-11 shows the page for creating a new standard view.

3.6.1. Sorting, Filtering, and Highlighting with Views

Figure 3-11 shows options for making the new view the default view for the list, making the view private, and which columns to include. You can also specify sorting, filtering, grouping, totals, and styles to customize the view. In other words, you can create views that highlight key items, such as the latest changes as shown by the New Today view in Figure 3-12.

animal 3-11. Creating a new standard view



animal 3-12. Using views to highlight items such as updates to the Shared Documents list

I included New Today as a web part on my home page to draw my editor's attention to the files I've been working on. I created the view by adding a new standard view to the Shared Documents list with the following options selected (Figure 3-11):

Option	Setting
Name	New Today
Create a Public View	Selected
Columns (selected columns)	Name (linked to document)
	Title
Sort	None
Filter	Show items only when the following is true:
	`Modified` is equal to [`Today`]
Style	Newsletter, no lines

There's no way to preview selected options, so you must create the view, then choose Modify settings and columns → New Today if you want to change it. I tried a few different column and style settings before I settled on those shown above.

3.6.2. Displaying Views in Web Parts

Once I'm happy with the view, I add it to my home page as a web part. To do that:

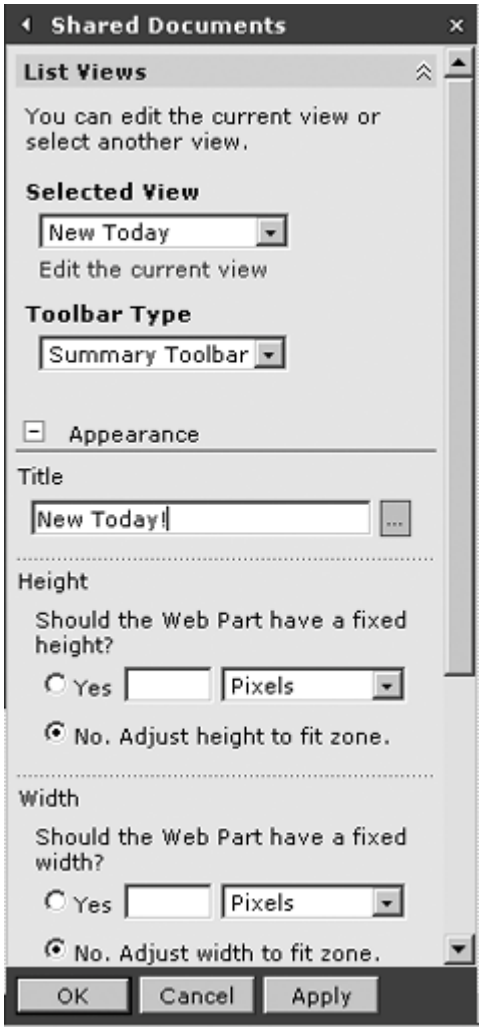
1. On the home page, choose Modify Shared Page → Add Web Parts → Browse. SharePoint displays the Add Web Parts task pane.

2. Select the Shared Documents web part, choose the web part zone, and click Add. SharePoint adds the list to the page as a web part using the default view.

3. Click the title bar on the new web part and select Modify Shared Web Part. SharePoint changes the task pane to edit the web part, as shown in Figure 3-13.

4. In the task pane, change the Selected View to New Today, change the title of the web part, and click OK. SharePoint makes the changes and closes the task panel.

animal 3-13. Changing the view for a list web part



Web parts that use views aren't updated if the view changes. Be sure you the view is final before you use it in a web part; otherwise you'll have to repeat the preceding procedure to see any changes.

3.6.3. Adding Views to List Templates

When you create a custom list template based on an existing list, SharePoint includes all the views you've defined for the list. If you download the template from the list template gallery and extract *manifest.xml* from the downloaded file, you can see the list view definition in XML. The following snippet shows the definition for the New Today view:

```
<?xml version="1.0" encoding="utf-8" ?>
<ListTemplate WebUrl="http://wombat1/Using SharePoint 2003">
    <Details> ... </Details>
    <Files> ... </Files>
    <UserLists>
        <List Name="{FCAA4E9E-BE51-49C9-8A34-C08691D2B0EB}"
        Title="Shared Documents" Description="Share a document with the team by
```

```
            adding it to this document library." Direction="0" BaseType="1"
            ServerTemplate="101" Url="Shared Documents" Version="0">
                <MetaData>
                ...
                    <View Name="{5B8C473E-B55F-4656-86E4-19F1C4C6F982}" Type="HTML"
                    DisplayName="New Today" Url="Shared Documents/Forms/New Today.aspx"
                    BaseViewID="1">
                            <ViewFields>
                                <FieldRef Name="LinkFilenameNoMenu"/>
                                <FieldRef Name="Title"/>
                            </ViewFields>
                            <Query>
                                <Where>
                                    <Eq>
                                        <FieldRef Name="Last_x0020_Modified"/>
                                        <Value Type="DateTime">
                                            <Today/>
                                        </Value>
                                    </Eq>
                                </Where>
                            </Query>
                            <RowLimit Paged="TRUE">100</RowLimit>
                            <ViewStyle ID="16"/>
                            <GroupByHeader>
                                <HTML> ... </HTML>
                        </View>
                    </Views>
                    <Fields>
                        <Field ColName="tp_ID" ReadOnly="TRUE" Type="Counter" Name="ID"
                         DisplayName="ID" FromBaseType="TRUE"/>
                        ...
                    </Fields>
                    <Forms> ... </Forms>
                    <Security> ... </Security>
                    <DocumentLibraryTemplate>Shared
                    Documents/Forms/template.doc</DocumentLibraryTemplate>
                </MetaData>
            </List>
        </UserLists>
        <WebParts> ... </WebParts>
    </ListTemplate>
```

The `ViewFields` element specifies the fields to include in the view. The `Query` element defines the filter applied to the view. You can get help on these and other elements in the SharePoint SDK.

3.7. Creating List Definitions

Site definitions include definitions of the lists they contain. Creating a new list definition is more complex than creating a custom list template, so it is easiest if you start by copying an existing list definition, renaming, and modifying it. List definitions are found on the SharePoint server in the *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\1033\sitedef\LISTS* folder, where *sitedef* is the folder containing the site definition.

To create a new list definition in a site definition:

1. Copy and rename the folder containing an existing list definition.
2. Modify the list definition. The key file to edit is *Schema.xml*.
3. Edit the site definition file (*ONet.xml*) to include the new list definition.
4. Restart SharePoint by running *iisreset.exe*.
5. View a site based on the site definition and create a list based on the new list definition to make sure it works.

For example, to create a new list definition for the DotSites site definition:

1. Copy the *.\DotSites\LISTS\ANNOUNCEMENT* folder and rename the new folder *NewAnnouncements*.
2. Change the list definition *Schema.xml* file as summarized here:

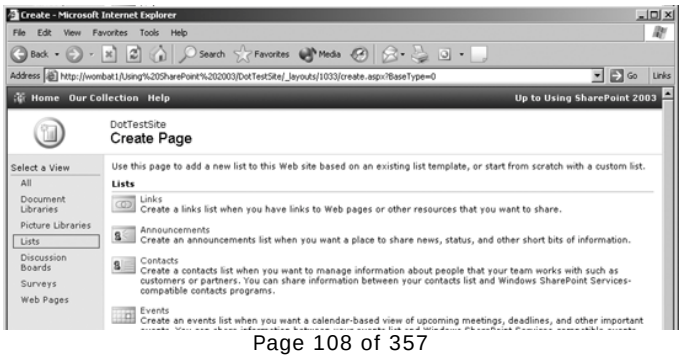
```
<?xml version="1.0" encoding="utf-8" ?>
<List xmlns:ows="Microsoft SharePoint" Name="NewAnnouncements" Title="New Announcements" Direction="0" Url="Lists/NewAnnouncements" BaseType="0" >
    <MetaData>
        <Fields>
            <Field Type="Note" RichText="FALSE" Name="Author" DisplayName="Author"
            Sortable="TRUE" >
            </Field>
            <Field Type="Note" RichText="TRUE" Name="Body" DisplayName="Body"
            Sortable="FALSE" >
            </Field>
            <Field Type="DateTime" Name="Expires" DisplayName="Expires"
            Format="DateOnly" FromBaseType="TRUE" >
            </Field>
        </Fields>
        <Views> ... </Views>
        <Forms> ... </Forms>
        <DefaultDescription>
            This is a demo showing how to create new list definitions.
        </DefaultDescription>
        <Toolbar Type="RelatedTasks"> ... </Toolbar>
    </MetaData>
    <Data>
        <Rows>
            <Row>
            ...
            </Row>
        </Rows>
    </Data>
</List>
```

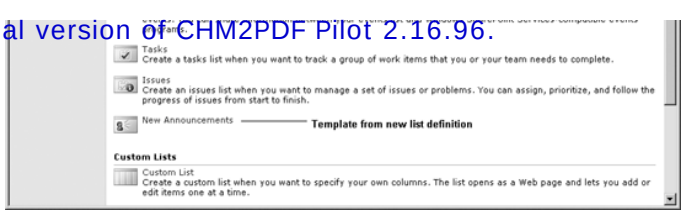
3. Edit *ONet.xml* in *.\DotSites\XML* to include the new list definition. The changes are made in several places, as shown by the following snippet:

```
<?xml version="1.0" encoding="utf-8" ?>
<Project Title="Team Web Site" ListDir="Lists" xmlns:ows="Microsoft SharePoint">
    ...
    <ListTemplates>
    ...
        <ListTemplate Name="NewAnnouncements" DisplayName="New Announcements" Type="204"
        BaseType="0" OnQuickLaunch="FALSE" SecurityBits="11" Description=""
        Image="/_layouts/images/itann.gif">
        </ListTemplate>
    </ListTemplates>
    <Configurations>
        <Configuration ID="0" Name="Default">
            <Lists>
                ...
                <List Title="New Announcements" Type="204"
                   Url="Lists/NewAnnouncements" />
            </Lists>
        </Configuration>
    </Configurations>
    ...
</Project>
```

4. Restart SharePoint.
5. Open a test site based on the DotSites template and create a new list based on the list definition to make sure the changes worked. The new definition shows up on the Create page, as shown in Figure 3-14.

animal 3-14. Testing a new list definition





PREV NEXT

3.8. Modifying Themes

Themes control the color scheme and backgrounds used by SharePoint sites. You can change a site's theme by selecting Site Settings → Apply theme to site. The themes that appear on the Apply Theme page come from the *.\TEMPLATES\THEMES* folder on the SharePoint server. Figure 3-15 illustrates the structure of the theme definitions.

To change a theme:

1. Edit the files in the theme's folder.
2. Restart SharePoint by running *iisreset.exe*.

Changes to a theme automatically appear in sites based on the theme after SharePoint restarts.

To create a new, custom theme based on an existing theme:

1. Copy and rename a theme's folder in *.\TEMPLATES\THEMES*.
2. Edit the files in the new theme folder.
3. Rename and edit the *.inf* file in the theme folder to match the theme's name.
4. Edit the file *SPTHEMES.xml* in the *\TEMPLATE\LAYOUTS\1033* folder. This file loads the theme definitions in SharePoint.

animal 3-15. Viewing themes on the server

The theme's *.inf* file contains the theme's title and localized name displayed for different languages, as shown here:

```
[info]
title=NewTheme
codepage=65001
version=3.00
```

```
format=2.00
readonly=true
refcount=0

[titles]
1031=Eis
...
```

The *SPTHEMES.xml* file tells SharePoint where to find the theme definitions. For example, the following snippet adds NewTheme (based on Ice) to the list of available themes:

```
<?xml version="1.0" encoding="utf-8" ?>
<SPThemes xmlns="http://tempuri.org/SPThemes.xsd">
    ...

    <Templates>
        <TemplateID>newtheme</TemplateID>
        <DisplayName>New Theme</DisplayName>
        <Description>Demo theme</Description>
        <Thumbnail>../images/thice.png</Thumbnail>
        <Preview>../images/thice.gif</Preview>
    </Templates>
</SPThemes>
```

You don't have to restart SharePoint for the new theme to be available.

3.9. Applying Style Sheets

SharePoint uses cascading style sheets (*.css*) to control the fonts and the background and foreground colors used by the sites on the SharePoint server. The default style sheet is *OWS.css* which is found in *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\TEMPLATE\LAYOUTS\1033\STYLES*.

Changes to *OWS.css* affect all SharePoint sites on the server. For example, making the following change turns all of the body text on the server red:

```
body {
    font-family: verdana, arial, helvetica, sans-serif;
    background: white;
    color: red;
}
```

Sites using themes are also affected by *THEME.css* found in the .\TEMPLATES\THEMES* folders. You can edit *THEME.css* to change the styles applied by a specific theme. For example, the following change to *THEME.css* in the *NewTheme* folder changes the page title font color for the theme created in the preceding section:

```
.ms-pagetitle{
color:black;
font-family:Verdana,Arial,Helvetica,sans-serif;
font-weight:bold;
}
```

Changing styles in a theme doesn't immediately change existing sites based on that theme, because style sheets are cached on the client. If you don't see the changes you've made, force a full refresh of the page by pressing Ctrl+F5.

Identifying the class name of styles that SharePoint uses on a page can be difficult. One way to determine the class name of the items on a page is to change the `body` element of *default.aspx* (or any other SharePoint page) to include the following event procedure:

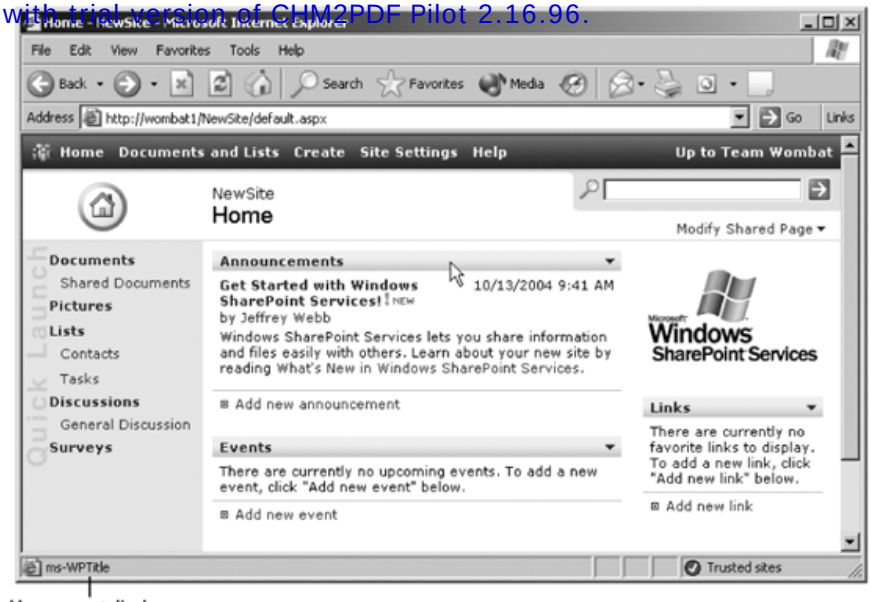
```
<body marginwidth="0" marginheight="0" scroll="yes" onmouseover="window.status = window.event.srcElement.className">
```

Now, when you move the mouse over an item on the page, the status bar displays the class name for that item, as shown in Figure 3-16.

The SharePoint SDK contains a table of the class names in the "CSS Class Definitions" topic.

When changing *default.aspx*, or any SharePoint page, you need to know whether or not that page is ghosted. Ghosted pages are shared among all sites and can be

animal 3-16. Finding an item's class name



Mouse event displays style class name in status bar

changed by editing the *.aspx* file directly in the site definition. Pages that are not ghosted are stored in the site's content database and can be edited by opening the site using FrontPage.

PREV NEXT

3.10. Changing the Default Icons

SharePoint gets its default icons from the *.\TEMPLATE\LAYOUTS\1033\IMAGES* folder. You can change the image files found there to customize the icons used throughout SharePoint.

3.10.1. Resources

To get	Look here
The SharePoint SDK	Search http://www.microsoft.com/downloads for "SharePoint SDK."
Information about built-in lists	SharePoint SDK Appendix, "Field Tables for Default Lists"
CabArc, *Extract*, and other cabinet file utilities	http://msdn.microsoft.com/library/en-us/dncabsdk/html/cabdl.asp
Internet Explorer Tools for Validating XML (*iexmltls.exe*)	Search http://www.microsoft.com/downloads for "Validating XML."
A list of class definitions used in SharePoint style sheets	http://msdn.microsoft.com/library/en-us/spptsdk/html/tsovCSSStyles.asp
A tutorial on how to create/use cascading style sheets (CSS)	http://www.w3schools.com/css/

Chapter 4. Sharing Contacts and Meetings with Outlook

Outlook can use SharePoint two main ways:

- Sharing address book entries with team members through contacts lists
- Automatically creating workspaces for meetings requested from the Outlook Calendar

This chapter discusses these tasks from a user's perspective. If you are an administrator, you can use this chapter to help educate Outlook users on use of the new SharePoint features with Outlook.

4.1. Sharing Contacts

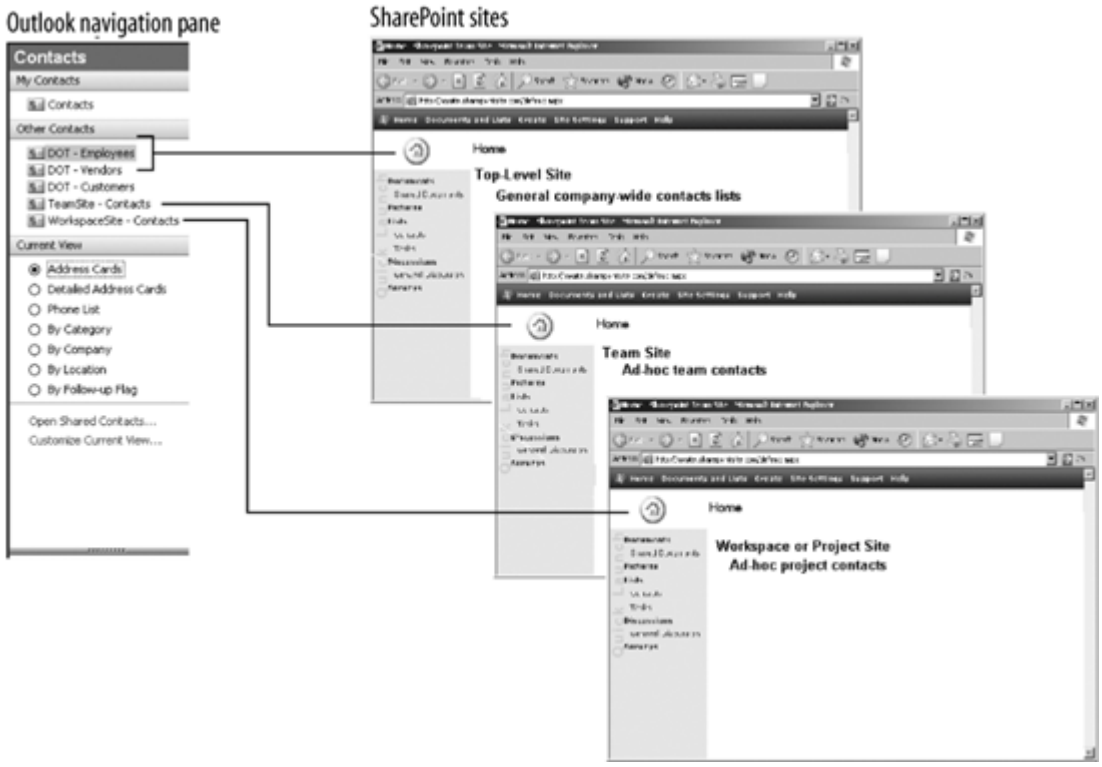
Outlook provides a set of tools for viewing and maintaining the list of contacts in your address book. If your company uses Exchange Server, you may already use a public list of contacts from Outlook to contact members of your organization. SharePoint provides another way to share contacts from your address book with others. Rather than providing a single public list containing everyone's information, SharePoint is focused more on team-based or project-based lists of contacts.

For example, a Team Site might include team members in the contacts list. Later, as new members join and lines of communication are established across groups, the contacts list grows. In this case, the contacts list is a way to share the collected knowledge of who the key people are and how to get a hold of them.

For a project site or a document workspace, the contacts list obviously includes everyone with responsibilities on the project. Outside resources, such as salespeople or customers, can be added as they become available. Of course, you can also use SharePoint to share a general, company-wide list of contacts. One advantage of that approach is that SharePoint contacts are easily shared over the Internet.

Finally, there's nothing stopping you from using all these approaches to help organize contacts by company, team, and project, as shown by the Outlook Navigation pane in Figure 4-1.

animal 4-1. Shared lists can organize contacts by company, team, or project



It's not a great idea to add the same contact to multiple lists because if the contact's information changes, it then has to be changed in all the lists. Instead, it's a good idea to follow rules about where you store contacts and how you use them. Here are some suggestions:

- Decide whether you are going to use SharePoint or Exchange Server to share company-wide contacts. It doesn't make much sense to use both.
- If you are using SharePoint for company-wide contacts, organize those contacts into one or more lists at the top-level site.
- Restrict who can add/change contacts in the top-level SharePoint lists.
- Use project/workspace contacts lists as temporary resources that have a limited lifetime.

A company might provide Employee, Customer, and Vendor contacts lists in their top-level site that can't be edited by most members, but also allow team members to create their own ad-hoc contacts lists in team sites and workspaces. Although the ad-hoc lists might become out-of-date, they allow members to organize the contacts that the team needs and perhaps include contacts that don't belong in the company-wide lists.

The following sections explain how to work with SharePoint contacts lists in Outlook.

4.1.1. Creating Contacts Lists

SharePoint team sites and document workspaces include a list of contacts by default. That list is simply named "Contacts," and it appears on the Quick Launch bar of the site's home page. You can rename that list or create new contacts lists for different types of contacts. To rename the default contacts list:

1. Display the list in the browser.
2. Select Modify settings and columns → Change general settings.
3. Type the new name in the Name field and click OK.

Changing the name of a list changes the name displayed on the list page and on the Quick Launch bar. It also determines the name of the list displayed in Outlook Navigation pane if you link the list to Outlook.

You may want to create more than one contact list for a site. For example, you might want to organize employees, customers, and vendors into separate lists so you can better control who has access to the different lists. To create additional contacts lists:

1. Select Create from the Navigation bar; then choose Contacts from the lists section of the Create page.
2. Enter a name for the new contacts list, select whether or not a link to the list should appear on the home page Quick Launch bar, and click Create.

This creates a new list based on the contacts template. Lists based on the contacts template can be linked to Outlook as shared lists. To control access to a list:

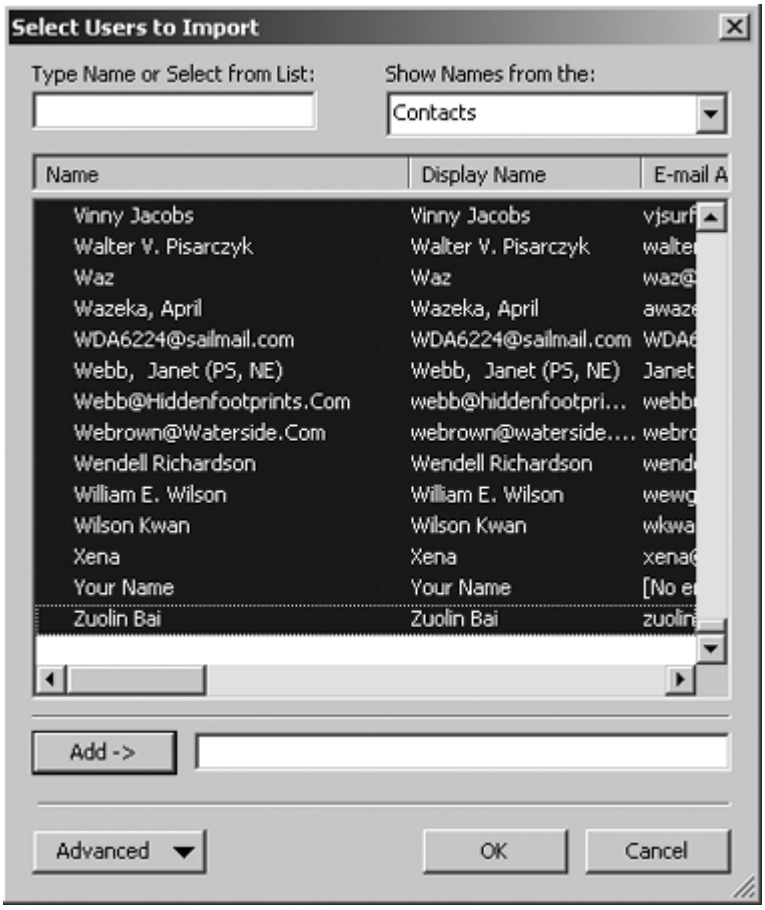
1. Display the list in the browser.
2. Select Modify settings and columns → Change permissions for this list.
3. Select a group and click Remove Selected Users to prevent members of that group from viewing the list; or select a group and click Edit Permissions of Selected Users to change the access privileges of those members.

4.1.2. Copying Contacts from Outlook to SharePoint

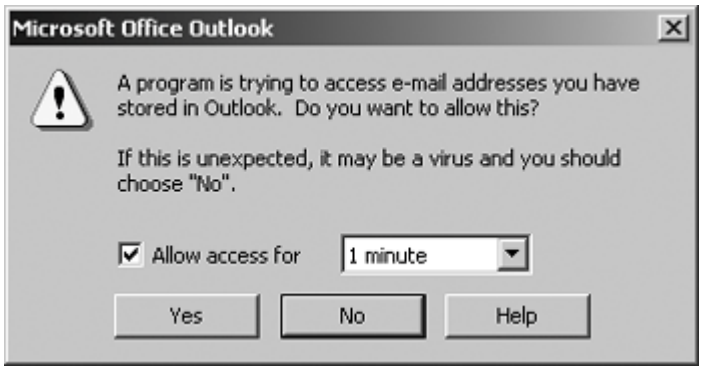
You can copy one or more contacts from Outlook to a SharePoint web site by following these steps:

1. Display the SharePoint site in your browser and click Contacts on the site's home page. You'll see the Contacts list for the site.
2. Click Import Contacts to see a list of the contacts from your local address book.
3. You can import some or all of your contacts. To import some of the contacts, hold down the Ctrl key while clicking on the contacts to import. To import all of the contacts, click the first contact, scroll to the end of the list, hold down the Shift key, and click the last contact. Figure 4-2 shows the importing all of the contacts.
4. Click OK to import the selected contacts. You'll see a security warning indicating that SharePoint is trying to access your address book, as shown in Figure 4-3. Select Allow access for 1 minute and click Yes to allow the import to continue.
5. When the operation completes, the new contacts appear in the Contact list as shown in Figure 4-4.

animal 4-2. Selecting the contacts to import



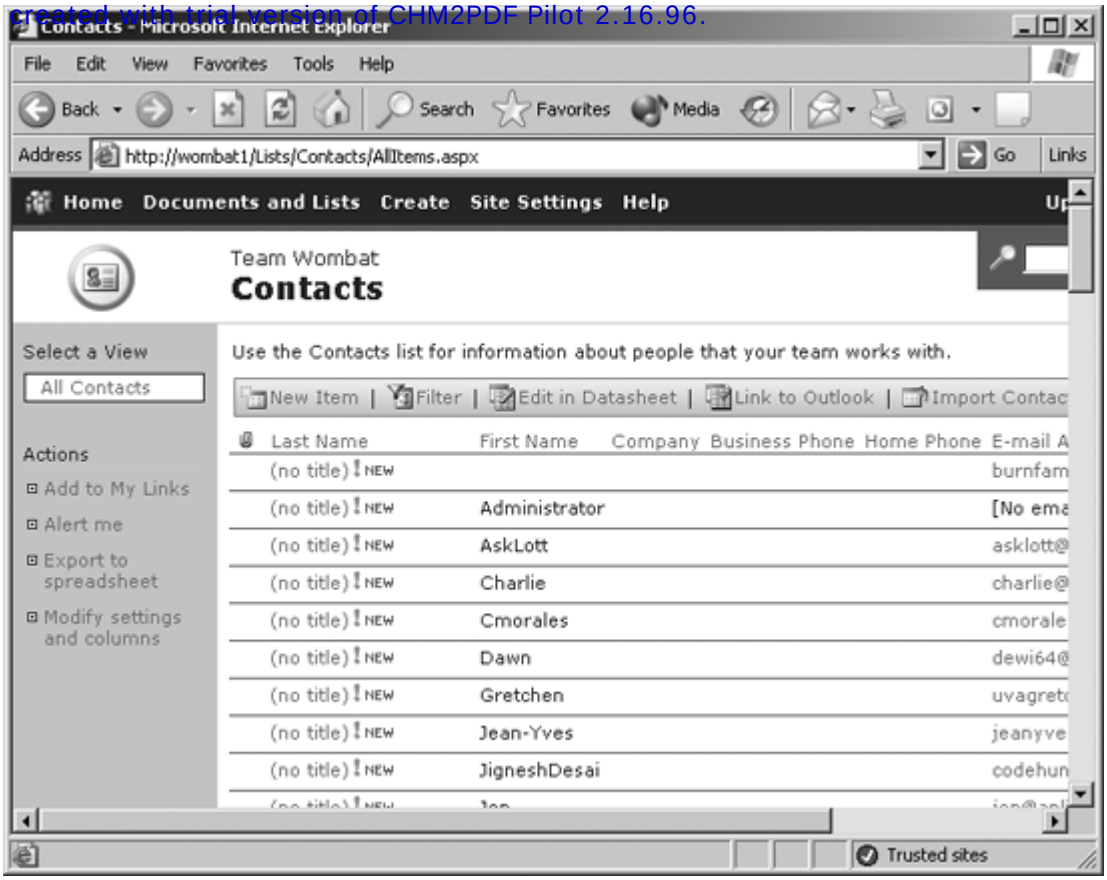
animal 4-3. Allowing access to the address book



4.1.3. Editing Shared Contacts Quickly

If you're like me, the contacts in your address book are kind of sloppy (see Figure 4-4). Many of my contacts don't have proper first and last names or address and phone information, something that often happens when I add contact information from a piece of email but don't bother to fill out all the fields correctly.

animal 4-4. Displaying the imported contacts



You can edit contacts in your Outlook address book before you import them into the Contacts list, or you can do it from SharePoint after importing. Perhaps the easiest way to clean up entries is to click Edit in Datasheet on the Contacts page shown in Figure 4-4. The datasheet view is a lot like a spreadsheetyou can select items to delete, drag names between fields, and do other edits quickly, as shown in Figure 4-5.

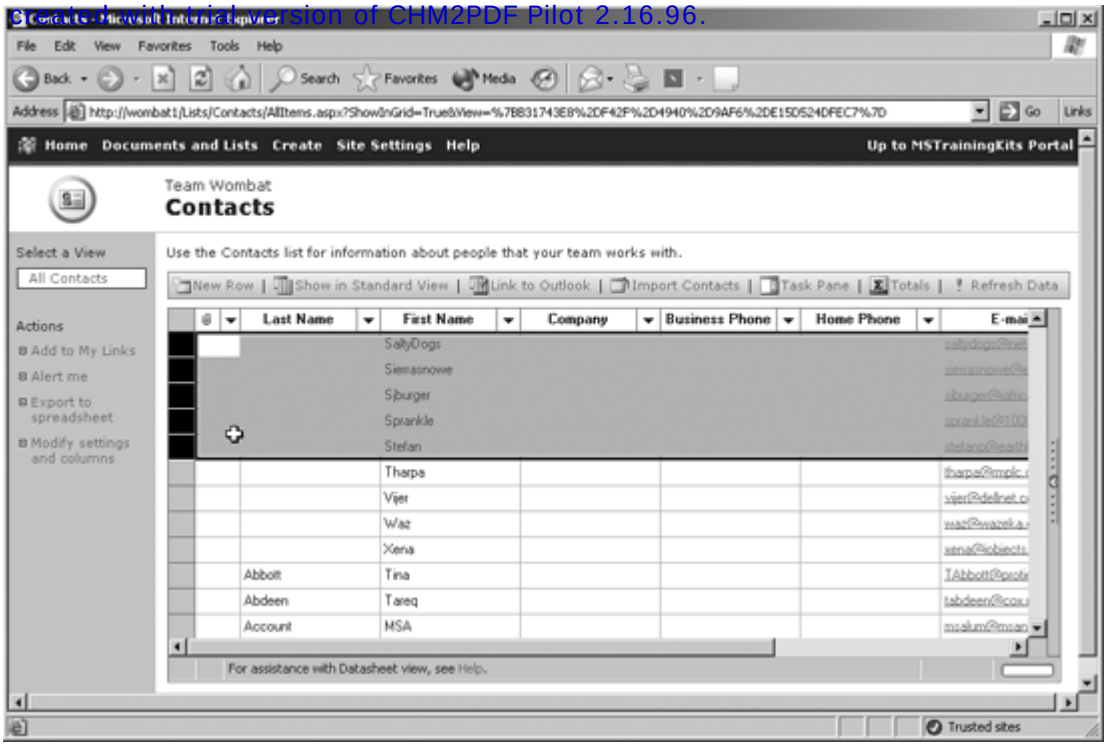
4.1.4. Linking SharePoint Contacts to Outlook

Once you've cleaned up the Contacts list, you can link the list back to Outlook. To do that:

1. Display the contacts list in the browser and click Link to Outlook.

2. Outlook displays a security warning. Click Yes to import the SharePoint Contacts list into Outlook. When finished, Outlook shows the shared list in the Other Contacts section of the Contacts Navigation pane as shown in Figure 4-6.

animal 4-5. Editing contacts quickly in the datasheet view



animal 4-6. Linking the SharePoint contacts back to Outlook

4.1.5. Editing Shared Contacts from Outlook

The SharePoint contacts list is stored on the SharePoint server so it can be used by all members of the site. That means you can't edit those contacts directly from Outlook. Instead, you must follow these steps:

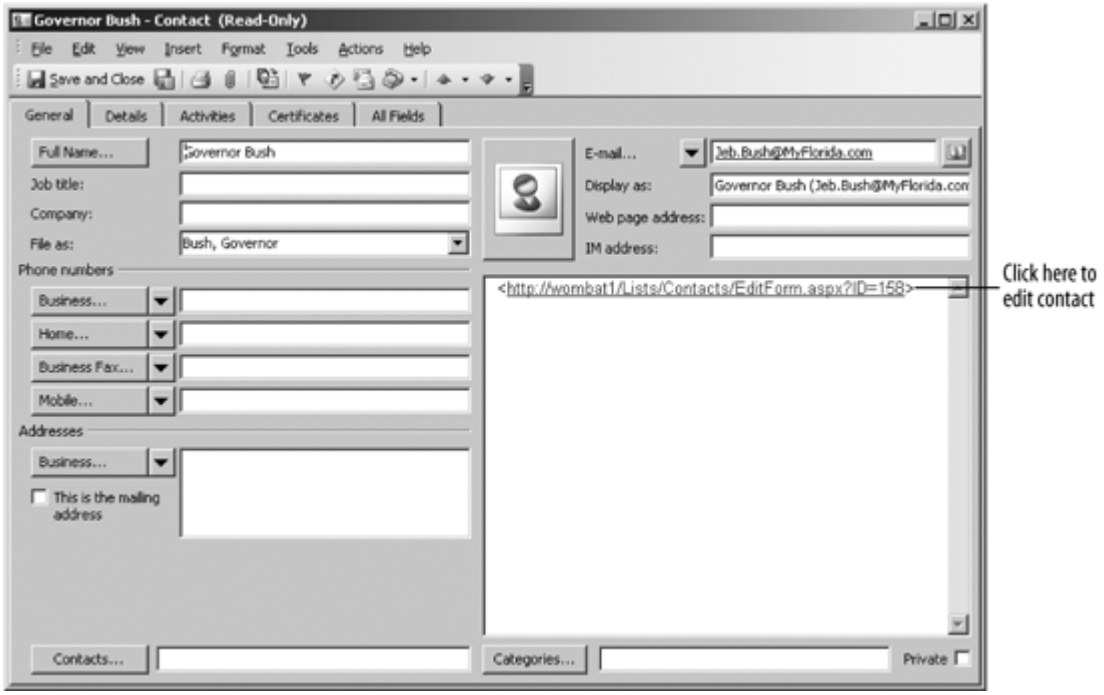
1. Open the contact to change in Outlook. Outlook displays the detailed contact information as read-only, as shown in Figure 4-7.

2. Click the link in the edit area. Outlook opens the SharePoint edit page for the contact, as shown in Figure 4-8.

3. Make your changes and click Save and Close to complete the change.

4. Return to Outlook and close the Contact dialog box.

animal 4-7. Changing shared contacts from Outlook

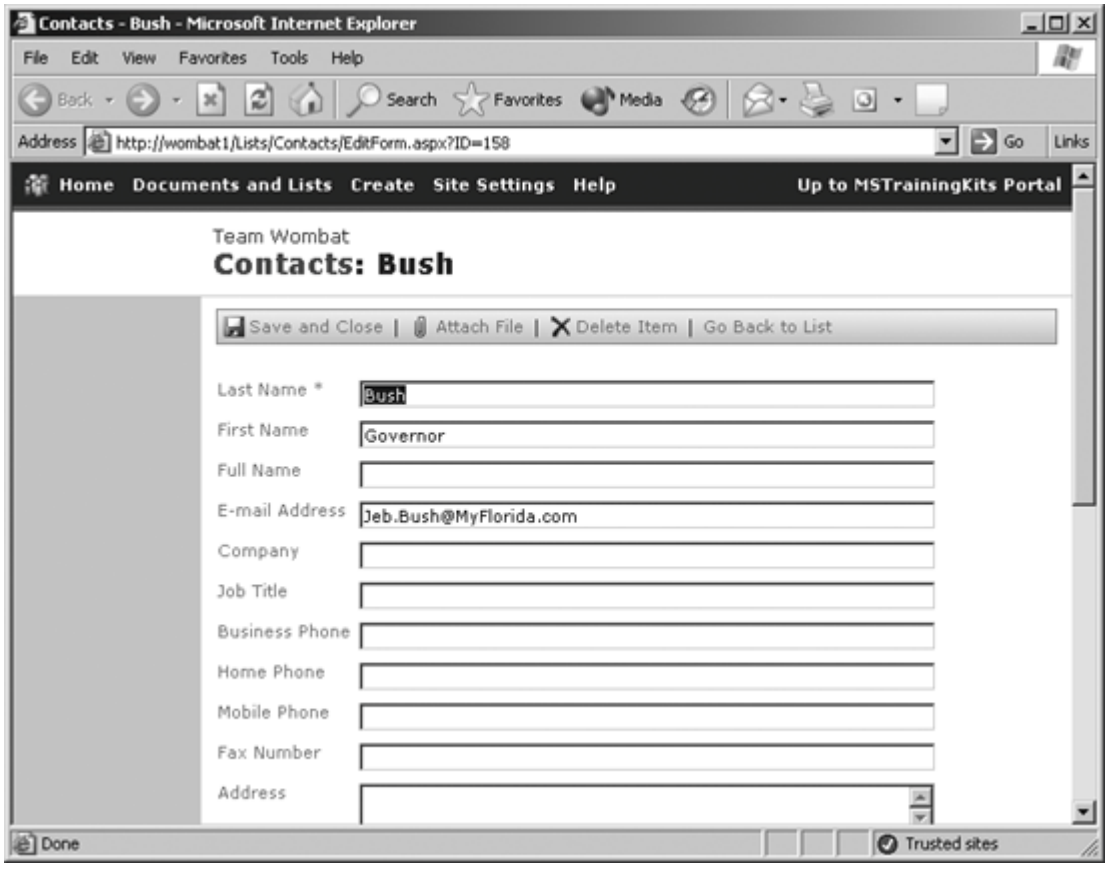


If you view the contact in Outlook right away, you won't see the changes you just made. Outlook updates its shared lists when the Outlook application starts and once every 20 minutes after that. To refresh the list immediately:

1. Right-click the shared contacts list in the Navigation pane (Figure 4-6).

2. Select Refresh from the context menu.

animal 4-8. Saving changes to the shared contact



4.2. Organizing Meetings

SharePoint provides a special type of site called a *meeting workspace*, which can be created from meeting requests sent from Outlook. Meeting workspaces are meant to prepare attendees by publishing the objectives and agenda before a meeting is held, and after the meeting takes place, they help record decisions and related documents.Workspaces are used to organize the meeting process like this:

1. Attendees receive a meeting request in Outlook. The request links to the SharePoint workspace.
2. Attendees can click on the link to see details about the meeting and add items as needed.
3. During the meeting a participant can optionally open the workspace from a laptop and make notes.
4. The meeting organizer can later use the workspace to record conclusions, assign follow-up tasks, or add key documents.

Meeting workspaces aren't meant to be online meeting places, but they can be used in conjunction with Microsoft NetMeeting, Exchange Conferencing, or other online meeting services.

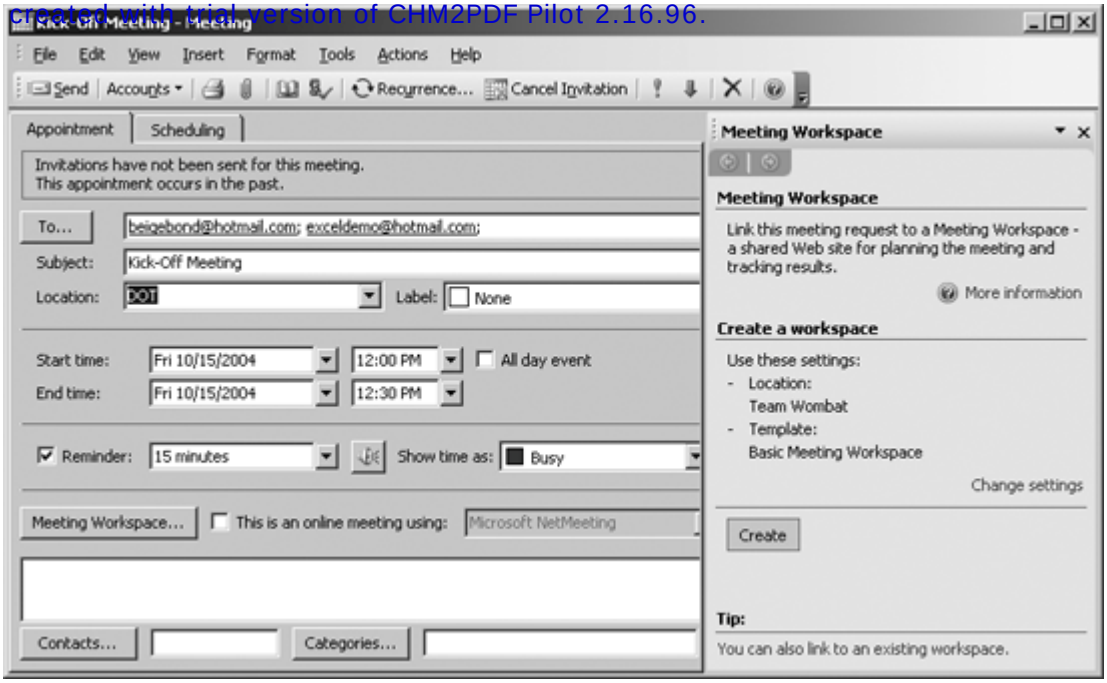
The following sections explain how to create SharePoint meeting workspaces from Outlook.

4.2.1. Creating a Meeting Workspace

To create a meeting workspace from Outlook:

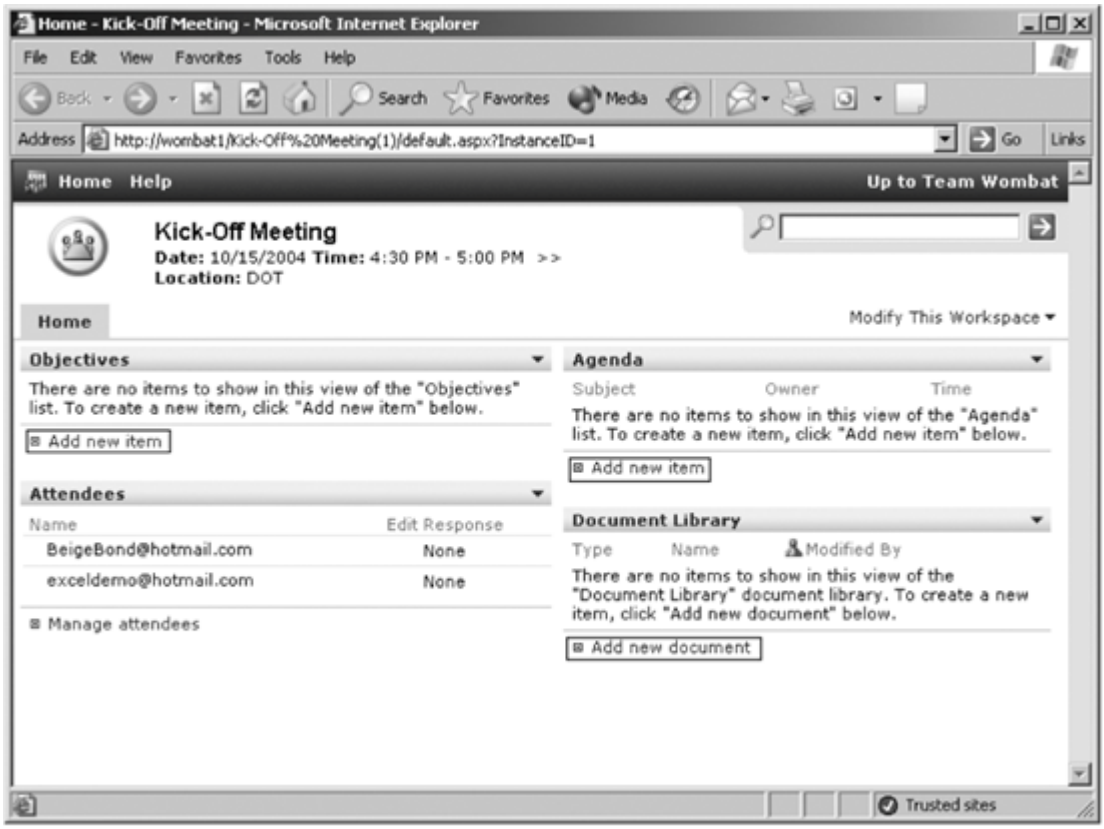
1. Select the Calendar in the Navigation pane to choose a date and time for the meeting.
2. Choose Action → New Meeting Request. Outlook displays the Meeting Request dialog box.
3. Complete the fields on the dialog box and click Meeting Workspace. Outlook displays a workspace task pane in the dialog box, as shown in Figure 4-9.
4. Click Create. Outlook creates a Meeting Workspace for the meeting and adds a link to the workspace in the dialog box.
5. Click the link and add objectives and agenda items to the workspace (Figure 4-10).
6. Return to Outlook and click Send to close the dialog and send the meeting request. The request includes a link to the meeting workspace so attendees can review the objectives and agenda, and add documents before attending.

animal 4-9. Creating a meeting workspace



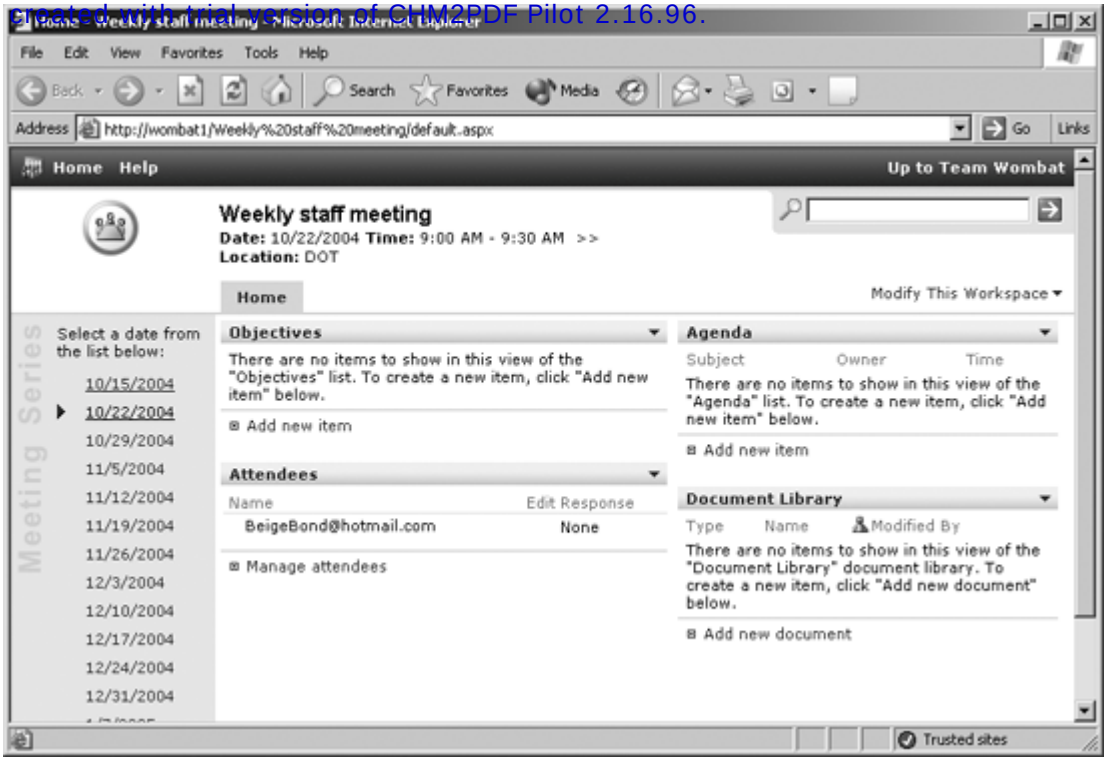
The workspace shown in Figure 4-10 is for a meeting that happens only once. Recurring meetings create a different type of workspace called a *meeting series*. To create a meeting series from Outlook, choose Action → New Recurring Meeting instead of Action → New Meeting Request in step 2 of the preceding procedure. Meeting series workspaces list recurring meetings by date, as shown in Figure 4-11.

animal 4-10. Adding objectives and agenda items for the meeting



Click Add new item to add Objectives and Agenda items before sending the meeting request.

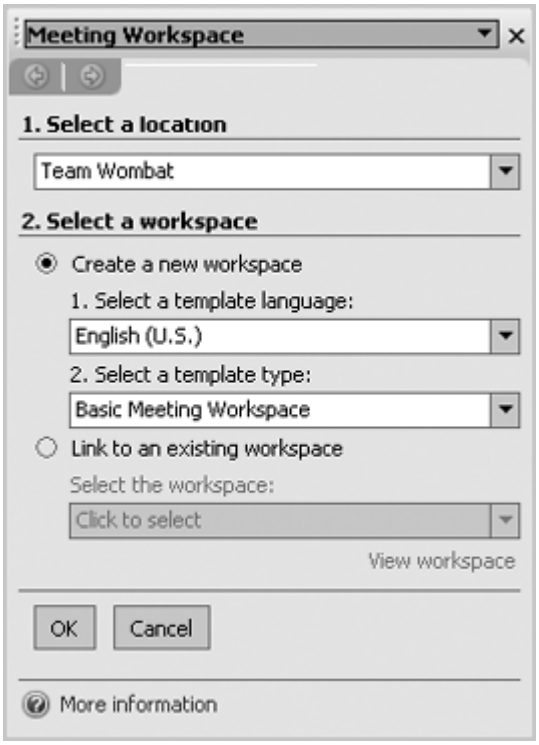
animal 4-11. Using a meeting series for recurring meetings



4.2.2. Creating Different Types of Workspaces

Outlook lets you create different types of meeting workspaces or link a meeting to an existing workspace. To see the different options before creating a meeting workspace, click Change Settings in the meeting workspace task pane (Figure 4-9). Outlook displays the options shown in Figure 4-12.

animal 4-12. Choosing other workspace options



By default, Outlook creates a Basic Meeting workspace that includes Objectives, Attendees, and Agenda lists. Figure 4-12 illustrates creating workspaces based on different templates, as described in Table 4-1.

Table 4-1. Other meeting workspace templates

Template	Use to

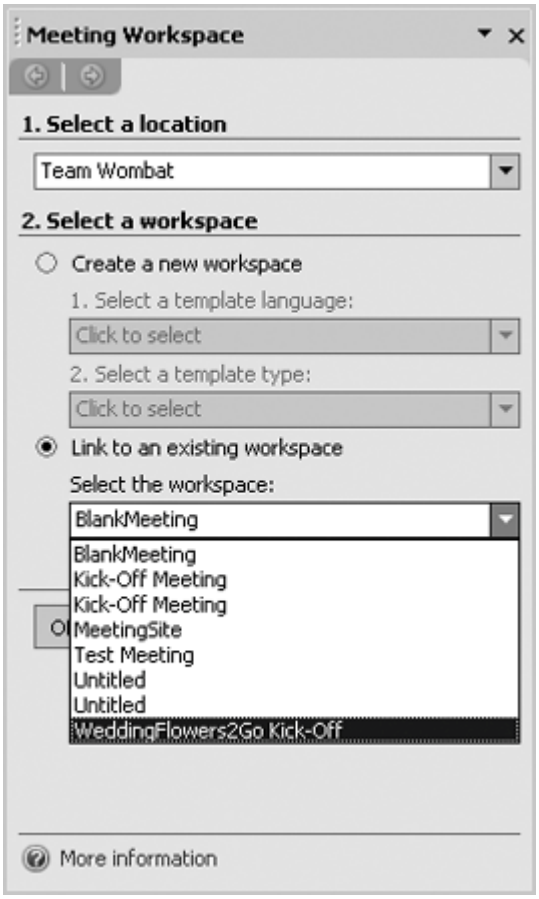
Blank Meeting	Start from scratch, adding your own lists and content.
Decision Meeting	Include Tasks and Decisions lists in addition to the basic lists.
Social Meeting	Include Directions, Things to Bring, Pictures, and Discussions lists.
Multipage Meeting	Include the basic lists and add two blank pages for additional information.
Custom template	Create a custom meeting workspace from a custom template installed on your site.

4.2.3. Linking to an Existing Workspace

In some cases, you may have already created the workspace before you send the meeting request in Outlook. In that case, you can use the options illustrated in Figure 4-12 to link the request to the existing workspace. To do that:

1. Select the team site that contains the meeting workspace in step 1 on Figure 4-12.
2. Select Link to an existing workspace in step 2.
3. Select the workspace from the drop-down list, shown in Figure 4-13, and click OK.

animal 4-13. Linking a meeting request to an existing workspace



4.3. Resources

To get	Look here
Information on changing the update interval for shared contacts	http://www.microsoft.com/office/ork/2003/three/ch8/OutC04.htm
Training kit for SharePoint	To view: http://usingsharepoint.com/sptraining/
Portal Server	To install: http://www.microsoft.com/sharepoint/downloads/components/detail.asp?a=631

Chapter 5. Sharing Workspaces and Lists with Excel

Excel can use SharePoint a number of different ways:

- Share workbooks through document workspaces
- Maintain version history of workbooks through document libraries
- Share parts of a worksheet through lists
- Publish completed workbooks as web pages
- Display worksheets on web pages through web parts
- Program SharePoint from Excel VBA

This chapter discusses these tasks from a user's perspective. If you are an administrator, you can use this chapter to help educate Excel users on new SharePoint features.

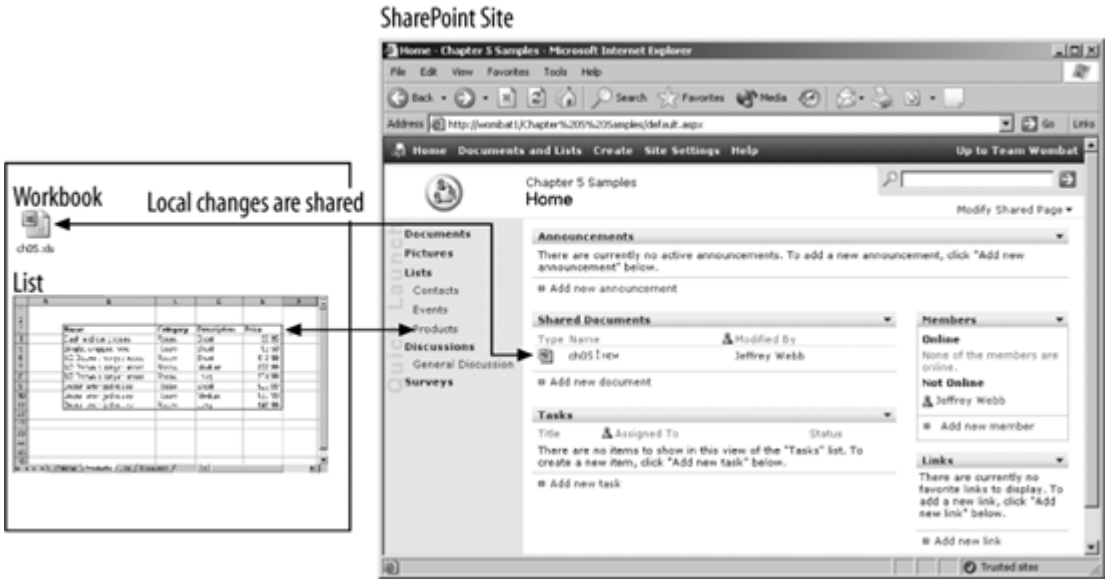
5.1. Getting Started with Excel and SharePoint

If you've ever tried to collaborate on a budget, project bid, or other team-oriented workbook in Excel, you know that just putting the *.XLS* file up on a public server doesn't cut it. Only one person can edit the workbook at a time, and if somebody leaves the file open and goes to lunch, the rest of the team is locked out until you find out who the culprit is and get him to close his session.

Earlier versions of Excel solved this problem with *shared workbooks*, which let more than one user have a single workbook open for editing. Changes were merged automatically, and conflicting changes could be resolved. That's a start, but it's a file-based system, so there is no way to manage the document's users, notify teammates of changes, assign tasks, or do other team-oriented work.

SharePoint solves this problem through shared workspaces and shared lists. From an Excel perspective, *shared workspaces* represent a workbook shared with team members; *shared lists* are ranges of cells shared through a SharePoint site. Figure 5-1 illustrates the difference.

animal 5-1. Sharing workbooks and parts of workbooks as lists



Before you begin using SharePoint in Excel, you'll need:

- The address of the site assigned to your team. SharePoint organizes sites into separate folders for each team or project. Your SharePoint administrator will probably have set up a server with a top-level site that everyone can view, but to create new workspaces from Excel, you'll need the address of an area where you are allowed to create new subsites.

- The Internet address of your site (optional). Usually the address you get from your administrator will be an *intranet address*that is, it will be located on your corporate network. However, if you travel or work from home, your administrator may also give you an address that can be accessed over the Internet.

- Your user name and password. This will usually be the same user name and password you use to sign on to the network. However, if you are using SharePoint over the Internet, you may have a new user name or password assigned.

Next, take these preliminary steps from Internet Explorer:

1. Navigate to your SharePoint site and add it to your Favorites folder or create a shortcut on your desktop.

2. If using SharePoint over the Internet, add the Internet site to your Trusted Sites list. From within Internet Explorer, select Tools → Internet Options → Security, and add the address to the Trusted Sites zone. Don't include the sitename or teamname in the addressjust use the domain

(for example, http://www.mycompany.com).

You're almost ready to start sharing workbooks and lists from Excel using SharePoint. How you do each of those things is different so, first-things-first, let's talk about sharing an Excel workbook by creating a SharePoint workspace.

5.2. Sharing Workbooks

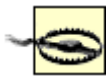
To share a workbook through SharePoint:

1. Open the workbook in Excel and select Tools → Shared Workspace. Excel displays the Shared Workspace task pane, as shown in Figure 5-2.

2. Type the address of your SharePoint site in the Location box and click Create.

3. The SharePoint site may ask you to sign in. Enter your user name and password and click OK.

4. Once the workbook is shared, Excel changes the task pane. You can click on Open site in browser to view the new shared workspace.

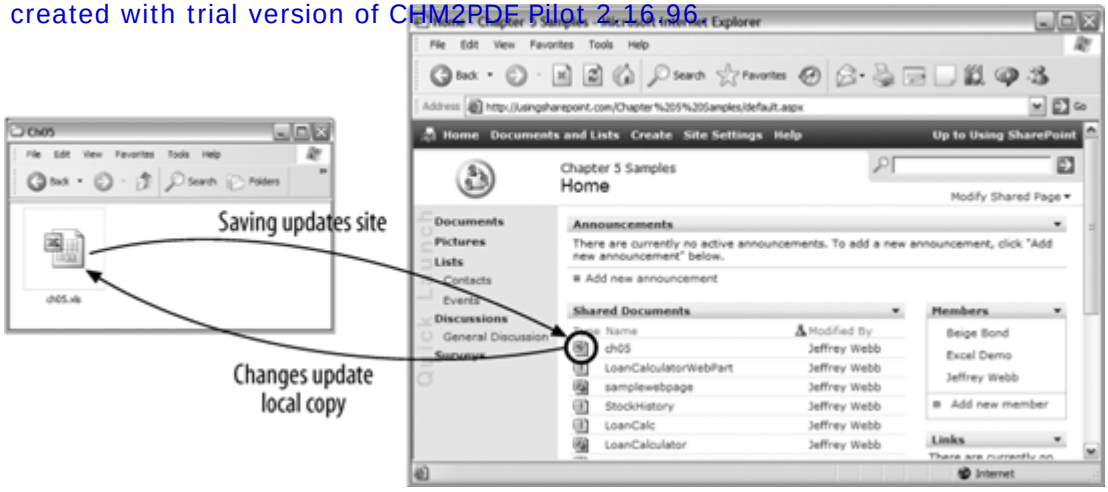


Don't confuse Tools → Shared Workspace with File → Save Workspace. The latter creates an *.xlw* file that stores your Excel windows and open documents. Also don't confuse it with Tools → Share Workbook. Share Workbook is the old way of allowing multiple authors to edit a workbook at the same time.

animal 5-2. Sharing a workbook using the Excel task pane

Excel connects to the SharePoint site and creates a new document workspace for each workbook you share from Excel. The workbook stored locally on your computer is now linked to the workbook stored on SharePoint. If you save the workbook, changes are saved locally and then sent to the server. If you close and then reopen the local workbook, Excel connects to SharePoint to get any changes from others as shown in Figure 5-3. See the section "Reconciling Changes and Viewing History" later in the chapter for an explanation of how to manage changes made by others.

animal 5-3. Sharing a workbook links it to the SharePoint site



Sharing the workbook is really only the first step for using SharePoint from Excel. After you share a workbook you can:

- Add other documents to the workspace. New workspaces contain only one document, but you will probably want to add others.
- Add members. By default, SharePoint includes only you in the workspace, and you must grant others access before they can edit documents in the workspace.
- Send alerts. SharePoint automatically notifies new members when you add them to a workspace. Alerts are typically used to notify members of changes or approaching deadlines.
- Assign tasks. Workspaces help track tasks assigned to team members.
- Check files in and out. Checking a file out prevents others from changing it.
- Reconcile changes. If a file is not checked out, members must reconcile their changes with the changes of others.
- View document history.

The following sections describe how to accomplish those tasks from Excel through the Shared Workspace task pane.

5.2.1. Using the Task Pane

The Shared Workspace task pane gives you access to the SharePoint site from within Excel. Each of the icons at the top of the task pane maps to an equivalent task that you can perform from the SharePoint site itself. Table 5-1 describes the icons and lists the tasks you can perform with them.

Table 5-1. Shared Workspace task pane buttons

Button	Name	Use to
	Status	Check whether the open document is up to date.
	Members	View or add team members to the workspace.
	Tasks	View or assign tasks for workspace team members.
	Documents	Add documents to the workspace or open other documents from the workspace.
	Links	Add or view links to related information.
	Document Information	View the document's revision history.

The Shared Workspace task pane is the same in Excel, Word, and PowerPoint.

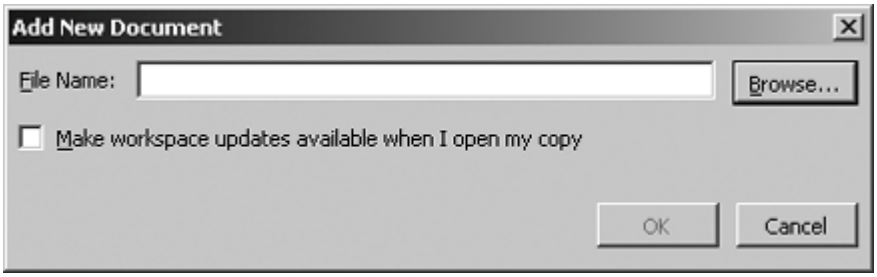
Because it's hard to include icons throughout a book like this, I instead use the names in Table 5-1 when referring to task pane buttons.

5.2.2. Adding Documents

Each new workspace has its own folder on the site. To add other documents to the workspace:

1. In the task pane, click Documents → Add new document. SharePoint displays Figure 5-4.
2. Click Browse to select a file to add from your local computer. Select Make workspace updates available to link the local copy of the document to the copy stored on SharePoint.
3. Click OK to copy the file to SharePoint and add it to the document workspace. The new document appears in the task pane.

animal 5-4. Adding other documents to the workspace

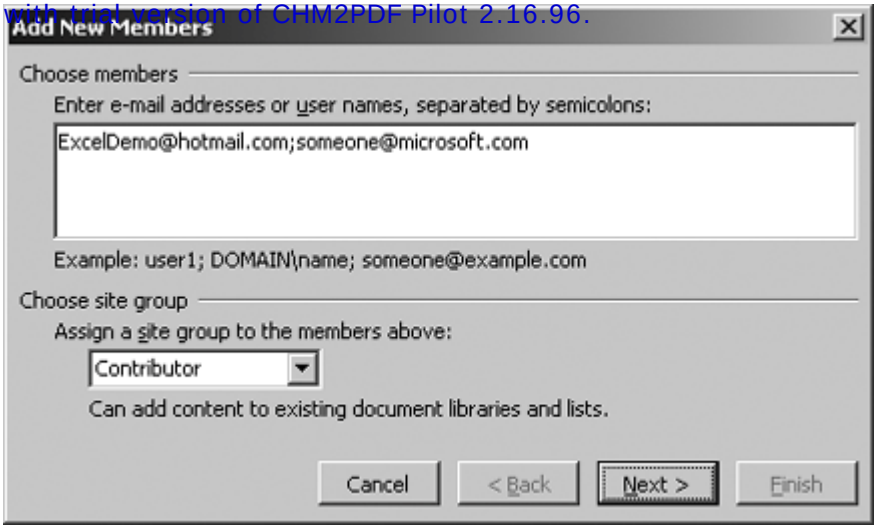


5.2.3. Adding Members

When you create a new workspace, only you are added to the list of members. For others to be able to view or edit files from the workspace, you'll need to add them as members. To add members to a workspace:

1. In the task pane, click Members → Add new members. Excel displays Figure 5-5.
2. Enter the email addresses of the users to add; select the level of access in Choose site group; click Next. Excel displays the list of members to add (Figure 5-6).
3. Click Finish to add the members. Excel displays a dialog telling you that the members were added and asking if you want to send them an email message.
4. Click OK to complete the task. If you left Send an email selected, Outlook displays the message so you can edit it before sending (Figure 5-7).

animal 5-5. Adding new members to a workspace



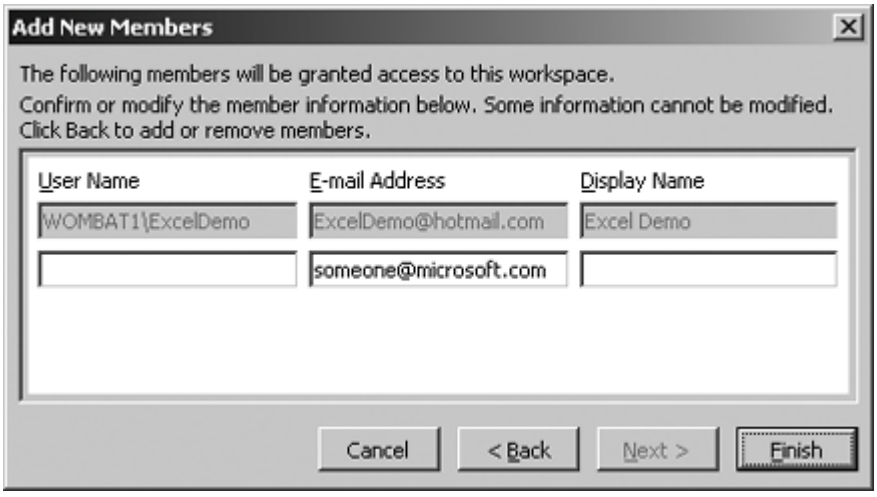
The site groups in Figure 5-5 determine the privileges granted to the new members. You can assign different levels of access, as described in Table 5-2.

Table 5-2. Site groups used to assign permissions to a workspace

Setting	Permission
Reader	Read-only access to the site.
Contributor	Add content to existing document libraries and lists.
Web Designer	Create lists and document libraries and customize pages in the workspace.
Administrator	Full control of the workspace.
Custom	You can define custom groups. Ask your site administrator about other levels of access if you need them.

SharePoint checks the email addresses of the members you entered in Figure 5-5. If it can't match the address with a valid account on the SharePoint server, the account information is left blank as shown in Figure 5-6. Typically, everyone on your network will have an account on the SharePoint server, but members outside your network will need to have an account set up by your SharePoint administrator.

animal 5-6. Identifying members SharePoint doesn't recognize

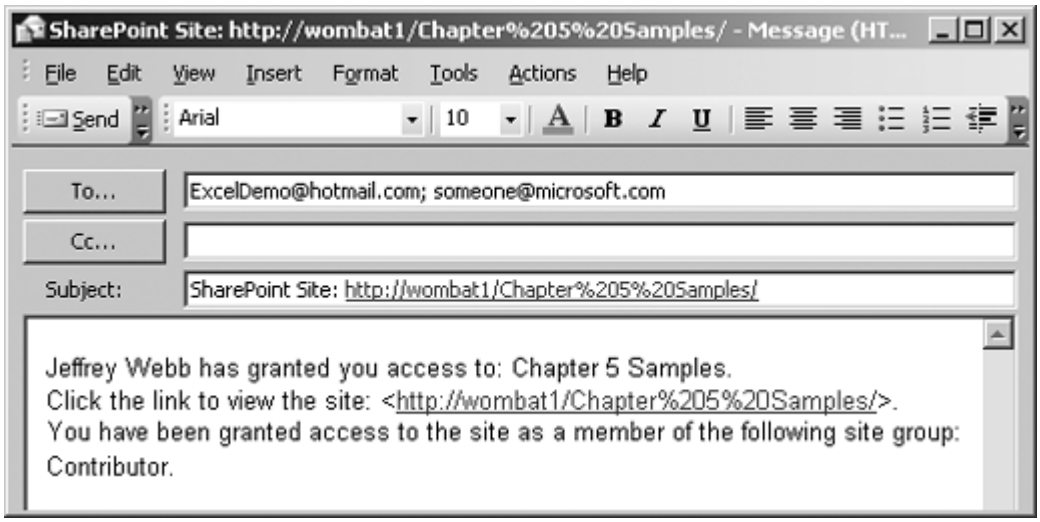


Excel can only send the email if you have Outlook 2003 installed as your default mail application. If you get an error at step 4, do this: from the Windows Start menu, choose All Programs → Microsoft Office → Microsoft Office Outlook 2003. If Outlook isn't your default mail application, you should see a dialog box asking if you want to make it the default. Click OK to make the change.

5.2.4. Sending and Receiving Alerts

Figure 5-7 shows the email sent when you add new members to a workspace. To send an email to members at any time, go to the Office SharePoint task pane and choose Members → Send e-mail to all members. Outlook displays a new message window with all the members in the To address.

animal 5-7. Editing the notification before sending



You can also create automated emails to notify you when a document changes. To receive this type of alert:

1. From the task pane, choose Document Information → Alert me about this document. Excel displays the page shown in Figure 5-8.

2. Choose the type of changes you want to be alerted about and how frequently you want to receive the alerts; then click OK.

3. SharePoint sends email confirming the creation of the alert, then sends email alerts as requested.

SharePoint sends email alerts directly from the server, so it doesn't require that you be running Outlook. The messages are sent through the SMTP server configured for SharePoint by the administrator. When you receive an alert, the From address is the SharePoint workspace address, and the message body include links to the workspace and document, as shown in Figure 5-9. To remove alerts:

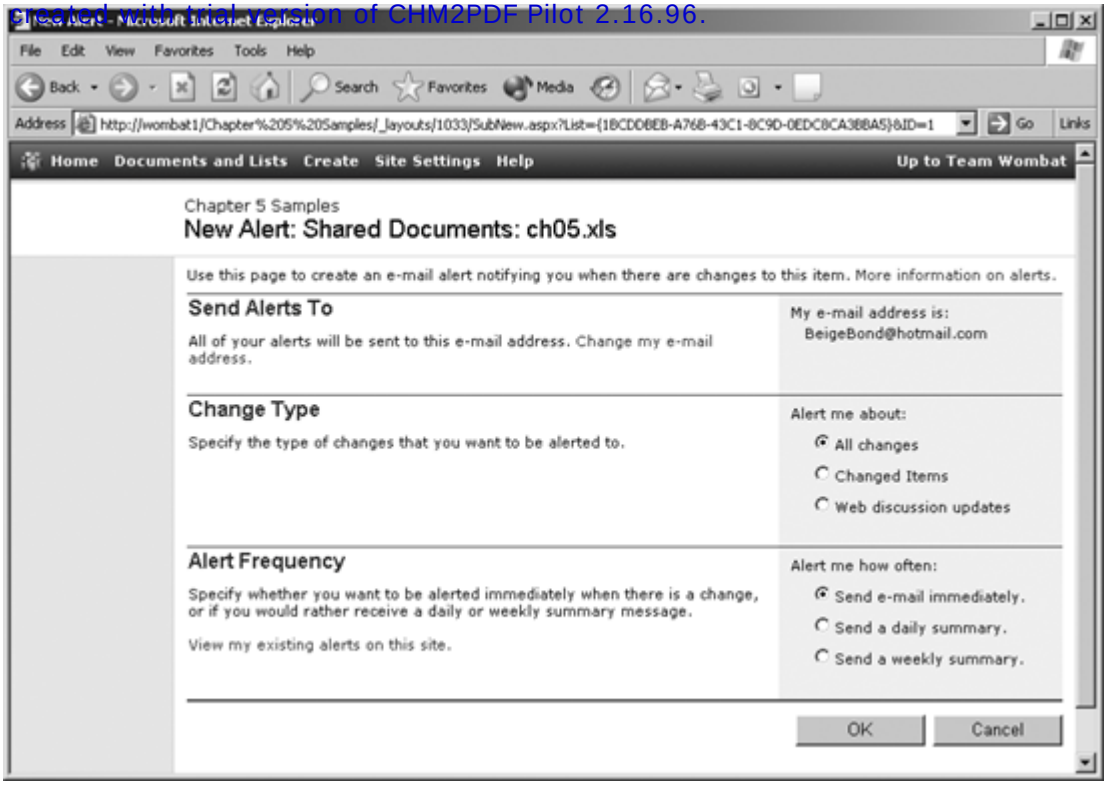
1. Click the My Alerts link illustrated in Figure 5-9. You'll see a list of your alerts in a browser window.

2. Select the alerts to remove and click Delete Selected Alerts.

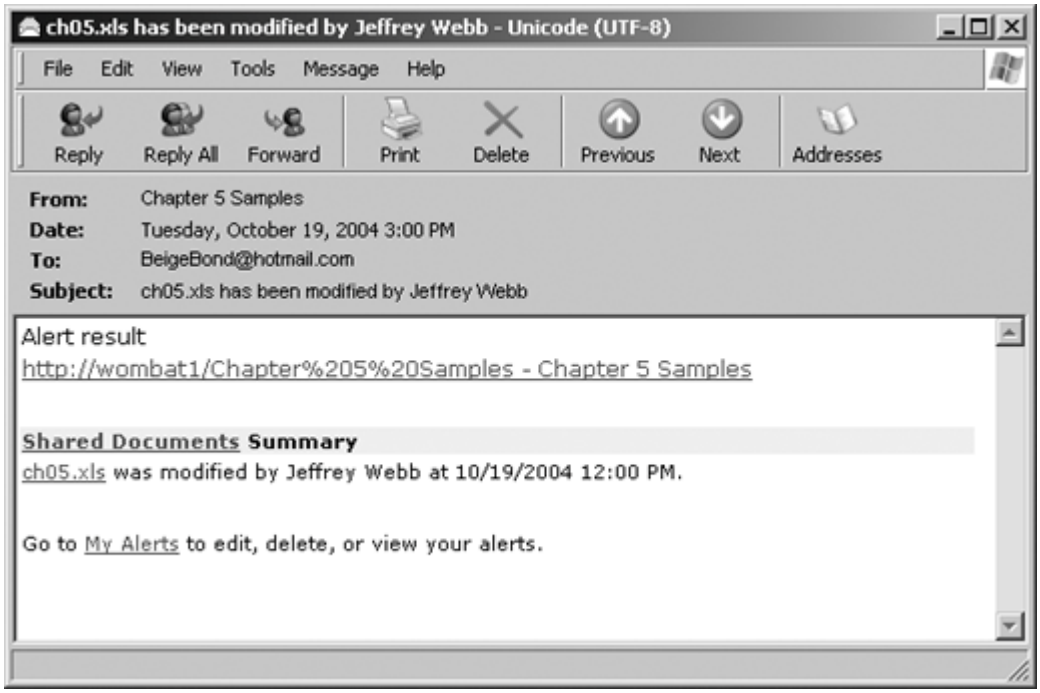
You can also see your alerts by choosing Site Settings → My alerts on this site.

Alerts are member-oriented, so you can add alerts for yourself but not for others. In order to add alerts for someone else, you must actually sign in as that member and then add the alert, pretending to be them.

animal 5-8. Requesting alerts when a document changes



animal 5-9. Receiving an alert



Members continue to receive alerts even if they are removed from a workspace. Therefore, it's important for the administrator to check alerts before removing a member. To check alerts set for a member:

1. Sign on as the administrator.
2. Choose Site Settings → Go to Site Administration → Manage user alerts.
3. From the drop-down list, select the member and click Update to view that member's alerts.
4. Select the alerts to remove and click Delete Selected Alerts.

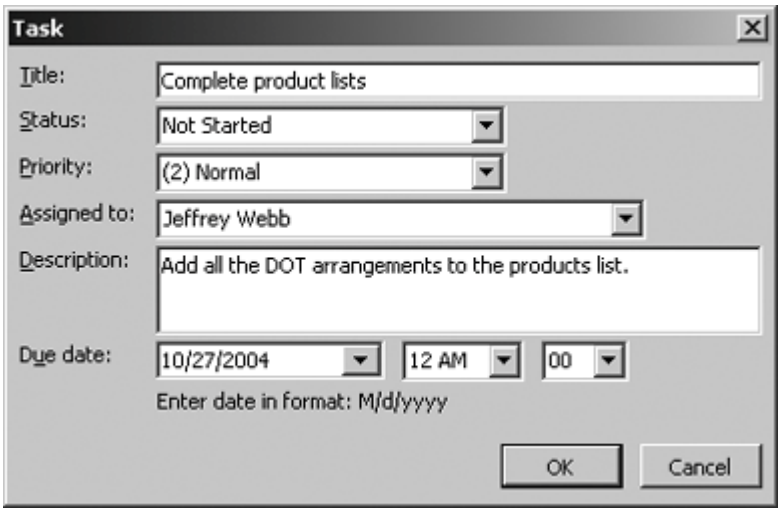
5.2.5. Assigning Tasks

You can track jobs assigned to members by assigning them as tasks within the workspace. To assign a task:

1. From the task pane, choose Tasks → Add new task. Excel displays Figure 5-10.

2. Complete the dialog box and click OK to assign the task.

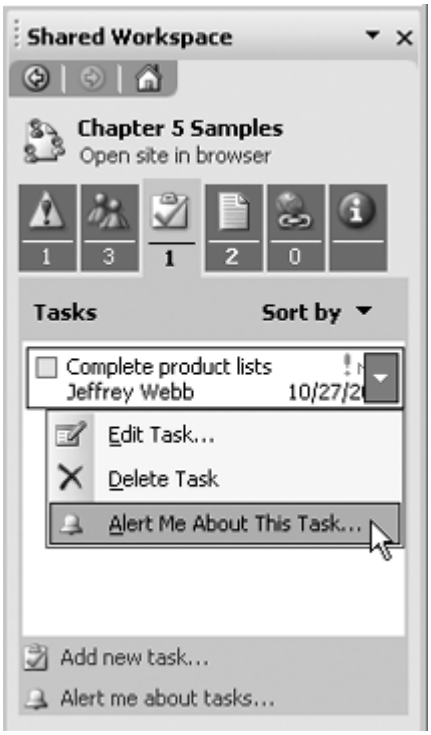
animal 5-10. Assigning tasks to members



Once assigned, the task appears in the task pane. You can edit, delete, or set alerts on the task from the task pane, as shown in Figure 5-11.

When you complete the task, select the checkbox next to the task title. Selecting the checkbox notifies SharePoint that the task is complete and changes the status of the task in the workspace.

animal 5-11. Changing a task



5.2.6. Controlling Updates

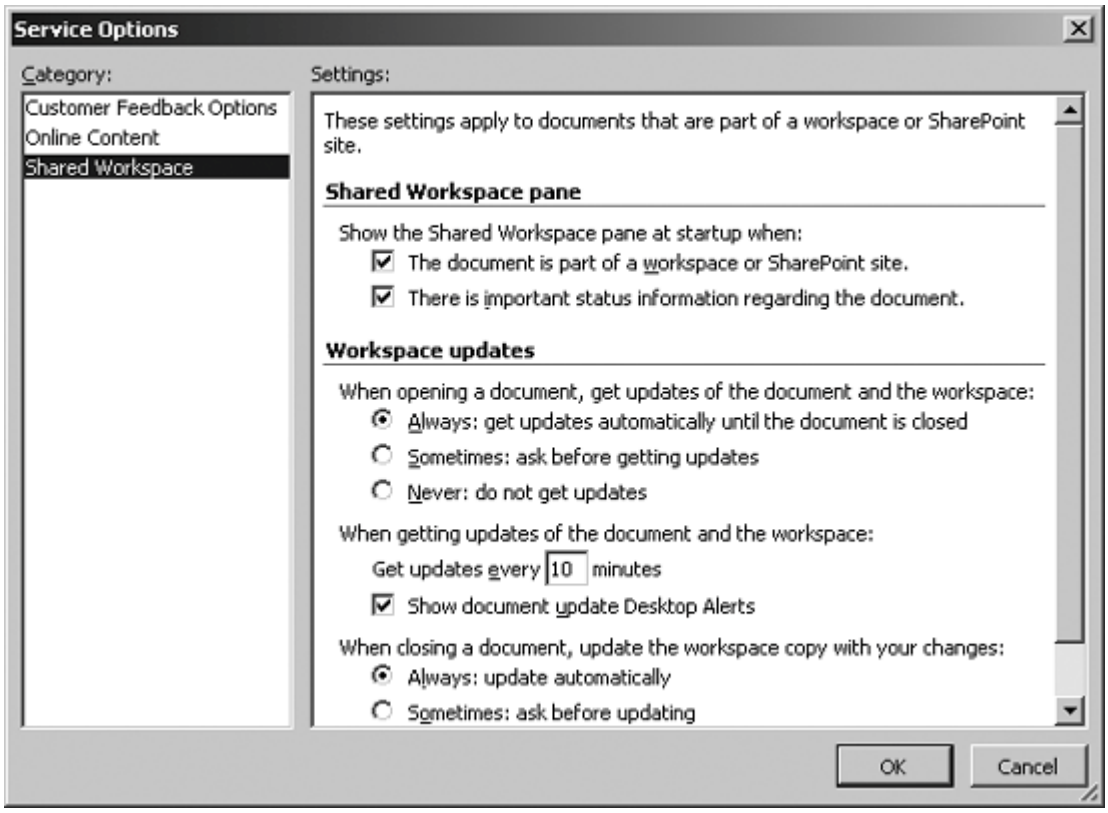
To control how the workbook displays the Shared Workspace task pane and how updates are handled, choose Options at the bottom of the task pane. Figure 5-12 shows the workspace options.

5.2.7. Opening a Shared Workbook

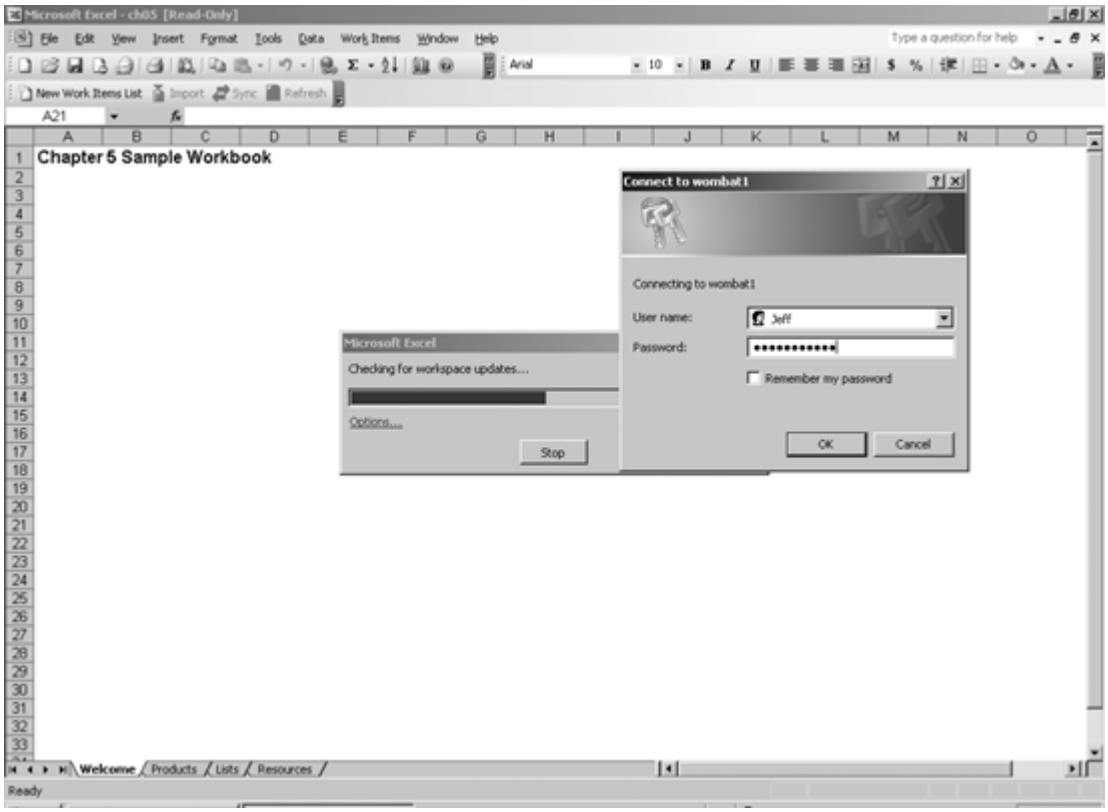
If the local file is linked to a shared workspace, opening that file automatically connects to the SharePoint site and updates the local file. You may have to sign on to the SharePoint site, and Excel displays the update status as shown in Figure 5-13.

If the workbook is not linked, or if the workbook is not stored locally, you can open the workbook from the SharePoint site. If you double-click on a workbook in the SharePoint site, Excel opens the workbook as Read-Only. To open the workbook for editing, select Edit from the pop-up menu on the site as shown in Figure 5-14.

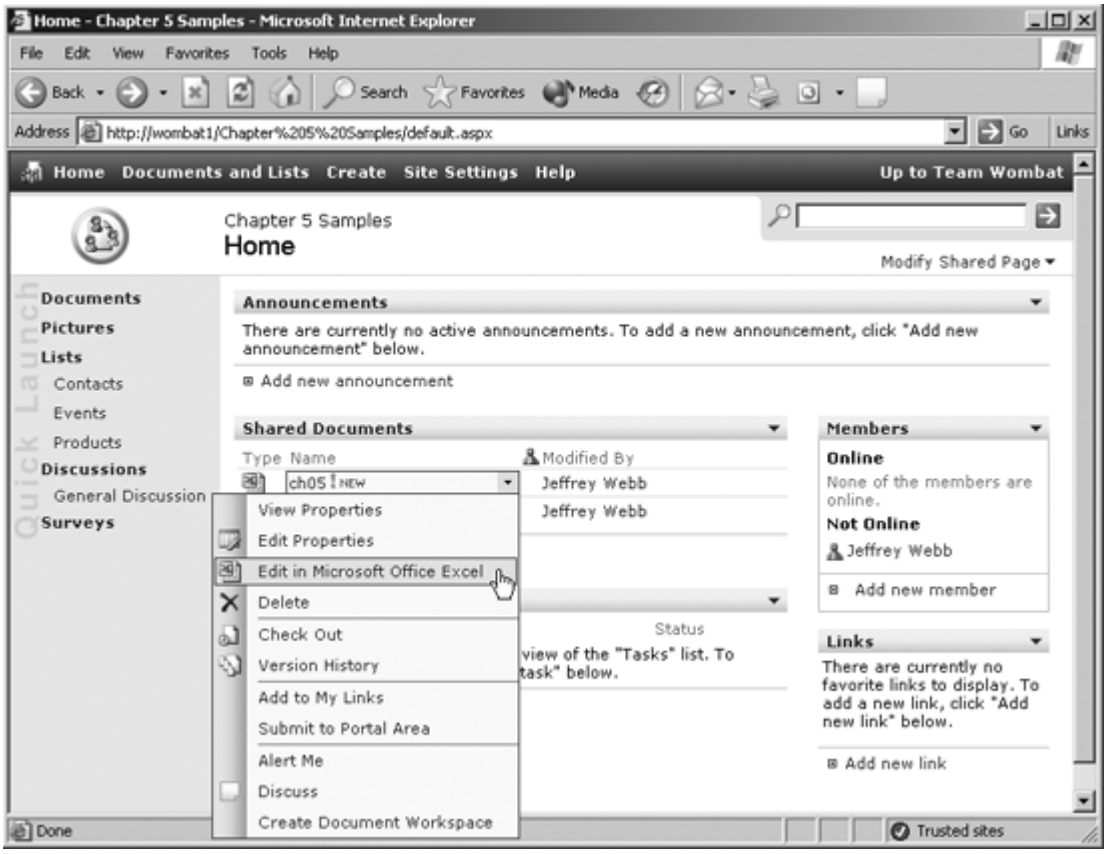
animal 5-12. Controlling how updates, alerts, and the task pane are handled



animal 5-13. Opening a workbook linked to a shared workspace



animal 5-14. Opening a shared workbook from the SharePoint site



5.2.8. Checking Files In and Out

More than one member can have a shared workbook open at the same time. To prevent conflicts among edits, you can check a document out to prevent others from making changes. You can't check documents in or out from the task pane, however. To check a document out:

1. Open the SharePoint site by clicking Open site in browser on the task pane.

2. From the workspace site, right-click the document and select Check Out from the context menu shown in Figure 5-14.

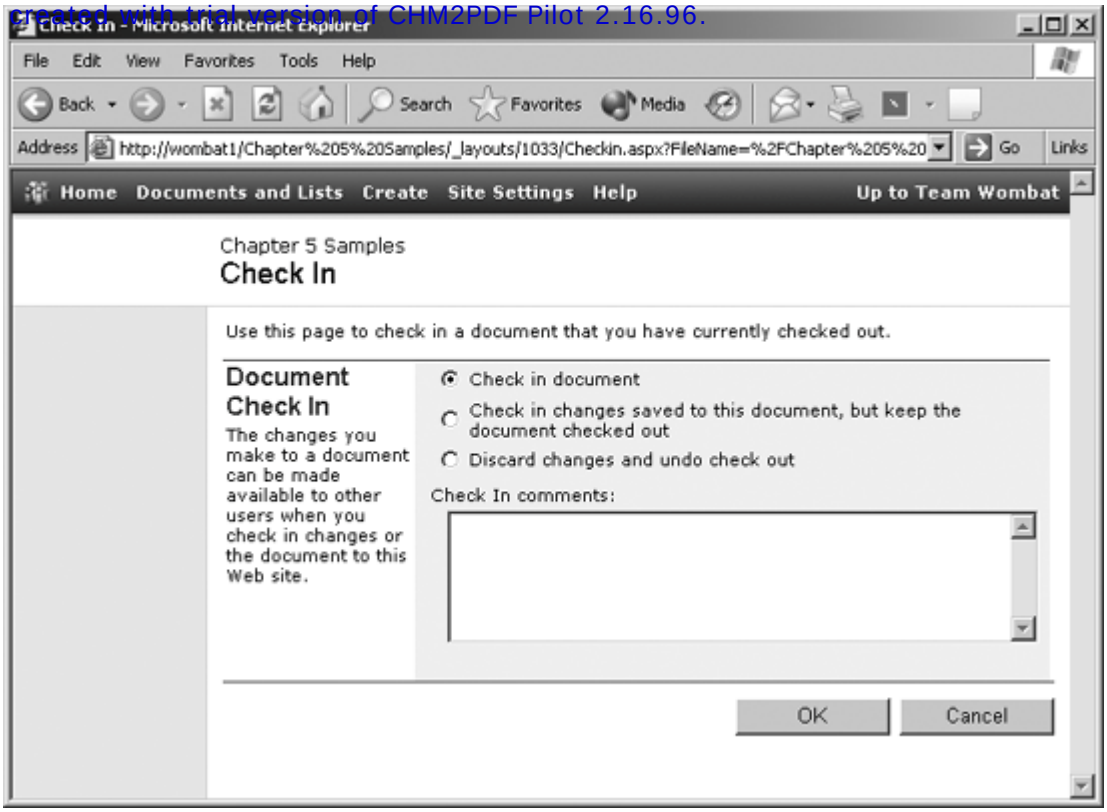
While the document is checked out, others can't open the document for editing from the workspace. Once you have completed your changes, check the document back in:

1. Save your changes and close the workbook.

2. From the SharePoint site, right-click the document and select Check In from the context menu. SharePoint displays Figure 5-15.

3. Add a comment describing the changes and click OK. Comments appear in the document history and are useful for tracking revisions.

animal 5-15. Checking a document in



5.2.9. Reconciling Changes and Viewing History

SharePoint doesn't require that you check files in or out before making changes, though it is a good idea to do so. The advantage of checking files in and out is that others can't edit the document while you have it checked out.

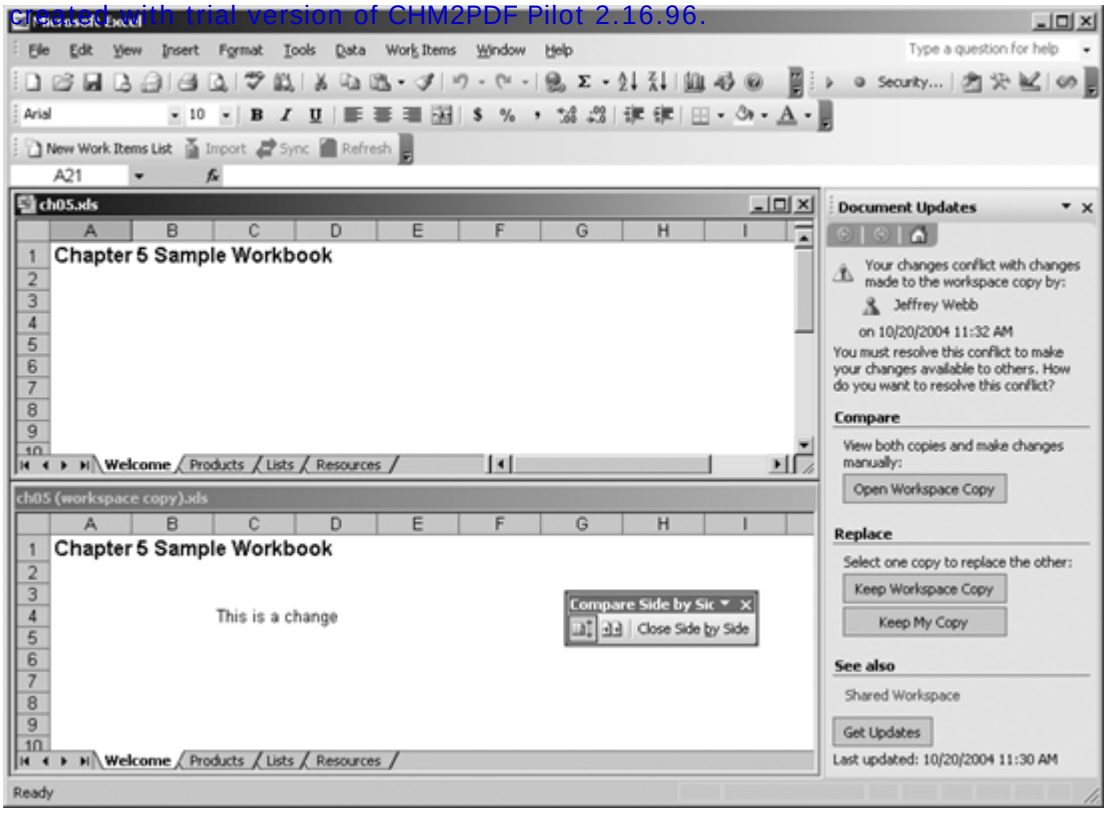
The same thing is true if you open the document directly from the SharePoint site. While you have a file open for editing from the site, others can open that file only in read-only mode. However, that *doesn't* happen when working with local copies of a document that are linked to the SharePoint site. In that case, members must resolve conflicting changes using the Document Updates task pane.

Whenever you open a document linked to a workspace, Excel checks whether your version is up-to-date. If others have made changes, Excel notifies you that your changes conflict with others. To reconcile your changes with those of others:

1. In the Document Updates task pane, click Open Workspace Copy.

2. Choose Window → Compare Side by Side. Excel displays the two versions for comparison, as shown in Figure 5-16.

animal 5-16. Reconciling changes using side-by-side comparison



SharePoint can track changes to documents in the workspace, but that feature is turned off by default. To turn on version-tracking:

1. From the workspace site's home page, select Shared Documents → Modify columns and settings → Change general settings.

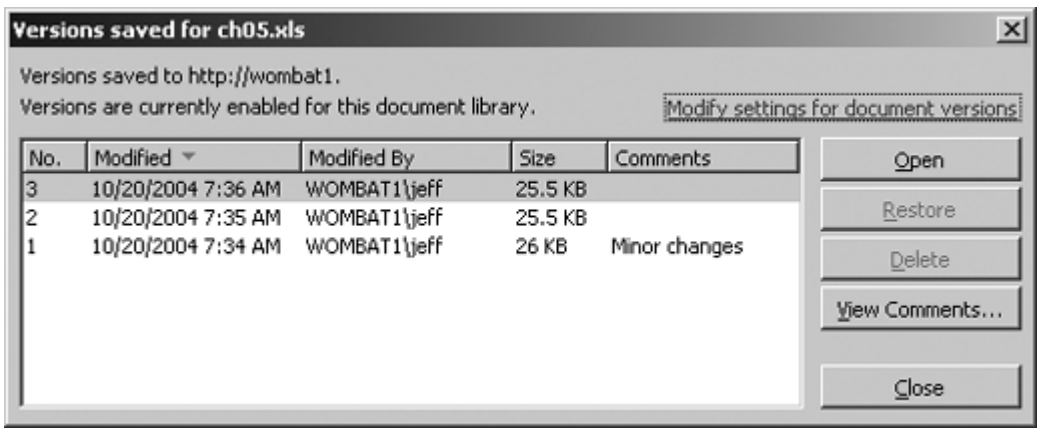
2. On the settings page, select Yes in the Document Versions section and click OK.

Once version tracking is on, a separate version of the document is saved each time a member updates the document. To see the version history, go to the Shared Workspace task pane and select Document Information → Version history. Excel displays Figure 5-17.

You can also view version history from the workspace home page by right-clicking the document and selecting Version History from the context menu.

You can view, restore, or delete previous versions of the document. Versions that were checked in or out usually have comments that were entered when the document was checked in. Changes made without checking in or out don't include comments.

animal 5-17. Viewing version history



5.3. Sharing Lists

In Microsoft Excel 2003, *lists* are ranges of cells that can easily be sorted, filtered, or shared. Lists are a little different from the AutoFilter feature available in earlier versions of Excel, in that lists are treated as a single entity rather than just a range of cells. This cohesion is illustrated by a blue border that Excel draws around the cells in a list.

Lists have these advantages over AutoFilter ranges:

- Lists automatically add column headers to the range.
- Lists display a handy List Toolbar when selected.
- It is easy to total the items in a list by clicking the Toggle Total button.
- XML data can be imported directly into a list.
- Excel can automatically check the data type of list entries as they are made.
- Lists can be shared and synchronized with team members through SharePoint.

That last item is the key advantage of listslists are really just ways to share information that fits into columns and rows.

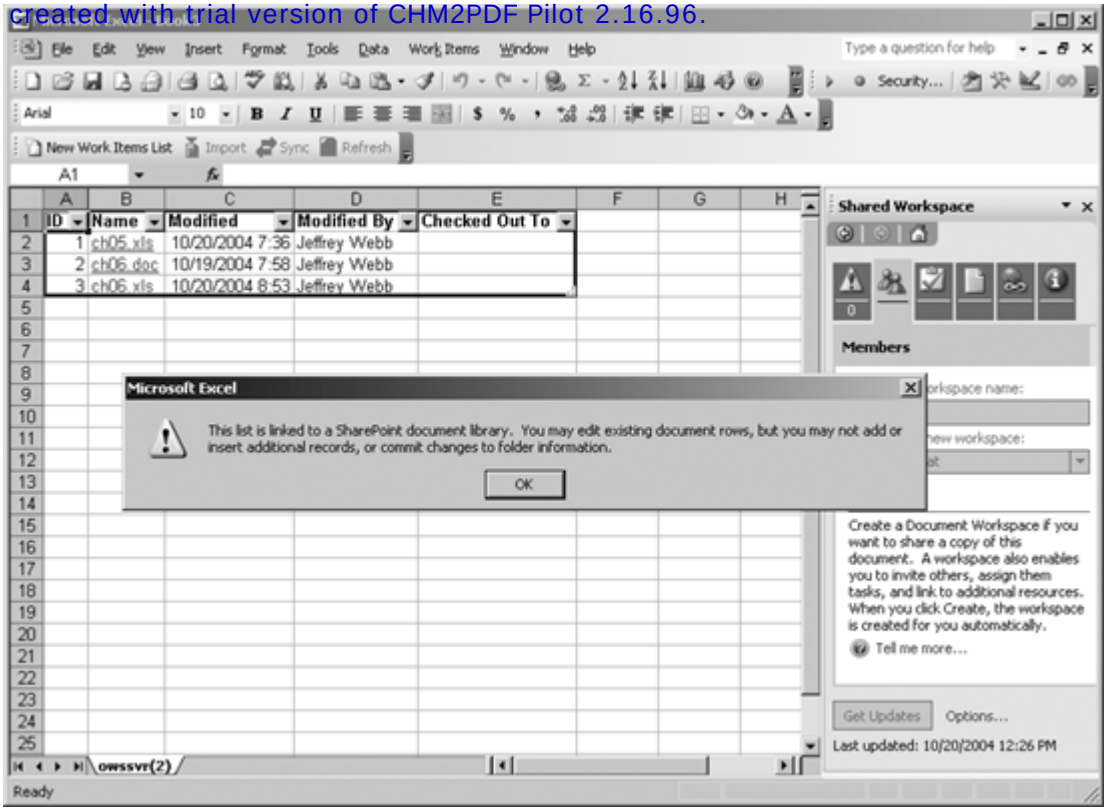
5.3.1. Viewing SharePoint Lists in Excel

There are many different ways to create lists in SharePoint. In fact, since SharePoint uses lists everywhere, most tasks involve either creating lists or adding new items to them. For example, the document workspace created in the preceding sections contains several lists: Announcements, Shared Documents, Members, Tasks, Links, Contacts, and General Discussion. To view any of those lists in Excel 2003:

1. Display the list in the browser. For example, select Shared Documents from the workspace home page.
2. Click Export to Spreadsheet. SharePoint creates an Excel query and displays the File Download dialog box.
3. Click Open to display the query results in Excel. Excel displays a security warning.
4. Click Open to run the query. Excel displays the Import Data dialog.
5. Excel creates a new workbook, inserts the list, and displays a list of limitations, as shown in Figure 5-18.

Other versions of Excel also allow you to import lists, but the results vary. See Appendix B for a table of version differences.

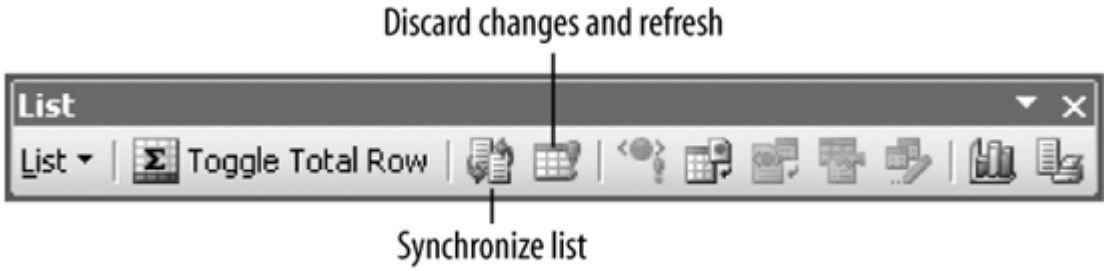
animal 5-18. Viewing the Shared Documents list in Excel



Excel imports the documents in the list as hyperlinks. To open one of the files, click on the file name in the worksheet. The list in Figure 5-18 is linked to the Shared Documents list in SharePoint. If a member adds a new document to the workspace, you can see that change in Excel by clicking one of the buttons on the Lists toolbar shown in Figure 5-19.

Refreshing a list discards local changes and updates the (local) worksheet list with data from the SharePoint server. *Synchronizing* updates both the worksheet list and the SharePoint list. In the case of the Shared Documents list, the two buttons in Figure 5-19 do the same thingall columns in the Shared Documents list are read-only when imported to Excel. SharePoint defines constraints for lists and those constraints are carried over when the list is imported.

animal 5-19. Refreshing a shared list in Excel



5.3.2. Importing Lists into Existing Workbooks

The preceding section showed you how to import a shared list into a new Excel workbook. If you import a list into an existing workbook, Excel deletes all of the Visual Basic code and ActiveX controls contained in the workbook, so if your workbook contains macros, they will stop working.

You can avoid this problem by importing the list using an Excel macro, rather than doing it from the SharePoint site. The following VBA procedure imports the Products list from a SharePoint site:

```
Sub ImportList(  )
    Const site = "http://wombat1/Chapter 5 Samples"
    Const list = "Products"
```

```
    Dim ws As Worksheet
    Dim src(1) As Variant
    Set ws = ThisWorkbook.Worksheets(1)
    src(0) = site & "/_vti_bin"
    src(1) = list
    ws.ListObjects.Add xlSrcExternal, src, True, xlYes, ws.Range("A1")
End Sub
```

5.3.3. Editing Lists and Reconciling Changes

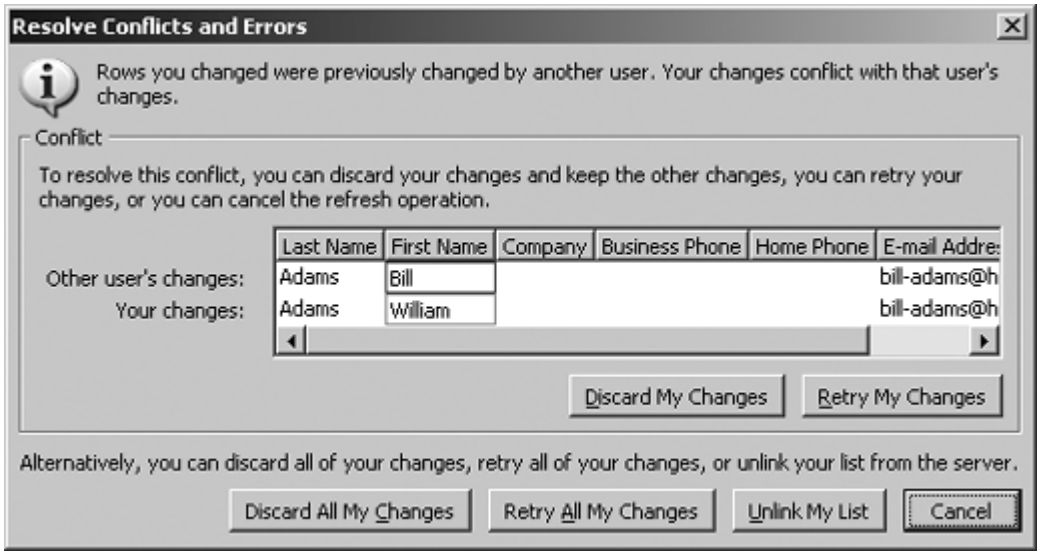
The Shared Documents list is read-only, but other types of shared lists can be imported into Excel, edited, then synchronized to send the changes from Excel back to SharePoint. To see how this works, repeat the preceding procedure with the Contacts list. Once imported into Excel, you can change contact information; then click Synchronize to update SharePoint.

As with workspaces, more than one member can edit a shared list at the same time. If your changes conflict with another member's changes, Excel displays the dialog box in Figure 5-20 to resolve the conflicts. To replace the other member's changes with yours, choose Retry My Changes.

5.3.4. Creating and Sharing Lists in Excel

The built-in SharePoint lists aren't really the best examples of the types of lists you work with in Excel. I used them to introduce shared lists because they are a quick

animal 5-20. Reconciling conflicting edits in a shared list



way to show you how some key features work. It's much more useful to create a list in Excel first and then share that list through SharePoint. To create a list from Excel:

1. Select a range of cells and then choose Data → List → Create List. Excel displays the Create List dialog.

2. If the range includes column headings, select My list has headers. If you want Excel to create column headings, deselect that option. Click OK to create the list.

3. Excel converts the selected range into a list and displays the Lists toolbar.

Excel calls sharing a list *publishing*. To publish a list in Excel, select the list and then choose Data → List → Publish List. Excel displays a series of steps that publish the list on the SharePoint server and display the address for the shared list (Figure 5-21).

When a list is published, SharePoint creates a new item in the Lists folder of the SharePoint site that members can use to view or modify the list's data. The list doesn't appear on the home page of the site by default, so there's no obvious way for other members to get to the list. To add the list to the Quick Launch bar on the

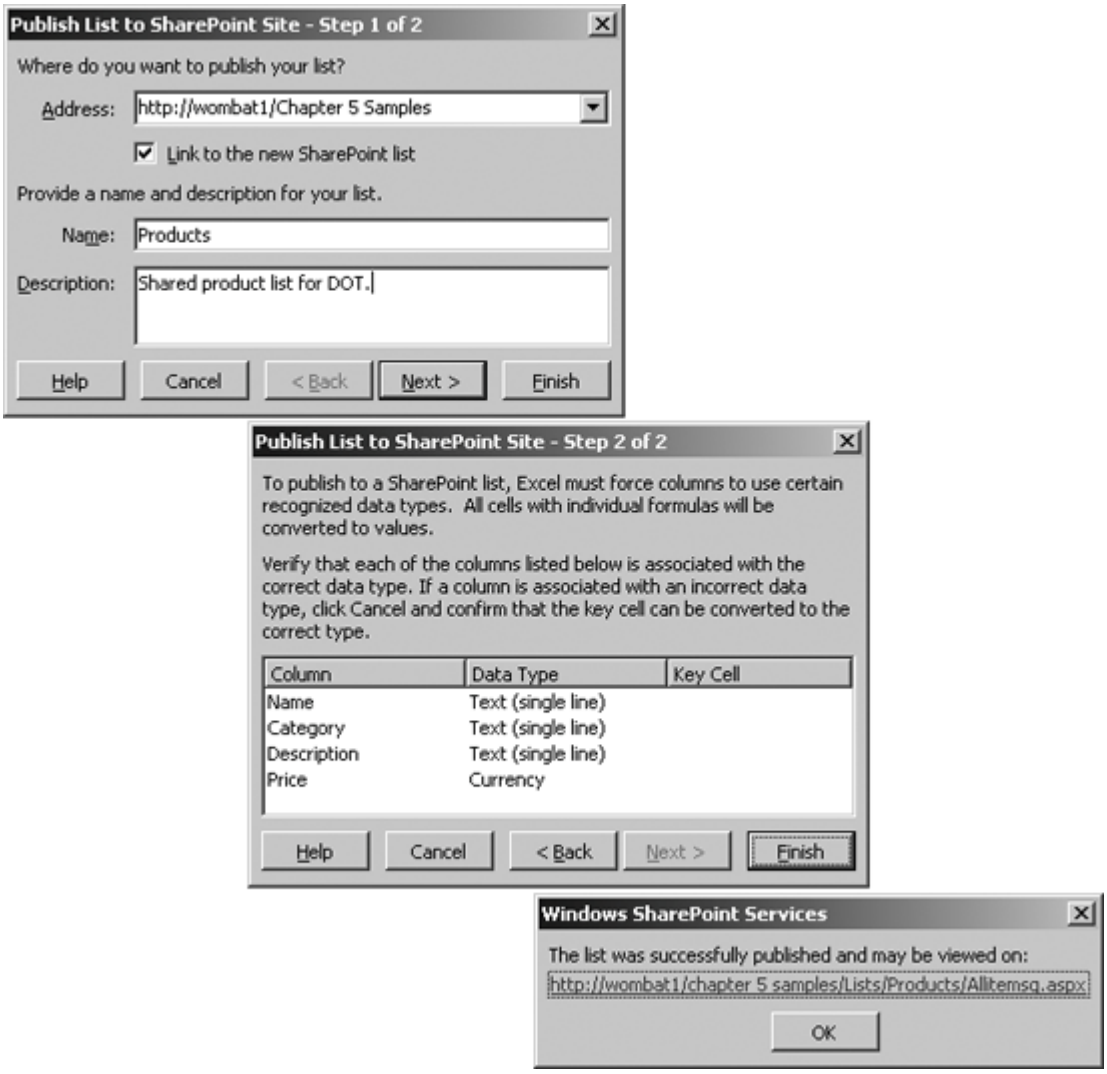
home page:

1. Display the list in the browser. To do that, either click the link in the last dialog box (Figure 5-21) or select the list in Excel and choose Data → List → View List on Server.

2. In the browser, choose Modify settings and columns → Change general settings.

3. Select Yes in the Navigation section and choose OK to make the change.

animal 5-21. Publishing a list from Excel



There are other ways to view lists not displayed in the Quick Launch bar. For example, choose Documents and Lists → Lists; then choose the list to display. The Quick Launch bar is for things members use every day.

5.4. Publishing as a Web Page

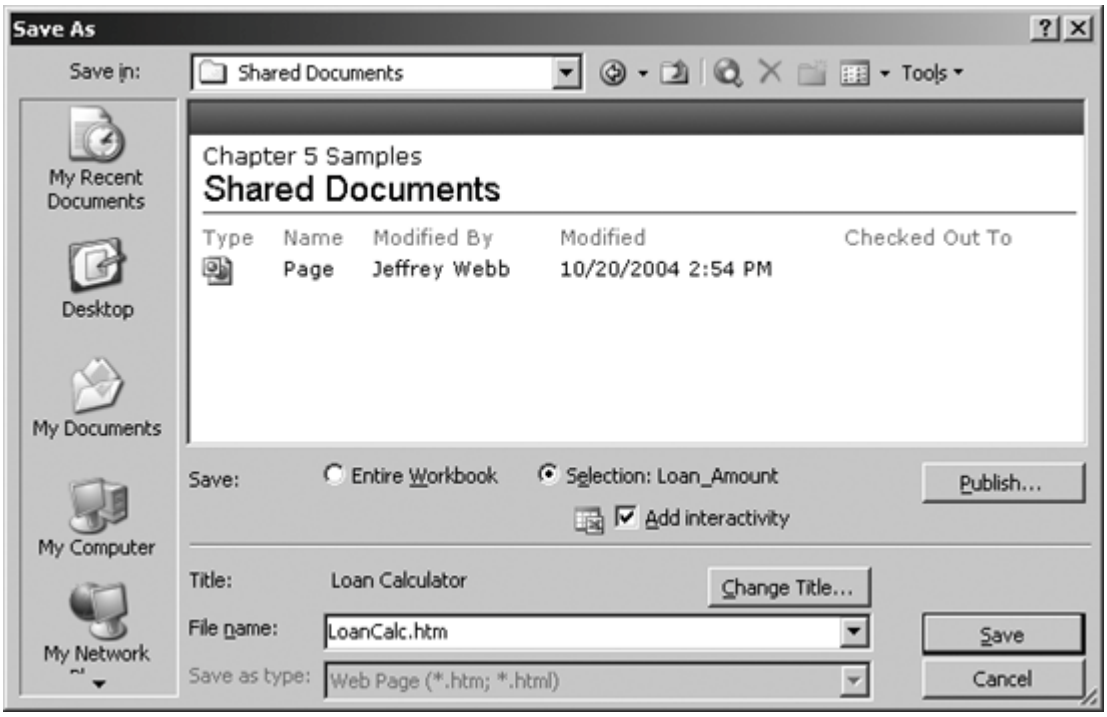
Publishing a workbook or a worksheet as a web page was introduced in Excel 2000. It allows you to save all or part of a workbook in HTML format so that others can view it from a web site. You can even save the page in an interactive format that allows other Excel users to use the page to perform calculations.

You can publish web pages from Excel to a SharePoint site using the same steps as you do for any other web site:

1. In Excel, open the workbook to publish as a web page.
2. Choose File ⟶ Save as Web Page. Excel displays Figure 5-22.
3. Enter the address of a document library in a SharePoint site (for example, http://wombat1/Chapter 5 Samples/Shared Documents/).
4. Select Add interactivity to allow other Excel users to change data on the page and perform calculations.
5. Enter a name for the page, and choose Save. Excel saves the workbook or worksheet as a web page in the SharePoint document library.

If you want to view additional web publishing options, in step 5, choose Publish rather than Save.

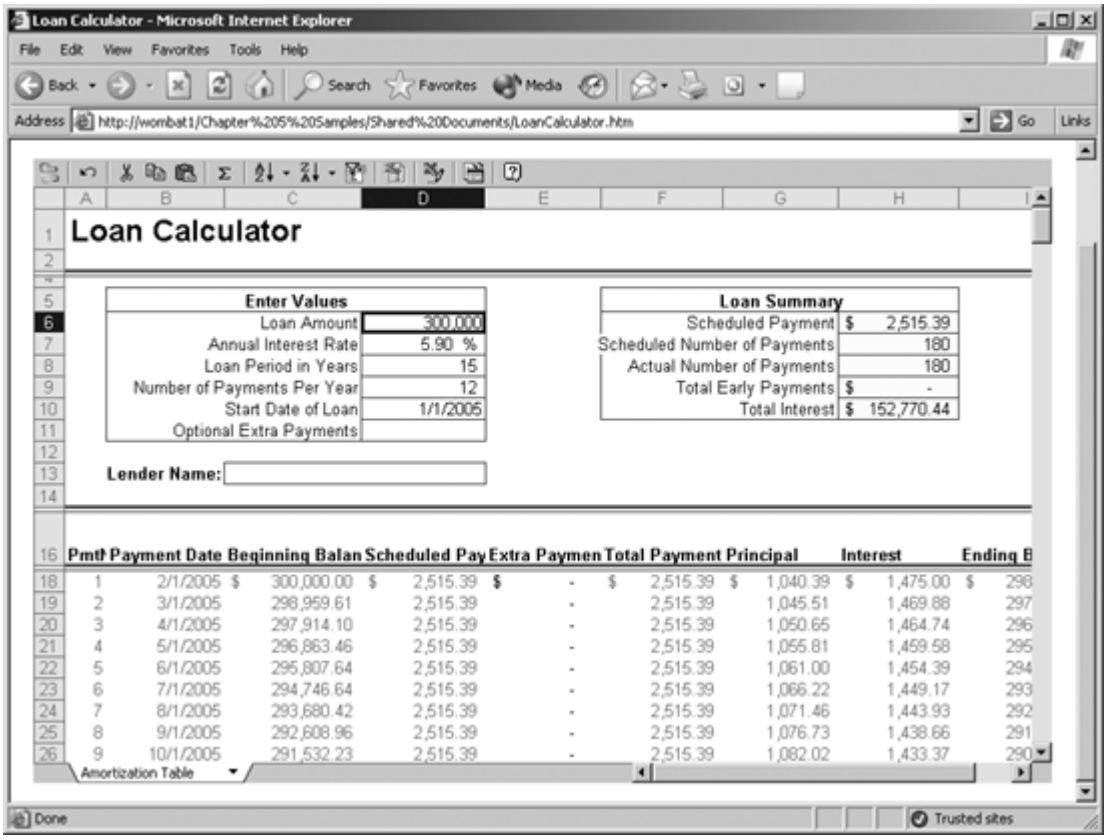
animal 5-22. Publishing a worksheet as a SharePoint web page



It's also possible to publish the page outside a document library. For example, if you specify http://wombat1/Chapter 5 Sample/LoanCalc.htm in step 3, the page will be published at the root level of the workspace. However, it's harder to edit or delete files saved outside of document libraries.

To view the newly published page in the browser, navigate to the document library and click on the new item created for the page. Figure 5-23 shows the loan calculator worksheet published from Excel.

animal 5-23. Displaying the published worksheet



The web page in Figure 5-23 uses Microsoft Office 2003 Web Components. Those components are available for free download from Microsoft. However, members who do not have Office 2003 installed will not be able to edit items within the page.

PREV NEXT

5.5. Using the Spreadsheet Web Part

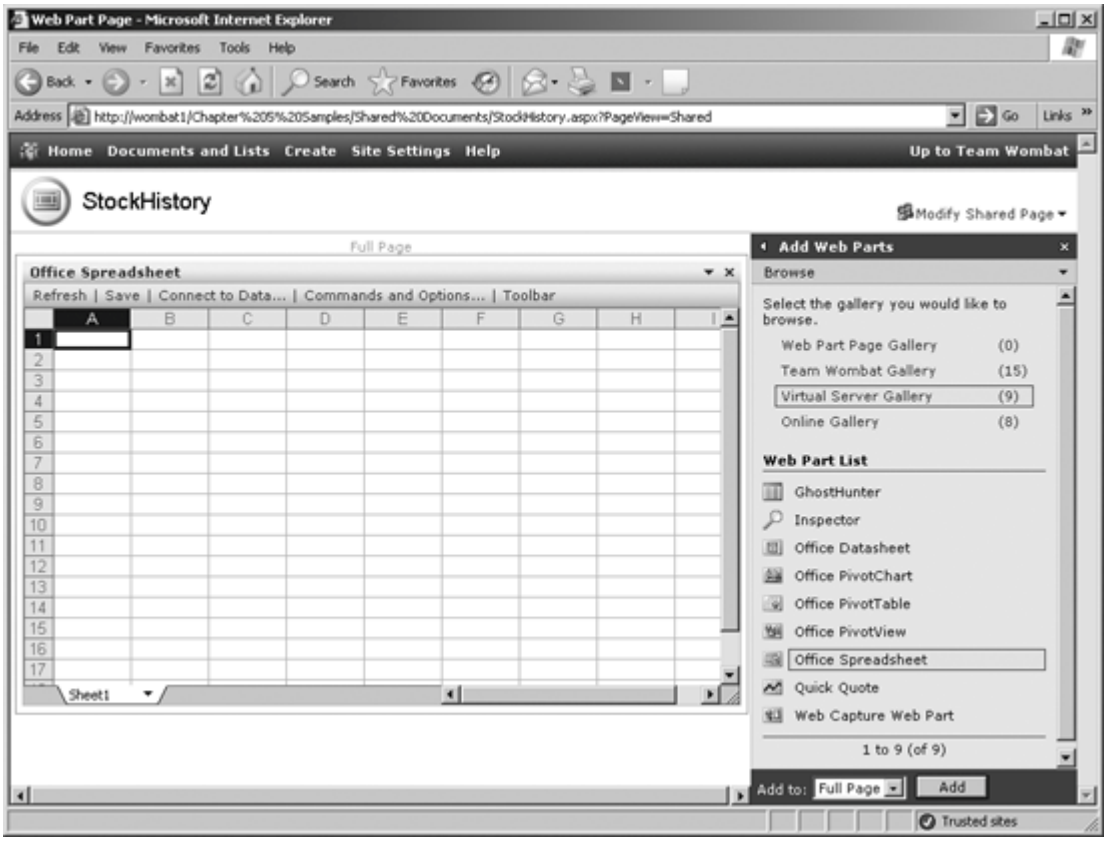
Publishing a spreadsheet as an interactive web page is cool, but unfortunately the data on the page can't be saved. You can view the page, make some changes, and see the result, but the next person to view the page won't see your work.

To create interactive spreadsheets that can save data, create a new web part page and add a spreadsheet web part. The spreadsheet web part is included with the Office 2003 web parts download available from Microsoft. Your SharePoint administrator must install that download on your SharePoint server before you can complete the tasks in this section.

To create a page with a spreadsheet web part:

1. From the SharePoint site home page, choose Create → Web Part Page.
2. Enter a name for the new page, choose a layout, select the document library to store the page in, and click Create. SharePoint displays the new page in edit mode.
3. In the web part task pane, select Virtual Server Gallery. You'll see a list of the web parts installed on your server.
4. Drag the Office Spreadsheet web part from the task pane to a web part zone on the page. SharePoint adds the web part to the page, as shown in Figure 5-24.
5. Close the web part task pane and click OK to save the changes.

animal 5-24. Creating a web part page containing a spreadsheet

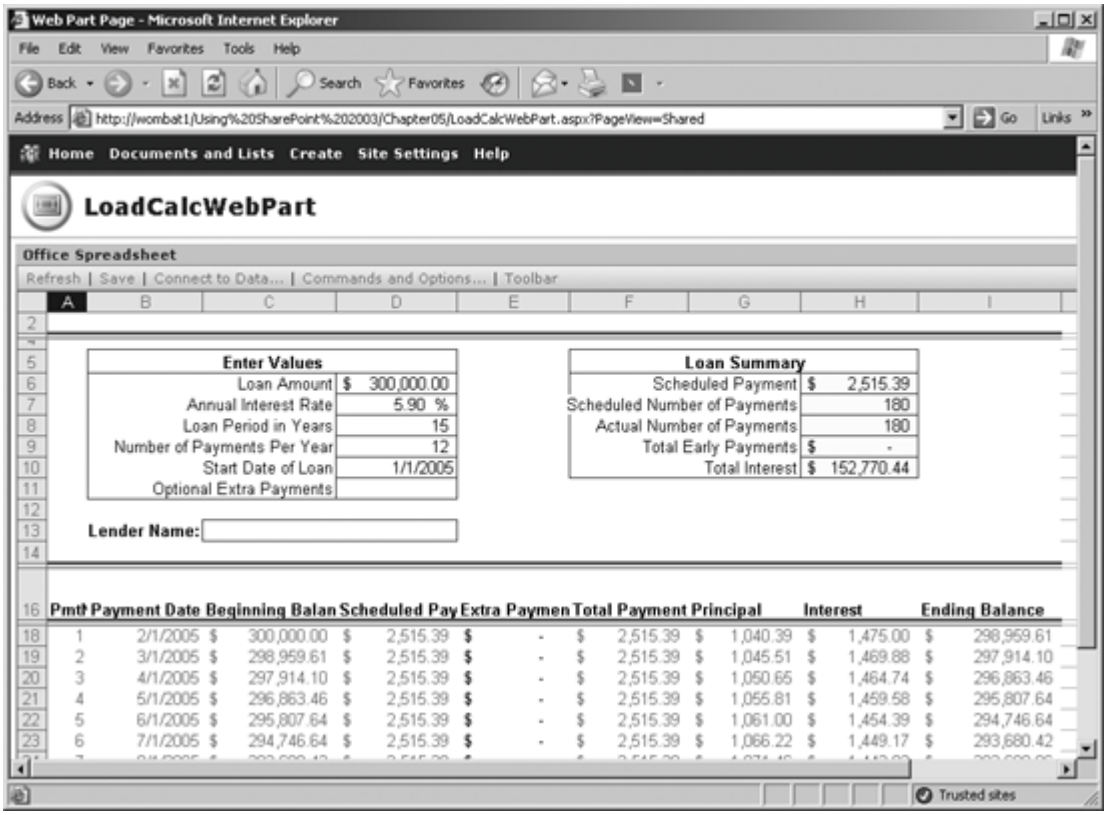


Once you've created a web part page containing a spreadsheet web part, you can edit the cells on the spreadsheet directly in the browser. Figure 5-25 shows a Loan Calculator created on the web part page. The main difference between this version of the Loan Calculator and the one shown in Figure 5-23 is that the data can be saved with the page.

There are also other differences between using the Office Spreadsheet web part and publishing a spreadsheet as a web page:

- The web part page includes a navigation bar.
- Only a limited amount of data can be saved in the Office Spreadsheet web part.
- You can cut and paste from Excel to an Office Spreadsheet web part, but some formulas may not work in the web part.

animal 5-25. Displaying a spreadsheet as a web part



The loan calculator example in Figure 5-23 is a somewhat tired example, but it does illustrate some of these differences well. In order to get it to work as a web part, I had to:

1. Limit it to 15 years of data (~180 rows). Showing a full 30-year amortization table resulted in a storage limit error when saving.
2. Define named ranges for Values_Entered and Number_of_Payments. Those names referred directly to formulas in Excel, but the web part can only store values if they are in a cell.

You may encounter other differences when using the Office Web Components as web parts rather than publishing as a web page.

Microsoft provides an Excel add-in for creating your own web parts from Excel. See Chapter 8 for more information.

5.6. Programming SharePoint in VBA

Many Excel users write macros in VBA. If you're not one of those power users, please skip the rest of this chapter. If you know VBA, you'll find the following tasks helpful when working with SharePoint.

5.6.1. Creating Workspaces

Use the `Workbook` object's `SharedWorkspace` property to work with shared workspaces in Excel. The `SharedWorkspace` property returns a `SharedWorkspace` object that you use to share the workbook, update the workbook, and navigate among other elements in the shared workspace. For example, this code uses the `SharedWorkspace` object's `CreateNew` method to create a new shared workspace and add the current workbook to it:

```
ThisWorkbook.Save
ThisWorkbook.SharedWorkspace.CreateNew "http://wombat1", _
    "Team Wombat"
```

You must save the workbook before adding it to a shared workspace, otherwise the `CreateNew` method will fail. The preceding code adds the current workbook to the SharePoint site on the Wombat1 server. If you click on Open site in browser from the Shared Workspace task pane, Excel displays the new workspace site created at http://wombat1/Team%20Wombat.

If you call `CreateNew` again, Excel will create another new SharePoint site and increment the site name to http://wombat1/Team%20Wombat(1). To add a workbook to an existing SharePoint site instead of creating a new site, follow these steps:

1. Open an existing document from the SharePoint site.
2. Get a reference to that document's **`SharedWorkspace`** object.
3. Add your workbook to the **`SharedWorkspace`** object's **`Files`** collection.
4. Close the document you opened in step 1.
5. Close the workbook you just added and reopen it from the SharePoint site.

The following code demonstrates how to add a workbook to an existing SharePoint site. The file *Blank.xls* is simply an empty workbook used to get a reference to the SharePoint site:

```
Set wb = Application.Workbooks.Open("http://wombat1/Team Wombat/Shared Documents/ &_
    Blank.xls")
If wb.SharedWorkspace.Connected Then
    wb.SharedWorkspace.Files.Add ThisWorkbook.Path & "\" & ThisWorkbook.Name
End If
```

Steps 4 and 5 are not shown in the code above because they require some explaining. Even though you have added the workbook file to the SharePoint site, the currently open workbook is the local version, not the shared version. You can't close the current workbook from code and then open it from the SharePoint site for two reasons: the code stops running the moment you close the current workbook, and you can't have two workbooks with the same name open at the same time.

There are a number of ways to work around this: you can save the shared workbook with a different file name, you can share the workbook from code running outside of the current workbook, or you can simply display the SharePoint site and allow the user to reopen the shared workbook from there. The following code demonstrates that last approach:

```
If MsgBox("Click Yes to close this workbook " & _
  "and then open the workbook from the SharePoint site.", vbYesNo, _
```

```
    "Workbook added to shared workspace.") = vbYes Then
        ' Open the SharePoint site in IE.
        ThisWorkbook.FollowHyperlink wb.SharedWorkspace.url, , True
        ' Close the temporary workbook.
        wb.Close
        ' Close this workbook.
        ThisWorkbook.Close
    End If
```

Now if the user clicks Yes, Excel displays the SharePoint web site and closes the current and temporary workbooks.

You can tell if a workbook belongs to a shared workspace by checking the `Connected` property. You should make sure the `Connected` property is `true` before using `SharedWorkspace` methods; otherwise an error may occur.

5.6.2. Opening Workbooks from a Shared Workspace

To open a workbook from a shared workspace in code, simply use the `Workbooks` collection's `Open` method with the address of the workbook from the SharePoint site. For example, the following code opens a workbook from the http://wombat1/Team Wombat/ site:

```
Application.Workbooks.Open "http//wombat1/Team Wombat/Shared Documents/temp.xls"
```

To check a file out from code, use the `Workbook` object's `CanCheckOut` property and the `CheckOut` method. For example, the following code attempts to check out a file, and if it is successful it opens the file in Excel:

```
fil = " http//wombat1/Team Wombat/Shared Documents/temp.xls"
If Application.Workbooks.CanCheckOut(fil) Then
    Application.Workbooks.CheckOut fil
    Set wb = Application.Workbooks.Open(fil)
    MsgBox wb.Name & " is check out to you."
End If
```

The `CheckOut` method doesn't open the workbook, so you need to add the `Open` method as shown above. Checking a file in automatically closes the file as shown here:

```
Set wb = Application.Workbooks("temp.xls")
If wb.CanCheckIn Then
    ' CheckIn closes the file.
    wb.CheckIn True, "Minor change"
    MsgBox "File was checked in."
Else
    MsgBox wb.Name & " could not be checked in."
End If
```

In some cases, a file may not be able to be checked in. For instance, you can't check in the current workbook from within its own code:

```
If ThisWorbook.CanCheckIn Then ' Always False!
```

In such cases, you can display the SharePoint site to provide a way to check the workbook back in.

5.6.3. Removing Sharing

There are two levels of removing sharing from a workbook stored in a shared workspace. You can:

- Delete the file from the SharePoint server. This method breaks the connection that other users share.

- Disconnect the file from the shared workspace. This method only breaks the connection between the local copy of the workbook and the shared workbook.

Use the `RemoveDocument` method to delete the current document from the shared workspace as shown by the following code:

```
If ThisWorkbook.SharedWorkspace.Connected Then _
    ThisWorkbook.SharedWorkspace.RemoveDocument
```

The preceding code leaves intact local copies that users have downloaded from the shared workspace, but all become disconnected since the shared workbook no longer exists. Alternately, you can leave the workbook in the shared workspace and disconnect only your own local copy with this code:

```
If ThisWorkbook.SharedWorkspace.Connected Then _
    ThisWorkbook.SharedWorkspace.Disconnect
```

Now, your local copy can no longer be updated from or send updates to the shared workbook. If you want an updatable copy, you must reopen the workbook from the shared workspace.

You can also use the `Files` collection to remove workbooks from a shared workspace. This technique works well if you want to remove a file other than the current workbook. For example, the following code removes *Security.xls* from the current workbook's shared workspace:

```
Dim file As Office.SharedWorkspaceFile
If ThisWorkbook.SharedWorkspace.Connected Then
    For Each file In ThisWorkbook.SharedWorkspace.Files
        If InStr(1, file.urlThisWorkbook, "security.xls") Then _
            file.Delete
    Next
End If
```

In the preceding case, you need to locate the file to remove using the `Instr` function because the `Files` collection doesn't provide an indexer to locate the file by name.

5.6.4. Responding to Updates

The `Workbook` object provides events you can use to respond to user actions. In order to use these events, write your code in the `ThisWorkbook` class of the workbook (in the Visual Basic editor, double-click on `ThisWorkbook` in the Project window). Visual Basic displays the `Workbook` events in the event list at the top of the Code window.

Use the `Sync` event to respond to document updates from SharePoint:

```
Private Sub Workbook_Sync(ByVal SyncEventType As Office.MsoSyncEventType)

End Sub
```

By default, document updates occur automatically every 10 minutes and any time a linked file is opened or closed.

5.6.5. Creating a List

Use the `Add` method of the `ListObjects` collection to create a list in code. The `ListObjects` collection is exposed as a property of the `Worksheet` object. The following code creates a new list for all the contiguous data, starting with the active cell:

```
ActiveWorksheet.ListObjects.Add
```

Use the `Add` method's arguments to create a list out of a specific range of cells. For example, the following code creates a list out the range A1:C3:

```
Dim ws As Worksheet
Dim rng As Range
Set ws = ThisWorkbook.Sheets("Sheet1")
Set rng = ws.Range("A1:C3")
ws.ListObjects.Add xlSrcRange, rng
```

When Excel creates the list, it automatically adds column headings to the list, either by converting the first row into column headings or by adding a new row and shifting the subsequent data rows down. It's hard to determine exactly what will happen because Excel evaluates how the first row was intended. You can avoid this unpredictability by supplying the `HasHeaders` argument:

```
Set rng = ws.Range("A2:C4")
ws.ListObjects.Add xlSrcRange, rng, , xlNo
```

The preceding code adds headers to the second row and shifts the range down a row.

Lists always include column headers. To avoid shifting the range down one row each time you create a list, include a blank row at the top of the source range and specify `xlYes` for `HasHeaders`:

```
Set rng = ws.Range("A1:C4")
ws.ListObjects.Add xlSrcRange, rng, , xlYes
```

Since column headers and new rows added to a list cause the subsequent rows to shift down, it is a good idea to avoid placing data or other items in the rows below a list.

When creating lists in code, it is also a good idea to name the list so that subsequent references to the list can use the list's name rather than its index on the worksheet. To name a list, set the Name property of the `ListObject`:

```
Dim lst As ListObject
Set rng = ws.Range("A1:C4")
Set lst = ws.ListObjects.Add (xlSrcRange, rng, , xlYes)
lst.Name = "Test List"
```

You can get a reference to a named list using the `Worksheet` object's `ListObjects` property:

```
Set ws = ThisWorkbook.Worksheets("Sheet1")
Set lst = ws.ListObjects("Test List")
```

5.6.6. Sharing a List

Once a list exists on a worksheet, you can share that list using the `Publish` method. The first argument of the `Publish` method is a three-element string array containing the address of the SharePoint server, a unique name for the list, and an optional description of the list. For example, the following code publishes the list created in the preceding section:

```
Set lst = ws.ListObjects("Test List")
Dim str As String
Dim dest(2) As Variant
dest(0) = "http://wombat.sharepointsite.com"
dest(1) = "Test List"
dest(2) = "A description goes here..."
str = lst.Publish(dest, True)
MsgBox "Your list has been shared. You can view it at: " & str
```

The Publish method returns a string containing the address of the published list. The preceding code displays that address in a message box, but you may want to navigate to that address or include a link to it somewhere on the sheet. To add a hyperlink to the list on the SharePoint server, add a hyperlink to a range:

```
Dim lnk As Hyperlink
Set lnk = ws.Hyperlinks.Add(Range("A6"), str)
```

After adding the hyperlink, you can display the Web page for the list by using the Follow method:

```
lnk.Follow
```

To navigate to the list without adding a hyperlink, use the FollowHyperlink method:

```
ThisWorkbook.FollowHyperLink str
```

The ListObject's SharePointURL property returns the address of the list, so it is easy to get the address of the shared list after it has been created, as shown here:

```
str = ws.ListObjects("Test List").SharePointURL
Set lnk = ws.Hyperlinks.Add(Range("A6"), str, , _
  "Click to display list site.", "View")
```

5.6.7. Inserting a Shared List

Once a list is published on a SharePoint site, you can insert that list into other worksheets using the ListObjects Add method and the SourceType argument xlSrcExternal:

```
Set ws = ThisWorkbook.Worksheets("Sheet2")
Dim src(1) As Variant
src(0) = "http://wombat.sharepointsite.com/_vti_bin"
src(1) = "0B803D34-FDA7-4E2D-A341-D1CF7FE95DE9"
ws.ListObjects.Add xlSrcExternal, src, True, xlYes, ws.Range("A1")
```

When SourceType is xlSrcExternal, the Source argument is a two-element array containing this information:

Element	Data
0	List address. This is the SharePoint address plus the folder name */_vti_bin*.
1	The name or GUID of the list. This is a 32-digit numeric string that identifies the list on the server.

To find the GUID of a list, view the list on the SharePoint server and choose Modify Columns and Settings on the list's web pagethe GUID of the list is displayed as part of the URL in the Address bar of the browser.

5.6.8. Refreshing and Updating

Use the ListObject's Refresh method to discard changes to the list on the worksheet and refresh it with data from the SharePoint server as shown here:

```
lst.Refresh
```

Use the UpdateChanges method to send data from the worksheet list to the SharePoint server and retrieve new and changed data from the SharePoint server:

```
tst.UpdateChanges xlListConflictDialog
```

As mentioned earlier, if two authors modify the same item in a list, a conflict will occur when the second author updates his or her list. The `iConflictType` argument determines what happens when a conflict occurs. Possible settings are shown here:

Setting	Result
`xlListConflictDialog`(the default)	Conflict displays dialog.
`xlListConflictRetryAllConflicts`	Worksheet data wins conflict.
`xlListConflictDiscardAllConflicts`	Server data wins conflict.
`xlListConflictError`	Conflict causes error.

5.6.9. Unlinking, Unlisting, and Deleting

Use these `ListObject` methods to unlink, unlist, or delete a list:

Method	Use to
Unlink	Remove the link between the worksheet list and the SharePoint list.
Unlist	Convert the worksheet list to a range, preserving the list's data.
Delete	Delete the worksheet list and all its data.

Once you have unlinked a list, you can't relink it. To reestablish the link, you must delete the list and insert it back onto the worksheet from the SharePoint list.

5.6.10. Resources

To get	Look here
Office 2003 web components	Search http://www.microsoft.com/downloads/ for "Office Web Components."
Information on Office 2003 web component licensing	http://support.microsoft.com/default.aspx?scid=kb;en-us;828949
Office 2003 web parts	Search http://www.microsoft.com/downloads/ for "Office Web Parts."
Help on the Office Spreadsheet web part	*C:\Program Files\Common Files\Microsoft Shared\Web Components\11\1033\OWCRSS11.CHM*
Programming information on the Excel ShareWorkspace objects	*C:\Program Files\Microsoft Office\OFFICE11\1033\VBAOF11.CHM*

Chapter 6. Using Document Libraries with Word

With the exception of lists, Word provides the same SharePoint features that Excel does. In fact, creating workspaces, adding documents, members, alerts, and tasks are all done the same way through the Shared Documents task pane in Word, Excel, and PowerPoint.

However, you'll rely on some SharePoint features more in Word than in Excel. In particular, document libraries are of key importance. This chapter shows how to use document libraries to create, organize, revise, and approve or reject Word documents.

Remember, you can use document libraries with any type of document (Excel, PowerPoint, web page, etc.), but it's hard to have a discussion in such general terms. I hope you'll find this practical approach useful and be able to apply it to other applications as well.

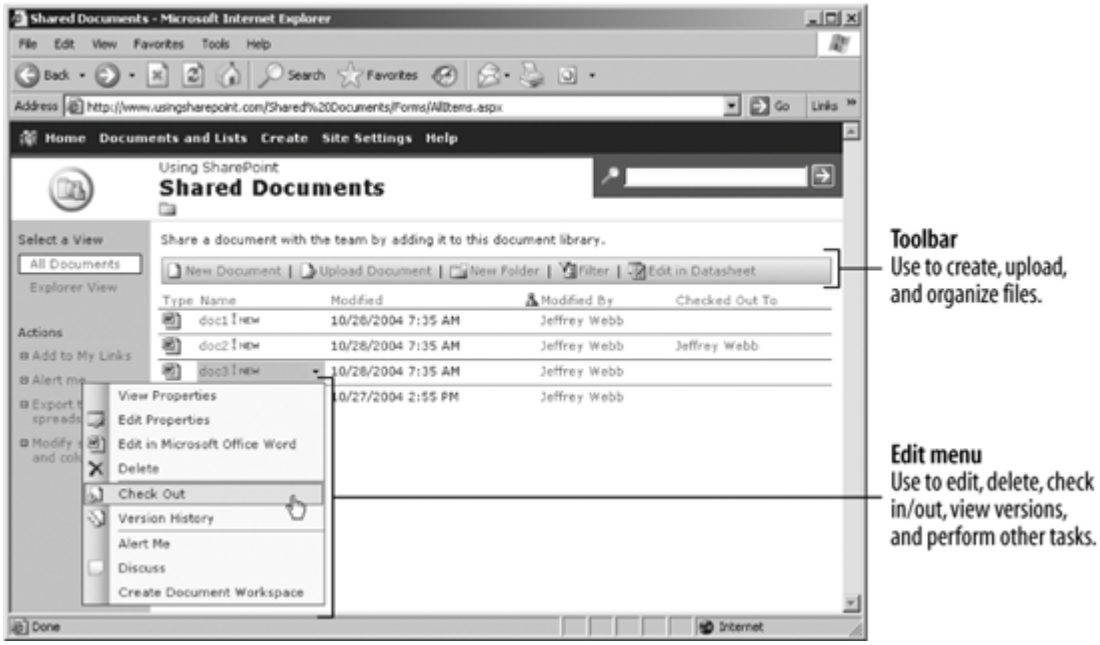
6.1. Understanding Libraries

Libraries organize content within a site. Technically, they are a special type of list that provides these key features:

- Templates for creating new documents
- A status field that indicates whether the document was approved or rejected
- Storage for previous versions of documents
- The ability to reserve documents by checking them in/out
- Synchronization of list columns with properties stored in the document

Most types of SharePoint sites include a Shared Documents library when they are created. You can create new documents, upload existing ones, create folders, check out files, and sort or filter the contents of the library using the toolbar and Edit menu as shown in Figure 6-1.

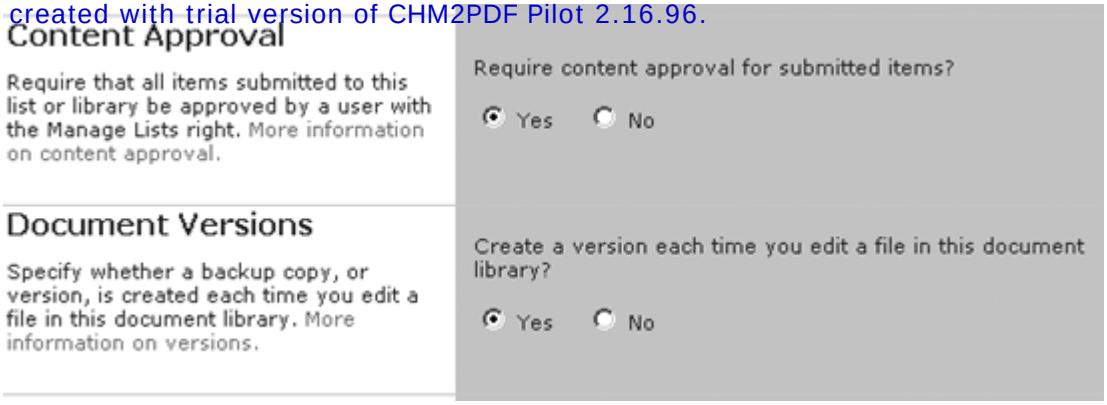
animal 6-1. Using the library toolbar and context menu



The standard Shared Documents library doesn't include approval status or version history. To enable those features:

1. Choose Modify settings and columns → Change general settings and select Yes for the Content Approval and Document Versions sections (Figure 6-2).
2. Choose OK to make the change.

animal 6-2. Enabling approval and versioning for a library



Sites and subsites can contain any number of document libraries. You should create a separate library for each type of document you want to share through the site. The term "type" is awfully vague; here are a couple factors to help you decide when to create a new library and where to put it:

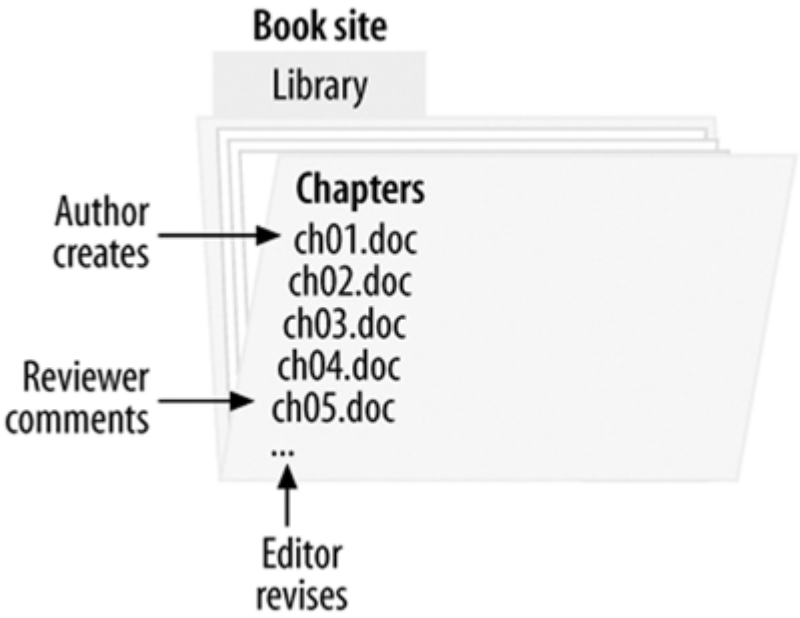
- Libraries can include a template for creating new documents. You can only have one template per library, so create separate libraries for each template you use.
- Templates are associated with applications (Word, Excel, PowerPoint). If you're using templates, create a separate library for each type of document (*.doc*, *.xls*, *.ppt*).
- SharePoint performs searches at the site level, so include all libraries you want members to be able to search in top-level sites rather than subsites.

Here are two different scenarios that illustrate some of the choices you face when creating document libraries:

- For large written works (such as this book), it's important to store documents in a central place where authors, editors, and reviewers have access. It's important to keep track of versions of each document and for multiple authors to be able to work on a single document.
- For shorter works (letters, forms, etc.), tracking versions is less important, but organization, templates, and approval become critical.

Figures 6-3 and 6-4 show these two scenarios.

animal 6-3. Using SharePoint to create a book



The following sections describe the specific tasks you can accomplish with document libraries.

6.2. Adding Documents to a Library

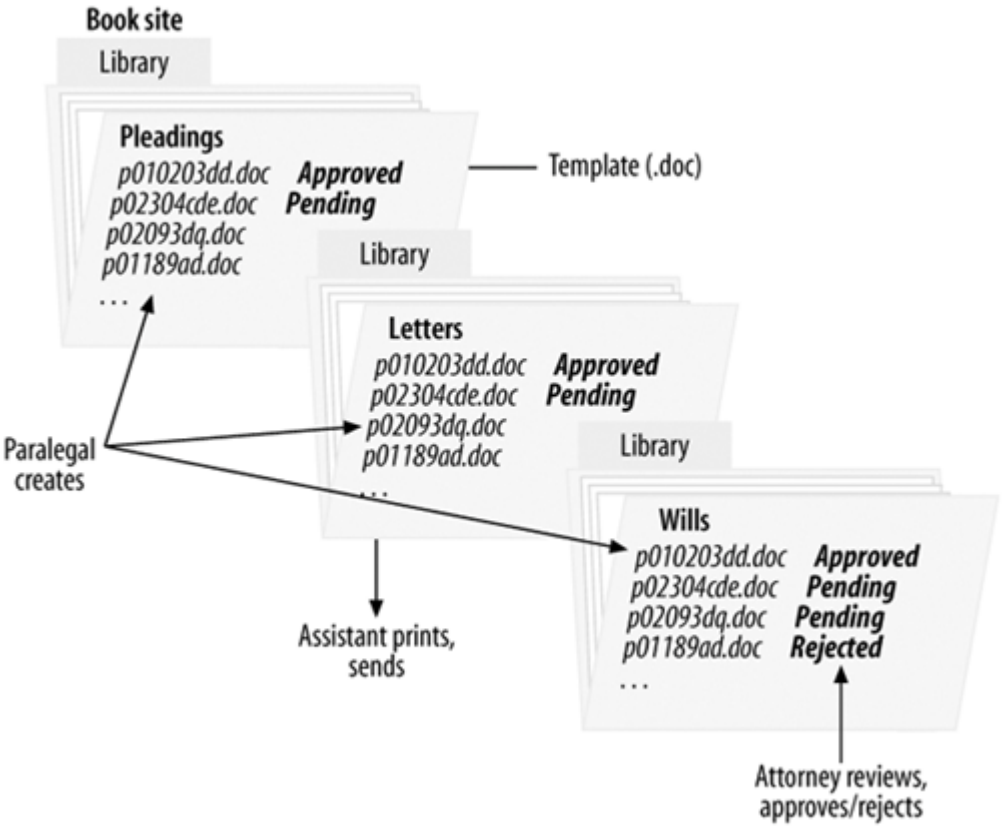
To copy documents from your computer to a SharePoint library:

1. Navigate to the library in the browser and choose Upload Document.
2. SharePoint displays the Upload page. Choose Browse to copy a single file or choose Upload Multiple Files to copy a group of files.
3. Select the file(s) to copy and choose Save and Close. SharePoint copies the files into the library.

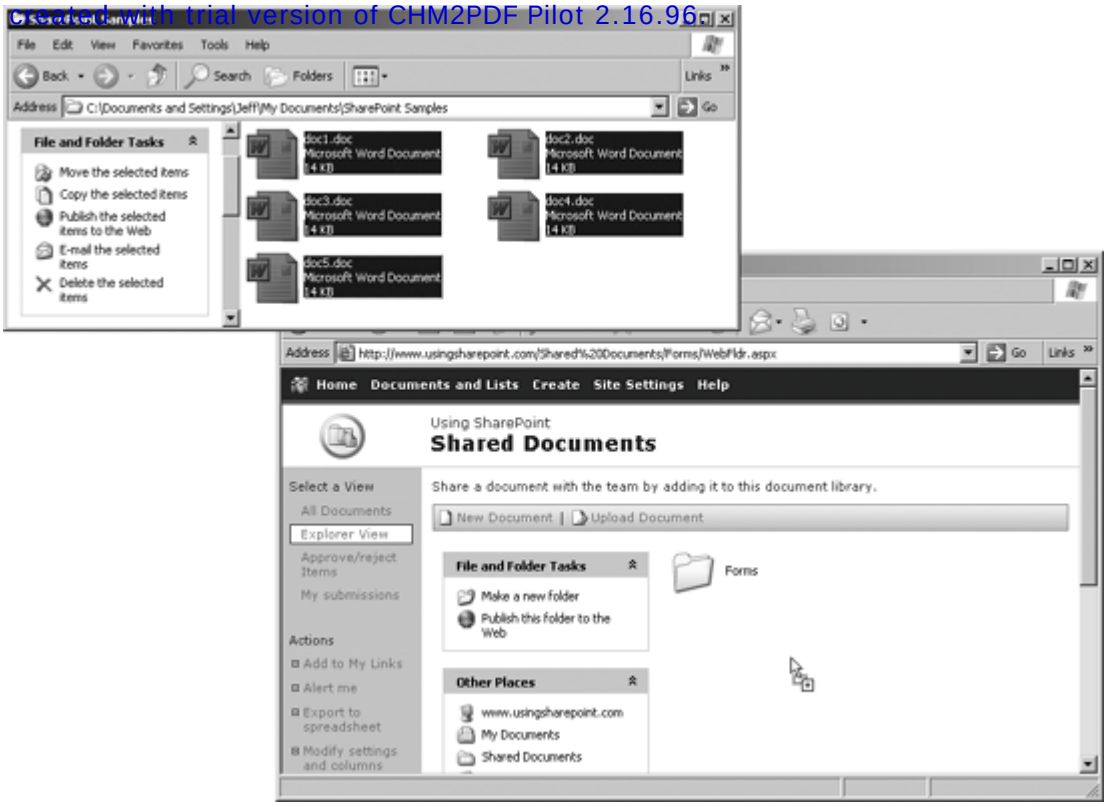
To copy groups of files into a SharePoint library quickly:

1. Navigate to the library and choose Explorer View.
2. Open an Explorer window on your computer, such as My Documents.
3. Drag and drop files from one window to the other as shown in Figure 6-5.

animal 6-4. Using SharePoint to manage legal documents



animal 6-5. Using the Explorer view to drag/drop files into a library



The Explorer view shown in Figure 6-5 is also handy for moving or copying files from one SharePoint library to another. This happens frequently when you are first setting things up and learning how to organize your work through SharePoint.

You can't drag and drop files from one SharePoint library to another, but you can use Copy (Ctrl+C), Cut (Ctrl+X), and Paste (Ctrl+V). For example, to move files from one library into another:

1. Navigate to the library you want to copy from and choose Explorer View.
2. Select the files you want to move and Cut (Ctrl+X).
3. Choose File → New → Window to open a new browser window. (This step is optional.)
4. Navigate to the library you want to copy to and choose Explorer View.
5. Paste (Ctrl+V) to move the files.

This may seem very simple, but you'd be surprised how many new users don't know about the Explorer View.

6.3. Creating New Documents

There are two main ways to create new documents in an existing library. They differ based on where you start:

- Creating a new library document from Word is familiar to most of usthere's little new to learn.
- Creating from the library itself can provide a template to ensure all the documents are consistent.

The following sections show the two approaches.

6.3.1. Starting from Word

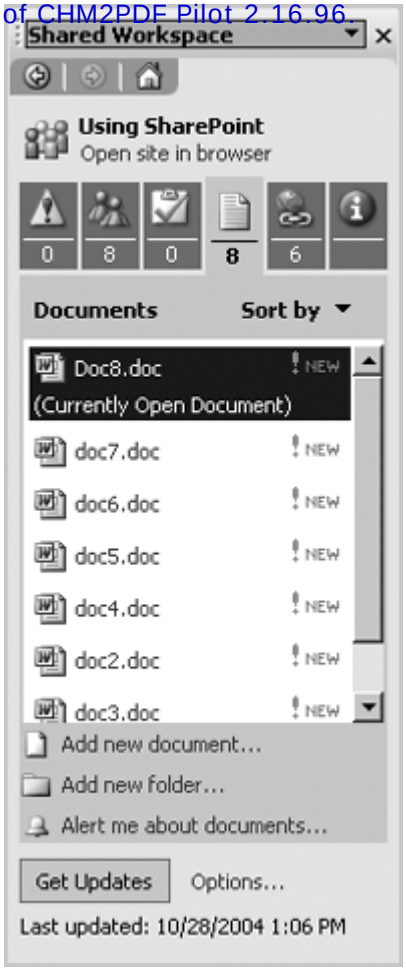
To create new document in a library from Word, simply save the document to a SharePoint library:

1. Create a new document in Word.
2. Choose File → Save.
3. Enter the address of the SharePoint library in File Name (for example, http://wombat/Shared Documents/). Word displays a list of the files in the library.
4. Enter a name for the file and choose Save. Word saves the file to the library.

Once you save a document to a library, Word can display information about the library in the Shared Workspace task pane. To see the other documents in the library from Word:

1. Choose View → Task Pane. Word displays the Shared Workspace task pane.
2. Choose the Documents icon to view other files in the library (Figure 6-6).

animal 6-6. Viewing the library from Word



Word refers to this task pane as a workspace, even though it displays a document library from a team site.

6.3.2. Starting from the Library

To create a new document from the library:

1. In your browser, navigate to the library and choose New Document.
2. SharePoint displays a security warning. Choose OK.
3. SharePoint starts Word and creates a new document based on the library's document template.
4. Save the new document. SharePoint sets the save location to the SharePoint library. The new document won't appear in the library until you save it there.

As with starting from Word, you can view other documents in the library by choosing View → Task Pane.

6.4. Adding Document Properties

When you add a document to a library, SharePoint reads the document's `Title` property and adds that information to the list. To see the title in the default view of the list:

1. Choose Modify settings and columns → All Documents (in the Views section). SharePoint displays the Edit View page.
2. In the Columns section, select Title, change Position from **`left`** to **`3`**, and choose OK.

Now when you display the list, the `Title` property appears as one of the columns. The interesting thing about this is that if you edit the property in SharePoint, it changes in the document as well. Try it:

1. Navigate to the library and choose Edit Properties from the document's Edit menu.
2. Change the Title property and choose Save and Close.
3. Open the document in Word.
4. Choose File → Properties, change the title, and choose OK.
5. Save and close the document in Word.
6. Return to the SharePoint document library and refresh the page to see the change.

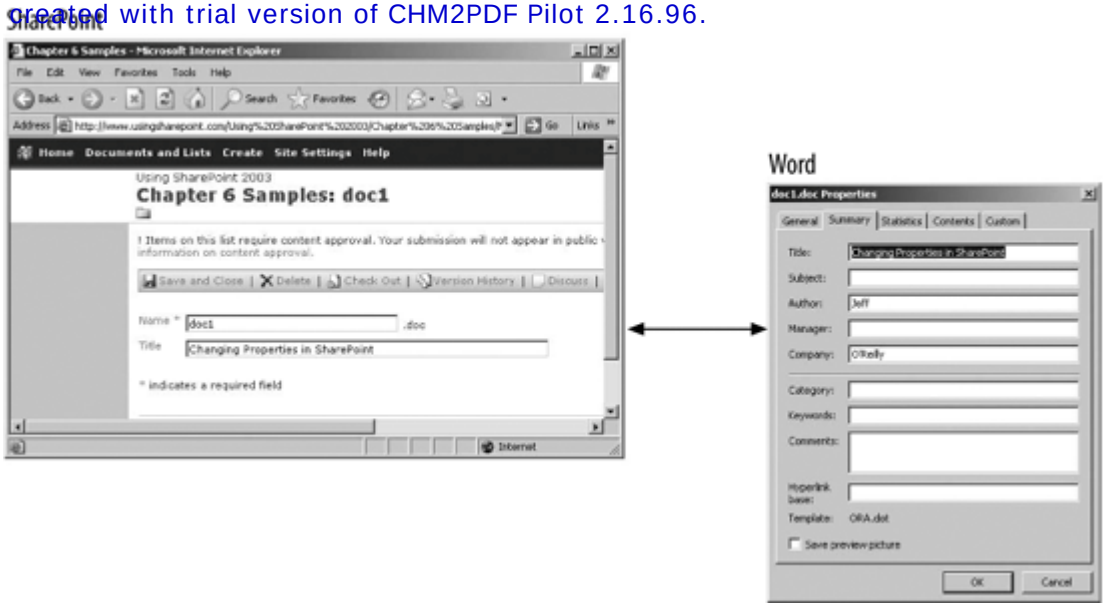
Figure 6-7 illustrates the connection between the properties in SharePoint and Word.

`Title` is the only built-in property that SharePoint maintains in this way, but you can add custom properties in SharePoint to keep additional information synchronized between the library and the documents it contains. For example, to add a Keywords custom property to the library:

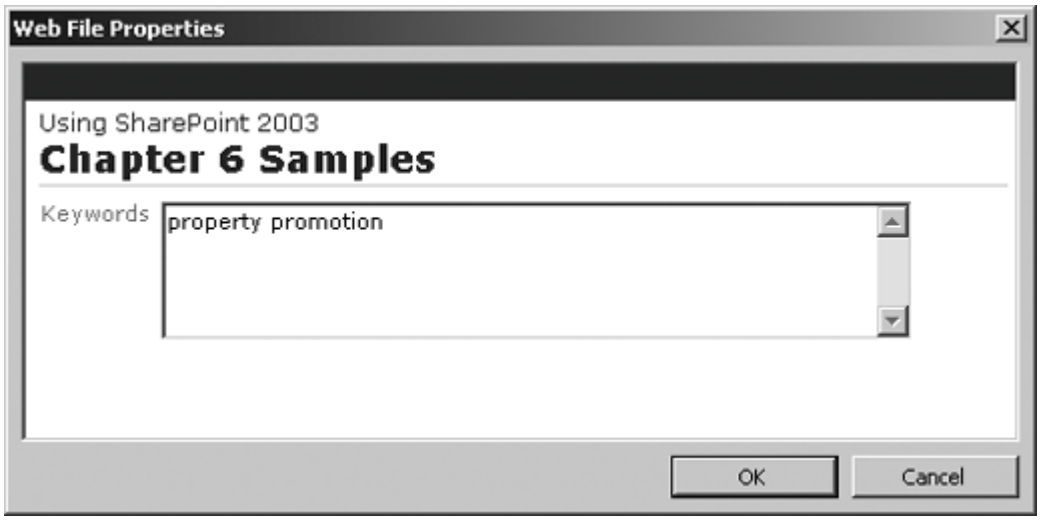
1. Navigate to the library and choose Modify settings and columns → Add a new column. SharePoint displays the Add Column page.
2. Enter **`Keywords`** in Column Name, select Multiple lines of text and choose OK.

Now when you choose Edit Properties in the library, SharePoint includes the Keywords field on the page. More interestingly, when you add a new file to the library, SharePoint now prompts you for the Keywords property as shown in Figure 6-8.

animal 6-7. Changing Title in SharePoint updates Word and vice versa



animal 6-8. Saving a document with a custom SharePoint property



After the custom property is set, you'll see the prompt shown in Figure 6-8 any time you choose File → Properties while editing the document, but Word adds a File Properties button that you can click if you want to see the standard Word Properties dialog box. The Keywords property is actually stored as a Word custom property named `Keywords0`.

SharePoint synchronizes document library columns with property settings in the document whenever the document is saved to the library. Microsoft calls this *property promotion*which is a useful search term for finding the details on how it works.

6.5. Changing the Library Template

The key advantage of creating new documents from the library is that SharePoint provides a template. The default template (*template.doc*) is really just a placeholder for your own template. SharePoint uses *.doc* files rather than *.dot* files for templates because it doesn't actually tell Word to create a new file when you select New Document, but just downloads the template as a sort of starting point.

To change the template used by a library:

1. Create the new template in Word as a *.doc* file. If you are starting with a Word template (*.dot*), just create a new file based on that *.dot* file.
2. Navigate to the library and choose Explorer View.
3. Open the Forms folder (which is hidden in Other Views).
4. Drag and drop or copy the new template to the Forms folder.
5. Choose Modify columns and settings → Change general settings.
6. Enter the name of the new template in the Templates section and choose OK.

Now when you choose New Document from the library, SharePoint uses your new template document rather than *template.doc*.

Document library templates are limited to the file types that SharePoint can open for editing: Word (*.doc*), Excel (*.xls*), or PowerPoint (*.ppt*). SharePoint also provides specific library templates for some other file types, such as FrontPage web pages (*.html*), Web Part pages (*.aspx*), Picture Libraries (image types), and InfoPath Form Libraries (*.xml*).

6.6. Linking Documents to Libraries

Keeping a local copy of a document linked to SharePoint is useful for any file you use a lot, because you can just open it from your desktop. It's also handy when traveling, since you don't need a network connection to access the file. Changes can be synchronized later, when you have access to the network again.

Whether you can link a SharePoint document to a local copy depends on where the document is stored in SharePoint and how it was created. Only documents that are part of a document workspace created from the Shared Workspace task pane can be linked with local copies of the document.

If you are working from a library that is part of a team site, you must create a workspace for the document before it can be linked. To do that:

1. Open the document from the SharePoint library.
2. In Word, choose File → Save As and save the file to a folder on your computer. Word displays the Shared Workspace task pane once the file is saved.
3. In the task pane, choose Create. Word creates a new document workspace and links the local copy.
4. Choose File → Properties → Custom and create a `_SourceUrl` property with the value of the source library's web address; for example, http://wombat1/Shared Documents/.

The last step is optional, but it enables you to publish the document to update the library with your changes. See the next section for details.

Although you can also create a document workspace from a library by using the context menu for the document, as shown in Figure 6-10, this method prevents linking.

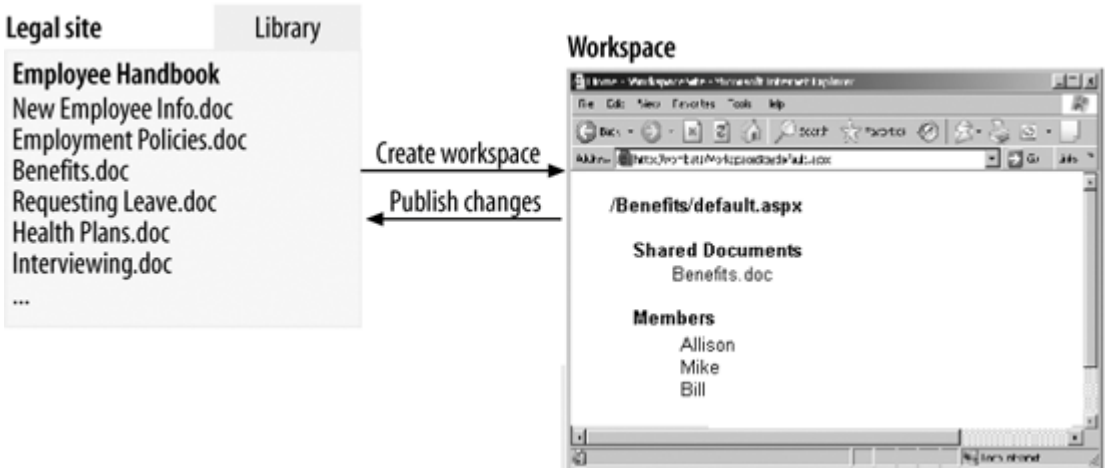
6.7. Making Revisions Privately

Document libraries may contain works in progress, or they may contain only final documents. Until now, we've mostly discussed the works-in-progress scenario, in which changes are made directly to the library. In the final-documents scenario, changes are made offline in a document workspace, reviewed, approved, and then published back to the library when complete.

Making changes through a document workspace restricts who can see the changes before the documents are complete because only members of the workspace can read or edit the document. Workspace members can also create local document copies that are linked to SharePoint, as described in the preceding section.

One example of a final-documents scenario is the HR site in Figure 6-9. All employees have access to the current version of the Employee Handbook library, but HR members can revise it out of the view of others, cycle it through the approval process, and publish it once revision is completed.

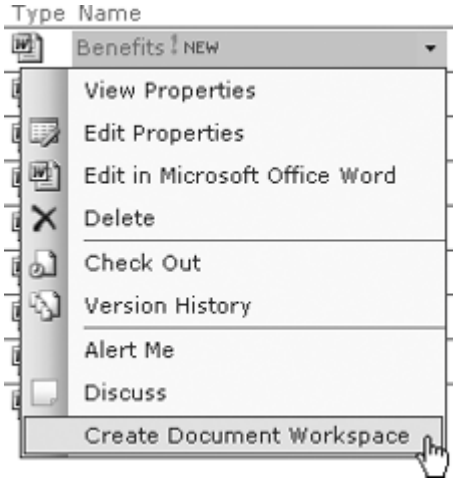
animal 6-9. Creating a workspace from a library to make revisions out of the view of others



There are two ways to create a workspace for a document from a library. The technique you choose depends on whether you want to allow members to create local copies that are linked to the workspace:

- To enable linking, follow the steps in the earlier section, "Linking Documents to Libraries."
- To prevent linking, choose Create Workspace from the context menu in the document library, as shown in Figure 6-10.

animal 6-10. Creating a workspace from a library (prevents linking)



Once you create the workspace, you can add other documents and members, set alerts, or assign tasks as described in Chapter 5. When the changes are complete, publish the document back to the library so everyone can see the new version.

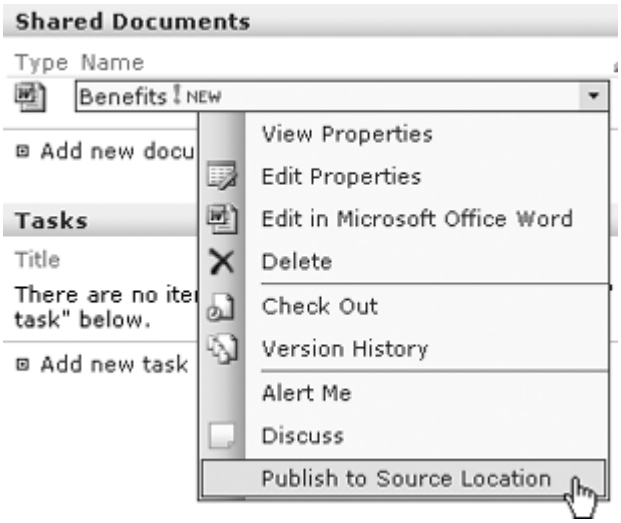
To publish a document from a workspace back to its library:

1. Navigate to the document workspace.
2. From the document's context menu, choose Publish to Source Location as shown in Figure 6-11.
3. SharePoint displays a confirmation page. Choose OK to publish the document.

After you publish the document, you may want to delete the document workspace you created for the revisions. Before deleting the workspace, you should notify the members like this:

1. Open the document from the document workspace.
2. In the Word Shared Workspace task pane, choose Members → Send email to all members. Word creates an empty message in Outlook.
3. Send a message informing the members that you are going to delete the workspace. This action disconnects any linked copies members may have created. Those copies still exist; they just won't update the SharePoint site.

animal 6-11. Publishing a final version back to the library



4. Close Word and wait a bit to see if anyone objects.

To delete the workspace:

1. Navigate to the workspace.
2. Choose Site Settings → Go to Site Administration → Delete this site. SharePoint displays a warning page.
3. Choose Delete to delete the workspace.

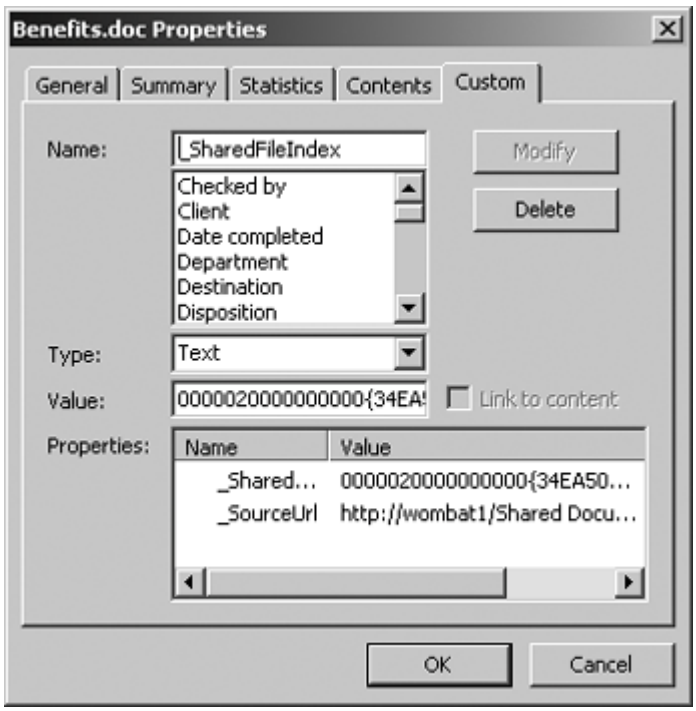
6.8. Linking and Publishing Custom Properties

Word documents use two custom properties to enable linking and publishing. First, `_SourceUrl` is the address of the source library that a document workspace uses when you choose Publish to Source Location. `_SharedFileIndex` is the site ID Word uses to locate the document workspace when you open a linked file from your computer.

Of the two, `_SourceUrl` is the most straightforward since it contains a simple web address. The `_SharedFileIndex` property is a much longer value that identifies the document in the workspace. Figure 6-12 shows both custom properties in Word.

In the previous section, "Linking Documents to Libraries," I instructed you to set the `_SourceUrl` manually to enable publishing from a linked document. That might seem like a hack, but it's the only way I know. Word doesn't enable both features, even though they are useful when combined.

animal 6-12. Linking and publishing custom properties



For you programmers out there: the value of `_SharedFileIndex` is a GUID identifying the attachment in the list. You can retrieve that information using the `FindDwsDoc` method of the DWS web service.

6.9. Discussing a Document

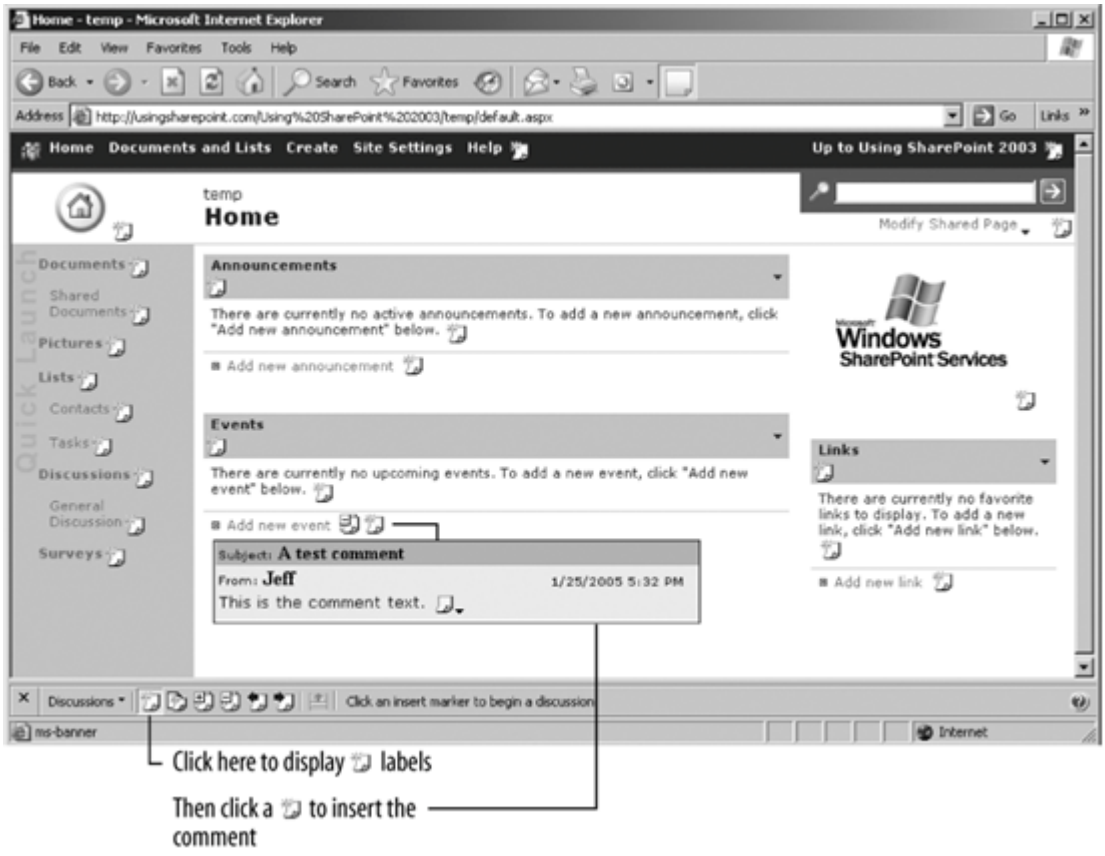
You can discuss a document online by selecting Discuss from the document's context menu, as shown in Figure 6-11. Doing so for a Word document opens the document in the browser and allows you to add comments in the Discussion pane at the bottom of the screen.

Comments entered as a discussion are saved when you navigate away from the document in the browser, but adding and reading items is so confusing, I'd recommend you avoid Discussions when working with Word documents. Instead, open the document in Word and use the Reviewing toolbar to enter comments and revisions.

Discussions seem better-suited for web pages (*.html*, *.aspx*). In those cases, the Discussion tags appear as Post-it? -like notes on the page as it is displayed in the browser, and it is easy to add or reply to comments in the discussion through the browser, as shown in Figure 6-13.

Discussions are maintained separately from the document library. To manage discussions in SharePoint, select Site Settings → Go to Site Administration → Manage Web Discussions.

animal 6-13. Adding Discussion comments to a SharePoint web page



6.10. Enabling Emailed Submissions

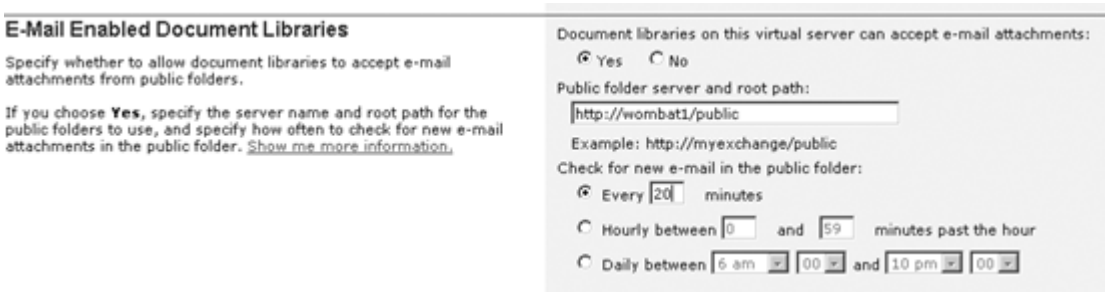
If your company uses Exchange Server, you can create document libraries that automatically add documents emailed to an address. Before you can use this feature, your Exchange Server administrator must:

1. Create a public folder.
2. Grant read access to the application pool account used by your SharePoint virtual server in IIS.
3. Grant an email distribution list permission to post to the public folder.
4. Create an email address for the public folder and enable the public folder to receive email messages.

Once the public folder is configured, your SharePoint Server administrator must enable the email-enabling feature:

1. From SharePoint Central Administration, choose Configure virtual server settings → the virtual server to enable → Virtual server general settings. SharePoint displays the general settings page.
2. Scroll down to the E-Mail Enabled Document Libraries section, select Yes, and enter the path to the public folder as shown in Figure 6-14.
3. Choose OK to make the change.

animal 6-14. Enabling email to document libraries from SharePoint Central Administration



Once these major administrative steps are complete, you can enable individual libraries to accept documents by email. To do that:

1. Navigate to the library.
2. Choose Modify settings and columns → Change advanced settings. SharePoint displays the advanced library settings page. If you don't see this link, email submissions have not yet been enabled for the site.
3. In the E-Mail Settings section, enter the path to the Exchange public folder from which the library should get emailed documents (Figure 6-18).
4. The path is relative to the root location shown in Figure 6-14, so if the library's public Exchange folder is http://wombat1/public/SharedDocs, enter **/SharedDocs**. Each public folder is associated with an email address in Exchange. Members can now submit files to the library by emailing them as attachments to that address.

6.11. Approving/Rejecting Documents

The section "Understanding Libraries" earlier in the chapter showed how to enable content approval for a library. If you select Yes in Content Approval (Figure 6-2), new documents must be approved before they appear to all members. Members of Web Designer and Administrative groups have the honor of approving or rejecting documents. To approve or reject pending documents:

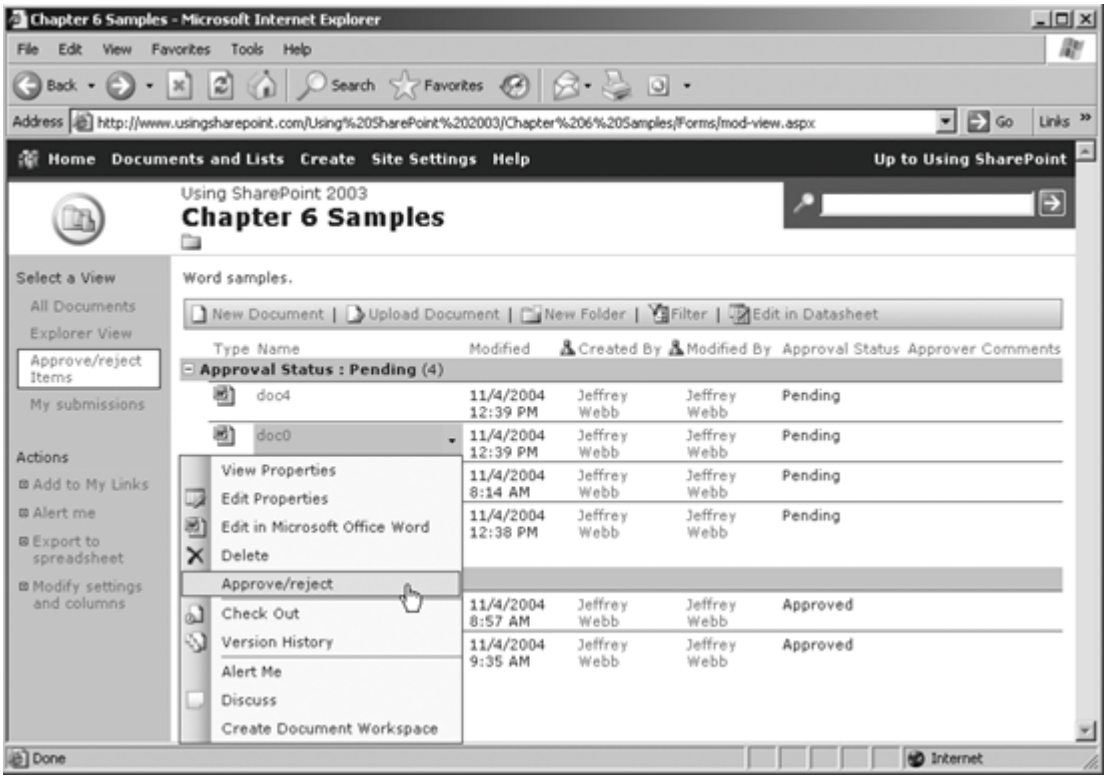
1. Navigate to the library and choose Approve/reject items.

2. Select a document and choose Approve/reject from the Edit menu as shown in Figure 6-15. SharePoint displays the approval status page.

3. Select a status (Approved, Pending, or Rejected), enter a comment, and choose OK. SharePoint changes the status of the document.

When you enable approval status, SharePoint adds views for changing the status and for checking documents that a member has submitted. The My Submissions view lets members check the status of their own documentsthe list is divided into Pending, Approved, and Rejected sections similar to those in Figure 6-15.

animal 6-15. Approving/rejecting new submissions



Members of the Contributor, Web Designer, and Administrator groups can add documents to a library, but only Web Designers and Administrators can approve or reject items. Your SharePoint administrator may want to create a Manager group that grants the Manage Lists right to the standard Contributor rights as shown in Figure 6-16, then members that need to approve/reject documents may belong to the Manager group rather than the Web Designer group.

6.12. Responding to Events

Chapter 5 showed how to create email alerts that notify you when a document changes. Email alerts are sent for several general types of changes. You can also respond to changes by enabling an event handler for the library. *Event handlers* are code blocks activated by specific occurrences, enabling you to respond to specific changes, such as document approval or rejection, and create a response that is more complex than sending email.

Event handlers are written by programmers. As a member, you need to know how they work and how to enable them for your library. Event handlers can respond to these types of events:

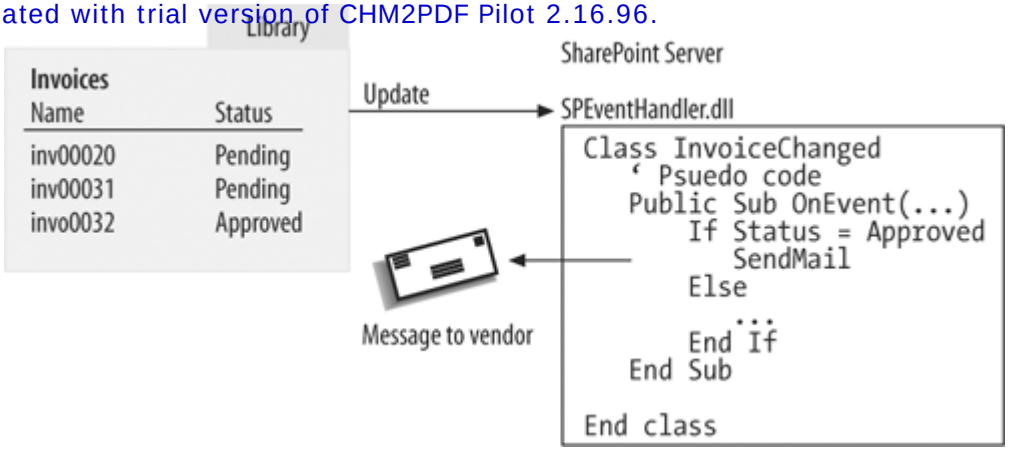
- Cancelling check-out
- Checking in
- Checking out
- Copying

animal 6-16. Creating a new member group that can approve/reject documents

- Deleting
- Moving or Renaming
- Updating

When an event occurs in the document library, SharePoint calls the event handler DLL and provides an `SPListEvent` object that includes the type of event, the document that changed, and the properties of the document before and after the change. The code in the DLL uses that information to decide what to do. For example, your company might use an event handler to send notification to a vendor when an invoice is approved. Figure 6-17 shows how the process would work.

animal 6-17. Using an event handler to notify a vendor



Before you can use event handlers, your SharePoint administrator must enable events for the server, and your programming staff must create and install the DLL that handles the event. Usually, they will create one DLL that contains a separate class for each library.

To enable an event handler for a library:

1. Get the DLL name and class name of the event handler from your programming staff.

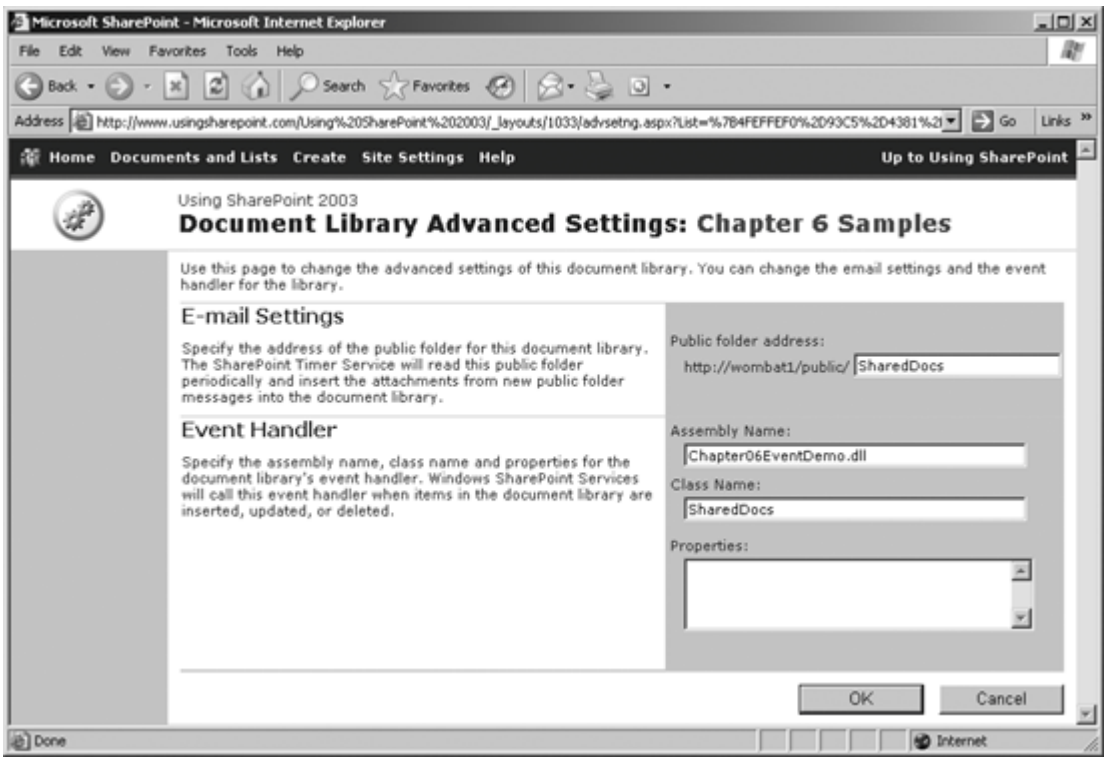
2. Navigate to the document library.

3. Choose Modify settings and columns → Change advanced settings. SharePoint displays the advanced library settings page. If you don't see this link, events have not yet been enabled for the server.

4. In the Event Handler section, enter the DLL name and class name you got in step 1, and choose OK.

Figure 6-18 shows the event handler settings for a document library. This is the same page used to enable email submissions.

animal 6-18. Setting advanced options on a library



The Properties field shown in Figure 6-18 is for special items the event handler may use. Ask your programming staff if you need to enter anything there.

6.13. Searching for Documents

If searching is enabled for your SharePoint server, a Search box appears in the upper-right corner of each home page, as shown in Figure 6-19. See Chapter 2 for information on enabling searching if that box does not appear.

animal 6-19. Searching a site

SharePoint searches the current site and displays the documents and lists that contain that term. The search can look inside of Office documents and other file types that implement search filters for Microsoft SQL Server 2000. SQL includes the filters for Word and Excel documents, but other document types, such as Adobe PDF files, must be installed separately.

Search excludes these types of list fields:

- Nontext fields
- Attachments
- Hidden fields or lists
- Surveys
- Built-in Office document properties such as Author and Keywords

Search also doesn't support expressions formed with operators such as `AND`, `+`, or `OR`. Using more than one term returns all documents that include *either* term, so the searching really works best when used with a unique term such as a customer's last name, an invoice number, or some unusual word you know was used in the document, such as "Hazelnootpasta."

6.14. Resources

To get	Look here
More information on enabling email for a library	"Configuring E-Mail Enabled Document Libraries" in WindowsSharePointServicesAdmin.chm or http://www.microsoft.com/resources/documentation/wss/2/all/adminguide/en-us/stse15.mspx

Chapter 7. Gathering Data

Collecting data from users is one of the central problems in the computer world. It's been addressed many different ways, usually through some sort of tool that displays data entry forms on screen, then validates and stores the entries. SharePoint provides two different approaches for gathering data:

- Lists collect simple tables of items.
- Form libraries collect complex data sets.

How do you know if the data you want to gather is simple or complex? Mostly, this has to do with the relationship between items. Lists are tables containing rows and columns. It's hard to create hierarchical relationships or complex entry forms, so lists are best for items where most columns contain simple values.

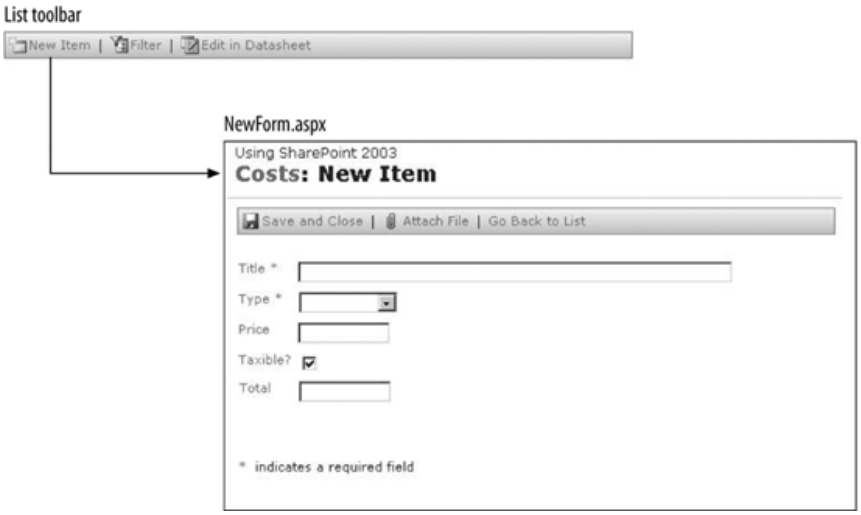
Form libraries collect pages of data using InfoPath form templates. InfoPath supports links to databases, complex validation rules, and spell-checking, and stores data in XML format within the form library. Form libraries are more useful when the entries are linked to a database, need to be sent through email, or contain items that include hierarchical relationships.

7.1. Using Lists to Gather Data

If you've read this far, you're probably already familiar with the simple Announcements and Contacts lists. You can enter data in those lists by choosing New Item or Edit in Datasheet from the list toolbar. For simple lists, Edit in Datasheet often works best because you can create new items simply by tabbing to the next row. For lists that contain long text fields or other types of data, New Item is often a better choice.

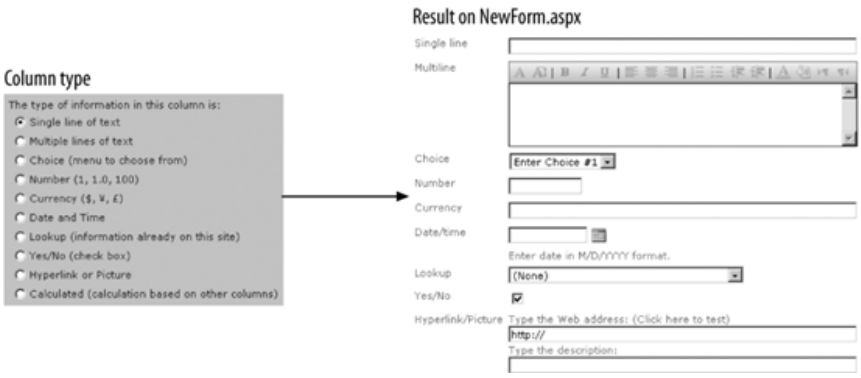
When you choose New Item, SharePoint displays the New Item page (*NewForm.aspx*). New Item displays text boxes, drop-down lists, or other controls for each field in the list. Required fields are indicated by an asterisk, as shown in Figure 7-1.

animal 7-1. Adding new items to a list



The New Item page is the data-entry form for the list. You complete the fields on the page, and choose Save and Close to store the data in the list. To exit without saving, choose Go Back to List or just navigate to another page. SharePoint generates the fields you see on the New Item page based on the types of columns found in the list, as shown by Figure 7-2.

animal 7-2. Using column type to create controls



There are a couple things you need to know about the column types:

- Multiline text columns can include formatting.
- Choice and Lookup column types generate drop-down lists.
- Numeric, currency, and date/time columns validate the data entry as that type. For example, you can't enter `five` in a numeric column.
- Calculated columns are read-only, don't appear on the New Item page, and can't be totaled in a standard view.

Even though lists seem very simple, you can do some significant data entry tasks with them. The following sections walk you through creating a list for collecting and categorizing expenses. I used it to figure out how much I spent building my boat, but you can modify it for other uses.

7.1.1. Building a Lookup Table

Before creating a data entry list, you need to consider what types of multiple-choice options members might need to enter. You can provide multiple-choice options in a list through either a Choice or Lookup column type, as appropriate:

- Use Choice to display options that aren't likely to change.
- Use Lookup to display a dynamic set of options from another list. Lookup columns can also link to data in other lists, so they are a good way to include details for complex options.

Lookup tables can't span web sites. If you are creating a list for use in multiple web sites, use the Choice type rather than the Lookup type.

The list in this tutorial categorizes expenses for a boat-building project. When I started the list I wasn't sure of all the categories I'd need, so I created a lookup table list for the main ones I could think of, then I added new categories as needed.

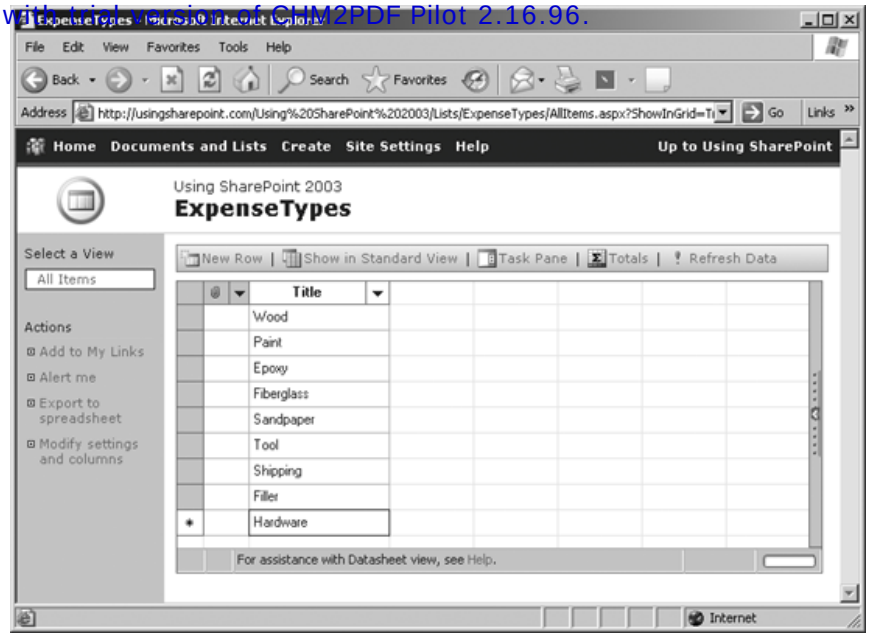
To create the lookup table list:

1. From the SharePoint site, choose Create → Lists → Custom List. SharePoint displays the New List page.
2. Name the list `ExpenseTypes`, select No in the Navigation section, then click Create. SharePoint creates a new single-column list.

This is a simple lookup table, so you don't need to add any new columns. The default Title column will do fine. Next, enter the different categories for the expenses. To add items to the lookup table:

1. Choose Edit in Datasheet.
2. Enter the items as shown in Figure 7-3.
3. Choose Show in Standard View to commit the changes.

animal 7-3. Entering lookup values



The great thing about lookup tables is that you don't have to think of all the categories right now. Later, you can return to this table and add new categories as they pop upthe changes will automatically appear in the data entry list.

7.1.2. Creating a Data List

Now that you've laid the foundation, you can create the data entry list itself. To create the list, follow these steps:

1. From the SharePoint site, choose Create → Lists → Custom List. SharePoint displays the New List page.
2. Name the list `Expenses`, select No in the Navigation section, and then click Create. SharePoint creates a new, single-column list.
3. Choose Modify settings and columns to add columns to the list. SharePoint displays the Customize page.
4. In the Columns section, choose Add a new column. SharePoint displays Figure 7-4.
5. Enter the information for the Date column, then repeat step 4 for each of the columns using the settings listed in Table 7-1.

animal 7-4. Adding columns to the list

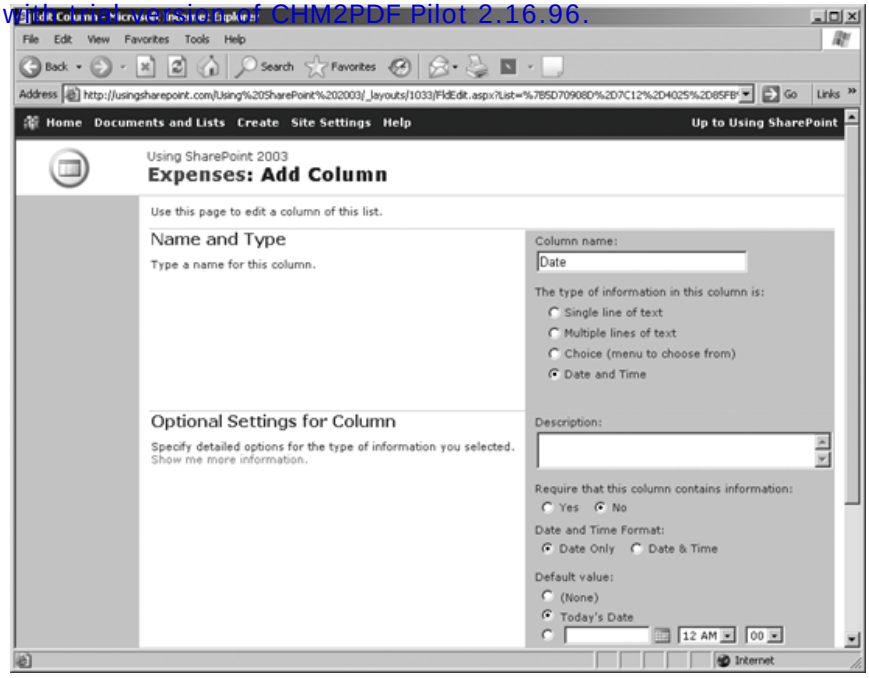


Table 7-1. Data list column settings

Name	Column type	Required?	Default value	Other settings
Date	Date/time	No	Today's date	
Type	Lookup	Yes		Get information from: `ExpenseTypes` In this column: `Title`
Price	Currency	Yes		
Taxable?	Yes/no	N/A	Yes	
Total	Calculated	N/A		Formula: `[Price] + [Price] * [Taxable?] * .06`[1]

[a]The sales tax rate in Florida is 6%. Change `.06` to your own sales tax rate.

When done, display the list and add a test entry to make sure it works as expected:

1. Choose New Item. SharePoint displays the New Item page for the list. You should see fields for all of the columns except Total, which is read-only.

2. Enter data and choose Save and Close. SharePoint returns you to the list and displays the new item. If the test entry was taxable, the correct Total should be displayed as shown in Figure 7-5.

animal 7-5. Adding a test entry to make sure the list works

Title	Type	Date	Price	Taxable?	Total
test! NEW	Epoxy	11/11/2004	$99.00	Yes	$104.94

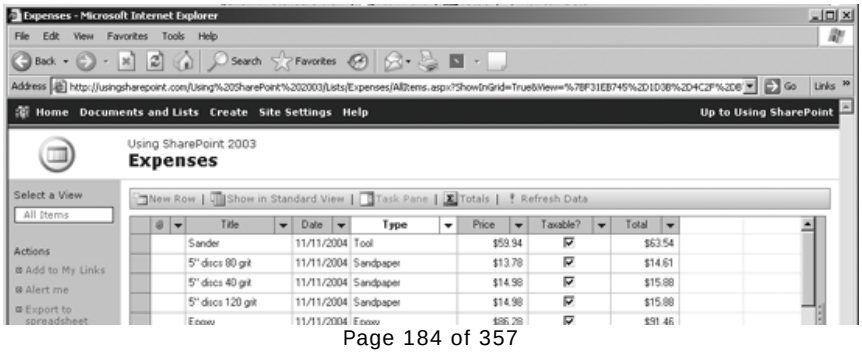
7.1.3. Adding Totals, Groupings, and Filters

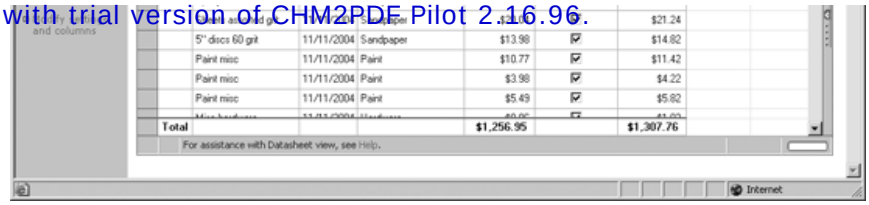
As you enter items in the Expenses list, you may want to view the data in different ways. For example, you may want a running total of what you've spent or you may want to see what you spent on epoxy. Lists can include totals, and you can filter or group items in various ways. There are two ways to view totals in a list. Here is the easiest and most effective:

1. Choose Edit in Datasheet.

2. Choose the Totals button on the list toolbar. SharePoint displays totals for all of the numeric columns as shown in Figure 7-6.

animal 7-6. Displaying totals in the Datasheet view





The Datasheet view works only in Internet Explorer and only if Office 2003 Professional Edition is installed.

The second way to view totals is to add Totals to the standard view. This approach isn't as useful as the Datasheet view because it doesn't total calculated columns. To add totals to the standard view.

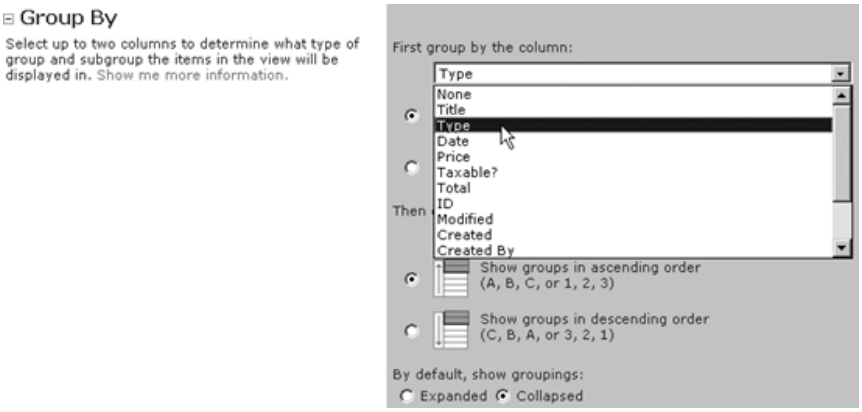
1. Choose Modify settings and columns.
2. In the Views section, choose the view to modify. The default view is All Items. SharePoint displays the Edit View page.
3. Expand the Totals section and select the type of total to display for each column as shown in Figure 7-7.
4. Choose OK when done. SharePoint adds the column total to the top of the standard view.

animal 7-7. Adding totals to a view

If you don't see the choices shown in Figure 7-7, check the data type of the column: different data types support different choices.

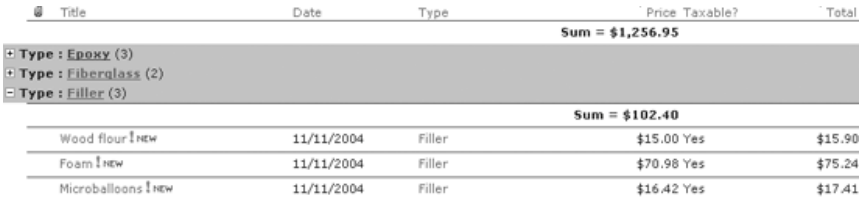
You can also modify the view to group items in interesting ways. For example, repeat the preceding procedure, but at step 3 expand the Group By section instead of Totals and make the changes shown in Figure 7-8.

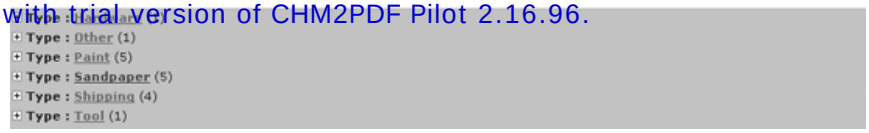
animal 7-8. Grouping items by type



Now when you display the list, you'll see a categorized list of expenses. You can expand each expense type to see the details (Figure 7-9).

animal 7-9. Viewing categorized expenses

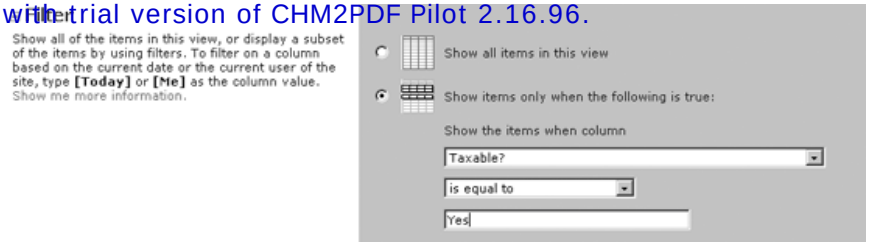




Use the Group By settings to create hierarchical views of lists. Use this view in a web part to create treeview-type controls. See http://usingsharepoint.com/Samples/TreeView.aspx for an example.

Finally, you can filter the list to display only items that meet a certain criteria. The easiest way to do that is to choose the Filter button on the list toolbar and then apply a filter to one of the list columns. However, you can also add filters to a view using the Filter section of the Edit View page, as shown in Figure 7-10.

animal 7-10. Adding a filter to a view



Adding a filter to a view prevents members from viewing excluded itemsin this case, nontaxable expenses. That's not that useful in this context, but you can use it in other lists to hide unpublished or unapproved items.

7.1.4. Modifying the New Item Form

If you add items to the Expenses list through the standard New Item form, SharePoint returns you to the standard list view after you choose Save and Close. That's OK if you only want to add one item, but if you're adding a large set of records it gets annoying. To make lists truly useful for data entry, you need to change that behavior. There are a number of ways to do that, but the easiest way I found is to follow these general steps:

1. Create a new, default view for the list.
2. Open *NewForm.aspx* in FrontPage and copy the list form web part.
3. Open the new view in FrontPage and paste the list form web part onto the view.
4. Edit the new view in FrontPage as needed.

Editing template pages such as *NewForm.aspx* or *Default.aspx* with FrontPage copies that page into the content database of the site. Afterward, changes to that page in the site template won't appear in the edited page. The page becomes unghosted.

Putting the list form web part on a default list view ensures that SharePoint returns to that view when the member chooses Save and Close. Including a list view on the data entry form is actually pretty handy, since you can filter the view to show only the member's new entries. The following sections show you how to do this for the Expenses list.

7.1.4.1 Creating a new default view

To create a new default view for adding items to the Expenses list:

1. Navigate to the Expenses list and choose Modify settings and columns.
2. Scroll down to the Views section and choose Create a new view → Standard View. SharePoint displays the Create View page.
3. Name the view `Add Items`, select the options listed in Table 7-2, and choose OK.

Table 7-2. Option settings for Add Item view

Section	Option	Setting
Name	Make this the default view	Select
Audience	Create a public view	Select
Columns	Title (linked to item)	Select
	Other columns	Clear
Sort	First sort column by the column:	Modified
Filter	Show items only when the following is true: *And* When column	Created by is equal to `[Me]` Select Created is equal to `[Today]`
Style	View Style:	Newsletter (no lines)
Item Limit	Number of items to display:	10
	Limit the total number of items returned to the specified amount	Select

When you are finished, the new view will look similar to the All Items view, but it probably won't display any items, since the filter settings hide any items not created today by you. You won't see the data-entry form yet; but don't worry, that's next.

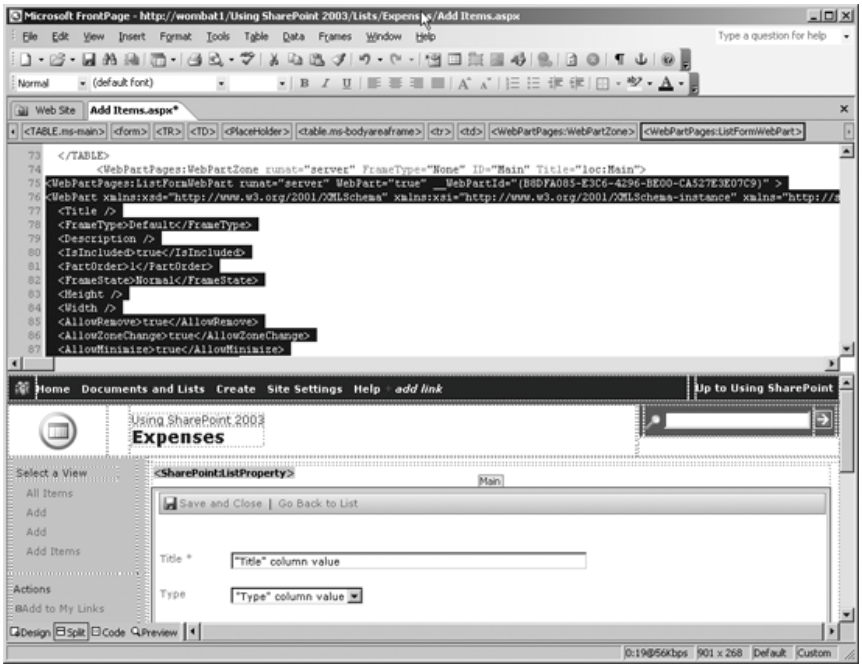
7.1.4.2 Add the list form web partlist form web part

To add the list form web part to the Add Items view:

1. Open the site in FrontPage 2003.
2. In the Folders view, select Lists → Expenses to open the folder that contains the list.
3. Open *NewForm.aspx* and copy the `WebPartPages:ListFormWebPart` element.
4. Open *Add Items.aspx*, find the `WebPartPages:WebPartZone` element and paste the copied section after it.
5. Save *Add Items.aspx*, return to the browser and view the Expenses list to verify that the form appears on the view.

Figure 7-11 shows the paste operation when complete in FrontPage.

animal 7-11. Adding the list form web part to the view in FrontPage (after paste)



Here is the list form web part in its entirety:

```
    <WebPartPages:ListFormWebPart runat="server" WebPart="true" _
_WebPartId="{B8DFA085-E3C6-4296-BE00-CA527E3E07C9}" >
    <WebPart xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.microsoft.com/WebPart/v2">
      <Title />
      <FrameType>Default</FrameType>
      <Description />
      <IsIncluded>true</IsIncluded>
      <PartOrder>1</PartOrder>
      <FrameState>Normal</FrameState>
      <Height />
      <Width />
      <AllowRemove>true</AllowRemove>
      <AllowZoneChange>true</AllowZoneChange>
      <AllowMinimize>true</AllowMinimize>
      <IsVisible>true</IsVisible>
      <DetailLink />
    <HelpLink>http://wombat1/Using%20SharePoint%202003/_vti_bin/help/1033/sts/html/dlistwps.htm</HelpLink>
      <Dir>Default</Dir>
      <PartImageSmall />
      <MissingAssembly />
      <PartImageLarge />
      <IsIncludedFilter />
      <ExportControlledProperties>false</ExportControlledProperties>
      <ConnectionID>00000000-0000-0000-0000-000000000000</ConnectionID>
      <ListName xmlns="http://schemas.microsoft.com/WebPart/v2/ListForm">{5D70908D-7C12-4025-85FB-2110E8525ADE}</ListName>
      <FormType xmlns="http://schemas.microsoft.com/WebPart/v2/ListForm">8</FormType>
      <ViewFlag xmlns="http://schemas.microsoft.com/WebPart/v2/ListForm">0</ViewFlag>
    </WebPart>
    </WebPartPages:ListFormWebPart>
```

The `ListName` element (shown in bold) connects the web part to the list. You can use the same `WebPartPages:ListFormWebPart` element in any list if you change the `ListName` setting to match the list's globally unique identifier (GUID).

To find a list's GUID:

1. Navigate to the list and choose Modify settings and columns.

2. Copy the GUID from the Address bar in the browser (Figure 7-12).

animal 7-12. Finding a list's GUID

The GUID identifies the list inside the content database. SharePoint uses GUIDs to identify lists, web parts, and many other things.

7.1.4.3 Editing the new view

Some edits you make in FrontPage are compatible with SharePoint's page management features and some aren't. Adding a new web part through FrontPage is an edit that SharePoint can maintain. So, if you add new columns to the Expenses list, SharePoint adds fields for those columns to the Add Item page automatically.

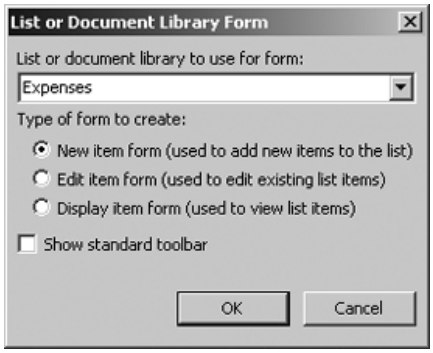
That's a great feature, but it limits you to working through web parts. If you want to provide a different user-interface for data entry to a list, you need to create a new web part and install it on the server. Those are programming tasks I cover in Chapter 8, but those tasks are too complicated for most members.

A more simple solution is to customize the web part directly in FrontPage. Doing that makes the web part staticSharePoint no longer updates it when the list changesso make sure the design of your list is final before you do this. The following tutorial removes the standard list form toolbar and replaces it with command buttons to make data entry tasks a little more obvious.

To remove the toolbar:

1. Open the site in FrontPage and create a copy of *Add Items.aspx* as a backup in case you break something.
2. Open *Add Items.aspx*, right-click the list form web part, and choose List Form Properties from the Edit menu. FrontPage displays Figure 7-13.
3. Clear the Show standard toolbar checkbox and choose OK. FrontPage removes the form toolbar from the web part.
4. Choose File → Save to save the change, and view the page in the browser to verify your changes.

animal 7-13. Removing the toolbar from the list form web part



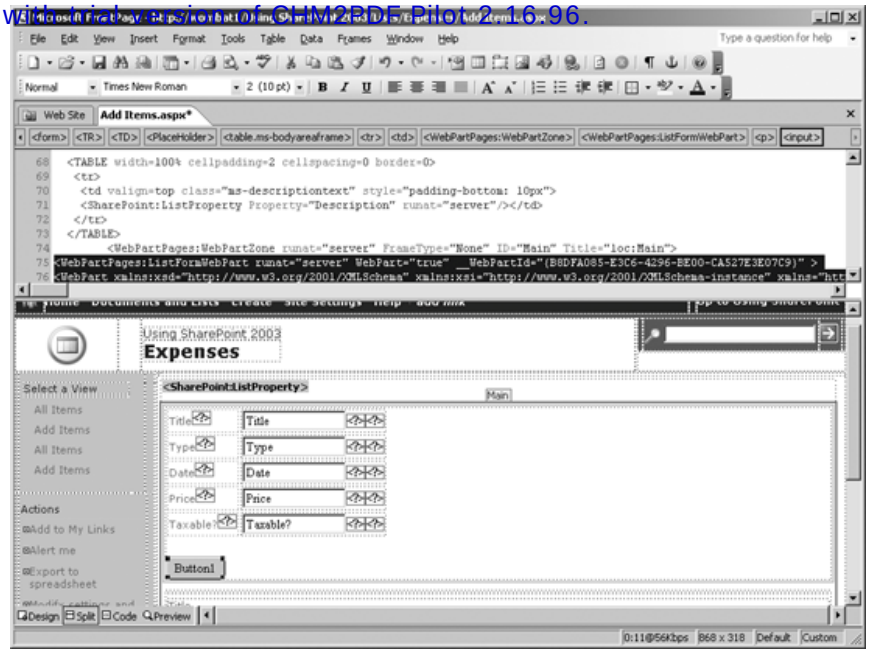
To add command buttons:

1. Right-click the list form web part, and choose Customize SharePoint List Form from the Edit menu. FrontPage switches the web part into edit mode.
2. Click after the last tag on the web part and choose Insert → Form → Push button to insert a button on the web part as shown in Figure 7-14.
3. Repeat step 2 to insert a second button.
4. Double-click the first button, label the button `Clear Form`, select the button type Reset, and choose OK.
5. Double-click the second button, label the button `Add Record`, select the button type Normal, and choose OK.
6. Edit the second buttons's HTML to include an `onclick` event:

```
<input type="button" value="Add Record" name="B2"
onclick="javascript:ClickOnce(  );">
```

7. Save the page and view the list in the browser to confirm that your changes worked.

animal 7-14. Adding a button to a web part



If you choose Revert SharePoint List Form from the web part's Edit menu, FrontPage removes the command buttons and restores the web part to its original state.

Finally, you'll also want to remove the toolbar from the data view web part that follows the data-entry form:

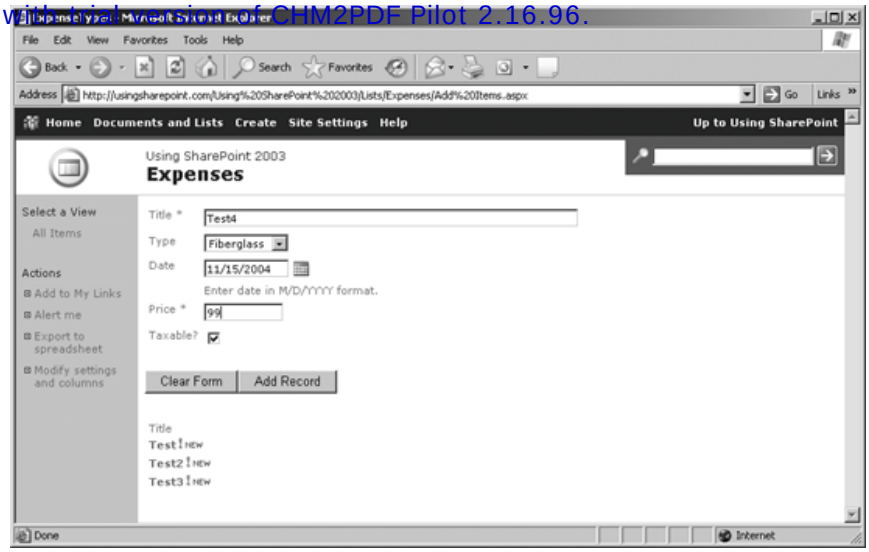
1. Right-click the data view web part and choose Convert to XSLT Data View from the Edit menu. FrontPage displays the web part in Edit mode.

2. Select the toolbar and delete it.

3. Save the page and view the list in the browser to confirm that your changes worked.

When you are done, your data-entry form should look like Figure 7-15.

animal 7-15. Testing the completed data-entry form



There are a number of key points you should remember when editing web part pages in FrontPage:

- Editing a page often means SharePoint no longer maintains it. Changes to the list or site won't automatically appear on the page.
- Selecting Revert from a web part's Edit menu in FrontPage discards your edits and returns the web part to a default state.
- Creating backup copies of files before you edit them is essential. Use the FrontPage Folders view to do so; then if the page stops working, you can delete the changed file and rename the backup copy to restore your prior version.

7.1.5. Changing Pages to View/Edit/Add Items

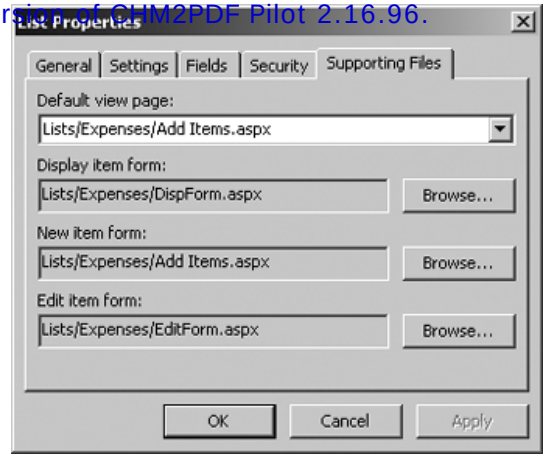
By building the data-entry form from a default list view in the preceding tutorial, I tricked SharePoint into redisplaying the form after adding each new record. I had set Add Items as the default view *before* any of the other steps, because once you modify the page in FrontPage, you can no longer change the view's properties from SharePoint. Try it:

1. Navigate to the Expenses list and choose Modify settings and columns.
2. In the Views section, choose the Add Items view. SharePoint displays a limited Edit View page. You can't add columns or change the view's settings.

If you set the default view to All Items or some other view, you can't switch the default view back to Add Items later. There's just no way to do that through SharePoint. Instead, use FrontPage to change the list's supporting file properties:

1. In the FrontPage Folders view, open the Lists folder.
2. Right-click on the Expenses list and choose Properties → Supporting Files.
3. Use the List properties dialog box to change the pages displayed to view, add, or edit items in the list, as shown in Figure 7-16.

animal 7-16. Changing the pages used to view, add, or edit list items



The List Properties dialog in FrontPage lets you change the page used to perform the built-in list actions. You can also use it to restore the original list settings, such as changing the new item form back to *NewForm.aspx*.

7.1.6. Saving the List as a Template

Part of the beauty of SharePoint is that you can easily reuse your work. In the case of the Expenses list, this means that you can save the list as a template, then use that template to create new lists for categorizing expenses on other projects. Customized pages and links to other lists are preserved.

To see how this works:

1. Navigate to the list and choose Modify settings and columns → Save site as template. SharePoint displays the Save as Template page.
2. Enter the settings in Table 7-3 and choose OK. SharePoint saves the list folder as a template and adds it to the server's list template gallery.
3. Choose Create → Lists → Expenses to create a test list based on the template.
4. Add items to the list to verify that it works.

Table 7-3. List template settings

Option	Setting
File name	Expenses
Template title	Expenses
Template description	List for categorizing expenses on a project
Include content	Clear

List templates are saved at the virtual-server level. That means the Expenses template is available to all sites on the virtual server; but since Expenses uses the ExpenseTypes lookup table, lists created from the template will only work correctly within the site that contains the ExpenseTypes list.

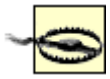
I warned you about this in "Building a Lookup Table": lookup tables can't span web sites. If you want to be able to use the list template from other sites, select the column type `Choice` for the Types column. (See step 5 of the earlier section, "Creating a Data List.")

7.2. Using Form Libraries to Gather Data

SharePoint form libraries are special document libraries for collecting data gathered through Microsoft InfoPath. Though purchased separately, InfoPath is part of the Microsoft Office System. It provides a platform for creating and displaying data-entry forms that may incorporate:

- Links to remote data sources such as SQL or Access databases
- Text fields that include simple formatting such as bold, italic, bulleted lists, etc.
- Office-like editing tools such as autocorrect and spell-checking
- Complex data validation
- Detail and summary views of data
- The ability to submit form data to a database or to an email recipient
- Property promotion (so that form data automatically becomes part of the searchable SharePoint form library)

InfoPath's form-creation tools are based on XML and its related standards. It is perhaps most useful for medium-to-large corporations that use XML schemas as part of their information architecture. If you're not familiar with XPath, XML, XSD, and XSLT, you may find designing InfoPath forms difficult. On the other hand, filling out an InfoPath form is very easy.

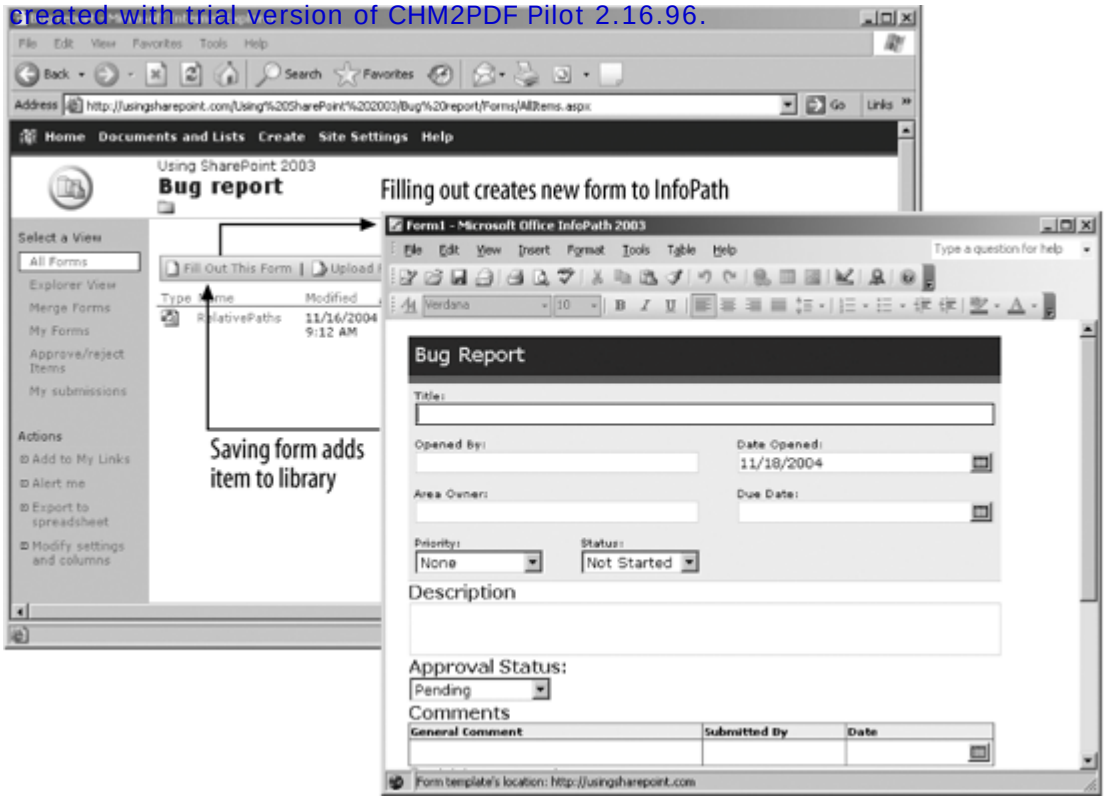


The following sections assume you have InfoPath 2003 SP1 installed on your computer. Members must also have InfoPath installed in order to fill out forms.

7.2.1. Understanding Form Libraries

A form library is basically a document library that uses an InfoPath form template (*template.xsn*) as a template rather than a Word, Excel, or PowerPoint file. To add a new item to the library, members choose Fill Out Form and SharePoint opens the template in InfoPath, as shown in Figure 7-17.

animal 7-17. Creating a new item in a form library



SharePoint comes with several sample templates that you can try. To create a test form library using one of the sample templates:

1. From a SharePoint site, choose Create → Form Library. SharePoint displays the New Form Library page.

2. Enter a name for the library and select one of the built-in samples from the Form template list. SharePoint creates a new library using the template.

SharePoint includes the built-in templates listed in Table 7-4.

Table 7-4. Built-in InfoPath form library templates

Form template	Use to
Absence Request	Request time away from work and to calculate remaining absence balances.
Applicant Rating	Rate job applicants, comment on applicants' strengths and weaknesses, and to provide a hiring recommendation.
Asset Tracking	Keep a record of your company's equipment and property. You can track information such as primary user, location, and purchase date for each asset. The form can also be sorted, allowing you to quickly find the information you're looking for.
Change Order	Explain and specify changes to existing orders or projects, as well as track total cost and time adjustments.
Expense Report (Domestic)	Create and submit an itemized list of expenses.
Expense Report (International)	Same as above, international travel.
Invoice (Multiple Tax Rates)	Document sales and transactions, and to bill customers for services rendered or equipment delivered.
Invoice (Single Tax Rate)	Same as above, one tax rate.
Invoice Request	Request the generation of an invoice for services rendered or equipment delivered.
Issue Tracking	Track an important issue, as well as provide details such as the due date, status, and

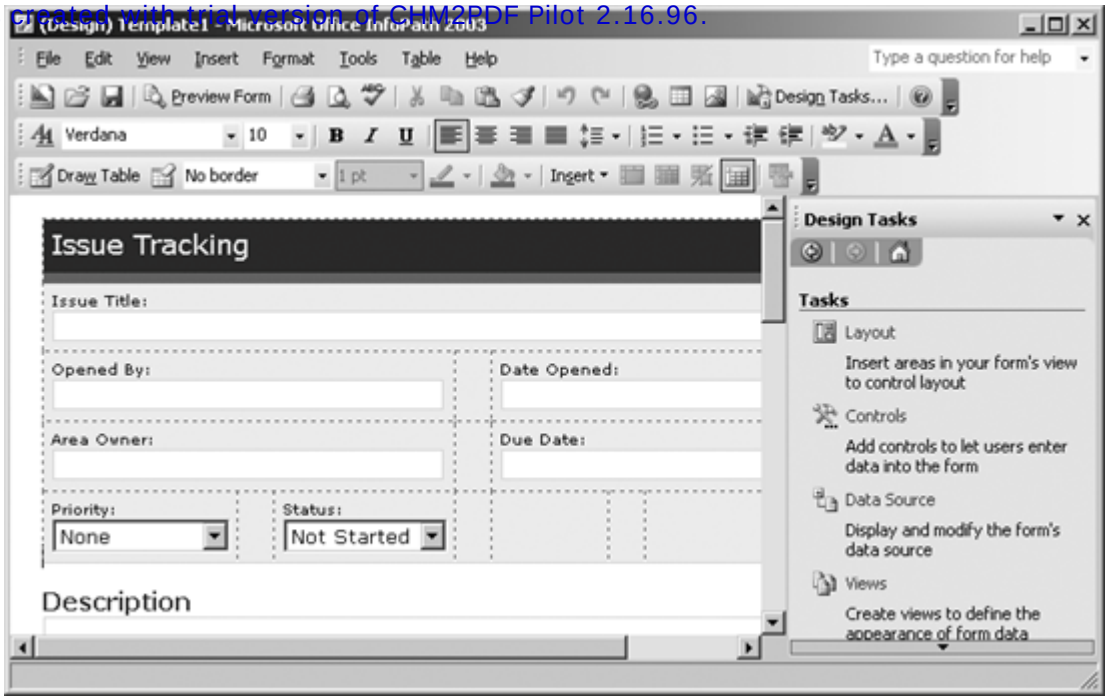
(Detailed)	owner for all action items that impact that issue.
Issue Tracking (Simple)	Provide and track the details of an important issue, including the progress, contributors, and date closed.
Meeting Agenda	Detail the agenda items, time allotments, guest speakers, attendees, and required materials for a meeting. This form can also be used to record meeting minutes, decisions, and action items.
Project Plan	Provide the details for a project, such as schedule, work items, materials, and budget.
Purchase Order	List the total items, total amount due, and required delivery dates for an order, and to authorize the delivery of the specified items.
Purchase Request	Create a request for items you want to purchase.
Sales Report	Record and track monthly sales of various items in different categories.
Status Report	Provide an update on the status of various projects. These reports can then be combined into a single report.
Service Request	Make requests for services or repairs.
Time Card (Detailed)	Track time worked, including times in/out per day, absences, and other payroll information.
Time Card (Simple)	Report the total hours worked during a specified time period.
Travel Itinerary	Create a schedule of events while traveling, which can include transportation arrangements, appointments, and various contacts.
Travel Request	Request the approval of a business trip and to provide details such as travel dates, destinations, itinerary information, and preferences so that accommodations can be made.
Vendor Information	Specify the products and services provided by a particular vendor, and to allow multiple individuals to rate the vendor according to cost, quality, and delivery.

When you save an InfoPath form to a form library, SharePoint saves the form data as an XML file in the library. SharePoint also maps elements from that XML file to columns in the SharePoint list using the *Properties.xfp* file found in the library's Forms folder. This process is similar to the way SharePoint promotes custom properties in Word and Excel document libraries. The advantage of this mapping is that you can use SharePoint to easily create different summary views of InfoPath data. These mapped fields are also included in searches of the site.

7.2.2. Designing a Form

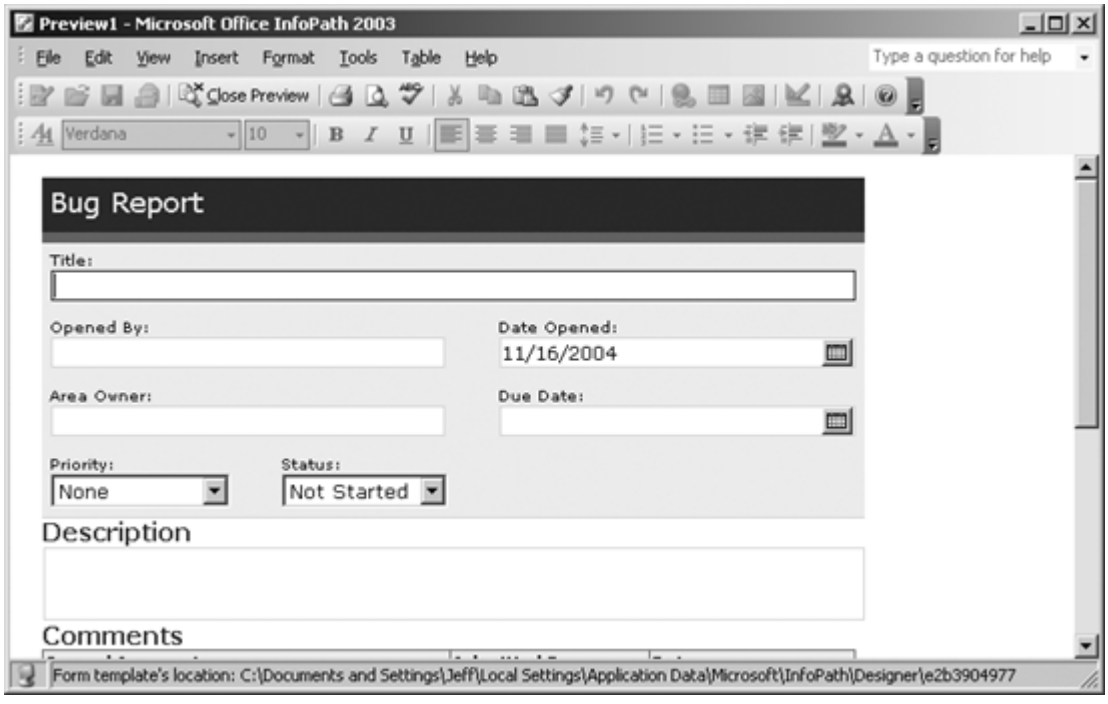
InfoPath is both a form designer and a form viewer. To design a data-entry form in InfoPath, start InfoPath, select one of the sample forms, and choose Design this form. InfoPath opens the form in design mode, as shown in Figure 7-18. You can modify the form by selecting any of the options in the task pane.

animal 7-18. Modifying the Issue Tracking form using Design mode



To preview the results of changes choose File ⟶ Preview Form ⟶ Default. InfoPath displays the form in Preview mode, as shown in Figure 7-19.

animal 7-19. Viewing changes in Preview mode



In preview mode, you can view changes and test controls but cannot save data entered on the form.

7.2.3. Creating a Form Library

Once you are satisfied with a form, you have a choice: you can save the form for local use or testing, or you can publish the form to create a new form library in SharePoint. To publish a completed form:

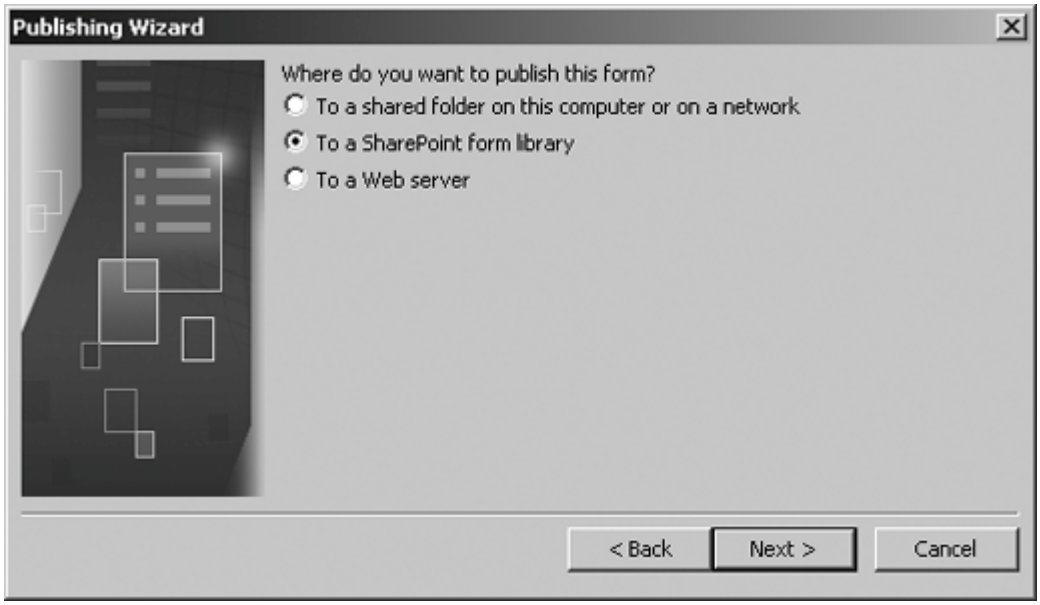
1. Choose File ⟶ Publish. InfoPath starts the Publish wizard to walk you through the process.

2. Click Next to choose the type of location to publish the form to, as shown in Figure 7-20.

3. If you chose to publish to a form libary, the Wizard asks if you want to create a new form library or update the form in an existing library. Select Create new form library and choose Next.

4. Enter the SharePoint site in which to create the form library and choose Next. InfoPath logs on to the site and you may be prompted for your user name and password.

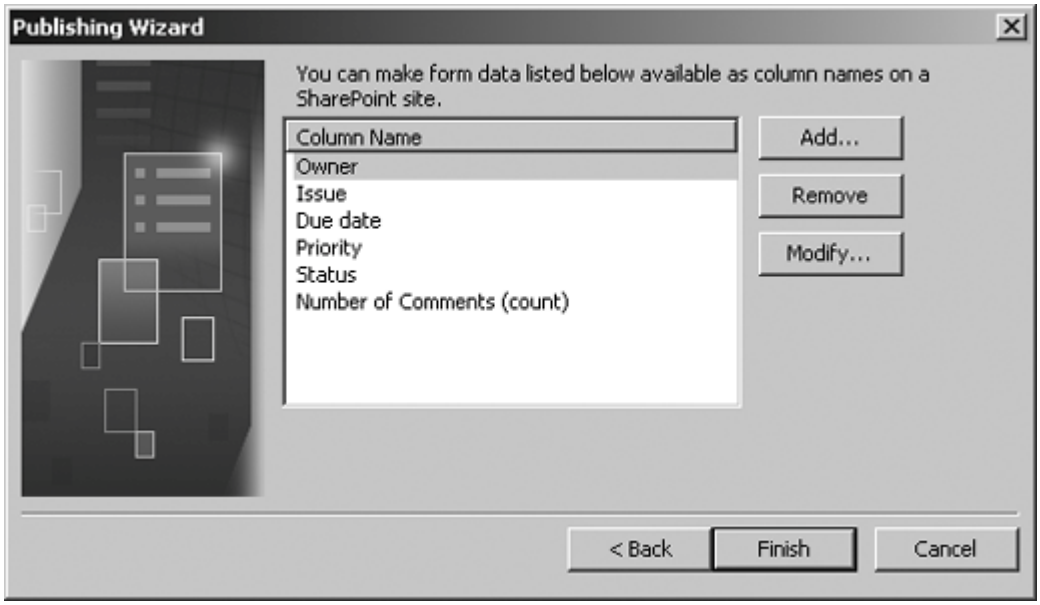
animal 7-20. Using the Publishing Wizard to create a form library



5. Enter a name and description for the form library and choose Next. The Wizard displays the mapping step (Figure 7-21). This step lets you map elements from the form's XML schema to columns in the form library. This mapping is maintained in the *Properties.xfp* file in the list's Forms folder.

6. Choose Add to create, Remove to delete, or Modify to change a mapping. Choose Finish to create the form library. InfoPath displays a success dialog box.

animal 7-21. Using this step to change the mapping between form data and form library columns



Once you've published the form, members can open it for data entry by choosing Fill Out This Form from the form library's toolbar. When members save the form InfoPath creates a new item in the form library as shown previously in Figure 7-17.

7.2.4. Emailing Form Data

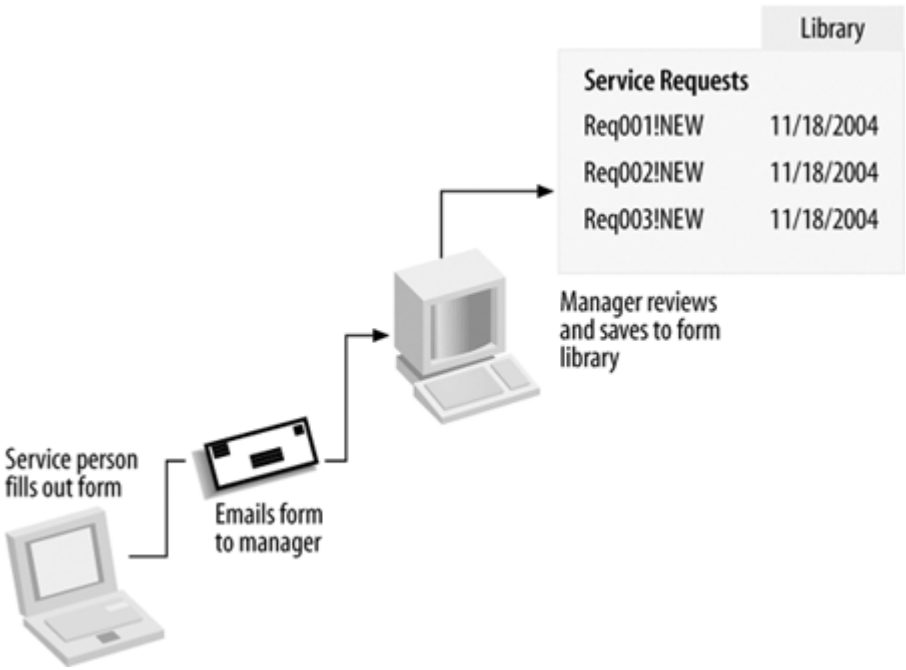
Form libraries are usually set up to collect data this way:

1. Choose Fill Out This Form from the library toolbar.
2. Complete, save, and close the form to create a new item in the library.
3. Open the item from the library to view/edit/modify the form as needed.

InfoPath also provides a Submit action for use in place of Save. Submit sends the form's data in XML form to a database or to an email recipient. Since SharePoint is really a database frontend, it's a bit redundant to submit to database from a form library.

However, submit via email does make sense. Consider this scenario: a support person completes a visit to a customer, fills out a service request, and emails it to the service manager. The manager views the request in Outlook; then saves it to a form library on SharePoint. Figure 7-22 illustrates this scenario.

animal 7-22. Submitting forms via email

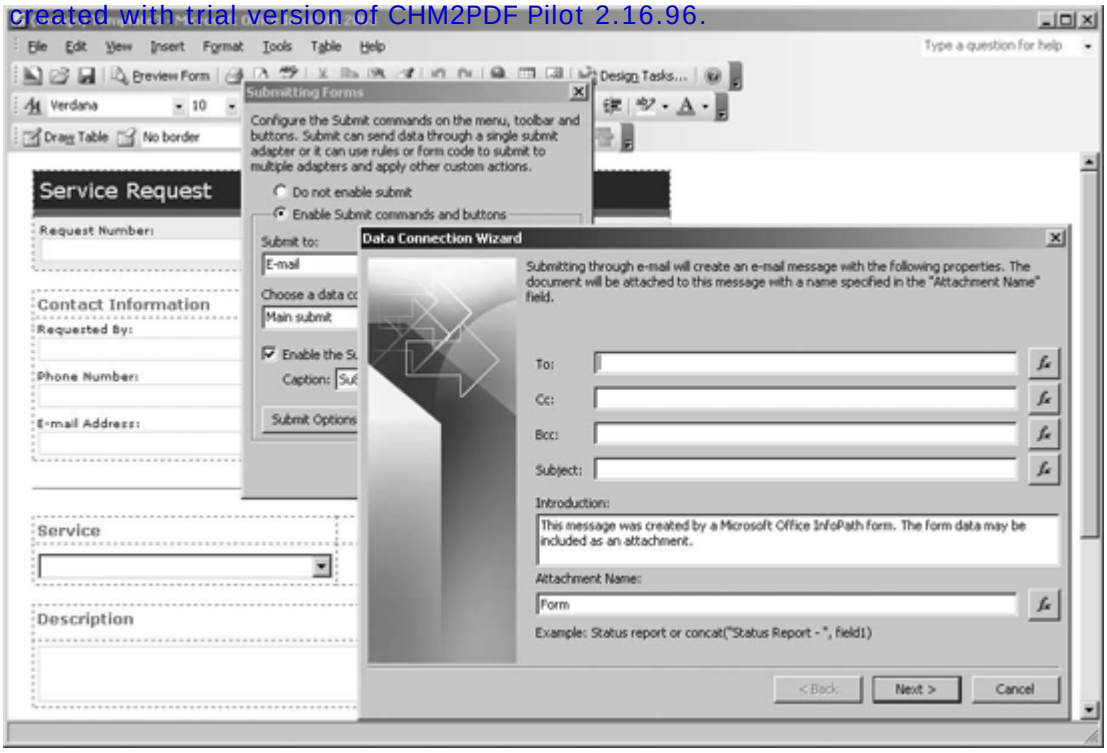


In this scenario, the InfoPath form template (*.xsn*) is installed on the service person's laptop or tablet as well as in the form library. That makes it possible to complete the form offline. The sample Service Request form doesn't come with submit enabled.

To add this feature to the form template:

1. Start InfoPath, select the Service Request sample form and choose Design this Form. InfoPath opens the form in design mode.
2. Choose Tools → Submitting Forms. InfoPath displays the Submitting Forms dialog.
3. Select Enable submit commands and buttons and choose Add. InfoPath starts the Data Connection Wizard shown in Figure 7-23.
4. Enter the email address, subject, and message body you want to include when sending the form, choose Next, and then choose Finish to complete the task.
5. Choose Submit Options, select Close the form, and click OK twice to close the dialogs.

animal 7-23. Enabling submit via email



Once you've enabled submit, you can save and publish the template to a form library as described in the earlier section "Creating a Form Library." You'll also want to copy the template to the service person's laptop.

Enabling submit adds a Submit item to the File menu in InfoPath. Once the form is complete, the service person chooses File ⟶ Submit to send the form. You can also add a command button to submit the form. To do that:

1. Open the template in design mode.
2. Choose Insert ⟶ More controls. InfoPath displays the controls task pane.
3. Drag the button control from the task pane onto the template.
4. Double-click the button. InfoPath displays the button's properties.
5. Select Action ⟶ Submit. InfoPath displays the submit properties.
6. Choose OK twice to close the dialogs.

7.2.5. Customizing Forms

To prevent members from opening a service request from the form library and accidentally submitting the request again, you might want to disable the Submit button whenever the form is opened from SharePoint.

You control the appearance of items on InfoPath forms using conditional formatting, but getting conditional formatting to do what you want can be tricky. Conditionally disabling the Submit button involves these major tasks:

1. Add a checkbox to enable/disable other controls.
2. Set conditional formatting on those controls based on the checkbox setting.
3. Write code that sets the checkbox value.
4. Test the form.
5. Hide the checkbox once the form is working correctly.

The following sections demonstrate those tasks, using the Service Request example.

7.2.5.1 Adding controls

To add a new checkbox control on the form:

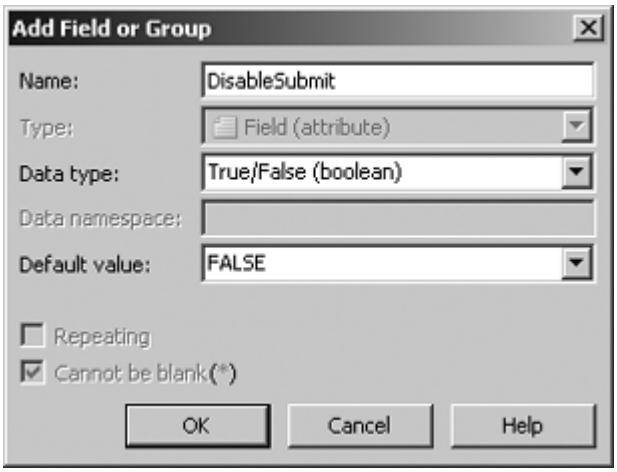
1. Open the template in design mode.
2. Choose Tools ⟶ Submit Options, clear the Enable the Submit menu item on the File menu, and choose OK. You can't enable/disable menu items from a form so you must remove the menu item if you want to control access to Submit.
3. Display the Data Source task pane and choose Add. InfoPath displays the Add Field or Group dialog.
4. Add a Boolean field named DisableSubmit, as shown in Figure 7-24. Choose OK; InfoPath adds an element to the form's XML schema.
5. Drag `my:DisableSubmit` from the task pane onto the form. InfoPath creates a checkbox control on the form.

7.2.5.2 Setting conditional formatting

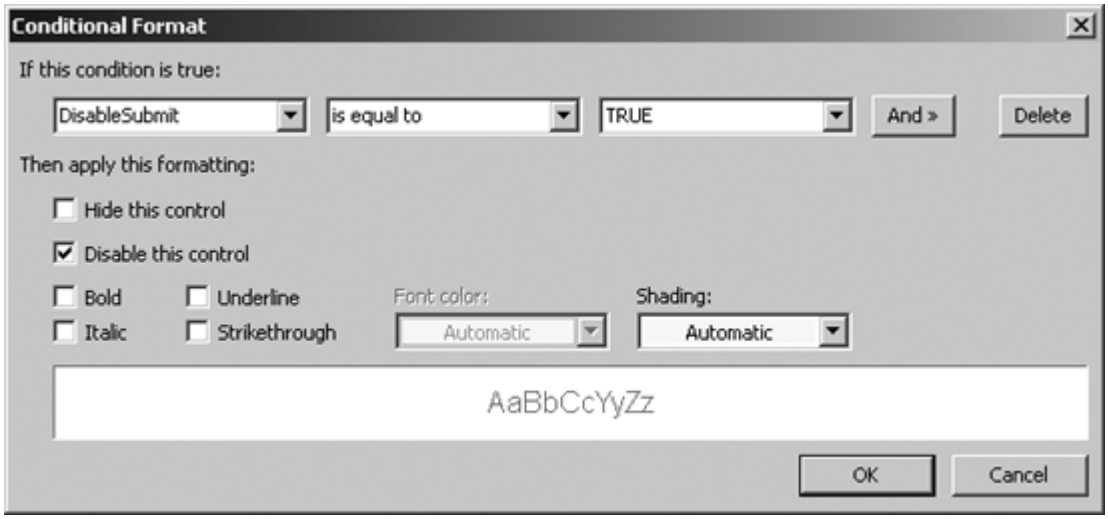
To enable/disable the Submit button based on the checkbox setting:

1. Double-click the Submit button and choose Display ⟶ Conditional Formatting ⟶ Add. InfoPath displays the Conditional Format dialog.
2. Add the condition shown in Figure 7-25 to disable the control when the checkbox value is `TRue`. Choose OK three times to close the dialogs.

animal 7-24. Adding a checkbox to the form



animal 7-25. Disabling the Submit button when the checkbox is selected



7.2.5.3 Writing code

To write code to control the checkbox value:

1. Choose Tools ⟶ Programming ⟶ On Load Event. InfoPath displays the script editor.

2. Add the following code to the `OnLoad` event:

```
// JScript code
var re = new RegExp("http:", "i");
// If opened from the web, disable Submit button.
if (re.test(XDocument.Solution.URI))
    setNodeValue(XDocument.DOM.selectSingleNode
        ("/svc:serviceRequest/my:Disablesubmit"),'true');
else
    setNodeValue(XDocument.DOM.selectSingleNode
        ("/svc:serviceRequest/my:DisableSubmit"),'false');
```

The sample forms use JScript as their programming language. It is *very* difficult to change the programming language after a form is created, but new forms can use JScript or VBScript. Choose Tools ⟶ Options ⟶ Design if you want to change the default language.

7.2.5.4 Testing a form

To test the form:

1. Choose File ⟶ Preview Form ⟶ Default to check your code. Compiler errors will display an error message and prevent the preview. Runtime errors will display a script debug error but allow the preview.

2. If the Preview is successful, select/clear the checkbox to make sure it enables/disables the Submit button.

3. Close the preview and choose File ⟶ Save to save the form to the SharePoint library you created previously in "Creating a Form Library."

4. Navigate to the form library and choose Fill Out This Form. Verify that the Submit button is disabled when opened from the Web.

If you are used to working in a programming language like C# or VB.NET, you'll probably find the script debugging tools frustrating. InfoPath uses JScript or VBScript as its built-in language, but you can also use C# or VB.NET using Visual Studio Tools for Office (VSTO).

7.2.5.5 Hiding controls

You can't directly hide interactive controls in InfoPath. Instead, create a section to contain hidden controls and then hide that section:

1. From the InfoPath Controls task pane, drag a section control onto the form.

2. Drag the Disable Submit checkbox onto that section.

3. Double-click the section and choose Display ⟶ Conditional Formatting ⟶ Add.

4. Create the condition `The expression 1 = 1` and select Hide this control. Choose OK three times to close the dialogs.

7.2.6. Populate a Control from a List

InfoPath forms can retrieve data from various types of data sources, including SharePoint lists. InfoPath's

access to lists is read-only; it can't directly modify items in a list. InfoPath refers to this type of access as a *secondary data source*. In InfoPath, *primary data source* refers to where the form data is stored.

Secondary data sources can populate controls, such as drop-down lists, and they can be used to create reports (or *views*) of lists from SharePoint. The major steps to populating controls from a SharePoint list are:

1. Create a data connection to the list.
2. Add controls and bind their values to list fields.
3. Add filters to display the appropriate data.

The following sections demonstrate these steps using a list on my site that contains InfoPath coding samples categorized by language.

7.2.6.1 Creating a data connection

To create a data connection from an InfoPath form to a SharePoint list:

1. Design a new, blank form and choose Tools ⟶ Data Connections ⟶ Add. InfoPath starts the Data Connection Wizard.
2. Enter the settings in Table 7-5 and choose Close when done.

Table 7-5. Data Connection Wizard settings

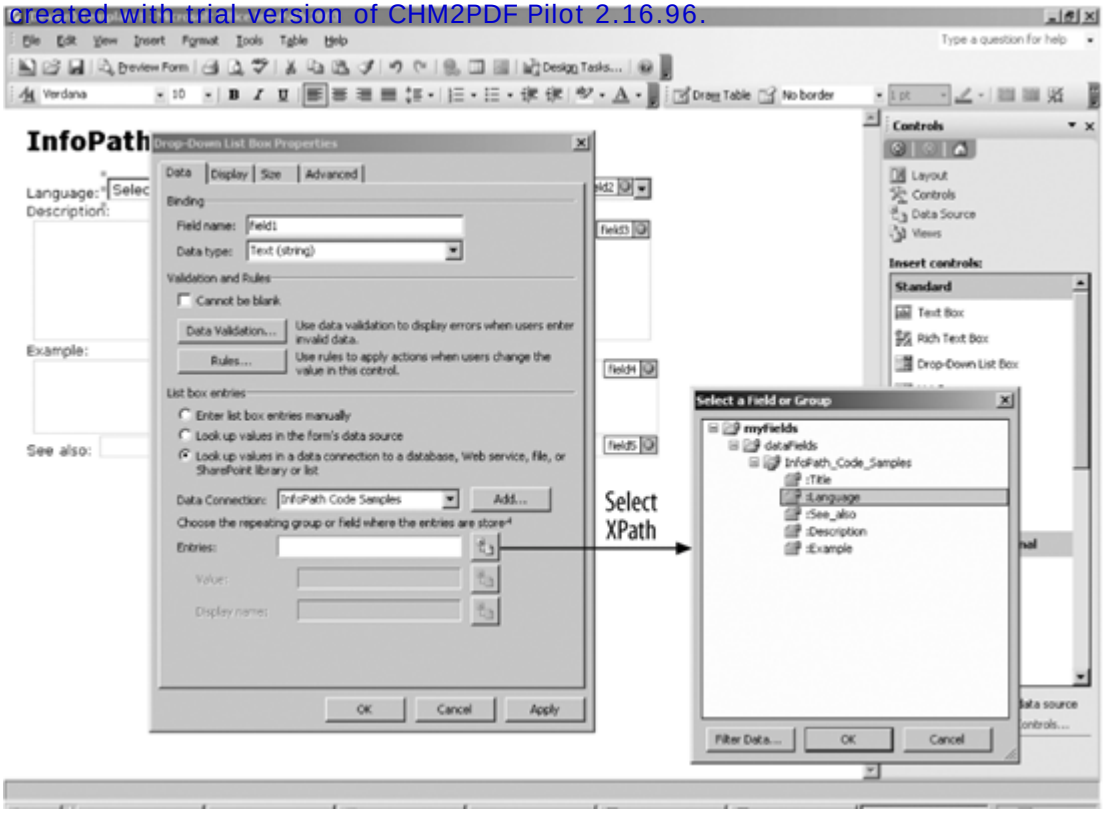
At Wizard step	Do this	And choose...
1	Select Receive data.	Next
2	Select SharePoint library list.	Next
3	Enter http://usingsharepoint.com/.	Next
4	Select the InfoPath Code Samples list.	Next
5	Select ID, Title, Language, Example, and Description fields.	Next
6	Select Automatically retrieve data when form is opened.	Finish

7.2.6.2 Adding bound controls

To add controls that display items from the data connection:

1. Display the Controls task pane and create two drop-down lists by dragging and dropping the controls from the task pane onto the form, as shown in Figure 7-26.
2. Double-click on Field; select Enter values manually; add the values VBScript, VB.NET, C#, Registry, and Other. Choose OK to close the dialog.
3. Double click on `field2`, select Look up values in a data connection, choose Select XPath, select the InfoPath_Code_Samples group, and choose Filter Data.
4. Specify the filter Language ⟶ is equal to ⟶ Select a field or group and select the `field1` element from the `Main` data source. Choose OK four times to return to the main dialog.
5. Choose Set XPath for the Value field and choose ID.
6. Choose Set XPath for the Display name field and choose Title. Choose OK to close the last dialog.

animal 7-26. Creating bound controls



Step 4 filters the values in field2 so that only the titles for the selected language are displayed. By binding field to InfoPath_Code_Samples you can display Title, but use the unique ID to filter other fields in the next step.

7.2.6.3 Filtering data

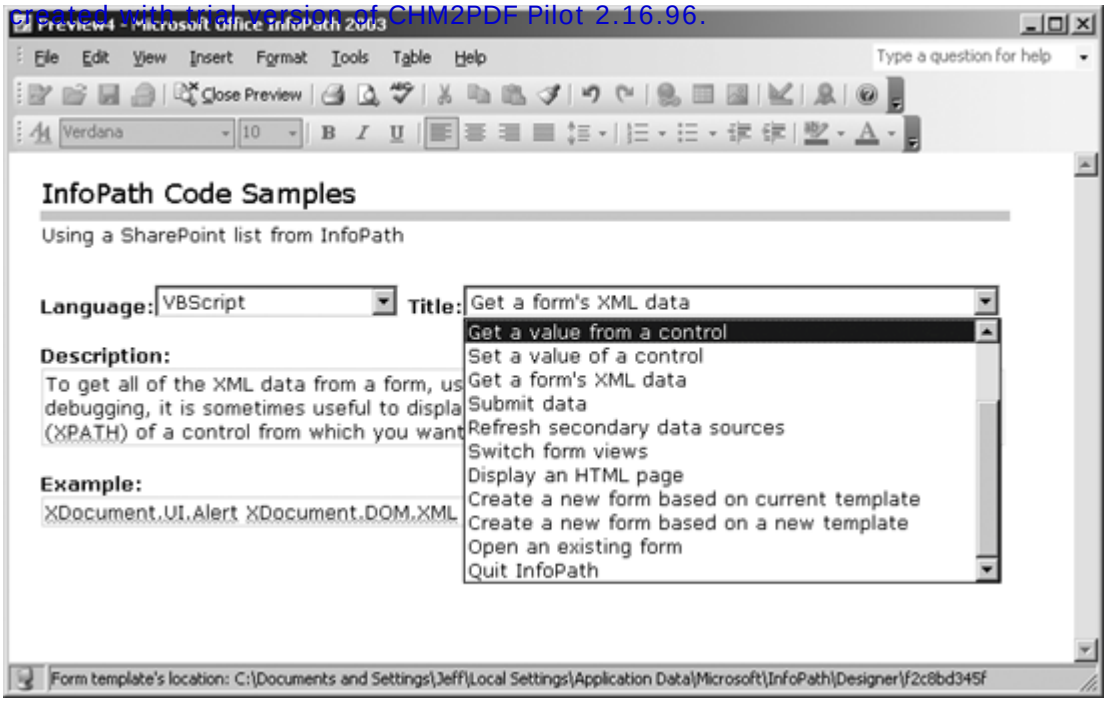
InfoPath list controls are built to receive arrays of values from data sources. Text box controls are a little different, however. In order to display a value from a list of repeating items, you must use a formula for the field's default value.

To see how that works:

1. Add a text box control to the form.
2. Double-click on the text box, choose Display, select Wrap text, and select Scrolling: Expand to show all text.
3. Choose Data, click the formula button after the Value field, choose Insert Field or Group, and select the Description field from the InfoPath_Code_Samples group.
4. Choose Filter Data → Add and add the filter ID is equal to field2.
5. Choose OK five times to close each of the open dialogs.
6. Repeat the preceding steps for the Example field from the InfoPath_Code_Samples group.

When complete, preview the form. It should look something like Figure 7-27.

animal 7-27. Previewing the form to test bound fields and filters



You may have noticed that InfoPath uses very long series of dialog boxes to complete tasks. I've tried to keep it simple, but describing tasks in InfoPath is difficult. You may have better luck learning from the completed samples.

7.2.7. Validating Data

One advantage of using Submit rather than Save is that Submit enforces data validation. InfoPath allows you to save forms containing invalid data, but you can't Submit the form until all fields are valid. InfoPath can validate fields using a variety of criteria. To add validation criteria to a control:

1. Display the control's properties and click Data Validation. InfoPath displays the Data Validation dialog.

2. Click Add. InfoPath displays the dialog box for validation criteria. Enter a condition that results in a validation error.

3. Click OK and then preview the form and test the validation.

You can combine multiple criteria by choosing And >>; however, you can only have one ScreenTip and one Message per control.

Validation rules are checked as the user enters data on the form. If an invalid entry is made in a control, that entry is flagged as soon as the focus moves to another control.

Forms containing validation errors can't be submitted, but members only receive a warning if they save or email a form containing validation errors. There is no built-in way to prevent the user from saving, emailing, or printing a form containing validation errors, but you can prevent it by following these steps:

1. Choose Tools → Form Options → Open and Save. Disable the Save, Print, and Send to Mail Recipient options. Choose OK to close the dialog.

2. Add buttons to the form to save, print, or submit the form.

3. Add a checkbox to enable/disable the save/print/submit buttons as described in the previous section, "Customizing Forms."

4. Write code to control the setting of the checkbox based on validation.

Only the last step above is new, so I'll explain it more. InfoPath creates an `Error` object for each validation error on a form. Within script, you can use the `XDocument` object's `Errors` collection to check whether a form is valid. For example, the following script checks whether the form is valid each time the focus changes from one control to another:

```
    ' Do validation checking.
    Sub XDocument_OnContextChange(eventObj)
        If eventObj.Type = "ContextNode" Then
            ' Get check box control
            Set x = XDocument.DOM.DocumentElement.SelectSingleNode("my:chkValid")
            If xdocument.errors.count = 0 Then
                ' Set value to True (valid).
                ' This enables Submit button through
                ' conditional formatting.
                x.text = "true"
            Else
                ' Not valid, set to False (disables Submit).
                x.text = "false"
            End If
            Exit Sub
        End If
    End Sub
```

This code sets the checkbox to `true` if the form contains no validation errors.

7.2.8. Preventing Changes to Form Templates

InfoPath both designs and displays forms. Only members of the Web Designer or Administrative groups can create form libraries or change the InfoPath form template used by the library. However, templates deployed to clients can be changed. In most cases, you won't want members opening those templates in Design mode and tinkering with them. There are two approaches to solving this problem:

- Enable form protection. This approach discourages members from changing form templates, but does not prevent them from doing so.
- Disable design mode on members systems. This approach keeps users from changing existing form templates and prevents them from creating new ones.

To protect a template from changes:

1. Open the template in Design mode.
2. Choose Tools → Form Options, select Enable protection, and choose OK.

Protected templates display the warning if the user opens them in design mode. That's weak protection at best, but if you sign the protected template with a digital signature, you can both discourage changes and detect changes if they are made (changes overwrite the digital signature, so the template will no longer be trusted).

A stronger solution prevents users from designing *any* templates. You can use the Custom Installation Wizard (CIW) to create a customized setup program for Office that omits the design features from InfoPath. In order to do that:

1. Run the CIW.
2. Open the InfoPath Windows Installer file (*INF11.msi*).
3. Configure Disable InfoPath Designer mode in step 10 of the Wizard.
4. Finish creating the custom installation and then use the generated Windows Installer to install InfoPath on the client machines.

Alternately, you can disable design mode by changing a system registry setting as follows:

```
    HKEY_CURRENT_USER\Software\Microsoft\Office\11.0\InfoPath\Designer\DisableDesigner=
0x00000001
```

Editing the system registry requires special knowledge. This is an advanced technique. Disabling design mode removes InfoPath's Design this Form button, Design a Form task pane, and the File, Design a Form menu option.

7.2.9. Generating HTML for Forms

If a member does not have InfoPath and they open a form data file, that file appears as XML in their browser. The InfoPath SDK provides a down-level tool (XDown) to convert a form file's XML to read-only HTML so others can view InfoPath output. To use XDown:

1. Download and install the InfoPath SDK. By default, the SDK installs XDown in the folder *C:\Program Files\Microsoft Office 2003 Developer Resources\Microsoft Office InfoPath 2003 SDK\Tools\XDown*.

2. Copy the files in the XDown folder to the project folder where you are working, or make them otherwise available for use from the command line.

3. Create a subfolder within the project folder to receive the output of XDown.

4. Run XDown from the command line on your InfoPath template (*.xsn*). XDown decompiles the template and converts the views from the template into XSL files that can be used to view form files in HTML.

5. Copy the generated XSL file to the form library's Forms folder.

6. Add processing instructions to form data files.

The generated XSL includes scripts, so you can't use it with the XML web part to perform the transformation within the SharePoint site.

For example, the following command line converts the views from *template.xsn* into XSL that can be used to display asset form data; output files are written to the Down subfolder:

```
XDown /d template.xsn Down
```

You can then copy the XSL files written to the Down folder to the form library's Forms folder and edit the form data files to include the following processing instruction:

```
<?xml-stylesheet type="text/xsl" href="Forms\view_1.xsl"?>
```

To include that instruction in all new forms created within a form library:

1. Create a new, blank form in the form library named *template.xml*.

2. Choose Explorer view, open *template.xml* in Notepad, and add the processing instruction.

3. Save the file to the Forms folder of the form library.

4. Choose Modify settings and columns → Change general settings, and change the Template URL: setting to `template.xml`.

After these changes, SharePoint will create new forms based on the form file rather than the InfoPath template file (*template.xsn*). Because *template.xml* includes an `xml-stylesheet` instruction, all subsequent form files also include the instruction. You can also use this procedure to prepopulate a form with default data entries.

XDown generates one XSL file for each view in a template. You can use any of those XSL files to display the form data, but if you want to switch between views you will have to create multiple versions of the form files, each with a different `xml-stylesheet` instruction.

If you edit the form with InfoPath after making this change, InfoPath preserves the `xml-stylesheet` instruction. In fact, members with InfoPath never see the HTML view of the form since the `mso-application` instruction causes the XML file to open in InfoPath if it is available. Remove the `mso-application` instruction if you want to

test the transformation on a machine that has InfoPath installed.

Forms displayed using the XDown-generated XSL files omit buttons and other user-interactive controls. They convert other controls, such as text boxes, into read-only HTML.

7.2.10. Programming in .NET

If you are creating templates that require more than a little code, you should plan to program InfoPath in .NET. Visual Studio .NET is a professional programming tool that gives you far more assistance than the script editor. Although the Visual Studio .NET environment is complicated, the command completion, syntax checking, and debugging tools make it a great deal easier to program with InfoPath.

Before you can program InfoPath in .NET, you must install the following tools:

- .NET Framework 1.1 must be installed *before* installing InfoPath
- Visual Studio 2003
- InfoPath 2003 Toolkit for Visual Studio .NET, available for free from http://www.microsoft.com/downloads

7.2.10.1 Creating a project

To create an InfoPath project with Visual Studio .NET:

1. Choose File → New Project and select InfoPath Form Template from the Microsoft Office InfoPath Projects group.
2. Visual Studio .NET starts the Project Wizard. Select whether to create a new template or use an existing one and click Finish.
3. Visual Studio starts InfoPath and creates a new project template in Visual Studio .NET.

It's a good idea to create a new template in InfoPath first, then specify that template in step 2. Creating the new template in InfoPath lets you use the data connection wizard or base a new template on an existing onefeatures that aren't available from Visual Studio .NET.

InfoPath and Visual Studio .NET are loosely coupled. You edit the template by switching to InfoPath. There, you can design the template's appearance. To write code, click Edit Form Code from control properties.

7.2.10.2 Responding to events

Visual Studio .NET creates code templates for any event you create in InfoPath, as shown here (bold code is mine):

```
<InfoPathEventHandler(MatchPath:="cmdOK", EventType:=InfoPathEventType.OnClick)> _
Public Sub cmdOK_OnClick(ByVal e As DocActionEvent)
    ' Write your code here.

    thisXDocument.UI.Alert("This template is " & _
      thisXDocument.Solution.URI)
 End Sub
```

To run this code, switch back to InfoPath and click Preview. Visual Studio .NET builds the underlying assembly automatically and then previews the form.

There are other entry points for creating events from InfoPath. For example, in InfoPath you can select Tools → Programming → On Load Event... to create a form load event procedure in Visual Studio .NET. Or you can add events from the validation dialog box by selecting an event and clicking Edit. Basically, all of the tasks that formerly started the Microsoft Script Editor (MSE) generate equivalent code in Visual Studio .NET.

There are a couple things to note that are illustrated by this simple project:

- You can use InfoPath Preview, as well as Visual Studio .NET's Debug → Start (F5) to run the project.
- The project is copied to a temporary location, not run from the project folder. This can lead to some complications while debugging.
- The InfoPath template is stored as component files, not as a compiled template (*.xsn*). Those component files are listed in the Solution Explorer. If you close InfoPath, you can directly edit those files as XML within Visual Studio .NET.
- If you place a breakpoint within cmdOK_OnClick, you can step through the code using the Visual Studio .NET debugging commands (unlike MSE).
- Visual Studio creates two object variables: `thisApplication` and *thisXDocument*. You can use these in place of the intrinsic Application and XDocument objects used in scripts.

7.2.10.3 Converting scripts to .NET

It is fairly simple to convert InfoPath scripts written in VBScript to Visual Basic .NET. Mainly, remember to add `this` before references to the XDocument and Application objects and remove `Set` from object assignments (.NET doesn't use `Set`). For example, the following changes convert some InfoPath VBScript to Visual Basic .NET:

```
    Dim x As DOMNode
    Set x = thisXDocument.DOM.DocumentElement.SelectSingleNode("my:field1")
x.text = "New value"
```

In the preceding and following code, bold indicates additions and `strikethrough` indicates deletions.

7.2.10.4 Debugging and deploying

The .NET code running in an InfoPath project inherits its security settings from the InfoPath template. To get some insight into this situation, try to run this code in your new InfoPath project:

```
    thisXDocument.Save(  )
```

InfoPath displays a security exception when you try to execute `Save` in preview mode. `Save` requires full trust, and preview runs in domain-level trust by default. Methods that require full trust are flagged in the InfoPath object model reference help as *level 3*. If you use any level-3 methods, you need to sign or install the form template to ensure it runs in full trust.

If you don't use level-3 methods or .NET methods that access the local resources, you don't need to ensure that your code runs in full trust. Domain-level trust is fine. In practice, however, that is pretty limiting.

You face two problems trying to assign full trust to InfoPath templates run from within Visual Studio .NET:

- You can't sign the template since Visual Studio .NET stores the template as component files and InfoPath can only sign compiled templates (*.xsn*).
- You can't install the template since you are still creating it and so would need to repeatedly uninstall/reinstall the template every time you made a change.

To solve this problem, follow these steps:

1. Close InfoPath and edit the template's manifest file (*manifest.xsf*) to remove the **`publishURL`** and

trustSetting attributes and require full trust as shown below.

```
<xsf:xDocumentClass
  solutionVersion="1.0.0.5"
  solutionFormatVersion="1.100.0.0"
  publishUrl="C:\IPWork\FTNet\ manifest.xsf"
  name="urn:schemas-microsoft-com:office:infopath:FTNet2:-myXSD-2004-04-23T16-52-59"
  productVersion="11.0.6250"
  trustSetting="automatic"
  requireFullTrust="yes"
```

2. Build the InfoPath project in Visual Studio .NET.

3. Run the following script from within the project folder to register the template's manifest file.

```
set ip = CreateObject("InfoPath.ExternalApplication")
Set fso = CreateObject("Scripting.FileSystemObject")
pth = fso.GetFolder(".").path
ip.registersolution pth & "\manifest.xsf"
set ip = nothing
set fso = nothing
```

Now, if you run the InfoPath project, the form previews running in full trust and the Save method (along with any other level-3 methods) will work. This procedure is for debugging only. Once you are ready to deploy the InfoPath project, follow these steps:

- Run the following script from within the project folder to unregister the template on your development machine:

```
set ip = CreateObject("InfoPath.ExternalApplication")
Set fso = CreateObject("Scripting.FileSystemObject")
pth = fso.GetFolder(".").path
ip.unregistersolution pth & "\manifest.xsf"
set ip = nothing
set fso = nothing
```

- From within Visual Studio .NET, choose Project → Publish Form. Visual Studio .NET compiles the component files into a template (*.xsn*), builds the assembly (*.dll*), for the project, and starts the InfoPath Publish Wizard.

- Both the template and assembly are placed in the destination folder you specified in the Publish Wizard. You can then open that template (*.xsn*) in InfoPath to sign it or use *RegForm.exe* to create an installer.

- Run the signed template or install the unsigned template to test that the form runs in full trust.

7.2.11. Resources

To get	Look here
Help on using InfoPath	*C:\Program Files\Microsoft Office\OFFICE11\1033\InfMain.chm*
Information about programming InfoPath and understanding form template components	*C:\Program Files\Microsoft Office\OFFICE11\1033\InfRef.chm*
The InfoPath SDK	Search http://www.microsoft.com/downloads for "InfoPath SDK."
Instructions on using XDown	http://msdn.microsoft.com/library/en-us/ipsdk/html/ipsdkUsingTheDownLevelTool_HV01081784.asp
Information about .NET programming with InfoPath	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odc_ip2003_tr/html/odc_INF_Lab_15.asp

Programming InfoPath	*C:\Program Files\Microsoft Office\OFFICE11\1033\InfRef.chm*
Advanced programming techniques	http://blogs.msdn.com/infopath
Support from the InfoPath programming community	Visit the newsgroup *microsoft.public.infopath*.
Information on installing templates using RegForm	http://msdn.microsoft.com/library/en-us/ipsdk/html/ipsdkUsingTheFormRegistrationTool_HV01073334.asp
Help creating fully trusted templates	http://msdn.microsoft.com/library/en-us/ipsdk/html/ipsdkUnderstandingFullyTrustedForms_HV01073332.asp

PREV NEXT

Chapter 8. Creating Web Parts

Web parts are based on ASP.NET web controls, but are otherwise fairly new. For that reason, the development tools for web parts are scattered across a number of different downloads. That makes developing web parts more confusing than it really should be. I try to address this problem by breaking my discussion of web parts into two chapters:

- This chapter covers setting up your development environment to make writing, debugging, and deploying web parts as easy as possible.
- Chapter 9 covers writing code to perform useful tasks in those web parts.

You can develop web parts in Visual Basic .NET or Visual C#. Both chapters show sample code in Visual Basic .NET because that is currently the most popular language. Equivalent Visual C# samples are available at http://usingsharepoint.com/samples/Ch08CS.aspx and *Ch09CS.aspx*.

8.1. Preparing to Develop

Developing a web part requires access to the SharePoint DLL, which runs only on Windows 2003. Therefore, you must be running Windows 2003 to do development. There are a couple ways to do this on your development machine:

- Upgrade to Windows 2003.
- Use Remote Desktop to access a server running Windows 2003 and do your development there.
- Use Microsoft Virtual PC to run Windows 2003 within Windows XP on your desktop.

The main advantages and disadvantages of each approach are summarized in Table 8-1.

Table 8-1. Possible web part development approaches

Approach	Advantage	Disadvantage
Upgrade	Best performance for development.	Changing your desktop OS is disruptive.
Remote Desktop	Server can be shared with other developers/testers.	Requires an additional physical server.
Virtual PC	Contained on single desktop, you can create/change server configurations without affecting others.	Performance is reduced.

Whichever approach you use, there are a few cautions to keep in mind. Don't develop on a production server. Doing development requires temporarily changing security settings that might open avenues for attack and running untested web parts (or any component) on an in-service server is never a good idea. Software requirements and licensing issues are the same for physical or virtual PCs.

The following sections explain how to use these different approaches and how to configure the server for development.

8.1.1. Using Remote Desktop

To use Remote Desktop to access a development server you must have an account on that server and the server must have remote access enabled. To enable Remote Desktop access to the server:

1. From the server, choose Control Panel → System → Remote.
2. Select Allow users to connect remotely to this computer and choose Select Remote Users.
3. Add your user account and the accounts of other developers that will have access. Choose OK twice to close the dialogs.

Then, to run Remote Desktop from Windows XP Professional:

1. From the Start menu, choose All Programs → Accessories → Communications → Remote Desktop Connection.
2. Select the server name and choose Connect. Remote Desktop connects to the server and displays the standard desktop sign-on dialog.

8.1.2. Using Virtual PC

To use Virtual PC to create a Windows 2003 development environment for web parts:

1. Install Virtual PC on your Windows XP Professional desktop.

2. In Virtual PC, choose File → New Virtual Machine Wizard. The wizard walks you through the creation process. For web part development you will want to allocate at least 512 MB of RAM for the virtual machine.

3. Start the new virtual machine and install Windows 2003. This takes quite a while, but it can run in the background while you perform other tasks.

4. Configure Windows 2003 as an Application Server. Select Enable ASP.NET, but don't select FrontPage 2002 Server Extensions since they aren't compatible with SharePoint.

5. Install Windows SharePoint Services from http://www.microsoft.com/downloads. The download file name is *STSV2.EXE*. For development purposes it is usually best to select the Typical Installation (WMSDE) option rather than Server Farm (SQL Server).

The Virtual PC window captures your keyboard and mouse movements while it has focus. Virtual PC uses the right-hand Alt key as *host key* to release control and switch back to the host operating system. Table 8-2 lists some useful key combinations for Virtual PC.

Table 8-2. Virtual PC key combinations

Key	Use to
Host key (right Alt key)	Release virtual machine's control of mouse pointer and keyboard. (Use to move mouse out of Virtual PC window and back to desktop.)
Host key+Delete	Send Alt+Ctrl+Delete to virtual machine.
Host+Enter	Switch between full-screen and windowed display.
Host key+P	Pause/resume virtual machine.
Host key+R	Reset virtual machine.
Host key+F4	Close virtual machine.
Host key+down arrow	Minimize virtual machine.

8.1.3. Configuring the Server for Development

Regardless of how you run Windows 2003, you must install the development tools within that Windows 2003 server and change some settings before you can develop web parts.

To install the development tools and configure the server:

1. Install Visual Studio .NET 2003. You can create web parts using Visual Basic .NET or Visual C#.

2. Install the Web Part Templates from http://www.microsoft.com/downloads. The download file name is *WebPartTemplatesforVSNetSample2.exe*.

3. Optionally, install FrontPage 2003. FrontPage makes it easier to create and modify test web part pages in SharePoint. You *cannot* edit SharePoint sites from Visual Studio 2003.

4. In Windows Explorer, create a root folder, such as *C:\WebParts*, for your web part projects. Creating this folder makes it easier to use required command-line tools.

5. Copy *vsvars32.bat* from *C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\Tools* to *C:\WebParts*. You'll use *vsvars32.bat* to access the command-line tools.

6. Add a *\bin* folder to the *C:\InetPub\wwwroot* folder. That is the location where you will write web part DLLs during development.

7. Open the *Web.config* file from *C:\InetPub\wwwroot* and change the trust level and debug settings as shown in this snippet:

```
<configuration>
    ...
 <system.web>
    ...
   <compilation batch="false" debug="true" />
  <trust level="WSS_Medium" originUrl="" />
 </system.web>
</configuration>
```

The web part templates are written to work with Visual Studio .NET 2003. They do not work with earlier or beta versions of Visual Studio.

While you have *Web.config* open, take a look at the `SafeControls` settings as well. You'll need to add a `SafeControl` element for each new web part project you create later on.

8.2. Creating a Web Part Project

Before you can start writing code for a web part, you need to set a few things up so that you can debug and test the component from within SharePoint:

1. Create a test page in SharePoint.
2. Create and configure a new project.
3. Build and sign the assembly; then extract the token.
4. Register the web part *Web.config*.
5. Add the web part to the test page to verify that everything works.

The following sections describe these tasks in more detail.

8.2.1. Creating a Test Page

Since web part projects are class libraries, they require a context within which to run. It is a good idea to create a document library of web part pages on your development server for debugging and/or testing purposes. Using a document library helps organize the test pages and makes it easier to create/modify them.

To create a test web part page library:

1. From SharePoint on the development server, choose Create → Document Library. SharePoint displays the New Document Library page.
2. Enter the name TestPages, select the Web Part Page document template, and choose Create. SharePoint creates a new document library.
3. Choose New Document to create a new, empty web part page named `Test1`.
4. Write down the address of the new page. You'll use that address when configuring the project.

8.2.2. Configuring a New Project

To create and configure a new web part project in Visual Studio:

1. From Visual Studio, choose File → New Project, select the Web Part Library template from the Visual Basic .NET or Visual C# project type, and choose OK. Visual Studio creates a new web part project.
2. Choose Project → Properties → Configuration Properties → Debugging.
3. Select Start URL and enter the address of the test page you created in the preceding section. You can use the localhost domain, for example http://localhost/Test Pages/Test1.aspx.
4. Select Enable ASP.NET debugging.
5. Choose Configuration Properties → Build.
6. Enter the Output path *C:\InetPub\wwwroot\bin* and choose OK. (You created the *\bin* folder in the section "Configuring the Server for Development" earlier in the chapter.)
7. Choose File → Save All to save the project settings.

8.2.3. Building and Signing the Assembly

Since you haven't written any code yet, this step might seem out of place. However, SharePoint requires that assemblies be signed with strong names before you can run, test, and debug them. You need to set up signing using the .NET Strong Name utility (*sn.exe*); build the assembly once; then extract the strong name token before you can register the web part with SharePoint. To sign and build the assembly:

1. From the server Start menu, choose All Programs → Accessories → Command Prompt. Windows opens command prompt window.

2. Enter `cd \WebParts` to change to the WebParts folder you created in the earlier section "Configuring the Server for Development" and run *vsvars32.bat*.

3. Enter `sn -k "key.snk"` to create a strong name key to use for signing the assembly.

4. From Visual Studio, open *AssemblyInfo.vb* or *AssemblyInfo.cs* and add an `AssemblyKeyFile` attribute referencing the key file. In Visual Basic it would look like this:

```
<Assembly: AssemblyKeyFile("C:\WebParts\key.snk")>
```

5. Choose Build → Build Solution. Visual Studio builds and signs the web part assembly.

6. Return to the command prompt and run *sn.exe* to extract the token from the signed assembly. Remember that you changed the build path, so the command line would be something like:

```
sn -T c:\InetPub\wwwroot\bin\WebPartLibrary1.dll
```

7. Record the public key token returned by *sn.exe* for use registering the web part next.

8.2.4. Registering the Web Part

You must register web part assemblies in the server's *Web.config* file to tell SharePoint that the assembly is safe to load. To register the web part:

1. From the server's *\InetPub\wwwroot* folder, open *Web.config*.

2. Add a `SafeControl` element to the `SafeControls` section. The values for each attribute come from the locations in Table 8-3.

3. Save *Web.config*.

The following element shows a typical `SafeControl` element. Each project will have its own unique settings. Table 8-3 describes the attribute settings and where you find them.

```
<SafeControls>
 <SafeControl Assembly="WebPartLibrary1,
  Version=1.0.0.0, Culture=neutral,
  PublicKeyToken=b03f5f7f11d50a3a"
  Namespace="WebPartLibrary1"
  TypeName="*" Safe="True" />
  ...
</SafeControls>
```

Table 8-3. SafeControl attributes

Attribute	Used to	Settings come from / notes
`Assembly`	Describe the web part assembly.	The assembly file name, minus the .DLL extension.
`Version`	Identify the version.	`Assembly: AssemblyVersion` attribute. Projects must have a static version number.
		`Assembly: AssemblyCulture` attribute or `neutral` if none

Culture	Identify the culture.	specified.
PublicKeyToken	Uniquely identify the assembly.	Output from sn.exe -T.
Namespace	Provide the root namespace for the web parts.	Project root namespace (top-level name in Solution Explorer).
TypeName	Include one or all types from the project.	The class name of the web part or * for all classes in the project.
Safe	Flag the web part as safe or unsafe.	Set to False to disable a web part.

8.2.5. Adding the Web Part to a Page

If you've completed the preceding tasks correctly, you should now be able to add the web part to a page, set a breakpoint, and step through the code in debug mode to verify that everything works. It's a good idea to do this check before adding code to the project, since debugging code and configuration problems at the same time can be very hard. To add the web part to a test page:

1. Load the Web Part project in Visual Studio and set a breakpoint on the `output.Write` line in the RenderWebPart procedure.
2. Choose Debug → Start or press F5. Visual Studio builds the assembly and displays the *Test1.aspx* page in the browser.
3. From the Test1 page in SharePoint, choose Modify Shared Page → Add Web Parts → Import. SharePoint displays the Import task pane.
4. Choose Browse, select the *WebPart1.dwp* file from the web part project folder, and choose Upload. SharePoint adds the web part to the task pane.
5. Drag the web part from the task pane to a web part zone as shown in Figure 8-1. Control switches to Visual Studio at the breakpoint.
6. Clear the breakpoint in Visual Studio and press F5 to continue. Control returns to the browser and SharePoint display the web part on the page.

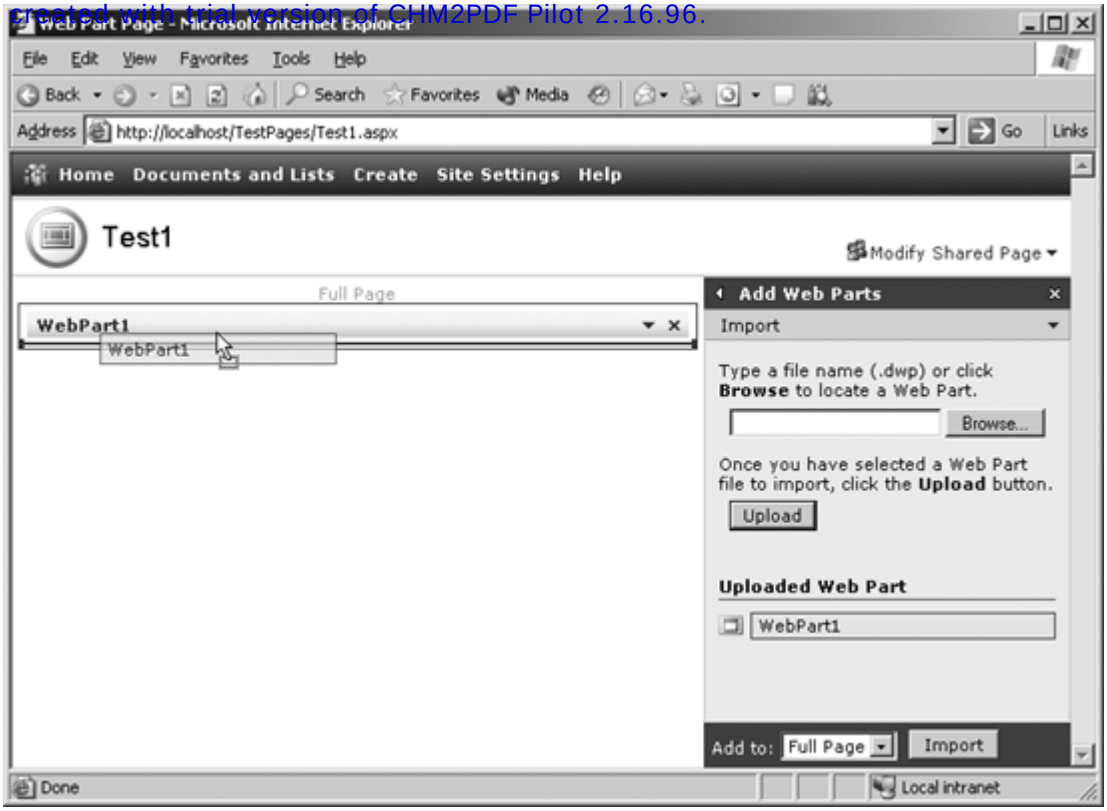
SharePoint saves the changes to *Test1.aspx*, so the web part is included on the page the next time you run the project. To see how that works, stop debugging and change the `_defaultText` constant as shown here:

```
Private Const _defaultText As String = "New test string"
```

Press F5 to run the project and the new string appears in the control (Figure 8-2).

You don't have to run the project to see the change. Just rebuild it and refresh the page in the browser.

animal 8-1. Adding a web part to a test page



animal 8-2. Changing properties in the assembly changes the web part appearance

8.2.6. Troubleshooting

If there is a problem with your web part, you'll usually see an error after dragging the part onto the web part page. Check the `SafeControl` settings in *Web.config*. These settings need to match the AssemblyInfo and public key token exactly, specifically:

- Errors loading the web part (Figure 8-3) are generally caused by a problem with the signature on the assembly. Verify that the `PublicKeyToken` attribute matches the value returned by running `sn.exe -T` on the assembly.

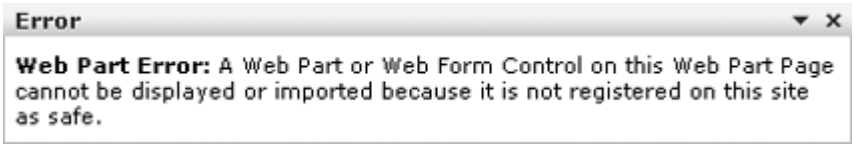
- Errors finding the web part (Figure 8-4) are generally caused by typos in the `Namespace` or `TypeName` attributes. Check that those match the entries in the *AssemblyInfo* file.

8.3. Deploying Web Parts

Importing a web part to a page installs the part in the web part page gallerythe web parts in that gallery are only available on that one page. That's usually what you want during development, since the web part isn't ready for general use. Once you

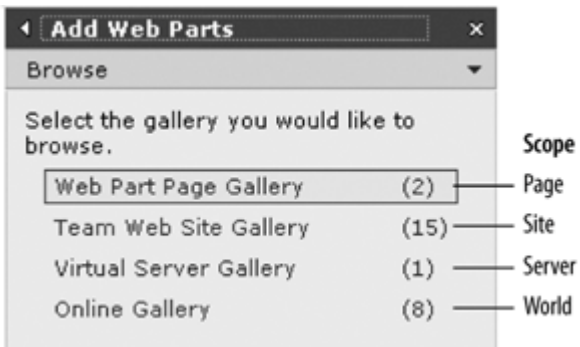
animal 8-3. Error loading the web part

animal 8-4. Error identifying the web part



move on to general testing and final deployment, you'll want to install the web part in one of the other galleries. Figure 8-5 shows the available web part galleries and explains the scope of each. The most common way to deploy web parts is to install them in the virtual server gallery. That makes the web part library available to all sites on the virtual server.

animal 8-5. Installing web parts in different galleries controls their scope



8.3.1. Deploying to the Virtual Server Gallery

To install a web part in the virtual server gallery, package the assembly, manifest, and description files as a CAB file and use the SharePoint Administration utility (*stsadm.exe*) to install the package. To make it easier to use *stsadm.exe*, add the following lines to the *vsvars32.bat* file you copied to the \WebParts folder earlier:

```
@set STSBinDir=C:\Program Files\Common Files\Microsoft Shared\web server extensions\
60\bin
@set PATH=%STSBinDir%;%PATH%;
```

Follow these steps to create and install the package from Visual Studio:

1. Open the web part project in Visual Studio.
2. Choose File → Add Project → New Project → Setup and Deployment Projects → Cab Project. Name the setup project `WpSetup` and choose OK.
3. Select the setup project in the Solution Explorer and choose Project → Add → Project Output. Visual Studio displays Figure 8-6.
4. Select Primary Output and Content Files from the web part project and choose OK. Visual Studio adds those items to the setup project.
5. Choose Build → Rebuild Solution. Visual Studio rebuilds the web part assembly and packages the assembly and content files in the CAB file.
6. Copy the resulting CAB file to the *\WebParts* folder.
7. Run *stsadm.exe* to install the CAB file on the server. For example, the following command line installs the web part library in the virtual server gallery:

```
stsadm -o addwppack -filename WpSetup.cab
```

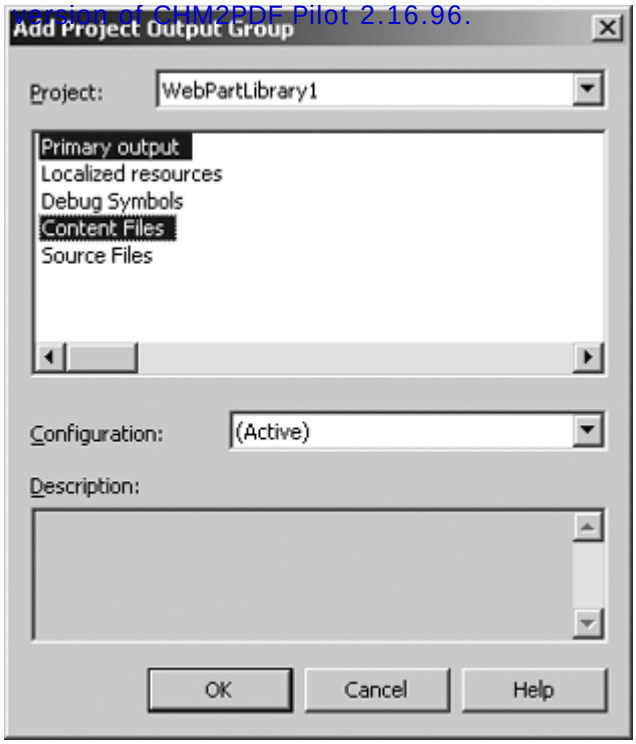
8.3.2. Adding to the Site Gallery

Once the web part library is deployed to the virtual server, you can add those parts to the Team Web Site gallery from within SharePoint. The site gallery allows you to categorize web parts so they can be filtered from the task pane, as shown in Figure 8-7.

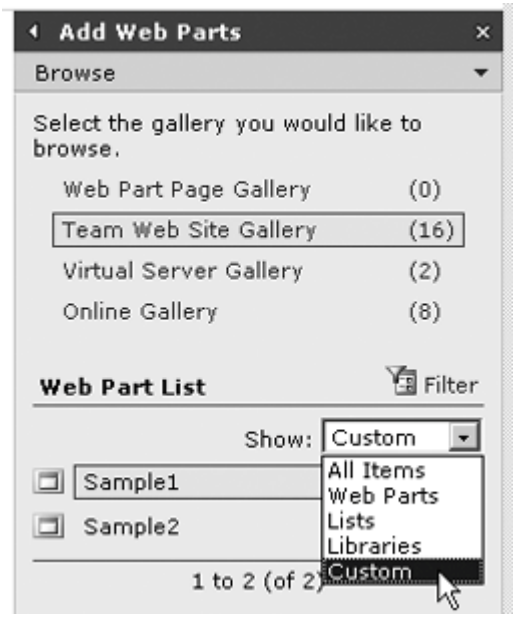
To add a web part to the Team Web Site gallery:

1. Choose Site Settings → Go to Site Administration → Manage Web Part Gallery. SharePoint displays the Web Part Gallery page which lists all of the web parts installed in the site gallery.
2. Choose New Web Part. SharePoint displays a list of all the web parts installed on the virtual server (Figure 8-8).
3. Select the web parts to add to the site gallery and choose Populate Gallery. SharePoint adds the web parts to the site gallery.

animal 8-6. Adding output and content to the CAB file



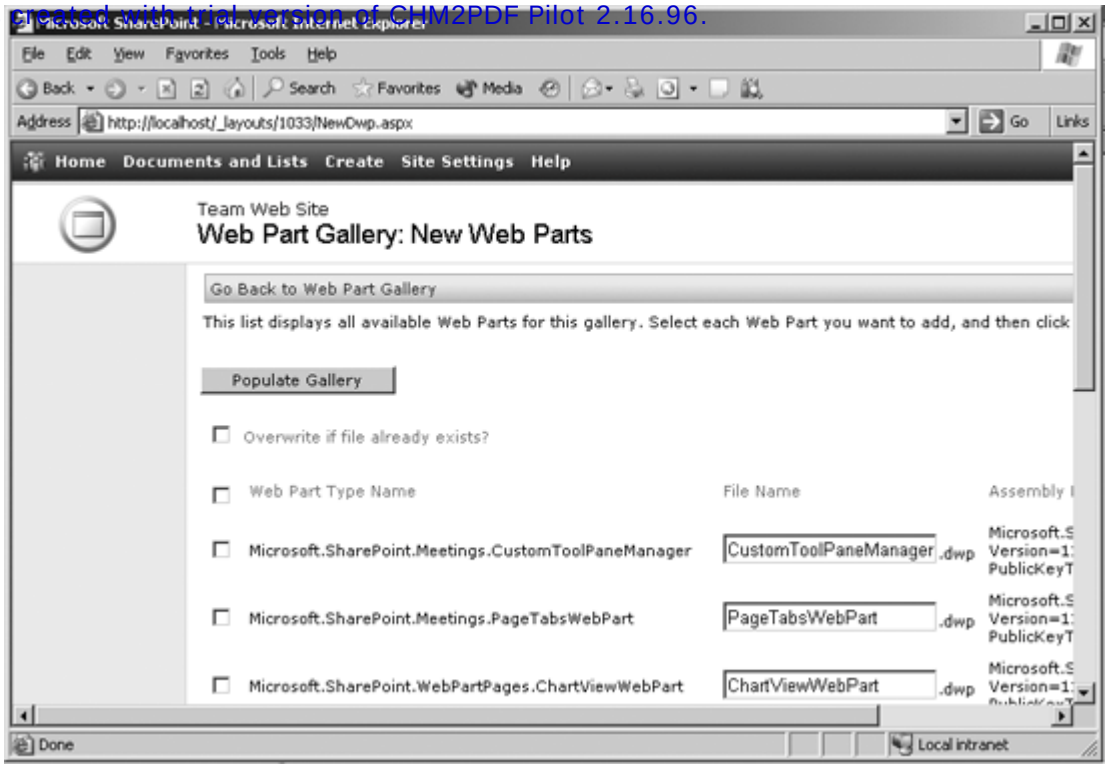
animal 8-7. Filtering web parts in the site gallery



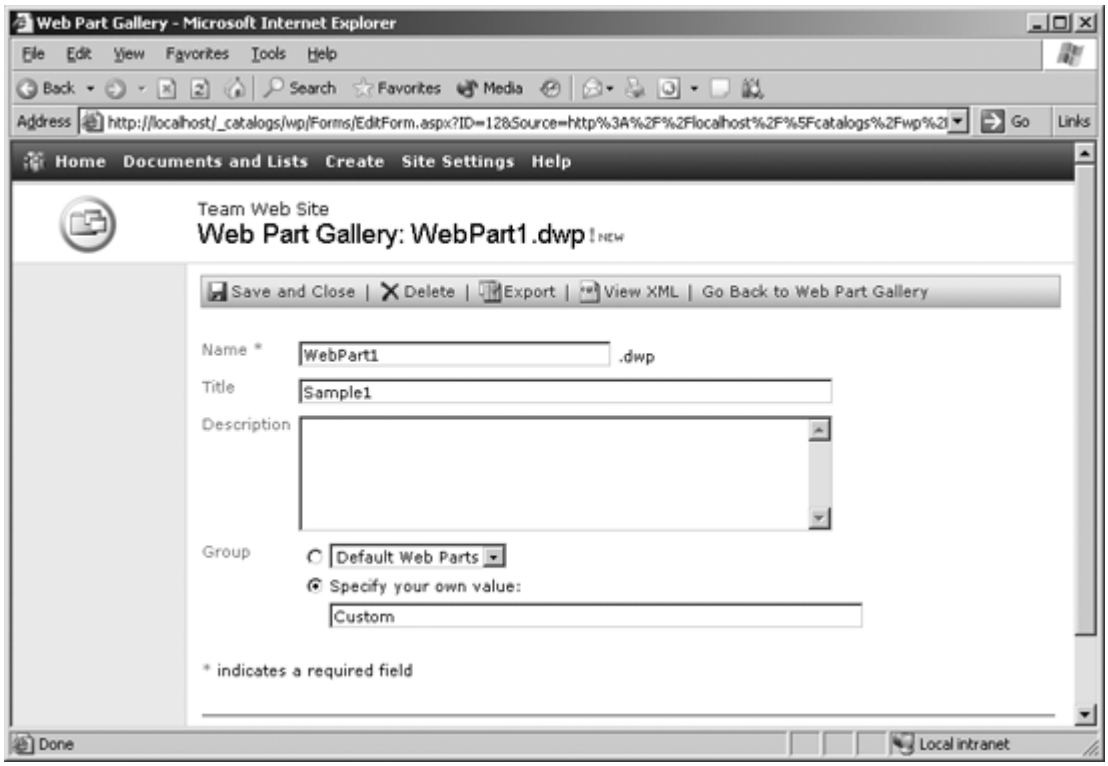
To add a category for the web part:

1. Choose the Edit Document Properties button on the Web Part Gallery page. SharePoint displays Figure 8-9.
2. Select Specify your own value and enter the name of the custom category.
3. Choose Save and Close to make the change and return to the gallery.

animal 8-8. Adding web parts to the site gallery



animal 8-9. Creating a category for a web part



8.3.3. Deploying as Windows Installer Files

Using CAB files is a simple way to deploy web parts. However if you want to include custom permission sets and allow uninstall from the Control Panel, you need to use the Web Part Packager utility (*wppackager.exe*) to create a Windows installer file (*.msi*) for deployment.

To use *wppackager.exe* to create an *.msi* file for a web part library:

1. Download the utility from http://www.microsoft.com/downloads.

2. Install the utility in the WebParts folder.

3. Copy the assembly from *\Inetpub\wwwroot\bin* to the project folder. The location of the assembly must match the `Assembly Filename` attribute in the project's *manifest.xml* file.

4. Copy the included packaging file (*wppackager.xml*) to the web part library project folder and edit it to reflect the web part project.

5. Run *wppackager.exe* from the project folder to create the Windows installer (*.msi*). For example:

```
..\wppackager wppackager.xml
```

6. Sign the *.msi* file with a public key. For example:

```
signcode wpsetup.msi
```

Once you've create the .msi file, you can install the web part library by running that file on the server. The installer walks you through the installation process providing a more friendly user-interface than the *stsadm.exe* utility.

The packaging file tells *wppackager.exe* what to include in the Windows installer file. For example, the following packaging file includes the web part library created earlier:

```
<?xml version="1.0" ?>
<Wppackager xmlns="http://schemas.microsoft.com/WebPart/v1/Wppackager">
  <Manifest FileName="Manifest.xml" />
   <CodeAccessSecurity AssemblyName="WebPartLibrary1" Version="1.0.0.0"
     PublicKeyBlob="a451359320dc76e6" >
      <PermissionSet class="NamedPermissionSet" version="1" >
        <IPermission
          class="System.Web.AspNetHostingPermission, System, Version=1.0.5000.0,
          Culture=neutral, PublicKeyToken=b77a5c561934e089"
          version="1"
          Level="Minimal"
      />
      <IPermission
          class="System.Security.Permissions.SecurityPermission, mscorlib,
          Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
          version="1"
          Flags="Execution"
      />
      <IPermission class="Microsoft.SharePoint.Security.WebPartPermission,
          Microsoft.SharePoint.Security, Version=11.0.0.0, Culture=neutral,
          PublicKeyToken=71e9bce111e9429c"
          version="1"
          Connections="True"
       />
       <IPermission class="Microsoft.SharePoint.Security.SharePointPermission,
          Microsoft.SharePoint.Security, Version=11.0.0.0, Culture=neutral,
          PublicKeyToken=71e9bce111e9429c"
          version="1"
          ObjectModel="True"
       />
       </PermissionSet>
   </CodeAccessSecurity>
  <MSI Name="WPSetup1" Version="1.0.0.0" Manufacturer="SP2003 Sample" />
 </Wppackager>
```

Table 8-4 describes the elements in packaging files.

Table 8-4. Packaging file elements

Element	Description

wppackager	Root element required for all packaging files.
Manifest	Identifies the location and name of the web part library's manifest.
CodeAccessSecurity	The permission set to be applied to the web part library.
PermissionSet	The minimal permissions needed to run a web part plus the SharePoint permission.
MSI	Determines the name of the MSI Installer and the Web Part package that is created.

More information about packaging files and custom permission sets can be found at:

- http://msdn.microsoft.com/library/default.asp?url=/library/enus/odc_sp2003_ta/html/sharepoint_deployingwebparts_msi.asp
- http://msdn.microsoft.com/library/default.asp?url=/library/enus/cpguide/html/cpconcodeaccesspermissions.asp
- http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnspts/html/sharePoint_WSSCodeAccessSecurity.asp

8.3.4. Deploying to the Online Gallery

The online gallery lists web parts deployed through a web service. By default, it is set to the Microsoft Office site through this setting in *Web.config*:

```
<OnlineLibrary Url="http://r.office.microsoft.com/r/hlidAwsGallery" />
```

You can change the setting to point to a web service provided within your organization. There is no information on implementing the online gallery web service at this time, but I will post anything I find at http://usingsharepoint.com/Samples/OnlineGallery.aspx.

8.4. Creating Web Parts from Excel

You can create custom Spreadsheet web parts from Excel 2003 using the Excel Office Web Part Add-In. Excel web parts can include data bindings, XML Maps, formatting, and complex formulas that are difficult to create from the Spreadsheet web part's browser interface.

The add-in doesn't create a new web part assembly, but rather customizes the existing Spreadsheet web part by adding a new web part description (*.dwp*) to the site gallery. Before you begin, download and install the components listed in Table 8-5 from http://www.microsoft.com/downloads.

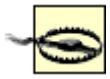
Table 8-5. Downloads for Office web parts

Download	Install on	Used to
Office Web Parts (*ststpkpl.exe*)	Server	Provide Spreadsheet, charting, and pivot table web parts.
Office Web Components (*owc11.exe*)	Clients	Provides interactive features from web parts for Office 2003 users and read-only capability for non-Office 2003 clients.
Excel Office Web Part Add-in (*Spreadsheet Add-In.EXE*)	Development machine	Create custom Spreadsheet web parts from within Excel 2003.

The add-in is currently in beta testing. By the time a release version of the add-in is available, some of these procedures and descriptions may have changed.

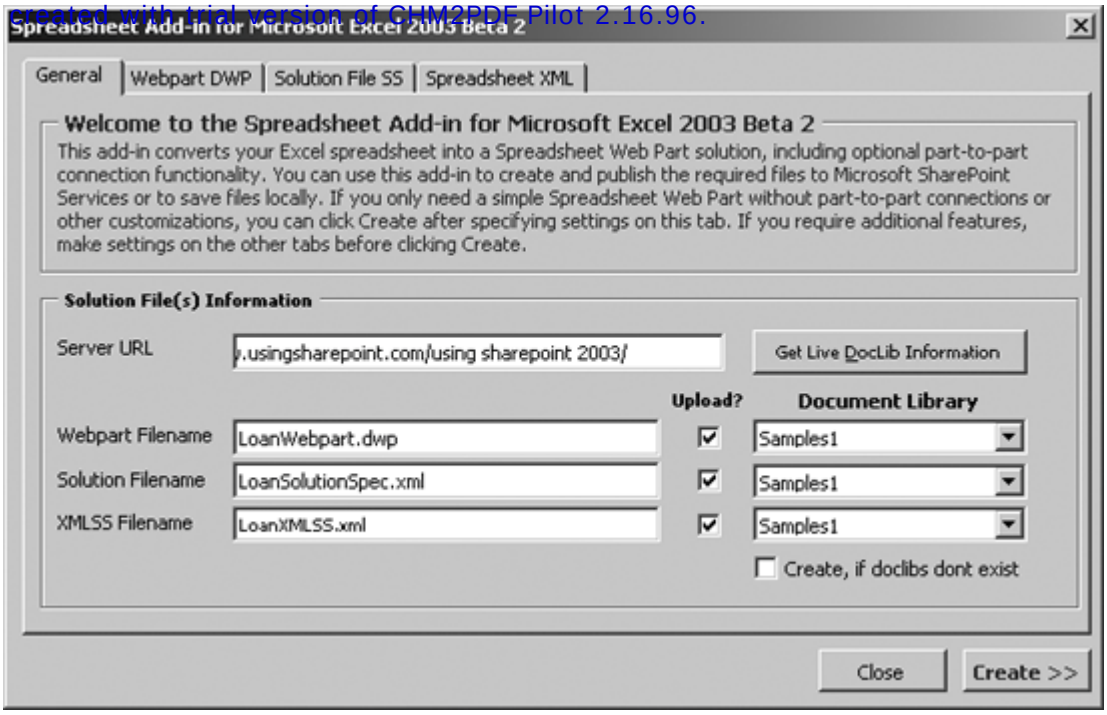
Installing the add-in appends the Create Add-In menu to Excel. To create a web part from an existing spreadsheet:

1. Open the workbook containing the spreadsheet in Excel 2003.
2. Choose Create Add-In. Excel displays the dialog in Figure 8-10.
3. Enter the name of the SharePoint server you wish to deploy the web part to and choose Get Live DocLib Information. The add-in populates the drop-down lists with lists of the libraries to which you can deploy the web part.
4. Select a library to deploy to and clear the "Create, if doclibs don't exist" option.
5. Choose Web Part DWP to add a title and description of the web part.
6. Choose Create to create and deploy the web part. The add-in displays Figure 8-11.

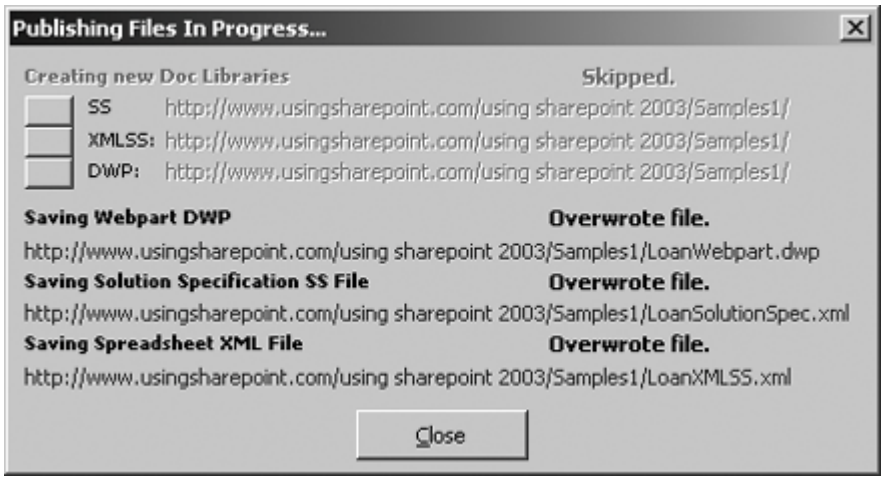


The Document Library name in Figure 8-10 must match the folder name of the library. If you get an error writing files in Step 6, check the actual address of the library. In this case, the Samples library was actually in the *Samples1* folder.

animal 8-10. Creating a spreadsheet web part from Excel



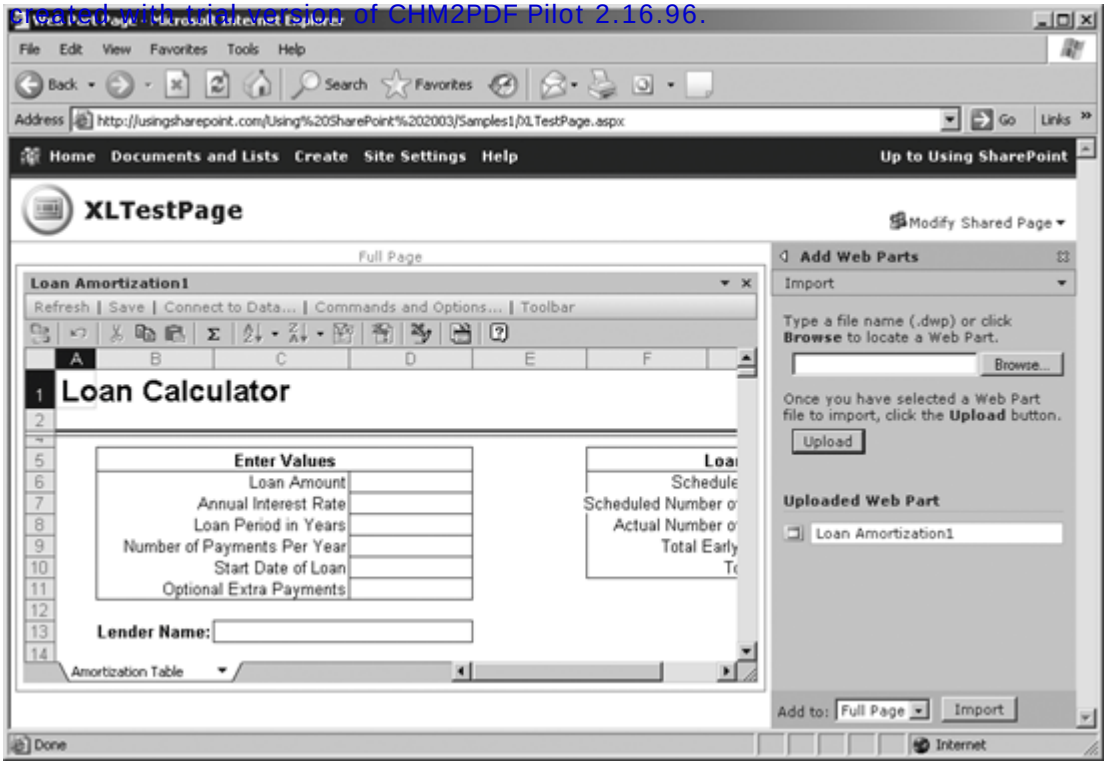
animal 8-11. The add-in displays the publishing status after you choose Create



To use the new web part from a web part page:

1. Navigate to the list where the web part description (*.dwp*) is published and select Explorer View.
2. Copy the *.dwp* file to a folder on your local computer. This step is necessary because you can't upload a *.dwp* file from a document library into SharePoint.
3. Create a new web part page and choose Modify Shared Page → Add Web Parts → Import. SharePoint displays the Web Part task pane.
4. Choose Browse and select the *.dwp* file you downloaded in step 2.
5. Choose Upload. SharePoint uploads the web part to the web part page gallery.
6. Drag the web part onto a web part zone. SharePoint adds the web part to the page shown in Figure 8-12.

animal 8-12. Importing a web part generated by the Excel add-in



Importing the web part makes it available on that one page. To make the web part available to all web part pages on the site, import the web part description into the site gallery:

1. Choose Site Settings ⟶ Go to Site Administration ⟶ Go to Top-level Site Administration ⟶ Manage Web Part Gallery.

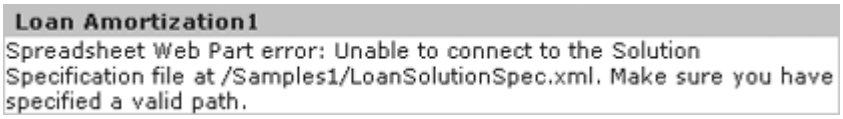
2. Choose Upload Web Part ⟶ Browse, select the web part's *.dwp* file, enter a group name to help categorize the part, and choose OK.

3. Verify that the web part works by choosing the Type icon next to the new web part in the gallery list. SharePoint displays a preview of the web part.

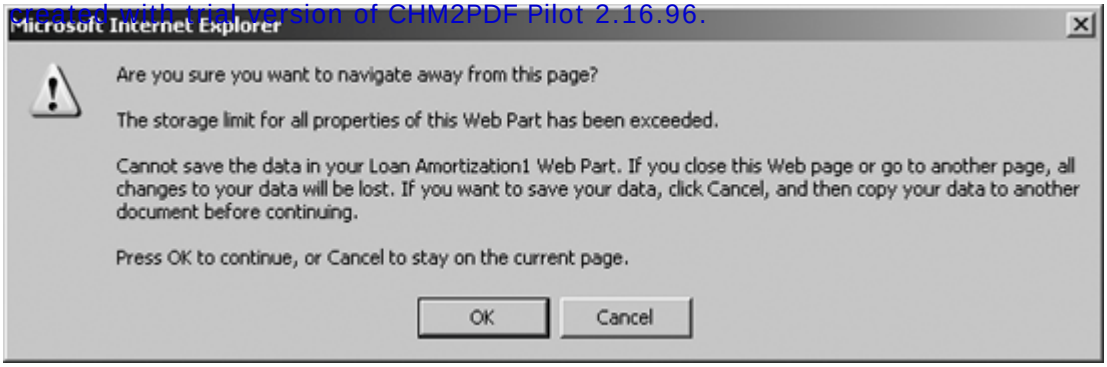
Since the add-in generated web part description references a solution file stored in a library, the new web part will only work within the site that contains the library. If you try to use the web part outside of that site, you'll see an error like the one in Figure 8-13.

You may also bump up against the web part property size limits when working with Spreadsheet web parts. The Loan Calculator sample saves quite a bit of data; if you enter values on the page in Figure 8-12, then go to another page you'll see the warning in Figure 8-14.

animal 8-13. You get an error if you use the web part outside its site



animal 8-14. Warning generated by web part trying to save too much data



To fix that, you must increase the `WebPartLimits PropertySize` attribute in the server's *Web.config* file, as shown here:

```
<WebPartLimits MaxZoneParts="50" PropertySize="2048576" />
```

Of course, increasing this limit requires additional storage on the server and may affect performance. Be sure to test your settings before deploying them to the production server.

8.5. Resources

To get	Look here
Visual Studio Web Part template	Search http://www.microsoft.com/downloads/ for "Web Part Template."
Excel Office web part add-in	Search http://www.microsoft.com/downloads/ for "Spreadsheet Web Part."
Office web components (client)	Search http://www.microsoft.com/downloads/ for "Office Web Components."
Office web parts (server)	Search http://www.microsoft.com/downloads/ for "Office Web Parts."
Spreadsheet web part schema documentation	Search http://www.microsoft.com/downloads/ for "Spreadsheet Web Part."
Office web component object model (VBA)	http://msdn.microsoft.com/library/en-us/owcvba11/html/ocwelcome_hv01136208.asp

13.14 <cstdarg>

The <cstdarg> header is the C++ version of the C standard <stdarg.h> header, which declares macros for accessing the arguments to a function that takes a variable number of arguments, that is, a function that is declared with an ellipsis as the last parameter.

A function that takes a variable number of arguments (called a *variadic function*) must have some way of knowing how many arguments have actually been passed to the function and what their types are. For example, the printf function (in <cstdio>) uses its format string to determine the number and type of arguments.

Example 13-8 shows how a function can use the <cstdarg> macros. The max function takes at least two arguments. The first is a count of the number of remaining arguments; the count must be positive. The template parameter specifies the type of each argument that follows the count.

Example 13-8. Finding the maximum value of any number of arguments

```
#include <cassert>

#include <cstdarg>


// Use a trivial wrapper class to ensure that va_end is called.

class varargs {

public:

  ~varargs(  ) { va_end(ap); }

  std::va_list& ap;

};


template <typename T>

T max(unsigned count, ...)

{

  assert(count > 0);

  varargs va;

  va_start(va.ap, count);

  T result = va_arg(va.ap, T);  // Get first argument.

  while (--count > 0) {

    T arg = va_arg(va.ap, T);   // Get successive arguments.

    if (arg > result)

      result = arg;             // Remember the largest.

  }

  return result;

}
```

```
int main(  )
{
  int a, b, c, d;
  ...
  int x = max<int>(4, a, b, c, d);
  int y = max<int>(2, x, 42);
  return y;
}
```

va_arg macro — *Gets next argument*

```
T va_arg(va_list ap, T)
```

The va_arg macro fetches the next argument, which must be of type T. The ap parameter must have been initialized by calling va_start. The type T must be a type that results from the standard type promotions (Chapter 3) or else the behavior is undefined. For example, T cannot be char, but must be int because the standard promotion of type char is to type int (or, in rare circumstances, unsigned int). The behavior is undefined if there is no next argument.

va_end macro — *Ends getting arguments*

```
void va_end(va_list ap)
```

The va_end macro finishes fetching arguments. You must call va_end once for each call to va_start. You cannot nest calls to va_start and va_end, but you can call them multiple times sequentially in the same function.

va_list type — *Argument list*

```
typedef ... va_list;
```

The va_list type is an opaque type that refers to the function's arguments. Declare a local variable of type va_list and supply the variable to the va_start, va_arg, and va_end macros.

va_start macro — *Starts getting arguments*

```
void va_start(va_list& ap, lastNamedParm)
```

The va_start macro initializes ap and prepares to fetch function arguments with va_arg. You must call va_end to clean up and finalize ap. The second argument to va_start is the name of the function's last parameter before the ellipsis. The last named parameter must not have a function, array, or reference type.

9.1. Understanding Web Parts

Web parts are based on ASP.NET web controls. In fact, the `WebPart` class inherits from `System.Web.UI.Control`. That means most of the programming issues of composition, lifetime, state, and server versus client-side processing are the same. In case you're not familiar with those concepts, here's a brief synopsis:

Composition

Web parts are made up of other componentsmainly ASP.NET web controls, HTML controls, literal strings, and client-side scripts. Components are assembled in a `Controls` collection and then rendered into HTML that is sent back to the client.

Lifetime

A web part instance is created on the server in response to a request from a client. That instance runs code, responds to events, renders the response, and then is destroyed soon after the response is sent back to the client. The next request creates a new instance which is in turn destroyed, and so on.

State

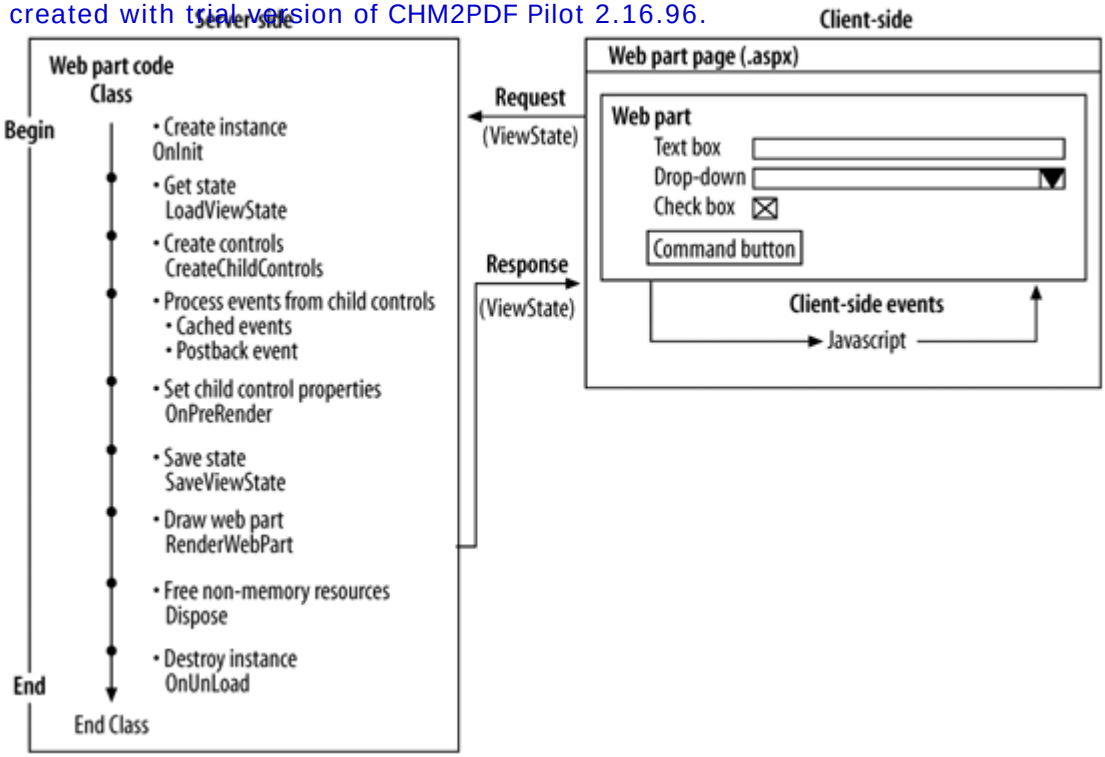
The properties of web controls are preserved beyond this lifetime by means of a built-in `ViewState` object. `ViewState` saves settings by means of hidden HTML controls rendered with the web part. The web part loads these values shortly after it is initialized with `LoadViewState` and saves any changes back just before it is rendered using `SaveViewState`.

Server/Client Code

The web part class runs on the server and that code only executes when the page is posted back to the server by means of a command button or some other postback control. You can also include JavaScript that runs on the clientthose scripts respond to events on the web part without returning to the server. That is more efficient and improves performance in many cases.

Figure 9-1 illustrates these concepts in terms of the life cycle of a web part on the server and how that is presented on the client.

animal 9-1. Key events in the life cycle of a web part



9.1.1. Extending ASP.NET

The previous section described high-level concepts that are constant for web parts and ASP.NET custom controls, but web parts extend the ASP.NET `Control` class to allow members to modify, save, and connect web part properties, as described here:

Modify

Members can set properties of web parts by choosing Modify Shared Web Part or Modify My Web Part from the web part menu and then entering their changes on the properties task pane.

Save

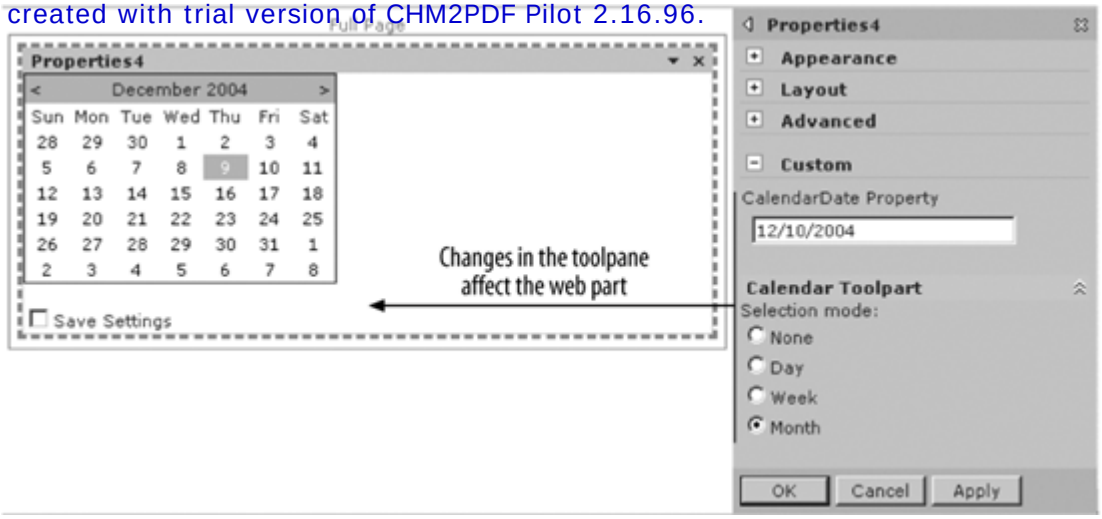
Web part properties can be saved to the content database. That allows members to modify the personal or shared view of a page and have those changes persist the next time they view the page.

Connect

Members can connect a property from one web part to a property on another web part. This creates a Lego-block type approach to building web part pages.

The SharePoint documentation calls web parts *designable*that is, authorized members can change their appearance and function from within the browser as shown in Figure 9-2.

animal 9-2. Designing a web part in the browser



Under ASP.NET those changes can be made only by editing a control's HTML, and there is no built-in way to save settings or connect parts. The SharePoint `WebPart` and `WebPartZone` classes provide the infrastructure for those features.

9.1.2. Using Web Parts

That said, not all web parts are designable. The navigation bar and Quick Launch areas on the default SharePoint home pages are examples of nondesignable `NavBar` web parts. Whether or not a web part is designable depends on where it is placed on a page:

- Web parts inside web part zones are designable and can be changed by members. The SharePoint documentation calls these *dynamic web parts*.
- Web parts outside of web part zones can't be modified by members. SharePoint calls these *static web parts*.

In Chapter 8, I showed you how to import dynamic web parts into a web part zone on a page. To add a static web part, edit the *.aspx* page in FrontPage to register the web part assembly and include an element for the web part. For example, the following line registers sample assembly used throughout this chapter:

```
<%@ Register TagPrefix="Ch09" Namespace="Ch09Samples" Assembly="Ch09Samples,
Version=1.0.0.0, Culture=neutral, PublicKeyToken=fb6919fe58e4ba63" %>
```

All the attributes of the `@ Register` directive match those defined for the web parts in the `SafeControl` element in *Web.config*, with the exception of `TagPrefix`. The `TagPrefix` attribute defines the prefix you'll use to qualify the web part element on the page. For example, the following element adds Webpart1 to the page:

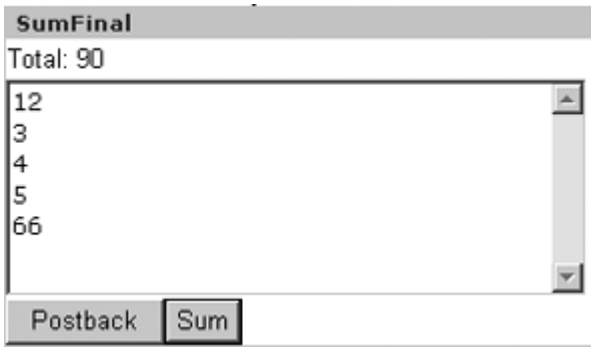
```
<ch09:WebPart1 text="New text property setting." title="WebPart1">
</ch09:WebPart1>
```

Properties of the web part are included as attributes in the element. In this case, the `text` attribute sets the `Text` custom property and the `title` attribute sets the web part's built-in `Title` property. Figure 9-3 shows the result.

animal 9-3. Setting a static web part's properties

In this context, the terms *static* and *dynamic* are misleading. Nondesignable (static) web parts can accept entries and display results dynamically the same way that designable parts can (Figure 9-4).

animal 9-4. Is this interactive part static or dynamic?



See the *StaticParts.aspx* sample for examples of including web parts outside of web part zones.

9.1.3. Programming Tasks

In ASP.NET you can create three types of custom controls: user controls, composite controls, and rendered controls. Because SharePoint doesn't directly support user controls (*.ascx*), all web parts are either made up of other web controls (composite controls) or rendered directly as HTML (rendered controls). In most cases, web parts use a combination of the two techniques, so that's what I show in my examples.

Table 9-1 summarizes the major tasks I cover in this chapter and provides a guide to some of the key members, clauses, attributes, and interfaces used to complete the task.

Table 9-1. Web part programming tasks

Task	Summary	Key members

Create user interface	Add ASP.NET controls to the `Controls` collection; then render those controls and additional HTML to an `output` object in `RenderWebPart`.	`Controls`
		`CreateChildControls`
		`RenderControls`
		`RenderControl`
		`RenderWebPart`

Add JavaScript and HTML	Write code to import and modify HTML control IDs to use the SharePoint-generated names.	`ClassResourcePath`
		`IsClientScriptBlockRegistered`
		`RegisterClientScriptBlock`
		`ReplaceTokens`
		`UniqueID`

Handle child control events	Hook up server-side event handling for ASP.NET controls in the `Controls` collection.	`AddHandler` `WithEvents` (clause) `Handles` (clause)
Create properties	Add property procedures to a web part and specify attributes that tell SharePoint how to display them in the property task pane and whether or not to save their values.	`Browsable` (attribute) `Category` (attribute) `DefaultValue` (attribute) `ElementName` (attribute) `SaveProperties`

		ShouldSerialize WebPartStorage (attribute) XmlElement (attribute) XmlRoot (attribute)
Add menus	Add items to the web part drop-down menu that respond to server or client events.	CreateWebPartMenu MenuItem MenuItems
Customize property task pane	Change the display of a web part's properties in the property task pane and control how the web part is updated from those settings.	ApplyChanges CancelChanges CustomPropertyToolPart GetToolParts ParentToolPane RenderToolPart WebPartToolPart
Make web parts connectable	Implement interface in the web part class to provide data to other web parts or consume data from other web parts.	ICellProvider ICellConsumer IRowProvider IRowConsumer IListProvider IListConsumer IFilterProvider IFilterConsumer IParametersInProvider IParametersInConsumer IParametersOutProvider IParametersOutConsumer

You can find reference information for these keywords in the SharePoint SDK or in the Visual Studio Helpthe SharePoint SDK omits items that aren't part of their namespace. For instance, it doesn't cover the `RenderControl` method because that is part of the .NET Framework.

PREV NEXT

9.2. Creating Web Part Appearance

The core of the web part template code is the `RenderWebPart` method. This method is called just before the control is disposed and it determines the appearance of the part on the page.

To create this appearance, you write HTML to the `output` object. For example, the following code displays a simple table containing information about the current user's identity:

```
' Requires this line at class level
' Imports System.Security.Principal
Protected Overrides Sub RenderWebPart _
  (ByVal output As System.Web.UI.HtmlTextWriter)
    ' Get the User identity.
    Dim user As IPrincipal = Me.Context.User
    ' Write table to output
    With output
         .Write("<TABLE id='tblUser'>")
         .Write("<TR><TD>Authenticated</TD><TD>")
         .Write(user.Identity.IsAuthenticated(  ))
         .Write("</TD></TR><TR><TD>User name</TD><TD>")
         .Write(user.Identity.Name(  ))
         .Write("</TD></TR><TR><TD>Authentication type</TD><TD>")
         .Write(user.Identity.AuthenticationType(  ))
         .Write("</TD></TR><TR><TD>Code is impersonating</TD><TD>")
         .Write(WindowsIdentity.GetCurrent(  ).Name)              .Write("</TD></TR><TR><TD>Request language: </TD><TD>")
         .Write(context.Request.UserLanguages(0))
         .Write("</TD></TR><TR><TD>Request host: </TD><TD>")
         .Write(context.Request.UserHostName)
         .Write("</TD></TR><TR><TD>Request IP: </TD><TD>")
         .Write(context.Request.UserHostAddress)
         .Write("</TD></TR></TABLE>")
    End With
End Sub
```

At runtime, the preceding web part displays user information in a two-column table, as shown in Figure 9-5.

animal 9-5. Rendering UserInfo web part at runtime (unauthenticated)

UserInfo	
Authenticated	False
User name	
Authentication type	
Code is impersonating	WOMBAT1\IUSR_JEFF-U0O1L3OFZF
Request language:	en-us
Request host:	67.8.219.122
Request IP:	67.8.219.122

HTML tables are a useful way to control the layout of web parts, but embedding all those table, row, and item tags in code results in code that is hard to debug, update, and localize. The only way to see the result of your HTML is to run the project, and it's easy to make errors in those string literals.

Fortunately, there's a better way. Instead of embedding HTML literals in code, create your tables as HTM files stored as resources, then load those and modify those files at runtime. This approach uses .NET's powerful string functions to substitute values from code into the table. To see how this works:

1. Create a new *.HTM* file in your web part project.
2. Set the file's Build Action property to Embedded Resource.
3. Edit the HTM page with Visual Studio's design tools to create the table, add headings, scripts, or controls as needed. Use literal placeholders (for example,: `{0}`) for items you will replace from code.
4. Create a procedure to load the resource by file name.
5. Create a procedure to fill in the table variables using the `String` object's `Format` or `Write` methods.
6. Write the result to the `output` object in the `RenderWebPart` event.

Figure 9-6 shows a user information table created in Visual Studio design mode that has seven placeholders for values to be filled in from code.

The following utility procedure loads the resource from the assembly and returns its HTML as a string. Resource names are case-sensitive, so be sure to correctly capitalize the file name when calling this procedure:

```
' Requires this line at class level:
' Imports System.Reflection
Friend Function GetHtml(ByVal fName As String) As String
```

animal 9-6. Creating a web part table in Visual Studio design mode

Authentication type	(1)
User name	(2)
Request language	(3)
Client browser	(4)
Client OS	(5)
Client IP address	(6)

```
    ' Get the web part assembly.
    Dim asm As [Assembly] = [Assembly].GetExecutingAssembly
    ' Build the full name of the resource (case-sensitive).
    Dim resName As String = asm.GetName.Name & "." & fName
    ' Declare a stream for the resource.
    Dim stream As IO.Stream
    Try
       stream = asm.GetManifestResourceStream(resName)
       ' Create a reader for the stream.
       Dim reader As New IO.StreamReader(stream)
       ' Read the stream and return it as a string.
       Return reader.ReadToEnd
   Catch ex As Exception
       Trace.Write(ex.ToString(  ))
       Return ""
   End Try
End Function
```

Next, the `BuildTable` procedure creates an array of values to plug in to the table's literal placeholders, gets the table from the assembly, and performs the substitution. I put this task in a dedicated procedure, rather than in `RenderWebPart`, to make it easier to extend `RenderWebPart` laterit's a good idea to keep that event uncluttered:

```
       ' Builds the UserInfo Table
    Private Function BuildTable(  ) As String
       ' Create an array for table variables
       Dim arr(6) As String
       ' Populate the array with info from the current context.
       With Me.Context
           arr(0) = .User.Identity.IsAuthenticated
           arr(1) = .User.Identity.AuthenticationType
           arr(2) = .User.Identity.Name
           arr(3) = .Request.UserLanguages(0)
           arr(4) = .Request.Browser.Browser
           arr(5) = .Request.Browser.Platform
           arr(6) = .Request.UserHostAddress
       End With
       ' Read the table resource.
       Dim sTable As String = GetHtml("userInfoTable.htm")
       ' Substitute the user info values into the table and return.
       Return String.Format(sTable, arr)
   End Function

   ' Draw the web part.    Protected Overrides Sub RenderWebPart _
    (ByVal output As System.Web.UI.HtmlTextWriter)
      ' Write the table to output
      output.Write(BuildTable)
   End Sub
```

At run-time, the web part loads the table from the assembly, fills in the placeholders, and displays the result as shown in Figure 9-7.

animal 9-7. Displaying a table from an embedded resource

UserInfoTable	
Authenticated	True
Authentication type	NTLM
User name	WOMBAT1\ExcelDemo
Request language	en-us
Client browser	IE
Client OS	WinXP
Client IP address	67.8.219.122

This approach isn't limited to tables. You can add client-side scripts, controls, styles, formatting, or any other type of valid HTML. Since the output is stored in an HTM file, you can preview the output in the browser, test the scripts, and edit/localize versions much more easily than if the output was embedded in code.

9.3. Adding Child Controls

The UserInfoTable web part isn't interactive: the information flows only from the server to the browser. To make a web part that can interact with members:

1. Declare the web controls to display in the web part.
2. Override the `CreateChildControls` event to set control properties.
3. Add each control to the controls collection.
4. Render the child controls in the `RenderWebPart` event.

The following code demonstrates the steps to create a Sum web part containing a textbox to receive a series of numbers input, a command button to perform the calculation, and a label to display the result:

```
     ' 1) Declare child controls.
     Dim _txt As New TextBox
     Dim _br As New Literal
     Dim WithEvents _btn As New Button
     Dim _lbl As New Label
     Dim _total As String

Protected Overrides Sub CreateChildControls(  )
     ' Create utility object for dimensions.
     Dim u As Unit
     ' 2) Set child control properties.
     With _txt
        .Width = u.Pixel(400)
        .Height = u.Pixel(200)
        .TextMode = TextBoxMode.MultiLine
    End With
    With _br
        .Text = "<br>"
    End With
    With _btn
        .Width = u.Pixel(60)
        .Height = u.Pixel(30)
        .Text = "Sum"
        .ToolTip = "Click here to get total."
    End With
    With _lbl
        .Width = u.Pixel(100)
        .Height = u.Pixel(30)
    End With
    ' 3) Add the controls in the order to display them
    Controls.Add(_txt)
    Controls.Add(_br)
    Controls.Add(_btn)
    Controls.Add(_lbl)
End Sub

' Display web part.
Protected Overrides Sub RenderWebPart _
  (ByVal output As System.Web.UI.HtmlTextWriter)
    ' 4) Write controls to output stream.
    RenderChildren(output)
End Sub
```

The `RenderChildren` method in step 4 renders the child controls collection to the `output` object. The order of the controls in the collection is preserved, so it's important that the controls are added in the correct order in step

3. Alternately, you can use the `RenderControl` method to write the controls to `output` one at a time in any order:

```
' Alternate approach, write controls one at a time.
Protected Overrides Sub RenderWebPart _
  (ByVal output As System.Web.UI.HtmlTextWriter)
    ' Use different order.
    _lbl.RenderControl(output)
    _br.RenderControl(output)
    _txt.RenderControl(output)
    _br.RenderControl(output)
    _btn.RenderControl(output)
End Sub
```

Using `RenderControl` makes it a little easier to intersperse HTML literal strings with the controls, but I try to avoid that. Instead, I tend to use literal controls such as `_br` in the preceding example. That's my attempt to keep control definitions organized.

The command button control (`_btn`) is declared `WithEvents` so that user actions raise events that can be handled in the web part's code. To respond to an event from a child control, create an event procedure with a `Handles` clause:

```
Private Sub _btn_Click _
  (ByVal sender As Object, ByVal e As System.EventArgs) _
  Handles _btn.Click
    ' Calculate the total.
    _total = Sum(  ).ToString
    ' Display the total.
    _lbl.Text = "Total: " & _total
End Sub

Public Function Sum(  ) As Double
    Dim total As Double
    ' Make sure there are numbers to add.
    If _txt.Text.Length Then
        Dim arr As String(  )
        arr = Split(_txt.Text, vbCr)
        For Each itm As String In arr
            Try
                total &= Convert.ToDouble(itm)
            Catch ' Skip if not a number.
            End Try
        Next
     Else
        total = 0
     End If
     Return total
 End Function
```

Sidebar 9-1. For C# Users

Visual C# doesn't have `WithEvents` or `Handles` clauses. Instead, associate the event with the handler, as shown here:

```
_btn.Click += new EventHandler(_btn_Click); // C#
```

At runtime, the Sum web part adds the series of numbers entered in the textbox and displays the result when the user clicks Sum, as shown in Figure 9-8.

animal 9-8. Responding to events from child controls

9.4. Working on the Client Side

The Sum web part performs its calculations on the server, but that's not really necessary or efficient. For high-volume applications, it's a good idea to do as much work on the client side as possible. For example, the following HTML creates an equivalent client-side web part that doesn't require a roundtrip to the server to perform the calculations:

```
<html>
    <head>
        <script id="clientEventHandlersJS" language="javascript">
  function _btn_onclick(  ) {
     var arr = new Array("");
     // Use getElementById, direct control refs don't work
     var _txt = document.getElementById("_txt");
     var _div = document.getElementById("_div");
     var total = 0, s = _txt.value;
     arr = s.split("\n");
     for (var i in arr)
     {
          total += parseFloat(arr[i]);
     }
     _div.innerText = "Total: " + total;
     return;
   }
          </script>
       </head>
      <body>
          <form id="_frm">
       <div id="_div">Total:
       </div>
       <TEXTAREA id="_txt" name="_txt" rows="10" cols="35">
       </TEXTAREA>
       <br>
       <INPUT id="_btn" type="button" value="Sum"
            onclick="return _btn_onclick(  )">       </form>
  </body>
</html>
```

This code is stored as an HTM file in a resource, then loaded and rendered by the following line:

```
' See "Creating Web Part Appearance" for GetHtml(  ) code.
output.Write(GetHtml("clientSum.htm")
```

At runtime this web part is visually identical to the server-side web part in Figure 9-8, but the calculation is done on the client computer via JavaScript. The result is much better performance and less network traffic, since the page isn't sent back to the server every time the member clicks the Sum button.

9.4.1. Using Scripts with Web Controls

The ClientSum web part is efficient, but you can't easily get values from the contained HTML controls once the page returns to the server. To see this limitation, enter some values in ClientSum and refresh with F5: the values you entered are cleared. That happens because the HTML controls don't automatically preserve their state the way that ASP.NET web controls do.

To solve this problem, use ASP.NET web controls rather than HTML controls. In other words, create a hybrid web part that uses both server-side controls and client-side scripts. Using client-side scripts with web controls requires these special steps:

1. Choose your web control type carefully; not all web controls are easy to access from client-side code. For

example, I had to change a label control to a read-only textbox in order to get its value both on the client side and server side.

2. Add an ID property for each web control. This property allows you to get a reference to the control from client-side scripts through the `getElementsByName` method.

3. Add the web controls to the Controls collection. This step ensures that the controls preserve their state.

4. Write code to get the web part's `UniqueID` and pass that value to scripts.

5. Write client-side scripts that combine the passed-in `UniqueID` with the element IDs created in step 2.

When SharePoint renders a web part, it includes a lot of special code to preserve the state of web controls in the Controls collection. To make sure that code gets the right values, SharePoint pre-appends a unique identifier to each `name` and `id` element in the generated HTML, as shown here:

```
<textarea name="FullPage:g_03d3b969_e9c0_4846_9cf5_b14b5e7f6aa7:_txt"
id="FullPage_g_03d3b969_e9c0_4846_9cf5_b14b5e7f6aa7_  _txt"
title="Enter a series of numbers to add." style="height:150px;width:300px;">
</textarea>
```

If you don't add an `ID` property to the control (step 2), SharePoint generates a `name` attribute and omits `id`. If you don't add the web controls to the Controls collection (step 3), SharePoint doesn't preserve the state of the control and consequently doesn't pre-append `UniqueID`. The following code shows these steps implemented for the Sum control:

```
' .NET code running on the server.
' Declare child controls.
Dim _txt As New TextBox
Dim _br1 As New Literal
Dim _br2 As New Literal  ' Added literal
Dim WithEvents _btn As New Button

Dim _lbl As New TextBox  ' 1) Changed to textbox

Protected Overrides Sub CreateChildControls(  )
    ' Create utility object for dimensions.
    Dim u As Unit
    ' Set child control properties.
    With _txt

       .ID = "_txt" ' 2) Added ID for scripts.
       .Width = u.Pixel(300)
       .Height = u.Pixel(150)
       .TextMode = TextBoxMode.MultiLine
       .ToolTip = "Enter a series of numbers to add."
     End With
     _br1.Text = "<br>"
     _br2.Text = "<br>"
     _btn.Text = "Postback"   ' Changed.
     With _lbl

        .ID = "_lbl" ' 2) Added ID for scripts.
        .Text = "Total: "
        .ReadOnly = True
        .BorderStyle = BorderStyle.None
      End With

      ' 3) Add the controls in the order to display them
      Controls.Add(_lbl)
      Controls.Add(_br1)
      Controls.Add(_txt)
      Controls.Add(_br2)
  End Sub
```

```
    ' Display web part.
    Protected Overrides Sub RenderWebPart _
      (ByVal output As System.Web.UI.HtmlTextWriter)
        ' Load client-side script.
        output.Write(GetHtml("sumFinal.js"))
        ' Write controls to output stream.
        RenderChildren(output)

        ' 4) Add button to run script.
        output.Write("<INPUT id='_btn' type='button' " & _
          "value='Sum' onclick='return _btn_onclick(""" & _
          Me.UniqueID(  ) & """)' name='_btn'>")
      End Sub
```

Step 4 above is a little tricky. I embedded the HTML button control in my code (despite telling you to avoid this) because it's only one control and because it makes it easier to include `Me.UniqueID` as an argument to `_btn_onclick`. SharePoint also provides a `ReplaceTokens` method to replace embedded tokens with SharePoint values. For example, it replaces `_WPID_` with the web part's unique ID:

```
' Alternate approach -- use ReplaceTokens
output.Write(ReplaceTokens("<INPUT id='_btn' type='button' value='Sum'" & _
  " onclick='return _btn_onclick(""FullPage:_WPID_"")' name='_btn'>"))
```

`ReplaceTokens` can replace the following literals. (See the SharePoint SDK for additional details.)

Token	Replacement value
`_WPR_`	`ClassResourcePath` property
`_WPQ_`	`Qualifier` property
`_LogonUser_`	`Request.ServerVariables( "LOGON_USER" )`
`_WPID_`	`ID` property (Control.ID)

The client-side script uses the passed-in `UniqueID` to get references to web controls through `getElementsByName`, as shown here:

```
 // JavaScript running on the client (sumFinal.js).
<script id="clientEventHandlersJS" language="javascript">
// Pass in unique page ID generated by SharePoint.
function _btn_onclick(uID) {
    var arr = new Array("");
    // Get elements using passed-in unique ID.

    var _txt = document.getElementsByName(uID + ":_txt")[0];
    var _lbl = document.getElementsByName(uID + ":_lbl")[0];
    var total = 0, s = _txt.value;
    arr = s.split("\n");
    for (var i in arr)
        total += parseFloat(arr[i]);
    _lbl.innerText = "Total: " + total;
    return;
 }
 </script>
```

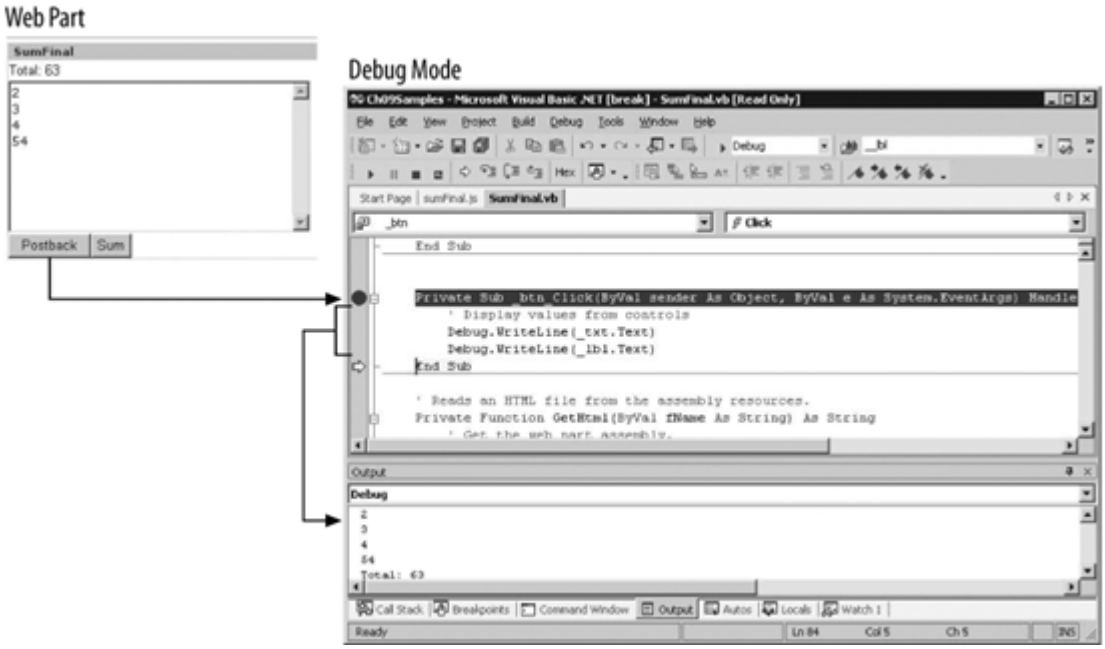
This completed control will now perform its calculation on the client side, and preserve its settings. Also, the values of the `_lbl` and `_txt` web controls are now available to the server. To test, add this code, set a breakpoint, and click the Postback button to see the values displayed in the Visual Studio Debug window:

```
Private Sub _btn_Click(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles _btn.Click
```

```
        Display values from controls
        Debug.Write(_txt.Text)
        Debug.WriteLine(_lbl.Text)
    End Sub
```

Figure 9-9 shows the completed control at runtime in Debug mode.

animal 9-9. Using debug mode to make sure control values are available on the server



9.4.2. Importing Script Blocks

I explained how to store HTML as a resource because it is a better way to build tables and other display elements of a web part. And I just used that same technique to insert a client-side script into a web part. That's OK for small scripts, but it means you have to rebuild the assembly each time you change the script.

You can import externally stored scripts directly into a web part using the `RegisterClientScriptBlock` method. That method imports scripts from a server folder at runtime so you can modify and debug scripts without rebuilding the assembly. Instead, you can simply refresh the page (F5) to get the changes.

To import script blocks into a web part:

1. Create a subfolder in the server's *wpresources* or *_wpresources* virtual folder for the scripts.
2. Copy the script to the new folder.
3. Add code to the web part's `PreRender` event to load the script from the new location.

The physical locations shown here are the defaults used when IIS and SharePoint are installed. Your locations may be different. Check the *wpresources* or *_wpresources* virtual folders in IIS for your actual paths.

Where you store the scripts depends on how the web part is installed. For web parts installed in a *\bin* folder, create a subfolder named after the web part assembly in the *C:\InetPub\wwwroot\wpresouces* folder. For example:

```
C:\InetPub\wwwroot\wpresouces\Ch09Samples
```

For web parts installed in the global assembly cache (GAC), create the assembly folder in *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\wpresources*; then create a sub folder in the new folder using the `Version`, `Culture`, and `PublickKeyToken` attributes from the web part's `SafeControl` element in *Web.config*. The name has the following form:

```
version_culture_token
```

Omit *culture* if the assembly culture is neutral. For example:

```
C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\wpresources\
Ch09Samples\ 1.0.0.0_  _fb6919fe58e4ba63
```

When you copy the script files to the new folder, remember to remove surrounding `<script>` tags from the file or you will get syntax errors when you import the script. Scripts should only be loaded once per page, so you need to create a unique key for each script file. Then determine whether the file has already been imported by checking `IsClientScriptBlockRegistered` before calling `RegisterClientScriptBlock`, as shown here:

```
Private Sub ImportScripts_PreRender(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles MyBase.PreRender
    Const sKey As String = "sumFinal.js"
    Dim sFile As String = Me.ClassResourcePath & "/sumFinal.js"
    Dim sBlock As String = "<script language='javascript' src='" & sFile & "'/>"
    ' Load client-side script.
    If (Not Me.Page.IsClientScriptBlockRegistered(sKey)) Then _
    Me.Page.RegisterClientScriptBlock(sKey, sBlock)
End Sub
```

In the preceding code, `ClassResourcePath` returns the location of the folder you created for script storage. The `RegisterClientScriptBlock` method inserts the script tag `sBlock` on the page and registers the script as `sKey` so it won't be loaded again. At runtime, SharePoint renders this output:

```
<script language='javascript'
src='http://localhost/wpresources/Ch09Samples/sumFinal.js'/>
```

9.5. Understanding Event Order

From an event perspective, web part programming is very different from Windows Forms programming. In Windows Forms applications, code runs as long as the form is displayed and responds as events occur. Web part code runs only short burstsbeginning when the request is received by the server and ending shortly after the response is returned to the client browser.

The performance difference between the server and client Sum controls illustrates a basic principle of web part design: *avoid unnecessary postbacks*. The button web control and submit button trigger postback events by default, but you can also set the `AutoPostback` property on textbox, list, and checkbox controls to cause postbacks. When a postback event occurs, the browser sends the current state of the page back to the server, which then processes web part events and overridden methods in the order shown in Table 9-2.

Table 9-2. Order of major server events/methods in a web part

Event/method	Use to
`OnInit` event	Create the web part.
`LoadViewState` method	Override the web part's base class behavior when reading control properties from the web part's `ViewState` object.
`CreateChildControls` method	Add web controls to the Controls collection.
`Onload` event	Initialize resources used by the web part.
Cached child control events	Process cached events from child controls, such as `TextChanged` or `SelectedIndexChanged`.
Postback child control event	Process the event that caused the postback, for example the button `Click` event.
`OnPreRender` event	Set child control properties and complete any processing before rendering the part.
`SaveViewState` method	Override the web part's base class behavior when saving control properties to the web part's `ViewState` object.
`RenderWebPart` method	Draw the web part as HTML.
`Dispose` method	Release resources used by the web part.
`OnUnload` event	Finalize the release of resources.

Web parts have many more events and methods than those in Table 9-2, but the ones there are critical to understanding how to code web parts and avoid unexpected results. That's also why I combined events and methods in the same tableit's important to know that the `CreateChildControls` method is called before the `OnPreRender` event occurs, for example.

The other thing that's important about the information in Table 9-2 is that cached events occur before the event that caused the postback. *Cached events* are server-side events that occur on a child control, but that don't cause a postback. SharePoint determines what cached events occurred by checking the web part's `ViewState`; then SharePoint raises those events after the web part loads.

See the *EventOrder.aspx* sample to test the sequence of events on a web part. That sample records the order of events and displays the results after a postback as shown in Figure 9-10. The code for the EventOrder simply appends a string to display as shown here:

```
Protected Overrides Sub OnInit(ByVal e As System.EventArgs)
    _msg &= "OnInit event<br>"
End Sub
```

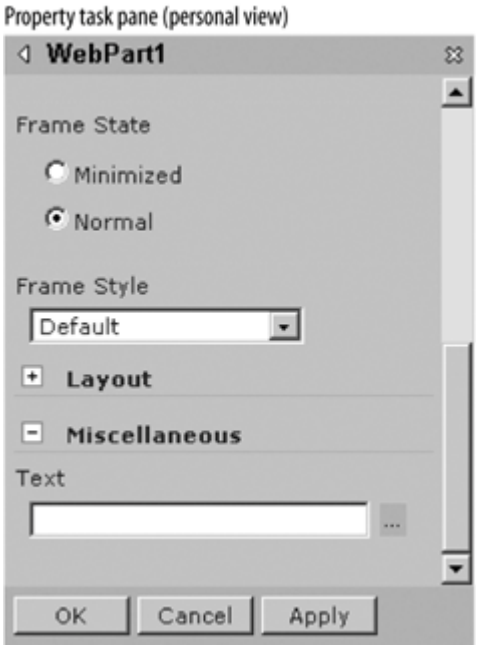
animal 9-10. Testing the order of events and method calls


```
    Protected Overrides Sub LoadViewState(ByVal savedState As Object)
        _msg &= "LoadViewState method<br>"
        ' Delegate back to base class.
        MyBase.LoadViewState(savedState)
    End Sub

...
```

9.6. Adding Properties

The default web part template comes with a `Text` property. That property appears at the bottom of the property task pane in the Miscellaneous category if you display the web part page in a personal view and then select Modify My Web Part, as shown in Figure 9-11.

animal 9-11. Displaying the custom Text property that comes with the web part template



The provided `Text` property is really only intended as a basic starting point for your own properties. Starting from `Text`, the most common next steps are:

1. Change the property storage from personal to `Shared`. Shared properties are available to everyone who views the page.

2. Serialize the property so that changes in the task pane are reflected in the web part.

3. Link the property setting to the value of a child control on the web part. That allows members to change the property and see the effects of the change.

4. Save the property to the content database; otherwise, changes are lost when the member navigates away from the page.

The following code illustrates these changes by extending the template code to create a simple web part with a textbox and a Save button. The textbox is linked to the `Text` property, so if the member clicks Save, the property is preserved the next time the page is viewed:

```
Private Const _defaultText As String = ""
Dim _text As String = _defaultText

' 1) Change storage to WebPartStorage(Storage.Shared).
'   and
' 2) Serialize: XmlElement(GetType(String), ElementName:="Text").
<Browsable(True), Category("Custom"), DefaultValue(_defaultText), _
  WebPartStorage(Storage.Shared), _
  FriendlyName("TextBox"), _
  Description("TextBox Text Property"), _
  XmlElement(GetType(String), ElementName:="Text")> _  .
  Property [Text](  ) As String
```

```
            Get
                Return _text
            End Get
            Set(ByVal Value As String)
               _text = Value
               ' 3) Link the property setting to a child control.
               _txt.Text = _text
            End Set
        End Property

        ' 2) Continued. Flag that the Text property should be
        ' stored in the content database using the property's
        ' XmlElement attribute settings.

        Public Function ShouldSerializeText(  ) As Boolean
            Return True
        End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        ' Web part UI definition.
        ' Web controls to include in web part.
        Dim WithEvents _btnSave As New Button
        Dim WithEvents _txt As New TextBox
         Protected Overrides Sub CreateChildControls(  )
            ' Set control properties
            _txt.ID = "_txt1"
            _btnSave.Text = "Save Settings"
            ' Add to controls collection
            Controls.Add(_txt)
            Controls.Add(_btnSave)
        End Sub

        'Render this Web Part to the output parameter specified.
        Protected Overrides Sub RenderWebPart _
         (ByVal output As System.Web.UI.HtmlTextWriter)
           ' Render view state controls.
          RenderChildren(output)
        End Sub

        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        ' Child control event procedures

        ' 3) Continued. Update the Text property if the text box changed.

        Private Sub _txt_TextChanged(ByVal sender As Object, _
          ByVal e As System.EventArgs) Handles _txt.TextChanged
            Me.Text = _txt.Text
        End Sub

        ' 4) Save all the property settings to the content database.

        Private Sub _btnSave_Click(ByVal sender As Object, _
          ByVal e As System.EventArgs) Handles _btnSave.Click
            Me.SaveProperties = True
        End Sub
```

The following sections explain these steps in greater detail.

9.6.1. Setting Property Attributes

The attributes for the property indicate whether or not the property appears in the task pane, what heading it is listed under, how it is stored, and other descriptive information. Table 9-3 explains the attribute settings.

Table 9-3. Web part property attribute settings

Attribute	Setting
Browsable	true displays the custom property in the property task pane. False hides the property.
Category	The task-pane section to include the property in. This defaults to Miscellaneous if omitted or if you specify one of the built-in categories, like Appearance.
DefaultValue	The default value of the property.
Description	The tool tip to display for the property in the task pane.
FriendlyNameAttribute	The caption to display for the property in the task pane.
ReadOnly	true prevents changes in the task pane. False allows changes. Only affects settings from the task pane.
WebPartStorage	Storage.Shared displays the custom property in the shared view of the page. Storage.Personal displays the property in the personal view. Storage.None to omit the setting and not store it.
ControlledExport	TRue allows users to export the property from the personal view of a web part. False prevents export.
HtmlDesignerAttribute	A custom property builder specified by a URL.
XmlElement	The XML element and data type to store the property as. This attribute is explained more in "Serializing Properties."

Specifying a default value reduces the web part's storage requirements because the property's value is only stored if it is different from the default. Because of this, it's important to ensure the DefaultValue attribute matches the setting used in code. The template property does that by initializing the _text variable with the _defaultText constant as shown here:

```
Private Const _defaultText As String = ""
Dim _text As String = _defaultText

<Browsable(True), Category("Custom"), DefaultValue(_defaultText), ...
```

It's a good idea to follow this practice in your own properties.

9.6.2. Serializing Properties

In order to save a property setting, SharePoint must have an XML namespace, element name, and data type to use when storing the value. The class XmlRoot attribute and the property XmlElement attribute define those things for the custom property. For example, the following attributes store the Text property as a string in <Text xmlns="Ch09Samples"></Text>:

```
<DefaultProperty("Text"), ToolboxData("<{0}:Properties runat=server></{0}:Properties>
 "), XmlRoot(Namespace:="Ch09Samples")> _
 Public Class Properties
     ...
   <Browsable(True), Category("Custom"), DefaultValue(_defaultText), _
       WebPartStorage(Storage.Shared), _
       FriendlyName("TextBox"), _
       Description("TextBox Text Property"), _
       XmlElement(GetType(String), ElementName:="Text")> _  .
       Property [Text](  ) As String
      ...
```

You can store any type that can be serialized. Just use GetType to specify the type in the property's XmlElement attribute. For complex types, you need to provide attributes to indicate how the types should be serialized. For example, the following Enum provides serialization information so that SharePoint can store the Color property correctly:

```
Enum enumColor
```

```
        <XmlEnum("Default")> [Default] = Drawing.KnownColor.ActiveCaption
        <XmlEnum("Black")> Black = Drawing.KnownColor.Black
        <XmlEnum("Blue")> Blue = Drawing.KnownColor.Blue
        <XmlEnum("Gray")> Gray = Drawing.KnownColor.GrayText
        <XmlEnum("Green")> Green = Drawing.KnownColor.Green
        <XmlEnum("Red")> Red = Drawing.KnownColor.Red
        <XmlEnum("White")> White = Drawing.KnownColor.White
    End Enum

    Private Const _defaultColor As enumColor = enumColor.Default
    Dim _color As enumColor = _defaultColor

    <Browsable(True), Category("Custom"), DefaultValue(_defaultColor), _
       WebPartStorage(Storage.Shared), _
       FriendlyName("Color"), _
       Description("TextBox Color Property"), _
       XmlElement(Type:=GetType(enumColor), _
       ElementName:="Color")> _
       Property Color(  ) As enumColor
         Get
            Return _color
         End Get

         Set(ByVal Value As enumColor)
             _color = Value
             _txt.ForeColor = Drawing.Color.FromKnownColor(_color)
         End Set
       End Property

       Public Function ShouldSerializeColor(  ) As Boolean
           Return True
       End Function
```

In this case, the `XmlEnum` attribute associates a string with each `Enum` value. SharePoint uses those strings when it serializes the property. For example:

```
<Color xmlns="Ch09Samples">Default</Color>.
```

The other part of serialization is the `ShouldSerialize`*PropName* method. This is a special type of method that allows you to turn serialization on or off for a property. To allow serialization, create the method and return `true` as shown here:

```
Public Function ShouldSerializeText(  ) As Boolean
    Return True
End Function
```

Note that the `ShouldSerialize`*PropName* doesn't override an inherited method; it's just a `Public` method with a special name. The `WebPart` class includes these other `ShouldSerialize` methods that can be overridden to control built-in properties:

See the SharePoint SDK for information on overriding these `WebPart` methods.

9.6.3. Linking Properties to Controls

So far, I've only dealt with storing the settings from the task pane to the custom property. To see these changes displayed in a child control, you need to set one of the control's properties from the property `Set` clause. For example, the following code updates the textbox when the custom `Text` property changes:

```
Dim _text As String = _defaultText
Dim WithEvents _txt As New TextBox

Property [Text](  ) As String
```

```
        Get
            Return _text
        End Get
        Set(ByVal Value As String)
            _text = Value
            ' 3) Link the property setting to a child control.

             _txt.Text = _text
        End Set
    End Property
```

On the other side, changes to the textbox should update the custom `Text` property, as shown here:

```
    Private Sub _txt_TextChanged(ByVal sender As Object, _
      ByVal e As System.EventArgs) Handles _txt.TextChanged
        Me.Text = _txt.Text
    End Sub
```

Now changes to the property are displayed in the control, and vice versa.

9.6.4. Saving Property Settings

Use the `SaveProperties` property to save properties to the content database. Without this step, only design-time changes made through the task pane are saved. You can think of task pane settings as a sort of second-level default: the first-level defaults are those settings built in to the web part; the second-level are those stored when you modify the web part.

The following code saves the web part's properties to the content database when the user clicks Save:

```
    Dim WithEvents _btnSave As New Button


    '  Save  the property settings
    Private Sub _btnSave_Click(ByVal sender As Object, _
      ByVal e As System.EventArgs) Handles _btnSave.Click
        Try
            ' Reequires write privileges.
            Me.SaveProperties = True
        Catch
             _msg &= "You must be signed in as a contributor to save changes. "
        End Try
    End Sub
```

Setting `SaveProperties` to `true` tells SharePoint to serialize property settings and save them to the content database after the page is rendered. You only need to set the property once, but the `_btnSave` command button causes a postback event when clicked, which re-renders the control and thus saves the settings.

In some cases, you may not want to save a runtime property. For example, if a web part performs a query on a database, it makes little sense to save the results since that data changes over time. For those properties, return `False` from the `ShouldSerial`*PropName* method.

Though not strictly required, it's a good idea to provide a `ShouldSerial`*PropName* method even if the property isn't serialized, since it allows you to override that behavior in a derived web part.

9.6.5. Controlling/Debugging Serialization

You can use any type of property that can be serialized. That includes numeric types, enumerations, arrays, and objects. SharePoint knows how to serialize the numeric types, but in other cases you need to include attributes

to specify how those types are serialized. To see how your web part serializes its properties:

1. Display the web part on a test page.

2. Save the properties.

3. Click the down arrow on the web part's title bar and choose Export. SharePoint displays a download dialog that lets you save a *.dwp* file containing the current property settings.

The `Enum` sample in the section "Serializing Properties" earlier in the chapter shows how to include serialization information needed to store an enumeration. Arrays of simple types are serialized using subelements of the correct type, such as this array property:

```
Dim _items As String(  ) = {"this", "that", "other"}

' No default value, always store value.     <Browsable(True), Category("Custom"), _
  WebPartStorage(Storage.Shared), _
  FriendlyName("Array"), _
  Description("String array property."), _
  XmlElement(GetType(String(  )), ElementName:="Items")> _
  Property Items() As String(  )
    Get
        Return _items
    End Get
    Set(ByVal Value As String(  ))
         _items = Value
    End Set
End Property

Public Function ShouldSerializeItems(  ) As Boolean
     Return True
End Function
```

is serialized as:

```
<Items xmlns="Ch09Samples">
    <string>this</string>
    <string>that</string>
    <string>other</string>
</Items>
```

Using `XmlArray` and `XmlArrayItem` attributes instead of `XmlElement` lets you specify how each item of the array is serialized. For example the following attributes serialize the array items as `Item` elements rather than `string` elements:

```
 <Browsable(True), Category("Custom"), _
      WebPartStorage(Storage.Shared), _
      FriendlyName("Array"), _
      Description("String array property."), _

XmlArray(ElementName:="Items"), _
XmlArrayItem(GetType(String), ElementName:="Item")> _
Property Items() As String(  )
```

Saving property settings to the content database can cause subtle errors as you develop your part. Changing the type or name of a property changes how it is serialized, which often results in orphaned entries in your content database. SharePoint provides an `UnknownXmlElements` collection as a way to cull these items from the database. The following code checks for orphaned items and removes them during development:

```
' Use this code to clean up database during development.
Public Overrides Sub AfterDeserialize(  )
#If DEBUG Then
```

```
        For Each itm As System.Xml.XmlElement In UnknownXmlElements
            Debug.WriteLine(itm.Name, itm.Value) ' < Set a breakpoint here.
            ' Delete the item.
            UnknownXmlElements.Remove(itm)
        Next
#End If
End Sub
```

You face a similar problem when you upgrade a web part in the field. Obsolete properties must be mapped to new properties. See the topic on the `AfterDeSerialize` method in the SharePoint SDK for an example of how to handle these upgrade issues.

9.6.6. Setting Properties in DWP

You can use the XPath created by the `XmlNamespace` and `XmlElementName` attributes to override the web part's default property settings in the web part description (*.dwp*). For example, the following *.dwp* file sets initial value for `Text` property:

```
<?xml version="1.0" encoding="utf-8"?>
<WebPart xmlns="http://schemas.microsoft.com/WebPart/v2" >
    <Title>Properties</Title>
    <Description>Demonstrates web part properties.</Description>
    <Assembly>Ch09Samples</Assembly>
    <TypeName>Ch09Samples.Properties</TypeName>
    <!-- Specify initial values for any custom properties here. -->

    <Text xmlns="Ch09Samples">Setting from DWP</Text>
</WebPart>
```

When a member imports this *.dwp* file to a page, `Setting from DWP` is displayed in the textbox. The Excel Office Web Part Add-In covered in Chapter 8 uses this technique to create customized web parts from the base Spreadsheet web part. In that case, the `SolutionFileLocation` property is set to a solution file containing the customizations made to the web part:

```
<?xml version="1.0"?>
<WebPart xmlns="http://schemas.microsoft.com/WebPart/v2"
   xmlns:ODP="http://schemas.microsoft.com/WebPart/v2/Spreadsheet">
     <Title>Loan Amortization1</Title>
     <Description>Enter description here.</Description>
     <Assembly>Microsoft.Office.Dataparts, Version=11.0.0.0, Culture=neutral,
      PublicKeyToken=71e9bce111e9429c</Assembly>
     <TypeName>Microsoft.Office.Dataparts.SpreadsheetCtl</TypeName>

     <ODP:SolutionFileLocation>/Samples1/LoanSolutionSpec.xml
     </ODP:SolutionFileLocation>
 </WebPart>
```

The solution file in turn identifies the Excel XML spreadsheet file to display in the part:

```
<?xml version="1.0"?>
<SolutionSpecification xmlns="http://schemas.microsoft.com/WebPart/v2/Spreadsheet/
 Solutionspecification">
     <WebPartSettings>

          <XMLSSFileLocation>/Samples1/LoanXMLSS.xml</XMLSSFileLocation>
     </WebPartSettings>
 </SolutionSpecification>
```

9.7. Adding Menus

To add a menu to a web part, override `CreateWebPartMenu` to add `MenuItem` objects to the web part's `MenuItems` collection. For example, the following code adds a Save Settings item to the web part menu:

```
Dim _mnuSave As MenuItem

Public Overrides Sub CreateWebPartMenu(  )
    _mnuSave = New MenuItem("Save Settings", "mnuSave", _
      New EventHandler(AddressOf _mnuSave_Click))
     Me.WebPartMenu.MenuItems.Add(_mnuSave)
End Sub

Private Sub _mnuSave_Click(ByVal sender As Object, ByVal e As System.EventArgs)
   Try
       Me.SaveProperties = Not _chkSave.Checked
  Catch
       _msg &= "You must be signed in as a contributor to save properties. "
  End Try
End Sub
```

The `MenuItem` constructor sets the displayed name, ID, and event handler for the menu item. In this case, the `_mnuSave_Click` event handler switches the `SaveProperties` setting for the web part on and off.

Menus don't save their state: they are redrawn from scratch each time the page is displayed. That's why a statement like `_mnuSave.Checked = Not _mnuSave.Checked` doesn't toggle the checkmark on a menu item. Instead you must use a web part property or a web control (such as the checkbox) to track the checked status of a menu item.

The `CreateMenu` and `CreateChildControls` methods are not called in a fixed order, so it's important to set web control properties in the `OnPreRender` event when working with web part menus. `OnPreRender` always occurs just before the web part is rendered. This code completes the Save Settings menu example by setting the control properties just before the web part is displayed:

```
 Dim WithEvents _cal As New Calendar
 Dim _chkSave As New CheckBox
 Dim _msg As String

Protected Overrides Sub CreateChildControls(  )
    ' Add to controls collection
    Controls.Add(_cal)
    Controls.Add(New br)
    Controls.Add(_chkSave)
End Sub

Protected Overrides Sub OnPreRender(ByVal e As System.EventArgs)
     ' Set control properties
     _chkSave.Text = "Save Settings"
     _chkSave.Checked = Me.SaveProperties
     _mnuSave.Checked = Me.SaveProperties
     _cal.SelectedDate = Me.CalDate
End Sub

'Render this Web Part to the output parameter specified.
Protected Overrides Sub RenderWebPart _
  (ByVal output As System.Web.UI.HtmlTextWriter)
    ' Render view state controls.
    RenderChildren(output)
    ' Display message if error.
    If _msg <> "" Then
        output.Write("<br><strong>" & _msg & "</strong>")
```

```
    End If
End Sub
```

The preceding code creates a new `br` object to add a break to the controls collection. The `br` class inherits from the `Literal` class and is defined in the *Ch09Samples* project.

At runtime the web part includes a Save Settings item on its menu, as shown in Figure 9-12. If you don't want to display the checkbox, you can always set its `Visible` property to `False`.

animal 9-12. Adding a menu item to a web part

The Save Settings item appears whether or not the member is signed on. Since saving settings requires authentication, you may want to check the user's status before adding the item, as shown here:

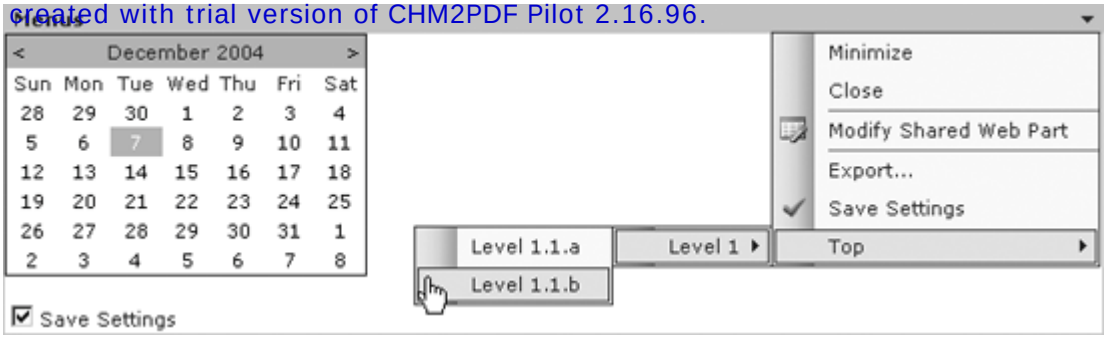
```
Public Overrides Sub CreateWebPartMenu(  )
    _mnuSave = New MenuItem("Save Settings", "mnuSave", _
      New EventHandler(AddressOf _mnuSave_Click))

    If Me.Context.User.Identity.IsAuthenticated Then _
      Me.WebPartMenu.MenuItems.Add(_mnuSave)
End Sub
```

The `MenuItems` collection `Add` method adds the item at the end of the menu. Use the `Insert`, `Remove`, and `Replace` methods to change the order of items. To create submenus, add the items to an existing item's `MenuItems` collection, as shown here:

```
Public Overrides Sub CreateWebPartMenu(  )
    ' Add multi-level menus
    Dim mnuTop As New MenuItem("Top", "mnuTop", "")
    Me.WebPartMenu.MenuItems.Add(mnuTop)
    Dim mnuL1 As New MenuItem("Level 1", "mnuL1", "")
    mnuTop.MenuItems.Add(mnuL1)
    Dim mnuL1a As New MenuItem("Level 1.1.a", "mnuL1a", "")
    mnuL1.MenuItems.Add(mnuL1a)
    Dim mnuL1b As New MenuItem("Level 1.1.b", "mnuL1b", "")
    mnuL1.MenuItems.Add(mnuL1b)
End Sub
```

The preceding code creates the multilevel menu shown in Figure 9-13.

animal 9-13. Creating multilevel menus



Menu items that include a server-side event handler cause postback events when they are clicked. Items that don't include the handler (such as Figure 9-13) don't cause postbacks.

You can call client-side scripts from a menu item by specifying a client event handler as a string in the second argument of the MenuItem constructor. For example, the following code adds an Add Numbers menu item to the client Sum web part created in the previous section, "Working on the Client Side":

```
Public Overrides Sub CreateWebPartMenu(  )
    ' Create a menu item that runs a client-side script.
    Dim mnuSum = New MenuItem("Add Numbers", _
      "return _btn_onclick(  );", "mnuSum")
    Me.WebPartMenu.MenuItems.Add(mnuSum)
 End Sub
```

9.8. Customizing the Property Task Pane

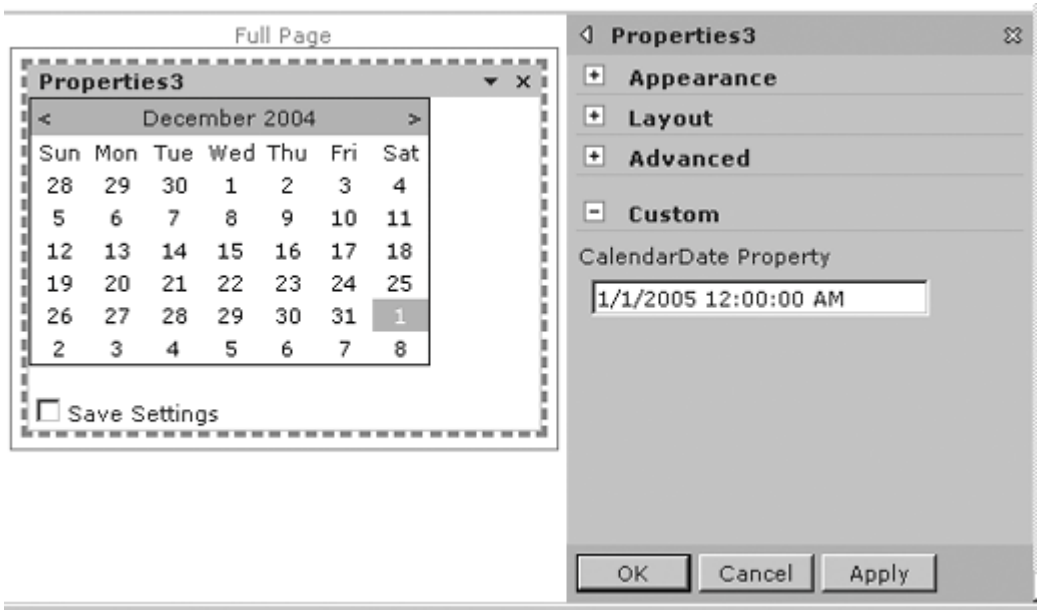
Members can change a web part by clicking the web part's down arrow and selecting Modify Shared Web Part or Modify Personal Web Part. That action displays the property task pane. You can customize a web part's property task pane by overriding the `GetToolParts` method. For example, the following code hides the built-in `AllowMinimize` property and expands the Custom section of the property task pane, as shown in Figure 9-14.

```
Public Overrides Function GetToolParts() As ToolPart(  )
    Dim toolParts(2) As ToolPart
    Dim wptp As WebPartToolPart = New WebPartToolPart
    Dim custom As CustomPropertyToolPart = New CustomPropertyToolPart
    ' Hide one of the built-in properties
    wptp.Hide(wptp.Properties.AllowMinimize)
    ' Expand the Custom section.
    custom.Expand("Custom")
    toolParts(0) = wptp
    toolParts(1) = custom
    Return toolParts
End Function
```

The `wptp` object defined in the preceding code represents the toolpart for the built-in propertiesthose properties inherited from the `WebPart` base class. The `custom` object represents the toolpart for all the properties you defined for the web part.

SharePoint generates the controls displayed in the `CustomPropertyToolPart` class, based on the data type of the custom property and whether or not it is marked as

animal 9-14. Customizing a web part's property task pane



`Browsable` in the property attributes. Table 9-4 lists the controls generated for various data types.

Table 9-4. Default CustomPropertyToolPart control types

Type	Generated control
String	Text box
Numeric	Text box
Date	Text box

Boolean	Checkbox
Array	Drop-down box
Enumeration	Drop-down listbox

You can override this behavior by creating your own toolpart class and adding it to the `toolParts` collection. To create a custom toolpart class:

1. Create a new class that inherits from the SharePoint `ToolPart` class.
2. Get a reference to the parent toolpart using `Me.ParentToolPane.SelectedWebPart` and converting (casting) the returned object to the web part's type.
3. Override `CreateChildControls` to add controls to the tool part.
4. Override `RenderToolPart` to draw the toolpart's user interface.
5. Override `ApplyChanges` and `CancelChanges` to commit the toolpart settings to the web part.

The following code illustrates these steps through a custom toolpart that allows users to change the `SelectionMode` of the Calendar control:

```
' 1) New toolpart class
Class CalToolPart
    Inherits ToolPart

    Dim _wp As Properties4
    Dim _opt As RadioButton(  )
    Dim _initialValue As Integer

    Private Sub CalToolPart_Init _
     (ByVal sender As Object, ByVal e As System.EventArgs) _
     Handles MyBase.Init
       ' 2) Get the parent web part and cast to the correct type.
      _wp = CType(Me.ParentToolPane.SelectedWebPart, Properties4)
      ' Save the inital state in case Cancelled.
      _initialValue = _wp._cal.SelectionMode
     End Sub

     Protected Overrides Sub CreateChildControls(  )
        ' 3) Create option-button controls.
        ReDim _opt(4)
        _opt(0) = New RadioButton
        _opt(0).Text = "None"
        _opt(0).GroupName = "SelectionMode"
        _opt(1) = New RadioButton
        _opt(1).Text = "Day"
        _opt(1).GroupName = "SelectionMode"
        _opt(2) = New RadioButton
        _opt(2).Text = "Week"
        _opt(2).GroupName = "SelectionMode"
        _opt(3) = New RadioButton
        _opt(3).Text = "Month"
        _opt(3).GroupName = "SelectionMode"
        ' Get the current selection mode.
        _opt(_wp._cal.SelectionMode).Checked = True
        ' Add controls to the toolpart's Controls collection.
        Controls.Add(_opt(0))
        Controls.Add(New br)
        Controls.Add(_opt(1))
        Controls.Add(New br)
        Controls.Add(_opt(2))
        Controls.Add(New br)
        Controls.Add(_opt(3))
```

```
        End Sub

        ' 4) Draw the toolpart
        Protected Overrides Sub RenderToolPart _
          (ByVal output As System.Web.UI.HtmlTextWriter)
            output.Write("Selection mode:<br>")
            renderchildren(output)
        End Sub
         ' 5) Apply the changes to the web part.
        Public Overrides Sub ApplyChanges(  )
            ' Find the selected option button
            ' and set selection mode to the index
            'of that button
            For i As Integer = 0 To 3
                If _opt(i).Checked Then _
                    _wp._cal.SelectionMode = i
             Next
         End Sub

         ' 5) (Continuted) Cancel changes and restore settings.
         Public Overrides Sub CancelChanges(  )
             ' Restore original settings.
             _wp._cal.SelectionMode = _initialValue
         End Sub

    End Class
```

In order to make the Calendar control's `SelectionMode` property available to the toolpart, I changed the scope of the control from `Private` to `Friend` in the web part class as shown here:

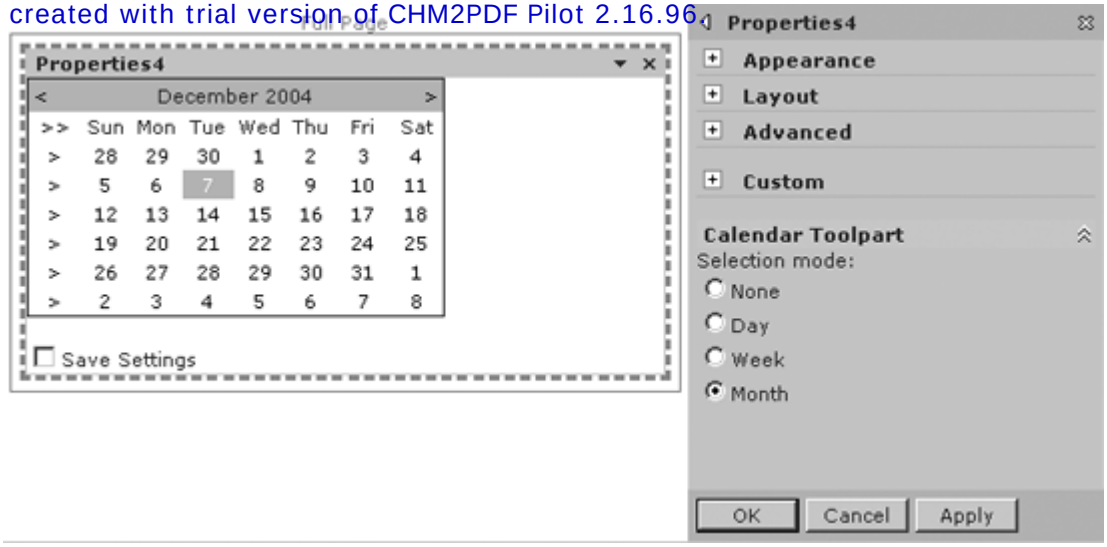
```
    ' Make child control properties available to toolpart.
    Friend WithEvents _cal As New Calendar
```

Then, to display the new custom toolpart I modified `GetToolParts` to create an instance of the new class and add the resulting object to the returned `toolParts` array:

```
    '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' Custom toolparts
    Public Overrides Function GetToolParts() As ToolPart(  )
        Dim toolParts(2) As ToolPart
        ' Toolpart for built-in properties.
        Dim wptp As New WebPartToolPart
        ' Toolpart for custom web part properties.
        Dim custom As New CustomPropertyToolPart

        ' New toolpart for Calendar control properties.
        Dim calToolPart As New CalToolPart
        calToolPart.Title = "Calendar Toolpart"
        ' Add toolparts.
        toolParts(0) = wptp
        toolParts(1) = custom
        toolParts(2) = calToolPart

        Return toolParts
    End Function
```

At runtime, the new toolpart appears in the property pane as shown in Figure 9-15. Members can change the calendar's selection mode by clicking one of the option buttons. The changes made in Figure 9-15 aren't saved after the member navigates away from the page. In order to save settings, you must serialize them by adding a `SelectionMode` property to the web part class and saving it as described earlier in "Adding Properties."

animal 9-15. Changing the calendar's selection mode through the toolpart



You also might want to hide the other toolparts. In that case, simply omit them from the `toolParts` array in `GetToolParts`. For example, the following code displays only the new toolpart:

```
Public Overrides Function GetToolParts() As ToolPart(  )
    Dim toolParts(0) As ToolPart
    ' New toolpart for Calendar control properties.
    Dim calToolPart As New CalToolPart
    calToolPart.Title = "Calendar Toolpart"
    ' Add toolparts.
    toolParts(0) = calToolPart
    Return toolParts
End Function
```

Finally, you need to be aware that causing postbacks from a toolpart redraws the entire page, discarding the changes made by the toolpart. For that reason, you can't easily use postback controls like command buttons or the Calendar control within a toolpart.

PREV NEXT

9.9. Connecting Parts

A web part that can exchange data with another web part is called *connectable*. You create a connectable part by implementing one of the SharePoint connection interfaces in your web part class. Since there are two sides to any connection, there are two interfaces for each type of connection: a provider and a consumer. The interfaces are described in Table 9-5.

Table 9-5. Web part connection interfaces

Interface pair	Use to share	Example
`ICellProvider`, `ICellConsumer`	Single item	Form web part (cell provider), Image web part (cell consumer)
`IRowProvider`, `IRowConsumer`	Row of values	List web part (row provider)
`IListProvider`, `IListConsumer`	List (table) of values	List web part (list provider)
`IFilterProvider`, `IFilterConsumer`	A list filter	Form web part (filter provider)
`IParametersInProvider`, `IParametersInConsumer`	Set of values as described by the web part	None
`IParametersOutProvider`, `IParametersOutConsumer`	Set of values as described by the web part	None

9.9.1. Creating a Simple Connection

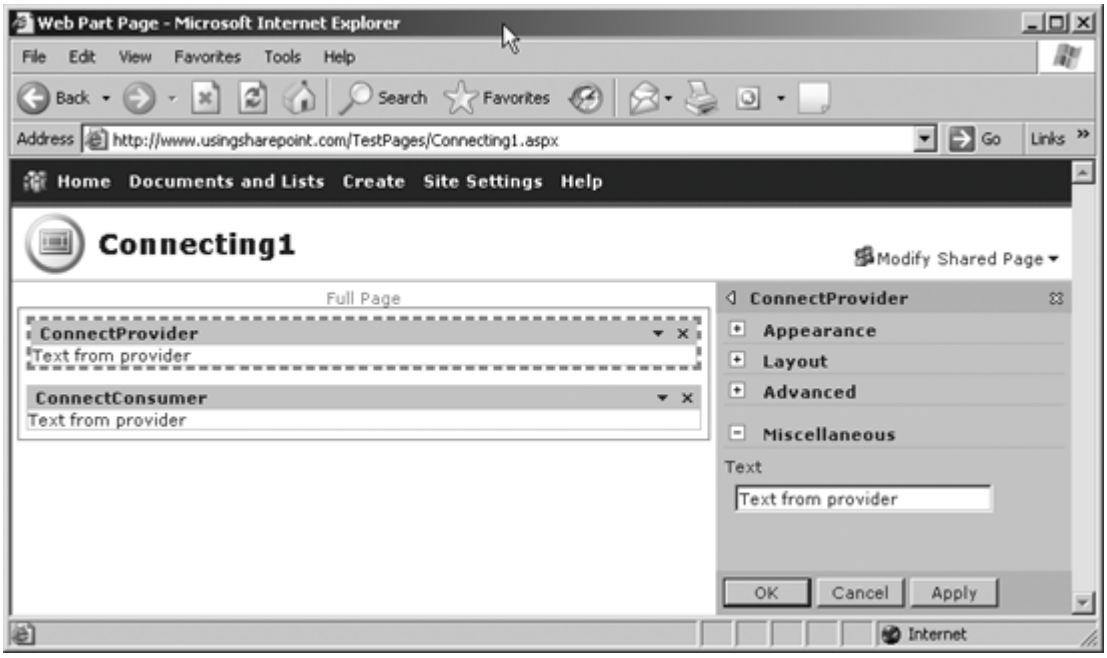
The Web Part Templates for Visual Studio includes templates for `ICellProvider` and `ICellConsumer` web parts. To add connectable web parts to an existing web part library project:

1. In Visual Studio, choose Project → Add → New Item and select the Provider Web Part template.
2. Create a *.dwp* file for the web part.
3. Repeat for the the Consumer Web Part template.
4. Build the project.
5. Create a test web page and import the two web parts onto the page.
6. Choose Modify Shared Page → Design this Page to switch into design mode.
7. Click the provider web part's drop-down menu and choose Connections → Provides a cell to → consumer web part.

The web part templates create connectable parts that share a single value through their `Text` property. Figure 9-16 shows provider and consumer web parts created from the templates running on a test page.

animal 9-16. Changing the Text property of the provider part to see the change in the

consumer part



9.10. Resources

For	Look here
Developer's Introduction to Web Parts: Sample Code	Search http://www.microsoft.com/downloads/ for "Web Parts Sample Code."
Windows SharePoint Services Software Development Kit (SDK)	Search http://www.microsoft.com/downloads/ for "SharePoint SDK."

Chapter 10. Remote Programming

Web parts use the SharePoint object model from the server side. Remote programming allows client-side applications to get and change content, create, and delete lists, and perform other tasks on the SharePoint server without installing or running new components on the server. There are several different approaches to remote programming:

- The Office 2003 object model provides SharePoint objects for workspaces, libraries, and members.
- SharePoint web services provide access to a more complete set of SharePoint tasks.
- URL commands provide a quick way to read SharePoint content in XML format using HTTP GET requests.
- Remote Procedure Call (RPC) methods allow you to get and change SharePoint content in XML format using HTTP POST requests.

This chapter compares these approaches to help you choose which to use, then demonstrates how to program SharePoint remotely using each technique.

10.1. Choosing an Approach

The approach you choose depends mainly on the type of client you want to create. Table 10-1 shows a set of recommendations for programming different types of applications.

Table 10-1. Choosing an approach based on the type of client

Type of client	Use this approach
Office application (or an application that integrates with Office 2003)	Office object model
Windows application	Web services
Non-Windows application	Web services
ASP.NET application	Web services, URL commands, or RPC methods
SharePoint web part page	URL commands or RPC methods

The last item in the table doesn't look like a remote client since the page exists on the SharePoint server. However, URL commands and RPC methods are sent from the client browser via HTTP GET or POST. The commands or methods reside on the client browser and therefore can be considered remote programming. You'll learn more about that later.

There is overlap in each of the remote programming approaches, but only web services and RPC methods offer complete sets of features. To explain the recommendations a bit more, Table 10-2 summarizes the advantages and disadvantages of each approach.

Table 10-2. Comparing remote programming approaches

Approach	Advantage	Disadvantage
Office object model	Built in to Office 2003 Available from VBA or .NET Easy to use	Limited to Office-related objects Requires Office 2003
Web services	Provides complete set of objects Available on many platforms Built-in authentication	More complicated than Office object model Adds overhead
URL commands	Easiest to use Can run from a SharePoint page	Limited to getting data Data returned in XML (sometimes an advantage) Uses GUIDs to identify lists
RPC methods	Provides access to most SharePoint tasks (similar to web services) Lower overhead than web services Can run from a SharePoint page	Difficult to learn Data returned in XML (sometimes an advantage) Uses GUIDs to identify lists Requires FormDigest control for authentication

For most programmers, the decision is simple: use the programming technique you are most familiar with and look at the other approaches when you hit a wall. For instance, Office programmers can do a lot with the Office object model, but will need to add web services in order to add attachments to a list.

10.2. Using the Office Object Model

The Office 2003 object library includes a set of objects for working with documents stored in document workspace sites and document libraries. You can use those objects from VBA, Visual Basic .NET, or C#.

The SharePoint objects are connected to the top-level Office document in each application. For example, the following lines each get a reference to the document workspace in Excel, Word, and PowerPoint, respectively, from within VBA:

```
Set wsXL = ActiveWorkbook.SharedWorkspace
Set wsWord = ActiveDocument.SharedWorkspace
Set wsPPT = ActivePresentation.SharedWorkspace
```

From the document object, the SharePoint objects are organized as shown in Figure 10-1.

animal 10-1. Office objects for SharePoint Services

10.2.1. From VBA

Use the `SharedWorkspace` object to create, get, or delete a SharePoint document workspace site. The technique is the same for any of the supported document types. For example, to create a new workspace:

1. Save the document.
2. Call the **`SharedWorkspace`** object's **`CreateNew`** method to create the document workspace.
3. Use the document's **`SharedWorkspace`** object to add members, tasks, links, files, or folders.

Chapter 5 demonstrates performing a wide range of SharePoint tasks with an Excel workbook using the `SharedWorkspace` object in VBA. It also shows how to share lists between Excel and SharePoint through VBA. Table 10-3 lists frequently asked questions to provide a quick reference for VBA programming.

Table 10-3. SharePoint VBA quick reference

How do I?	Do this
Open a document library file?	Use the `Open` method and specify the document's web address.
Add a file to a document library?	Use the `SaveAs` method and save to the document library's web address.
Display the document's site?	Use the `FollowHyperLink` method to open *document*`.SharedWorkspace.Url`.
Include multiple documents in a shared workspace?	Open a document from the workspace and use *document*`.SharedWorkspace.Files`. `Add` to add the files.
Send mail to site members?	Use the `Member` object's `Email` property to get the address; then use `FollowHyperLink("mailto:"` & *address*`)`.
Delete a workspace?	Notify workspace members; then use *document*`.SharedWorkspace.Delete`.
Remove a file?	Get the file from the `Files` collection; then use *file*`.Delete`.
Respond to update events?	Write code for the document class's `Sync` event.

10.2.2. From .NET

Office 2003 Professional provides *primary interop assemblies* (PIAs) to allow you to use the Office object libraries from Visual Basic .NET and C#. These assemblies are installed in your *global assembly cache* (GAC) when you install Office 2003 Professional with .NET Programmability selected (which is the default).

To use the PIAs from Visual Studio .NET:

1. Add references to the Microsoft Office 11.0 Object Library, Microsoft Excel 11.0 Object Library, and so on. The libraries are found on the COM tab of the References dialog.

2. Include **`Imports`**/**`using`** declarations for the **`Office.Interop`** and **`Office.Core`** namespaces.

3. Use the object model to get a reference to a document object.

The following .NET console application demonstrates using the Office object model to create a document workspace; add members and tasks; send mail to members; and delete the workspace:

```
Imports Microsoft.Office.Interop
Imports Microsoft.Office.Core

Imports System.Reflection

' Simple demo of using Office SharePoint objects from .NET.
Module Module1
    ' Address of SharePoint site.
    Const SPSite As String = "http://wombat1/"

    ' From Microsoft.Interop namespace.
    Dim WithEvents _wb As Excel.Workbook
    ' From Microsoft.Office.Core namespace.
    Dim _sp As SharedWorkspace

    Sub Main(  )
        OpenWorkbook(  )
        GetWorkspace(  )
        AddMember(  )
        AddTask(  )
        SendMail(  )
        DeleteWorkspace(  )
    End Sub

    Sub OpenWorkbook(  )
        ' Start  Excel
        Dim xl As Excel.Application = GetObject(, "Excel.Application")
        xl.Visible = True
        ' Get the workbook object.
```

```
            ChDir(".")
            _wb = xl.Workbooks.Open(CurDir(  ) & "\xlSPDemo1.xls")
        End Sub

        Sub GetWorkspace(  )
            ' Check if workspace exists; if it doesn't,
            ' create it.
            Try
                Debug.Write(_wb.SharedWorkspace.URL)
            Catch ex As Exception
                _wb.SharedWorkspace.CreateNew(SPSite, "xlSPDemo")
            End Try
            ' Get the workspace object.
            _sp = _wb.SharedWorkspace
        End Sub

        Sub AddMember(  )
            ' Note: 'wombat1\ExcelDemo' must be a valid account.
            _sp.Members.Add("ExcelDemo@hotmail.com", "wombat1\ExcelDemo", _
              "Excel Demo", "Contributor")
        End Sub

        Sub AddTask(  )
            ' Note: 'wombat1\ExcelDemo' must be a valid account.
            _sp.Tasks.Add("Task1", _
              MsoSharedWorkspaceTaskStatus.msoSharedWorkspaceTaskStatusInProgress, _
              MsoSharedWorkspaceTaskPriority.msoSharedWorkspaceTaskPriorityHigh, _
              "wombat1\ExcelDemo", _
              "Some task", _
              Today)
        End Sub

        Sub SendMail(  )
            Dim toAddress As String
            ' Build address string.
            For Each mem As SharedWorkspaceMember In _sp.Members
                toAddress = toAddress & mem.Email & ";"
            Next
            ' Send mail from client.
            _wb.FollowHyperlink("mailto:" & toAddress & _
              "?Subject=Deleting " & _sp.URL)
        End Sub

        Sub DeleteWorkspace(  )
            ' Delete the workspace.
            _sp.Delete(  )
        End Sub

    End Module
```

There are some tricks in the preceding code that merit some explanation. I use `GetObject` to connect to a running instance of Excel if it is already loaded. If Excel is not running, `GetObject` starts Excel. Using `CreateObject` or `Dim As New` always starts a new instance of Excel, which uses a lot of memory. I also use a `TRy/Catch` block to see if the workbook is already shared. The way the PIAs work, you can't use `IsNothing` to test whether the object exists.

You can't get help on Office objects from within Visual Studio. It's a good idea to create a shortcut to the Office VBA help files listed in Table 10-4 and open those files manually when you need reference information on Office objects.

Table 10-4. Help files for SharePoint and Office models

For help on	Open this file
SharePoint and other core Office objects	*C:\Program Files\Microsoft Office\OFFICE11\1033\VBAOF11.CHM*

Excel objects	*C:\Program Files\Microsoft Office\OFFICE11\1033\VBAXL10.CHM*
Word objects	*C:\Program Files\Microsoft Office\OFFICE11\1033\VBAWD10.CHM*
PowerPoint objects	*C:\Program Files\Microsoft Office\OFFICE11\1033\VBAPP10.CHM*

Microsoft sells a programming toolkit for working with Office 2003 from .NET called Visual Studio Tools for the Microsoft Office System (VSTO for short). VSTO provides project templates for creating document-based applications in .NET.

VSTO applications offer two key advantages over VBA-based applications. First, the application assemblies can be deployed to a trusted network address, and so can be maintained from a single location, allowing users to automatically get the latest release without having to install the assembly locally. Second, using .NET assemblies rather than VBA allows you to lock down macro security in Office, prohibiting users from running macros while still allowing automation.

However, there are also some disadvantages of using VSTO over VBA. VSTO only supports Office 2003 applications, and performance is generally slower since object access is through the .NET PIAs rather than a more direct path through COM.

PREV NEXT

10.3. Using Web Services

SharePoint web services allow remote applications to get and modify SharePoint sites and content. These services are more complete than the VBA object model and can be used from any web-service aware programming language such as Visual Basic .NET, C#, C++, VBA, Java, and so on.

Table 10-5 lists the web services SharePoint provides. All services except *Admin.asmx* are installed in the *_vti_bin* virtual folder, which maps to *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\ISAPI* under the default SharePoint installation.

Table 10-5. SharePoint web services

Service	Use to
Administration (*Admin.asmx*)	Manage SharePoint sites (for example, create or delete sites). This service is only installed for the SharePoint Central Administration site in the *_vti_adm* folder.
Alerts (*Alerts.asmx*)	Get or delete alerts on a site.
Document Workspace (*Dws.asmx*)	Manage document workspace sites and the data they contain.
Forms (*Forms.asmx*)	Get forms used in the user interface when working with the contents of a list.
Imaging (*Imaging.asmx*)	Create and manage picture libraries.
List Data Retrieval (*DspSts.asmx*)	Perform queries against lists.
Lists (*Lists.asmx*)	Work with lists and list data.
Meetings (*Meetings.asmx*)	Create and manage meeting workspace sites.
Permissions (*Permissions.asmx*)	Work with the permissions for a site or list.
Site Data (*SiteData.asmx*)	Get metadata or list data from sites or lists.
Sites (*Sites.asmx*)	Get information about the site templates for a site collection.
Users and Groups (*UserGroup.asmx*)	Work with users, site groups, and cross-site groups.
Versions (*Versions.asmx*)	Work with versions within a document library.
Views (*Views.asmx*)	Work with views of lists.
Web Part Pages (*WebPartPages.asmx*)	Get web part pages; get, add, delete, or change web parts.
Webs (*Webs.asmx*)	Work with sites and subsites.

Detailed reference information about these web services and their methods is available in the SharePoint SDK.

10.3.1. From VBA

The Office 2003 object library provides objects for working with SharePoint document workspaces and document libraries. In general, you'll use those objects when programming SharePoint from VBA. However, there are some cases where you may need to use the SharePoint web services to access lower-level tasks not available through the object library.

To use a web service from VBA:

1. Install the Web Services Toolkit from Microsoft at www.microsoft.com/downloads.

2. Close and restart any running Office applications

3. Open the Visual Basic Editor and select Web References from the Tools menu. Visual Basic displays the Microsoft Office 2003 Web Services Toolkit references dialog (Figure 10-2)

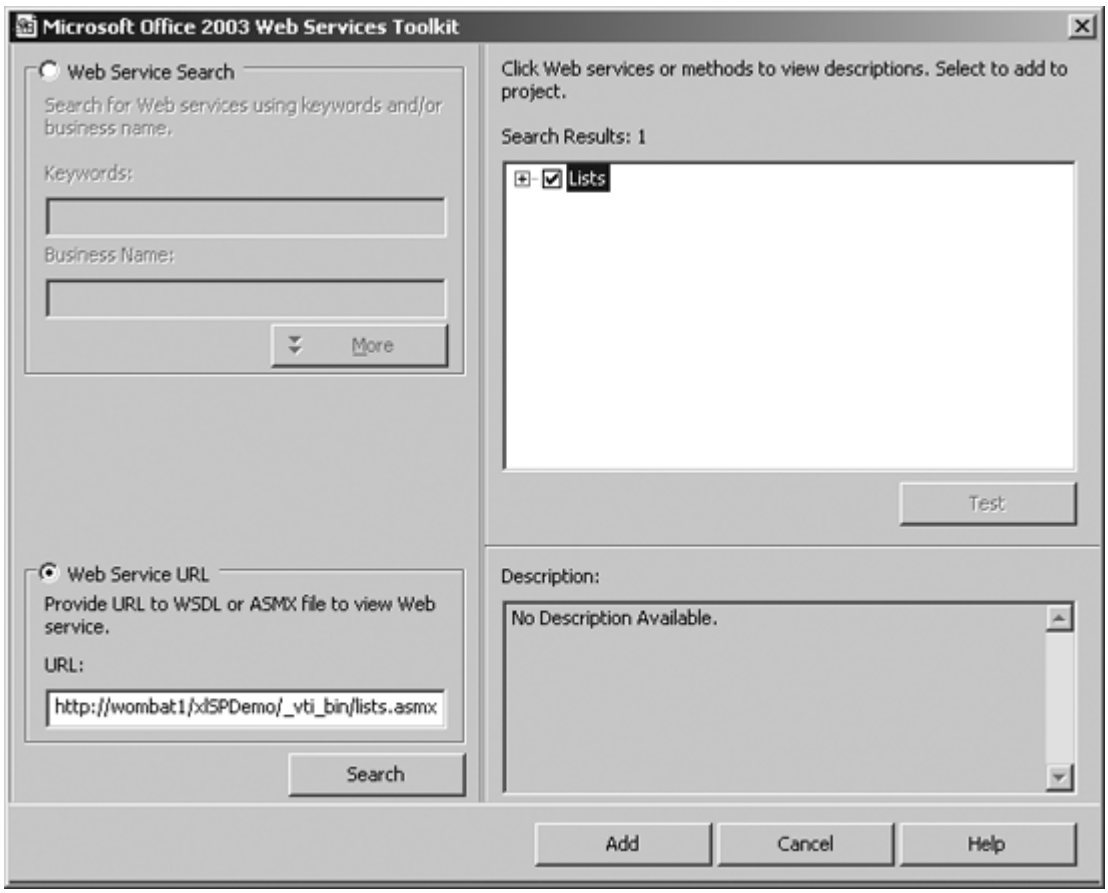
4. Select the Web Service URL option button and type the address of the web service using this form: *http://sharepointURL/_vti_bin/service.asmx* (for example, **http://wombat1/xlSPDemo/_vti_bin/Lists.asmx)**.

5. Click Search. The Toolkit should find the web service and display it in the Search Results list.

6. Select the checkbox beside the service in Search Results and click Add.

7. The Toolkit generates a proxy class named **clsws_***Service* and adds it to the current project.

animal 10-2. Adding a web reference in VBA



The SharePoint server must authenticate the user before you can call any of the web service methods. If the user has not been authenticated, a "Maximum retry on connection exceeded" error occurs. In Visual Basic .NET or C# .NET, you authenticate the user from code by creating a `Credentials` object for the user. For example, the following .NET code passes the user's default credentials to a web service:

```
wsAdapter.Credentials = System.Net.CredentialCache.DefaultCredentials
```

Unfortunately, you can't do that directly in VBA. Instead, you must use one of the following techniques to connect to the SharePoint server through the Office application:

- Update or refresh a document that is shared on the server.
- Insert an existing SharePoint list on an Excel worksheet. This can even be a dummy list placed on the server solely for the purpose of establishing connections.
- Navigate to the SharePoint site in code.

Any of these techniques displays SharePoint's authentication dialog box and establishes a user session for the Office application. Afterward, you can call web service methods, and they will be authorized using the current session.

10.3.2. VBA Programming Tips

The generated proxy classes hard-code the site address as a constant in the generated proxy class modules:

```
Private Const c_WSDL_URL As String = _
  "http://wombat1/xlSPDemo/_vti_bin/lists.asmx?wsdl"
```

You can change the `c_WSDL_URL` constant to target other sites or subsites. SharePoint creates a *_vti_bin* virtual folder for any new web site. That folder mirrors *_vti_bin* at the top-level site.

One thing you will notice quickly when using the generated proxy classes is that the error reporting is minimal. When a method fails on the server side, you receive only a general error. To receive more detail, change the proxy class's error handler. The following code shows how to add details to the Lists web service error handler:

```
Private Sub ListsErrorHandler(str_Function As String)
    If sc_Lists.FaultCode <> "" Then
        Err.Raise vbObjectError, str_Function, sc_Lists.FaultString & _
      vbCrLf & sc_Lists.Detail ' Add detail
    'Non SOAP Error
    Else
         Err.Raise Err.Number, str_Function, Err.Description
    End If
 End Sub
```

10.3.3. Working with Lists

One of the most useful scenarios for using web services in VBA is calling the Lists web service from Excel. The Lists web service lets you perform tasks on the server that you cannot otherwise perform through Excel objects. Specifically, you can use the List web service to:

- Add an attachment to a row in a list.
- Retrieve an attachment from a row in a list.
- Delete an attachment.
- Delete a list from a SharePoint server.
- Perform queries.

The following sections demonstrate how to perform those tasks in VBA using the Lists web service.

10.3.3.1 Adding attachments

Use the Lists web service `AddAttachment` method to add a file attachment to a row in a list; then use `GetAttachmentCollection` to retrieve attachments from within Excel. For example, the following code attaches the image file *joey.jpg* to the second row of a shared list:

```
' Requires Web reference to SharePoint Lists.asmx
Dim lws As New clsws_Lists, src As String
src = ThisWorkbook.Path & "\joey.jpg"
dest = lws.wsm_AddAttachment("Excel Objects", "2", "joey.jpg", FileToByte(src))
```

The `AddAttachment` method's last argument is an array of bytes containing the data to attach. To convert the image file to an array of bytes, the preceding code uses the following helper function:

```
Function FileToByte(fname As String) As Byte(  )
    Dim fnum As Integer
    fnum = FreeFile
    On Error GoTo FileErr
    Open fname For Binary Access Read As fnum
    On Error GoTo 0
    Dim byt(  ) As Byte
    ReDim byt(LOF(fnum) - 1)
    byt = InputB(LOF(fnum), 1)
    Close fnum
    FileToByte = byt
    Exit Function
FileErr:
    MsgBox "File error: " & Err.Description
End Function
```

10.3.3.2 Retrieving attachments

Use the Lists web service `GetAttachmentCollection` method to retrieve an attachment from a list. The `GetAttachmentCollection` method returns an XML node list that contains information about each attachment for the row. The following code retrieves the location of the file attached in the previous section:

```
Dim lws As New clsws_Lists ' Requires Web reference to SharePoint Lists.asmx
Dim xn As IXMLDOMNodeList  ' Requires reference to Microsoft XML
Set xn = lws.wsm_GetAttachmentCollection("Excel Objects", "2")
ThisWorkbook.FollowHyperlink (xn.Item(0).Text)
```

Notice that the returned XML node list is a collection since rows can have multiple attachments. Since the preceding example only attached one file, this sample simply retrieves the first item from the node list. The `Text` property of this item is the address of the attachment on the SharePoint server.

10.3.3.3 Deleting attachments

Finally, it is very simple to delete an attachment using the `DeleteAttachment` method:

```
Dim lws As New clsws_Lists  ' Requires Web reference to SharePoint Lists.asmx
lws.wsm_DeleteAttachment "Excel Objects", "2", _
SPSITE & "/Lists/Excel Objects/Attachments/2/joey.jpg"
```

Since `DeleteAttachment` requires the fully qualified address of the attachment, it is useful to save the address of each attachment somewhere on the worksheet or to create a helper function to retrieve the address from the SharePoint server, as shown here:

```
Function GetAttachment(ListName As String, ID As String) As String
    Dim lws As New clsws_Lists ' Requires Web reference to SharePoint Lists.asmx
    Dim xn As IXMLDOMNodeList  ' Requires reference to Microsoft XML
    Set xn = lws.wsm_GetAttachmentCollection(ListName, ID)
    GetAttachment = xn.Item(0).Text
End Function
```

10.3.3.4 Performing queries

You don't commonly need to perform queries through the Lists web service. Most of the operations you want to perform on the list data are handled through the Excel interface or through the Excel list objects.

However, advanced applicationsor especially ambitious programmersmay use the Lists web service to exchange XML data directly with SharePoint. For instance, you may want to retrieve a limited number of rows from a very large shared list. In this case, you can perform a query directly on the SharePoint list using the `GetListItems` method. For example, the following code gets the first 100 rows from a shared list:

```
Dim lws As New clsws_Lists  ' Requires Web reference to SharePoint Lists.asmx
Dim xn As IXMLDOMNodeList  ' Requires reference to Microsoft XML
Dim query As IXMLDOMNodeList
Dim viewFields As IXMLDOMNodeList
Dim rowLimit As String
Dim queryOptions As IXMLDOMNodeList
rowLimit = "100"
Dim xdoc As New DOMDocument
xdoc.LoadXml ("<Document><Query /><ViewFields />" & _
  "<QueryOptions /></Document>")
Set query = xdoc.getElementsByTagName("Query")
Set viewFields = xdoc.getElementsByTagName("Fields")
Set queryOptions = xdoc.getElementsByTagName("QueryOptions")
Set xn = lws.wsm_GetListItems("Shared Documents", "", query, _
  viewFields, rowLimit, queryOptions)
```

The results are returned as XML. To see them, you can simply display the root node of the returned object as shown here:

```
Debug.Print xn.Item(0).xml
```

The key to the preceding query is XML supplied the `LoadXml` method. You create conditional queries using the Query element and determine the columns included in the results using the `ViewFields` element. Perhaps the simplest way to create these queries is to write them as a text file in an XML editor (or Notepad), then load them from that file using the `Load` method shown here:

```
xdoc.Load ("query.xml")
```

The query file takes this form:

```
 <Document>
<Query>
   <OrderBy>
       <FieldRef Name="Column 1" Asending="FALSE"/>
    </OrderBy>
   <Where>
      <Gt>
         <FieldRef Name="_x0031_" />
         <Value Type="Value">6</Value>
      </Gt>
   </Where>
</Query>
<ViewFields>
       <FieldRef Name="ID" />
       <FieldRef Name="_x0031_" />
       <FieldRef Name="_x0032_" />
       <FieldRef Name="_x0033_" />
</ViewFields>
<QueryOptions>
   <DateInUtc>FALSE</DateInUtc>
   <Folder />
   <Paging />
   <IncludeMandatoryColumns>FALSE</IncludeMandatoryColumns>
   <MeetingInstanceId />
   <ViewAttributes Scope="Recursive" />
   <RecurrenceOrderBy />
   <RowLimit />
   <ViewAttributes />
   <ViewXml />
</QueryOptions>
</Document>
```

Notice that the FieldRef elements sometimes use the internal SharePoint names to identify columnslists don't always use the titles displayed in the columns as column names. You can get the internal column names by examining the list's XML. To see the list's XML, use the GetList method, as shown here:

```
Dim lws As New clsws_Lists
Dim xn As IXMLDOMNodeList ' Requires reference to Microsoft XML
Set xn = lws.wsm_GetList ("Shared Documents")
Debug.Print xn(0).xml
```

10.3.4. From .NET

Working with SharePoint web services from Visual Studio .NET is much the same as working from VBA, except you are using a somewhat different language (VB.NET or C#) and you use the .NET Framework objects rather than the Microsoft Soap and XML type libraries.

To use a web service from Visual Studio .NET:

1. Choose Project → Add web reference, enter the URL of the web service, and choose Go. Visual Studio displays Figure 10-3.

2. Choose Add Reference to add the reference. Visual Studio adds a namespace containing the class provided by the web service. The format is *server.class*. For example:

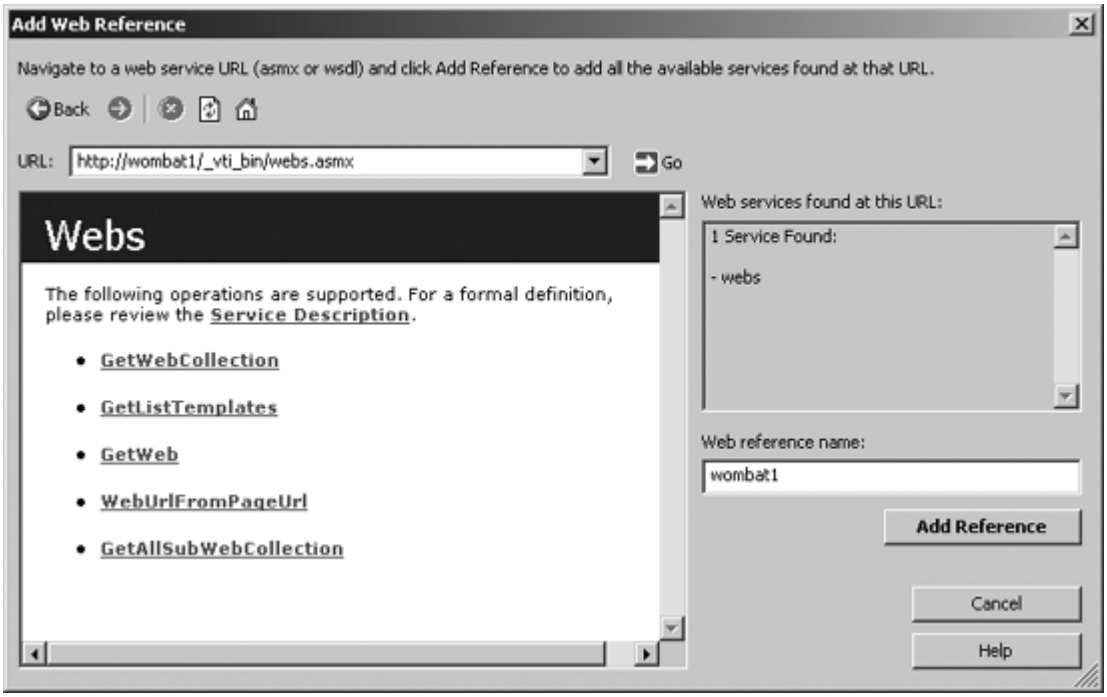
```
wombat1.Webs
```

3. Create a new object from the class.

4. Set the object's **Credentials** property to the user's default credentials.

5. Call the web service methods from the object.

animal 10-3. Adding a web reference in .NET



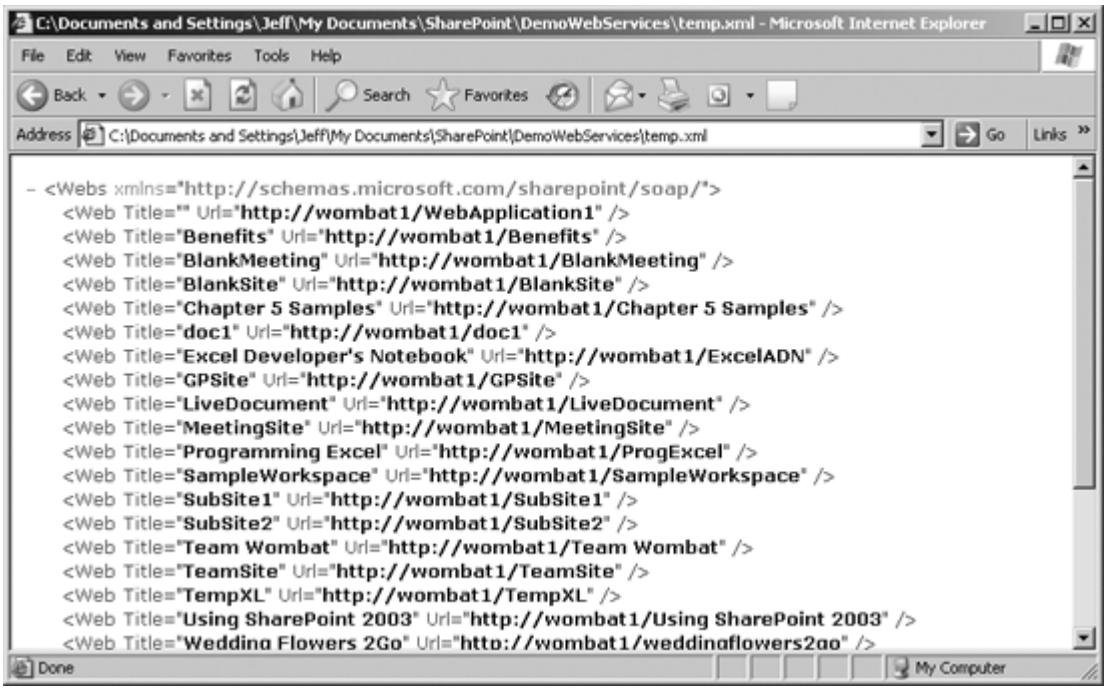
For example, here is a very simple console application that gets all of the web sites on the SharePoint server and displays their descriptions:

```
Module Module1
    Sub Main(  )
        ' Create a new object from the web service.
```

```
            Dim webs As New wombat1.webs
            ' Pass the service the default credentials.
            webs.Credentials = System.Net.CredentialCache.DefaultCredentials
            Dim xdoc As New Xml.XmlDocument
            ' Load the results in an XML document.
            xdoc.LoadXml(webs.GetWebCollection.OuterXml)
            ' Save the results
            xdoc.Save("..\temp.xml")
            ' Display results in the default XML editor.
            Process.Start("..\temp.xml")
        End Sub
    End Module
```

The last line displays the resulting file, using whatever editor is registered on your system for displaying XML. On my system, that is Internet Explorer, which displays the result as shown in Figure 10-4.

animal 10-4. Displaying the result of the web service



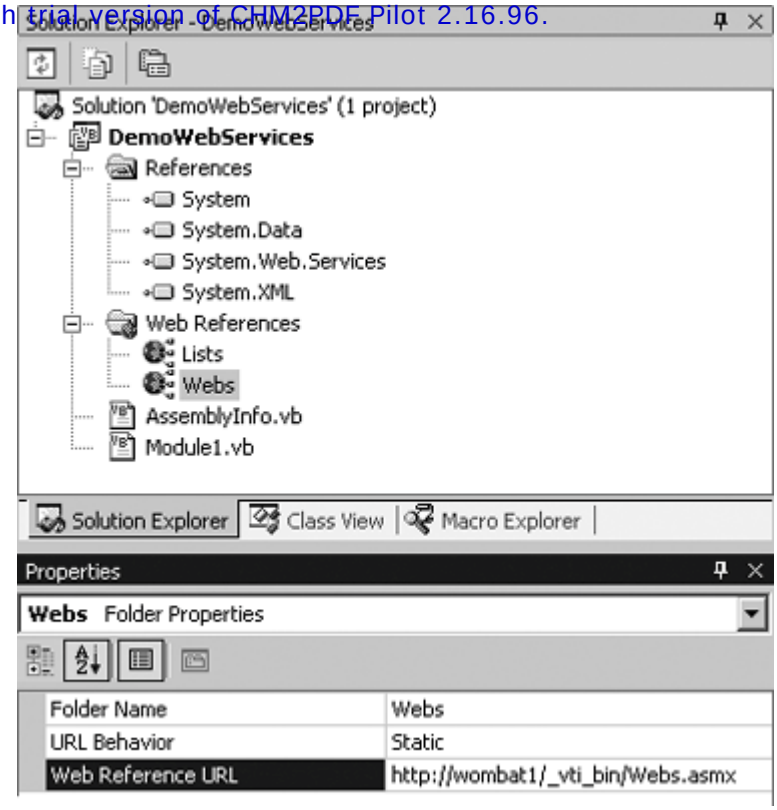
`Process.Start` is the .NET equivalent of `Shell`; using it to open documents in their default editors is a handy trick to know.

10.3.5. .NET Programming Tips

You can change the site that the web service acts on by changing the Web Reference URL property for the web service. You can also rename the namespace so it is more descriptive.

To set the web reference properties, select the item under Web References in the Solution Explorer. Then set the properties in the properties window as shown in Figure 10-5.

animal 10-5. Changing web reference properties

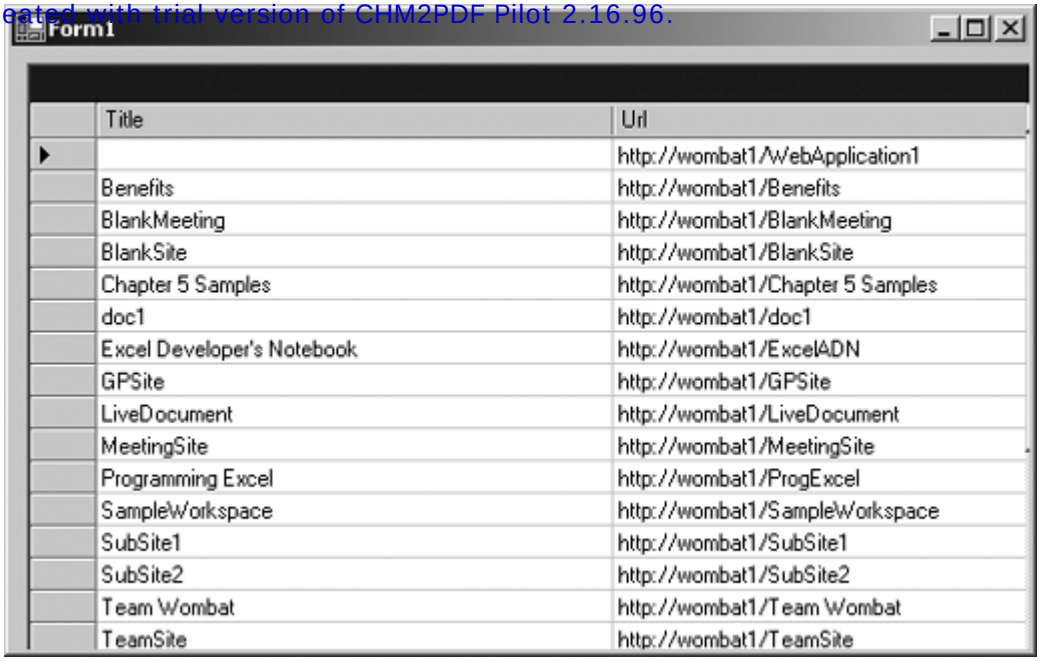


The .NET Framework contains many useful classes for working with XML and integrates its database support with XML. For instance, you can load XML directly into a `DataSet` and then use that object to easily iterate over elements or load the items into a data grid. The following code shows how to display the results from a web service in a data grid on a Windows form:

```
Private Sub Form1_Load(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles MyBase.Load
    ' Create a new object from the web service.
    Dim webs As New wombat1.Webs
    Dim ds As New DataSet, str As String
    ' Pass the service the default credentials.
    webs.Credentials = System.Net.CredentialCache.DefaultCredentials
    ' Get the list of sites.
    str = webs.GetWebCollection.OuterXml( )
    ' Read the string into a stream
    Dim sr As New IO.StringReader(str)
    ds.ReadXml(sr)
    ' Display results.
    dGrid.DataSource = ds.Tables(0)
    sr.Close( )
End Sub

Private Sub Form1_Resize(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles MyBase.Resize
    dGrid.PreferredColumnWidth = Me.Width / 2
    dGrid.Width = Me.Width
    dGrid.Height = Me.Height
End Sub
```

Figure 10-6 illustrates the results from this very simple program.

animal 10-6. Quick results from using data sets with XML



Often in .NET you can assemble a few very powerful classes to create quick results with a few lines of code. In fact, `Form1_Load` is deliberately verbose in order to separate the steps. Here's how the code would more typically be written:

```
Private Sub Form1_Load(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles MyBase.Load
    ' Create a new object from the web service.
    Dim webs As New wombat1.Webs, ds As New DataSet
    ' Pass the service the default credentials.
    webs.Credentials = System.Net.CredentialCache.DefaultCredentials
    ' Shortened form of load/read.
    ds.ReadXml(New IO.StringReader(webs.GetWebCollection.OuterXml(  )))
    ' Display results.
    dGrid.DataSource = ds.Tables(0)
End Sub
```

10.3.6. From ASP.NET

When using SharePoint web services from ASP.NET applications on other servers, you must still provide valid credentials of authorized users. In most cases, you'll want to do that by impersonating the current user and passing his credentials to the service. To impersonate the current user in an ASP.NET web application, change the *Web.config* of the application as shown here:

```
<authentication mode="Windows" />
<authorization>
    <deny users="?" /> <!-- Deny unauthenticated users -->
</authorization>
<identity impersonate="true" />
```

These settings enforce Windows authentication and then run the ASP.NET code under the current user's identity. You can use other authentication and authorization techniques, but the concept is the samethe code invoking the web service must impersonate a user authorized to make the request on the SharePoint server.

Once the code is impersonating an authorized user, invoking the web service is basically the same as shown previously with the addition that the DataGrid web control requires a `DataBind` method to load data:

```
Dim ds as DataSet

Private Sub Page_Load(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles MyBase.Load
    ' Create a new object from the web service.
```

```
    Dim webs As New wombat1.Webs
    ' Pass the service the default credentials.
    webs.Credentials = System.Net.CredentialCache.DefaultCredentials
    ' Shortened form of load/read.
    ds.ReadXml(New IO.StringReader(webs.GetWebCollection.OuterXml(  )))
    ' Display results.
    dGrid.DataSource = ds.Tables(0)
    ' Bind data (required for DataGrid web control).
    dGrid.DataBind(  )
End Sub
```

10.3.7. Using the Admin Service

The *Admin.asmx* web service is not installed on the root SharePoint site, but rather is provided as part of the Central Administration site which SharePoint installs on a separate port number. To create a web reference for this service, find the port number for the Central Administration site. To do that, open the site in the browser and record the port number used in the Address bar. Then specify the port number and the *_vti_adm* folder in the URL of the web reference. For example:

```
http://wombat1:2933/_vti_adm/Admin.asmx
```

The Admin service provides two main methods: `CreateSite` and `DeleteSite`. These methods require administrative privileges to use, but are otherwise straightforward. For example, the following code adds a Delete site feature to the previous ASP.NET data grid example:

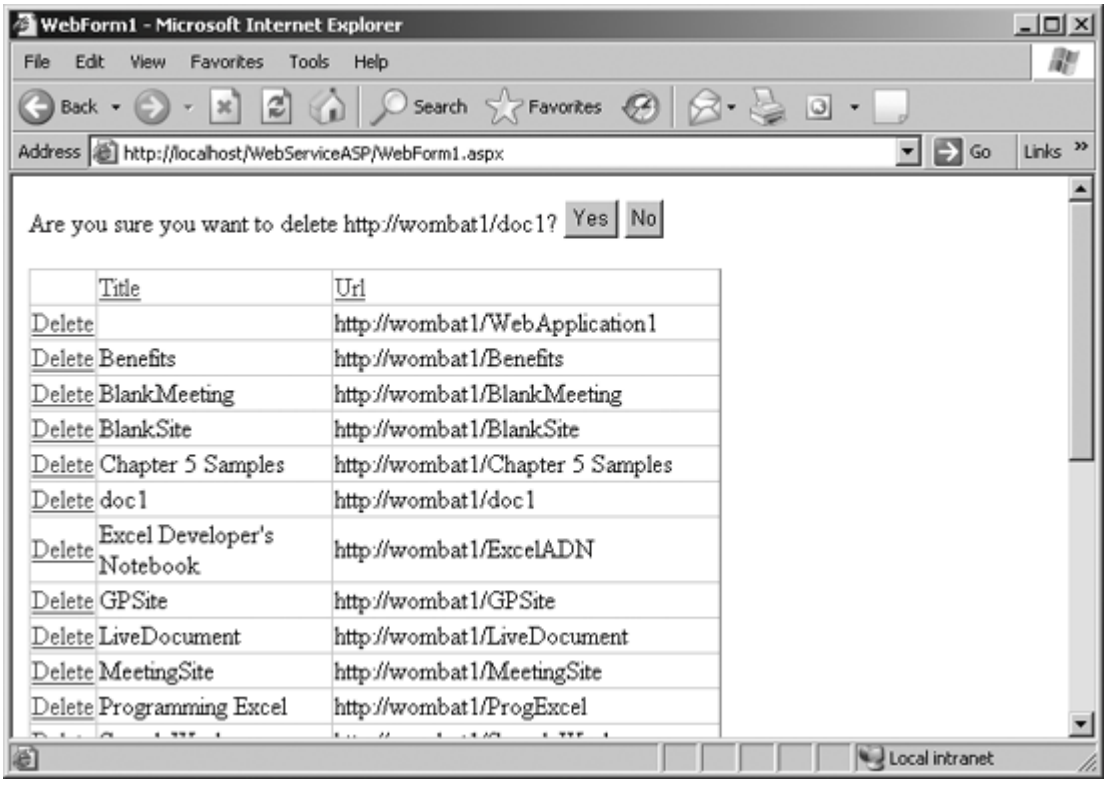
```
Private Sub dGrid_DeleteCommand(ByVal source As Object, _
  ByVal e As System.Web.UI.WebControls.DataGridCommandEventArgs) _
  Handles dGrid.DeleteCommand
    ' Display the delete panel.
    pnl.Visible = True
    ' Show the site to delete.
    lbl.Text = ds.Tables(0).Rows(e.Item.ItemIndex).Item(1)
End Sub

Private Sub cmdDelete_Click(ByVal sender As System.Object, _
  ByVal e As System.EventArgs)
    ' Create admin object.
    Dim adm As New wombat1Adm.Admin
    ' If an item is selected
    If lbl.Text <> "" Then
        Try
            ' Delete the site.
            adm.DeleteSite(lbl.Text)
            ' Display success.
            status.text = lbl.Text & " deleted."
        Catch ex As Exception
            ' Otherwise, note the error.
            status.text = "Error: " & ex.Message
        End Try
    End If
    ' Hide panel.
    pnl.Visible = False
End Sub

Private Sub cmdNo_Click(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles cmdNo.Click
    ' Clear the site and hide the panel.
    lbl.Text = ""
    pnl.Visible = False
End Sub
```

In addition to the data grid, the preceding code uses some controls defined in the following ASP.NET web form:

```
    <form id="Form1" method="post" runat="server">
        <asp:Panel id="pnl" runat="server" Visible="False">
          Are you sure you want to delete
        <asp:Label id="lbl" runat="server"></asp:Label>?
        <asp:Button id="cmdYes" runat="server" Text="Yes"></asp:Button>
        <asp:Button id="cmdNo" runat="server" Text="No"></asp:Button></asp:Panel>
        <asp:Label id="status" runat="server"></asp:Label><br>
        <asp:DataGrid id="dGrid" runat="server" Width="432px" AllowSorting="True">
            <Columns>
                <asp:ButtonColumn Text="Delete"
                  CommandName="Delete"></asp:ButtonColumn>
            </Columns>
        </asp:DataGrid>
    </form>
```

At runtime, the user can select a site from the grid and delete it as shown in Figure 10-7.

animal 10-7. Using the Admin service to delete a site



10.4. Using URL Commands

URL commands get XML results from a SharePoint server through HTTP GET requests. You invoke the commands by specifying them as query strings in an address that has this form:

```
http://server/subsite/_vti_bin/owssvr.dll?Cmd=cmdname&param=value&param=value ...
```

URL commands don't alter the content database, so they don't require authentication. The RPC protocol provides a similar approach, but uses HTTP POST instead of GET, and so can include authentication information and thus change content. Table 10-6 lists the commands that can be invoked through HTTP GET.

The List and View parameters in Table 10-6 are specified as GUIDs. See the following section, "Getting GUIDs," for how to get those values from the list or view names. The URL commands also accept the set of optional parameters in Table 10-7 to modify the returned results.

Table 10-6. SharePoint URL commands

Command	Parameters	Use to
`Dialogview`	`dialogview`, `location`, `FileDialogFilterValue`	Open the view used in a dialog box for opening or saving files to a document library; or open the custom property form that is used when saving a file to a document library.
`Display`	`List`, `XMLDATA`	Run a database query against the list and return XML or HTML.
`DisplayPost`	`PostBody`, `URLBase`	Render the Collaborative Application Markup Language (CAML) assigned to the PostBody parameter.
`ExportList`	`List`	Export in CAML format the schema of the list.
`GetProjSchema`		Request the XML schema for a web site.
`GetUsageBlob`	`BlobType`	Get information about web site usage.
`HitCounter`	`Page`, `Image`, `Custom`, `Digits`	Render a hit counter in an `img` element on a page.
`RenderView`	`List`, `View`, `URLBase`	Request the contents of a view for the list.

Table 10-7. URL optional parameters

Parameter	Use to
`FileDialogFilterValue`	Set filters for a view, and to return the list of all files of a specified type from a document library according to filename extension (for example, *.*doc*, *.*ppt*, or *.*xls*).
`FilterField`*n*	Specify the name of a field in the database, where *n* is an integer that is limited only by the number of fields allowed in the database table or by the length allowed for the URL field.
`FilterValue`*n*	Specify the string value on which to filter a field, where *n* is an integer that is limited only by the length allowed for the URL field.
`SortField`	Specify the name of the field on which to sort.
`SortDir`	Indicate an ascending (asc) or descending (desc) sort order.
`Using`	Specify a particular file containing CAML for the server to evaluate and render.

The HitCounter command is special. It renders an image used in an `img` element as shown here:

```
<img src= "../_vti_bin/owssvr.dll?Cmd=HitCounter&Page=TestPages/URLDemo.aspx&Image=4&Digits=3">
```

10.4.1. Getting GUIDs

The List and View parameters of the URL commands are the unique identifiers (GUIDs) SharePoint uses internally. There's no URL command to get a GUID from a name, so you have to use the SharePoint object model or a web service to get the values. The following sections demonstrate how to get the List and View GUIDs in these three different scenarios:

- Server .NET code in a web part using the SharePoint object model
- Remote .NET code using web services
- Remote VBA code using web services

The return values of all the procedures are the same, but as you will see from the code, how you get the results varies.

10.4.1.1 Using SharePoint objects

If you're working within SharePoint, you can get GUIDs through the SharePoint object model. The following procedures get a reference to the current web using the `GetContextWeb` shared method, then use that object to get the list or view by name from the `Lists` or `Views` collections. The `ID` property is a GUID, which must be converted to a string in order to be used from the URL command:

```
' Requires: Imports Microsoft.SharePoint.Webcontrols
Function GetListGuid(ByVal lName As String) As String
    Try
        ' Get the web from the current context.
        Dim web As SPWeb = SPControl.GetContextWeb(context)
        ' Get the list by name.
        Dim lst As SPList = web.Lists(lName)
        ' Get the GUID of the list.
        Dim guid As System.Guid = lst.ID
        ' Format the GUID as a string.
        Return ("{" & guid.ToString & "}")
    Catch ex As Exception
        Debug.Write(ex.Message)
        Return ""
    End Try
End Function

Function GetViewGuid(ByVal lName As String, _
  ByVal vName As String) As String
    Try
        ' Get the web from the current context.
        Dim web As SPWeb = SPControl.GetContextWeb(context)
```

```
        ' Get the view's GUID.
        Dim view As SPView = web.Lists(lName).Views(vName)
        ' Get the GUID of the list.
        Dim guid As System.Guid = view.ID
        ' Format the GUID as a string.
        Return ("{" & guid.ToString & "}")
    Catch ex As Exception
        Debug.Write(ex.Message)
        Return ""
    End Try
End Function
```

The error handling in these and subsequent procedures is very basic and simply returns "" if the list or view was not found. You may want to change that in your own code.

10.4.1.2 Using web services (.NET)

The Lists web service `GetList` method returns an XML description of the list, which contains an `ID` attribute with the list's GUID. You can use `SelectSingleNode` to extract the `ID` attribute from the XML, as shown here:

```
' Requires a Web reference to the Lists.asmx.
Function GetListGUID(ByVal lName As String) As String
    Dim xn As Xml.XmlNode, lws As New Lists.Lists
    Try
        ' Get the XML result from the web service.
        xn = lws.GetList(lName)
        ' Get the GUID (it's the ID attribute of the root element).
        Return xn.SelectSingleNode("//@ID").InnerText
    Catch ex As Exception
        Return ""
    End Try
End Function
```

The Views web service `GetViewCollection` returns an XML description listing the views available for a list. Each view has a `Name` attribute containing the GUID of the view. In order to extract that information, you need to find the view by its `DisplayName` attribute using an XPath expression in `SelectSingleNode` as follows:

```
' Requires a Web reference to the Views.asmx.
Function GetViewGUID(ByVal lName As String, ByVal vName As String) As String
    Dim xn As Xml.XmlNode, vws As New Views.Views
    Try
        ' Get the XML result from the web service.
        xn = vws.GetViewCollection(lName)
        ' Get the GUID (it's the Name attribute)
        ' where @DisplayName matches the view name.
        Return xn.SelectSingleNode("//*[@DisplayName='" & _
          vName & "']/@Name").InnerText
    Catch ex As Exception
        Return ""
    End Try
End Function
```

10.4.1.3 Using web services (VBA)

Getting the GUIDs from VBA is similar to using the web services from .NET; however, the XML objects and error handling are different than those available in .NET:

```
' Requires web reference to Lists.asmx.
Function GetListGUID(lName As String) As String
    On Error Resume Next
    Dim lws As New clsws_Lists
    Dim xn As IXMLDOMNodeList, guid As String
    ' Get the list.
    Set xn = lws.wsm_GetList(lName)
    ' Extract the GUID (it's the ID attribute).
    guid = xn(0).selectSingleNode("//@ID").Text
    ' Return "" if not found.
    If Err Then guid = ""
    ' Return the GUID.
    GetListGUID = guid
End Function

' Requires web reference to Views.asmx.
Function GetViewGUID(lName As String, vName As String) As String
    On Error Resume Next
    Dim vws As New clsws_Views
    Dim xn As IXMLDOMNodeList, guid As String
    ' Get the list's views.
    Set xn = vws.wsm_GetViewCollection(lName)
    ' Extract the GUID (it's the Name attribute).
    guid = xn(0).selectSingleNode("//*[@DisplayName='" & _
      vName & "']/@Name").Text
    ' Return "" if not found.
    If Err Then guid = ""
    ' Return the GUID.
    GetViewGUID = guid
End Function
```

10.4.2. Executing URL Commands

You can use the URL commands as part of a link rendered on a page. For example, the following link displays the schema of a list:

```
<a href="http://wombat1/_vti_bin/owssvr.dll?Cmd=ExportList&List={70F9FF01-15E5-4129-A370-9A31090204E9}">Show list schema</a>
```

Or you can get the resulting XML in code using the .NET XML objects, as shown here:

```
' .NET: Use URL protocol to get a list's XML.
Function GetListSchema(ByVal lName As String) As String
    Dim xdoc As New Xml.XmlDocument
    Dim guid As String = GetListGUID(lName)
    ' Create a reader for the URL.
    Dim xr As New Xml.XmlTextReader("http://wombat1/_vti_bin/owssvr.dll" & _
      "?Cmd=ExportList&List=" & guid)
    ' Load the response.
```

```
        xdoc.Load (url)
        ' Return the XML as a string.
        Return xdoc.OuterXml
    End Function
```

Here's the same code in VBA:

```
    ' VBA: Use URL protocol to get a list's XML.
    Function GetListSchema(ByVal lName As String) As String
        Dim guid As String, xdoc As New DOMDocument
        guid = GetListGUID(lName)
        ' Load the response.
        xdoc.Load ("http://wombat1/_vti_bin/owssvr.dll" & _
          "?Cmd=ExportList&List=" & guid)
        ' Return the XML as a string.
        GetListXML xdoc.Text
    End Function
```

You can experiment with the URL commands to get the right combination of parameters. It is often easier to compose the command in the browser's address bar or as an HTML link before using it in code. For example, the first link below displays the visible list items in XML, whereas the second includes all of the hidden fields because it includes the parameter `Query=*`:

```
    <a href="http://wombat1/_vti_bin/owssvr.dll?Cmd=Display&List={70F9FF01-15E5-4129-A370-9A31090204E9}&XMLDATA=TRUE">Display list XML (minimal)</a><br>

    <a href="http://wombat1/_vti_bin/owssvr.dll?Cmd=Display&List={70F9FF01-15E5-4129-A370-9A31090204E9}&XMLDATA=TRUE&Query=*">Display list XML (full)</a>
```

For complex queries, it is often easier to use web services or RPC.

10.5. Using FrontPage RPC

The FrontPage Remote Procedure Call (RPC) methods provide yet another way to change SharePoint sites and get content remotely. The RPC methods are similar to the URL commands discussed earlier, except RPC uses HTTP POST rather than GET to send requests.

Don't confuse FrontPage RPC with .NET RPC. They are very distinct technologies that unfortunately share the same acronym.

Use RPC methods when you want to compose your changes or queries in Collaborative Markup Language (CAML) rather than through web service methods. CAML is a declarative approach to programming SharePoint using XML rather than a procedural programming language such as Visual Basic .NET. There are several advantages to this approach:

- RPC methods can be included on a web page as content rather than as server-side code.
- Users can compose and run queries in CAML without installing web parts or other code on the server.
- SharePoint templates and descriptions are in CAML, so understanding it helps you create custom site definitions.
- RPC incurs less overhead than web services.

Table 10-8 lists the methods available through RPC. The URL commands are also available through RPC but aren't included in this table since they are already listed in Table 10-6.

Table 10-8. SharePoint RPC methods

Method	Parameters	Use to
Cltreq	UL, STRMVER, ACT, URL	Perform web discussion operations such as adding, editing, or deleting a discussion associated with a web page or with a document stored in a document library.
Delete	ID, List, NextUsing, owsfileref	Delete an item in a list.
DeleteField	List, Field, owshiddenversion	Delete a field from the list.
DeleteList	List, NextUsing	Delete a list.
DeleteView	List, View	Delete the view.
ImportList	Title, RootFolder, ListSchema	Create a list based upon a specified XML schema.
ModListSettings	List, OldListTitle, NewListTitle, Description, ReadSecurity, WriteSecurity, SchemaSecurity	Change the properties of the list.
NewField	List, FieldXML, AddToDefaultView, owshiddenversion	Add a new field to the list specified.
NewList	ListTemplate, Description, displayOnLeft, VersioningEnabled, GlobalMtgDataList, AllowMultiVote, showUsernames	Create a list of a specified type, such as Contacts, Discussions, or Survey.
NewViewPage	List, PageURL, DisplayName, HiddenView	Add a new view page to the list.
NewWebPage		Create a new web part page or a new basic page in the document library.
ReorderFields	List, ReorderedFields	Change the order in which fields in the list are displayed on the data entry form.
Save	ID, List, NextUsing	Save a new list or save modifications to an existing list.
SiteProvision	CreateLists	Add the default set of lists to an existing SharePoint site.

UpdateField	List, FieldXML, showHiddenVersion	Modify the schema of an existing field in the list.

10.5.1. Preparing a Page for RPC

Because RPC methods can change content, the post must include user information so SharePoint can authenticate and authorize the method. That means you must take a few special steps to execute RPC commands on a web page:

1. Add a FormDigest control to the page.
2. Create a **form** element that includes the method to execute and that posts its contents to *http://server/site/_vti_bin/owssvr.dll*.
3. Create a client-side script to insert the value from the FormDigest control into the form before it is posted.
4. Add a Submit button to post the form.

When creating a page that includes RPC commands, it's easiest if you start with a web part page generated by SharePoint, since those pages include the `@ Register` directive for the SharePoint web controls.

For example, I created *RPCDemo.aspx* in the TestPages document library and edited it using FrontPage to create a platform for testing RPC commands. Then, I added the following script to the `HEAD` element at the beginning of the page:

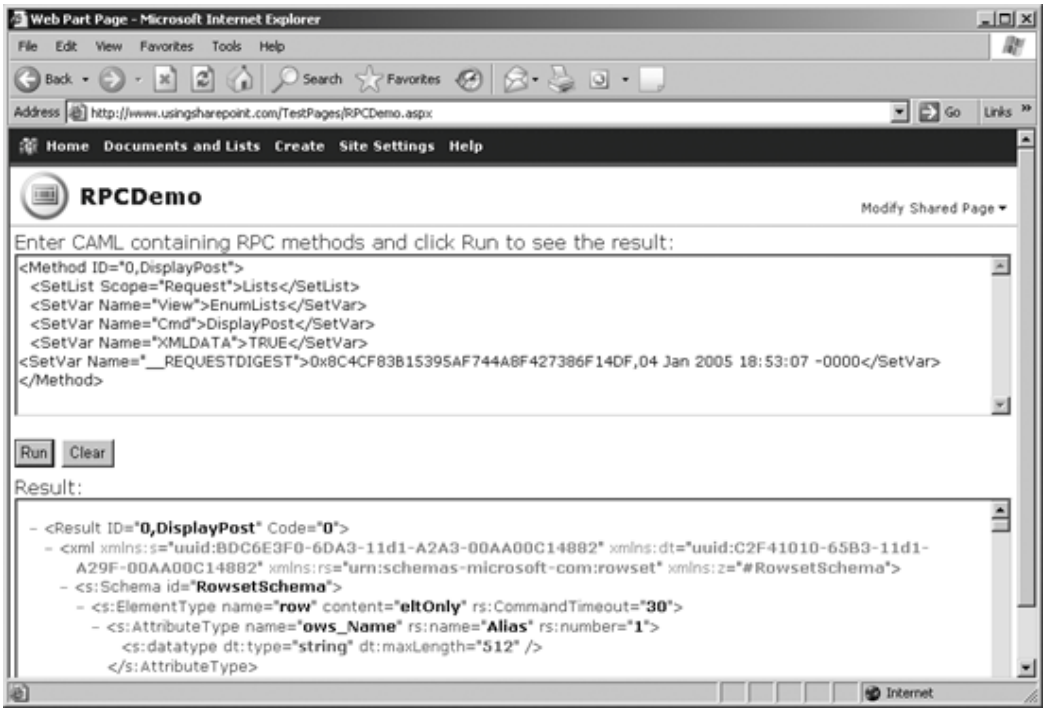
```
    <!-- This directive registers the FormDigest WebControl -->
    <%@ Register Tagprefix="SharePoint" Namespace="Microsoft.SharePoint.WebControls"
Assembly="Microsoft.SharePoint, Version=11.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c" %>
    <html dir="ltr">
    <HEAD>
    <!-- Add this script to include user info in the CASML -->
    <script type="text/javascript" language="JavaScript">
    function InsertSecurityValidation(oForm)
    {
        var sFormDigest = '<SetVar Name="_  _REQUESTDIGEST">' +
          oForm.elements["_  _REQUESTDIGEST"].value + "</SetVar>\n";
        var oPostBody = oForm.elements["PostBody"];
        var rePattern = /<\/Method>/g;
        oPostBody.value = oPostBody.value.replace(rePattern, sFormDigest + "</Method>");
    }
    </script>
    </HEAD>
```

Next, I added the following form after the closing form tag generated by SharePoint:

```
    <!-- SharePoint web part zones omitted here -->
    <\form>
    <!-- Form used to post RPC commands to SharePoint -->
    <form class="ms-formbody" method="post"
      action="http://wombat1/_vti_bin/owssvr.dll"
      onsubmit="InsertSecurityValidation(this);"
      target="result" >
      <!-- This control provides user information -->
      <SharePoint:FormDigest runat="server"/>
      Enter CAML containing RPC methods and click Run to see the result:
      <input type="hidden" name="Cmd" value="DisplayPost" />
      <!-- Source for CAML -->
      <textarea name="PostBody" style="width=100%;height=200">
      </textarea>
      <br>
      <input type="submit" value="Run" />
      <input type="reset" value="Clear" />
      <br>
      Result:
      <!-- Target for results -->
      <IFRAME name="result" src="::blank.htm" style="width=100%;height=200"/>
    </form>
```

Figure 10-8 shows the resulting web part page. You can enter CAML in the textbox, click Run, and see the results in the IFRAME as shown in the figure.

animal 10-8. Using RPCDemo.aspx to test RPC methods



I created *RPCDemo.aspx* as a test bed for composing RPC method calls. If the CAML is incorrect, an error displays in the result. You can cut and paste samples from the SharePoint SDK into the textbox, modify the parameters, and run the commands to see how they work.

Once you're satisfied with the results, you can hard-code the CAML onto a new web page using a hidden input control as shown here:

```
<!-- Embedded RPC -->
<form class="ms-formbody" method="post"
  action="http://wombat1/_vti_bin/owssvr.dll"
  onsubmit="InsertSecurityValidation(this);"
  target="_blank" >
  <!-- This control provides user information -->
  <SharePoint:FormDigest runat="server"/>
  Click Run to see the result in a pop-up window.
  <input type="hidden" name="Cmd" value="DisplayPost" />
  <!-- Source for CAML -->
  <input type="hidden" name="PostBody" value="
      <Method ID='0,DisplayPost'>
          <SetList Scope='Request'>Lists</SetList>
          <SetVar Name='View'>EnumLists</SetVar>
          <SetVar Name='Cmd'>DisplayPost</SetVar>
          <SetVar Name='XMLDATA'>TRUE</SetVar>
    </Method>" />
  <br>
  <input type="submit" value="Run" />
</form>
```

Don't forget to add the InsertSecurityValidation script to the new page's header. This sample displays the results in a new pop-up window (target="_blank") rather than on the page in an inline frame.

10.5.2. Common RPC Tasks

The following sections make up a quick tour of some common tasks you might want to perform using RPC. In each of these samples, you'll need to replace the GUIDs provided for lists and views with values from your own sites.

10.5.2.1 Identifying lists

RPC methods identify the list they act on by the list's GUID. You can get the GUIDs of all the lists in a site through RPC by calling the `DisplayPost` method with the `View` parameter set to `EnumLists` as shown here:

```
<Method ID="0,DisplayPost">
  <SetList Scope="Request">Lists</SetList>
  <SetVar Name="View">EnumLists</SetVar>
  <SetVar Name="Cmd">DisplayPost</SetVar>
  <SetVar Name="XMLDATA">TRUE</SetVar>
</Method>
```

Once you find the GUID of the list you want, you specify that value in the `SetList` element. For example, the following method exports the Announcements list:

```
<Method ID="0,ExportList">
  <SetList Scope="Request">{41D7D046-1E0A-4FB2-A096-C063D210D552}</SetList>
  <SetVar Name="Cmd">ExportList</SetVar>
</Method>
```

10.5.2.2 Combining multiple methods

Use the `Batch` element to combine two or more RPC methods in a single post. For example, the following CAML adds two items to the Announcements list:

```
<ows:Batch Version="6.0.2.5608" OnError="Return">
  <Method ID="Anouncement1">
    <SetList>{41D7D046-1E0A-4FB2-A096-C063D210D552}</SetList>
    <SetVar Name="ID">New</SetVar>
    <SetVar Name="Cmd">Save</SetVar>
    <SetVar Name="urn:schemas-microsoft-com:office:office#Title">Annoucment Title1</SetVar>
    <SetVar Name="urn:schemas-microsoft-com:office:office#Body">Annoucement text</SetVar>
    <SetVar Name="urn:schemas-microsoft-com:office:office#Expires">2005-09-14T00:00:00Z</SetVar>
  </Method>
  <Method ID="Announcment2">
    <SetList>{41D7D046-1E0A-4FB2-A096-C063D210D552}</SetList>
    <SetVar Name="ID">New</SetVar>
    <SetVar Name="Cmd">Save</SetVar>
    <SetVar Name="urn:schemas-microsoft-com:office:office#Title">Annoucement Title2</SetVar>
    <SetVar Name="urn:schemas-microsoft-com:office:office#Body">Announcment text</SetVar>
    <SetVar Name="urn:schemas-microsoft-com:office:office#Expires">2005-12-18T00:00:00Z</SetVar>
  </Method>
</ows:Batch>
```

The `Batch` element's `OnError` attribute determines what happens if one of the methods fails. The `Return` setting causes the methods to stop as soon as an error occurs; `Continue` causes SharePoint to skip the methods with errors and continue with the next method.

10.5.2.3 Querying lists

Use the Display method to perform a query on a list and return only a specific set of fields. For example, the following query returns three fields from a list using the filter criteria `introduced=2000` (note that field names are case-sensitive):

```
<Method ID="0,Display">
  <SetList Scope="Request">{5E6561D4-7048-45AB-BFA1-2D1991BAF3B1}</SetList>
  <SetVar Name="Cmd">Display</SetVar>
  <SetVar Name="XMLDATA">False</SetVar>
  <SetVar Name="Query">name Title introduced</SetVar>
 <SetVar Name="FilterField1">introduced</SetVar>
 <SetVar Name="FilterValue1">2000</SetVar>
</Method>
```

For queries that require filtering or sorting, you can also specify a view that defines those criteria. For example, the following query returns items from the list specified by one of the list views:

```
<Method ID="0,Display">
  <SetList Scope="Request">{5E6561D4-7048-45AB-BFA1-2D1991BAF3B1}</SetList>
  <SetVar Name="Cmd">Display</SetVar>
  <SetVar Name="XMLDATA">False</SetVar>
  <SetVar Name="View">{21E03766-D0CF-4942-B2C0-DF4E3706DD50}</SetVar>
</Method>
```

10.5.2.4 Creating lists

Use the NewList method to create a new list or document library on a site. The `ListTemplate` parameter determines what type of list is created, as shown in Table 10-9.

Table 10-9. ListTemplate settings

Setting	List type	Setting	List type
100	Generic	101	Document library
102	Survey	103	Links
104	Announcements	105	Contacts
106	Events list	107	Tasks
108	Discussion board	109	Picture library
110	Data sources	111	Site template gallery
113	Web part gallery	114	List template gallery
115	Form library	120	Custom grid for a list
200	Meeting series	201	Meeting agenda
202	Meeting attendees	204	Meeting decisions
207	Meeting objectives	210	Meeting textbox
211	Meeting things to bring	212	Meeting workspace pages
1100	Issue tracking		

For example, the following method creates a new document library:

```
<Method ID="0,NewList">
  <SetVar Name="Cmd">NewList</SetVar>
  <SetVar Name="ListTemplate">101</SetVar>
  <SetVar Name="Title">New Document Library</SetVar>
</Method>
```

10.5.2.5 Creating pages

To add a new web page to the document library created in the preceding section, get the library's GUID; then use the NewWebPage method:

```
<Method ID="0,NewWebPage">
  <SetList Scope="Request">{28F54F4C-BA98-43E9-A60F-8C81E5365560}</SetList>
  <SetVar Name="Cmd">NewWebPage</SetVar>
  <SetVar Name="ID">New</SetVar>
  <SetVar Name="Type">WebPartPage</SetVar>
  <SetVar Name="WebPartPageTemplate">3</SetVar>
  <SetVar Name="Overwrite">true</SetVar>
  <SetVar Name="Title">TempPage</SetVar>
</Method>
```

The `WebPartPageTemplate` parameter determines the layout of the new page, as shown in Table 10-10.

Table 10-10. WebPartPageTemplate settings

Setting	Page layout
1	Full page, vertical
2	Header, footer, 3 columns
3	Header, left column, body
4	Header, right column, body
5	Header, footer, 2 columns, 4 rows
6	Header, footer, 4 columns, top row
7	Left column, header, footer, top row, 3 columns
8	Right column, header, footer, top row, 3 columns

10.5.2.6 Deleting items

The following code deletes the web page created in the preceding section:

```
<Method ID="0,Delete">
  <SetList Scope="Request">28F54F4C-BA98-43E9-A60F-8C81E5365560</SetList>
  <SetVar Name="Cmd">Delete</SetVar>
  <SetVar Name="ID">1</SetVar>
  <SetVar Name="owsfileref">http://wombat1/New Document Library/TempPage.aspx</SetVar>
</Method>
```

10.5.2.7 Deleting lists

Use this code to delete the document library created earlier:

```
<Method ID="0,DeleteList">
  <SetList Scope="Request">28F54F4C-BA98-43E9-A60F-8C81E5365560</SetList>
  <SetVar Name="Cmd">DeleteList</SetVar>
</Method>
```

10.5.3. Resources

To learn about	Look here
Installing the Office Web Services Toolkit	Search http://www.microsoft.com/downloads for "Web Services Toolkit."
Lists web service	http://msdn.microsoft.com/library/en-us/spptsdk/html/soapcLists.asp
DOMDocument	http://msdn.microsoft.com/library/en-us/xmlsdk/html/xmobjPMEXMLDOMDocument.asp
IXMLDOMNodeList	http://msdn.microsoft.com/library/en-us/xmlsdk30/htm/xmobjxmldomnodelist.asp
Query element	http://msdn.microsoft.com/library/en-us/spptsdk/html/tscamlquery.asp
ViewFields element	http://msdn.microsoft.com/library/en-us/spptsdk/html/tscamlviewfields.asp
QueryOptions element	http://msdn.microsoft.com/library/en-us/spptsdk/html/tscSPQuery.asp
Batch element	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/spsdk11/caml_schema/spxmlelbatch.asp

Appendix A. Upgrading

Upgrading to SQL 2000

The default installation of SharePoint uses WMSDE as the data engine for the content and configuration databases. WMSDE runs locally on the server and limits the total size of SharePoint sites to 2 GB per content database. However, WMSDE does not limit performance, which is a change from previous versions of the data engine.

Upgrading from WMSDE to SQL 2000 (SP3a) on a shared server increases the capacity of SharePoint sites and simplifies maintenance since you can use SQL Server Enterprise Manager to configure automated back-ups and perform other tasks.

Upgrading the databases to a dedicated server running SQL 2000 can significantly improve performance as well as capacity, particularly for high-volume sites.

If you install SQL 2000 on a shared server after installing SharePoint, setup should upgrade the WSMDE server instance to use SQL 2000. To verify the data engine used by a server instance:

1. Start SQL Server Enterprise Manager.
2. Register the server instance. SharePoint names the instance *machinename*`/SHAREPOINT` by default.
3. Select the server instance and choose Action → Properties.
4. Check the Product name at the top of the General tab of the Properties dialog. The name should be `SQL Server` *editionname*.

If SharePoint was installed using the Typical Installation option on a server already running SQL Server 2000, Setup still configures SharePoint to use WMSDE. To upgrade without reinstalling SQL Server:

1. Use SQL Enterprise Manager to copy the content and configuration databases from the WMSDE server instance to your SQL Server instance. SharePoint names its configuration database `sts_config` and content databases `sts_`*machinename*`_`*n*.
2. Use the SharePoint Central Administration site to set your configuration database to the new database you just copied.
3. Use SharePoint Central Administration to add the content database you just copied and remove the old one.

Those last two steps require a little more detail. To change the configuration database:

1. Display the SharePoint Central Administration home page.
2. Select Set configuration database server.
3. Enter the new server name; select Connect to an existing configuration database and click OK. Since you didn't rename the database when you copied it, you can use the same database name.

To change the content database for a virtual server:

1. Display the SharePoint Central Administration home page.
2. Choose Configure virtual server settings and select the virtual server to change.
3. Select Manage content databases → Add a content database.
4. Select Specify database server settings; then enter the Database server name, the Database name, and the two capacity settings (standard settings are `9000` and `15000`, respectively).
5. Click OK to add the database. If the operation succeeds, SharePoint returns you to the Manage Content

Databases page.

6. Click on the previous database (it appears above the one you just added).

7. Select Remove content database and click OK.

To upgrade to a dedicated database server, install the SQL Server client tools on the SharePoint server. This allows you to copy the databases from the current shared server to the dedicated server. Then follow the procedures above for moving the configuration and content databases to the dedicated server. When upgrading to Portal Server, it is a good idea to move the database to a dedicated SQL Server.

PREV NEXT

Upgrading to Portal Server

If you create web sites using SharePoint Services, then decide to upgrade the server to SharePoint Portal Server, you'll need to restore the SharePoint settings for each virtual server.

SharePoint Services installed in Active Directory mode or in host header mode can't be upgraded to Portal Server.

To upgrade a SharePoint Services site to SharePoint Portal Server, follow these general steps:

1. Record the database settings used by each virtual server.
2. Record the Application Pool settings used by use top-level site in IIS.
3. Install and configure SharePoint Portal Server.
4. Re-extend each virtual server.
5. Connect the existing sites to the portal server.
6. Link the existing sites to the Sites Directory in the Portal.

You must have an account with Administrative privileges to perform these tasks. The following sections explain the previous steps in more detail.

Recording Settings

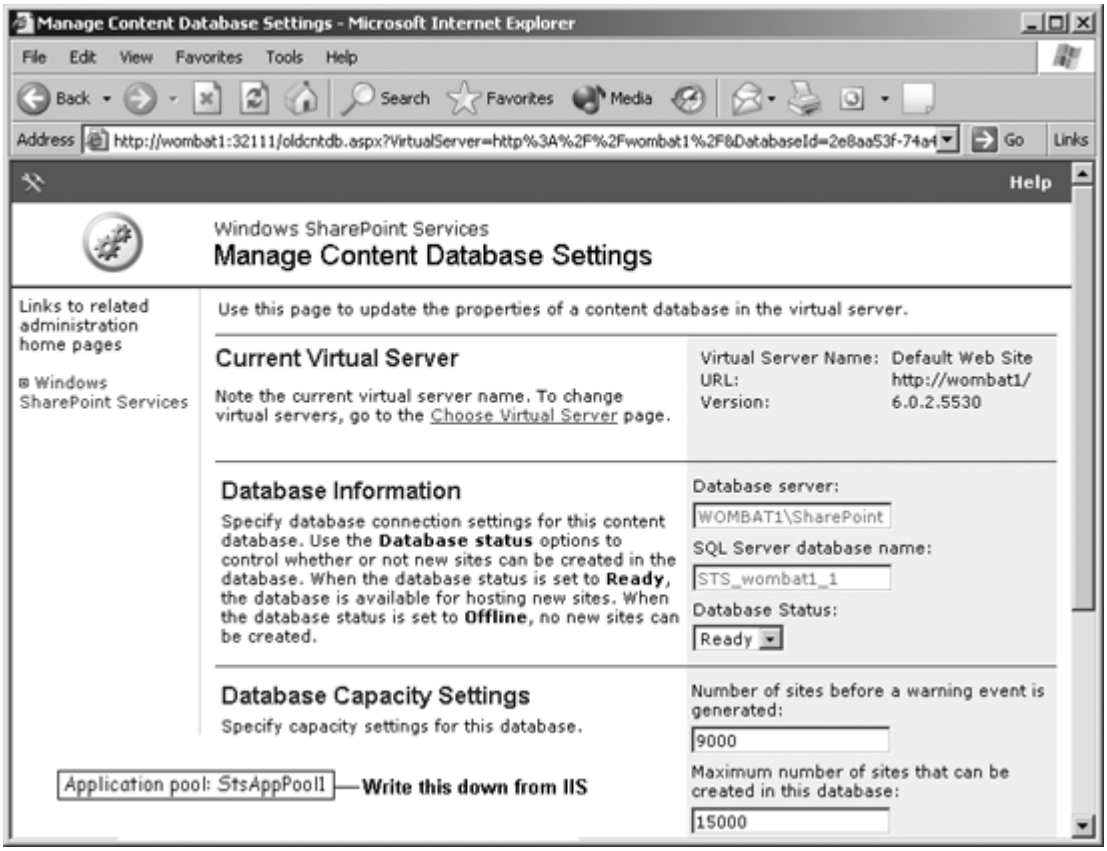
To record database and application pool settings:

1. Display the SharePoint Central Administration page in the browser.
2. Click Configure virtual server settings → choose the virtual server from the displayed list → Manage content database → choose the database from the displayed list. The browser displays the database settings as shown in Figure A-1.
3. Record the setting or simply print the page for later reference.
4. Repeat these steps for each virtual server on the physical server.
5. Use the Remote Desktop utility to log on to the server hosting the site.
6. From the server's Start menu, choose Adminstrative Tools → Internet Information Services (IIS).
7. In IIS, select the default web site and click Action → Properties → Home Directory to display the application pool setting (for example, `StsAppPool1`).
8. Record the setting and repeat step 7 for each top-level SharePoint site on the server (top-level web sites in IIS are virtual servers in SharePoint).

Installing and Configuring Portal Server

Installing Portal Server halts SharePoint Services and web sites on the server during installation. Send an alert to users before beginning an installation or schedule the installation for a time when no one is using the server. You should also back up the server before installing the new software.

animal A-1. Print the Database Settings page to record your settings; then write the application pool setting from IIS on the printout



To install and configure SharePoint Portal Server on a server already running SharePoint Services:

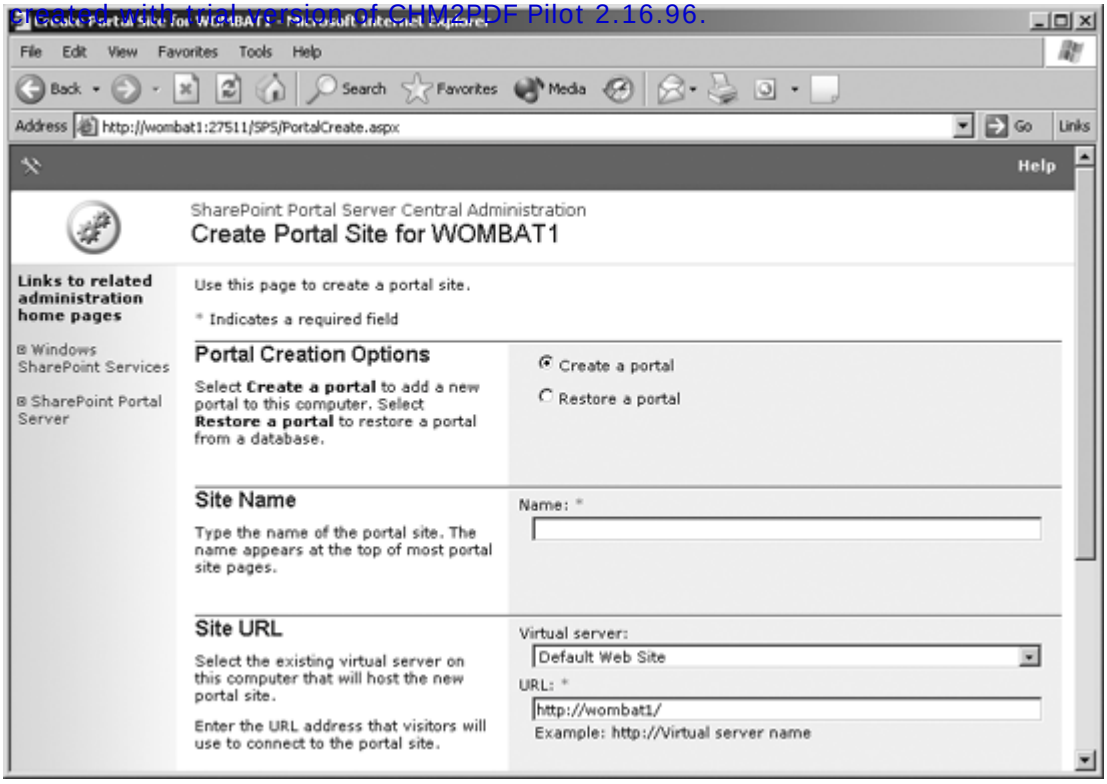
1. Log on with administrative privileges to the server using the Remote Desktop utility and run the SharePoint Portal Server installation from CD or from a download. The installation is straightforward and takes about 30 minutes.

2. When install is complete, Portal Server displays the Configure Server Farm page. If you are configuring a single server, you can simply enter contact information and click OK. SharePoint displays the Create Portal Site page shown in Figure A-2.

3. You can create a portal now, or later. A portal site organizes the other SharePoint sites in your organization as shown in Figure A-3.

The portal site is installed as a new virtual server (in IIS terminology it is a top-level web site). If you have other existing virtual servers, you will need to configure those sites to use a different port or IP address through IIS. For example, to change the port of an existing site, start IIS, select the top-level site and click Action → Properties → Web Site. Then change the TCP port used by the site and click OK.

animal A-2. Creating a portal site



animal A-3. The default home page of a portal site

The default port for a web server is 80. Other settings require users to enter the port number to view the site, (for example http://wombat1:40/)displays the top-level site at port 40 on my server whereas http://wombat1/ displays the site at port 80. If your server has multiple IP addresses, you can use those to distinguish among the sites rather than among port numbers.

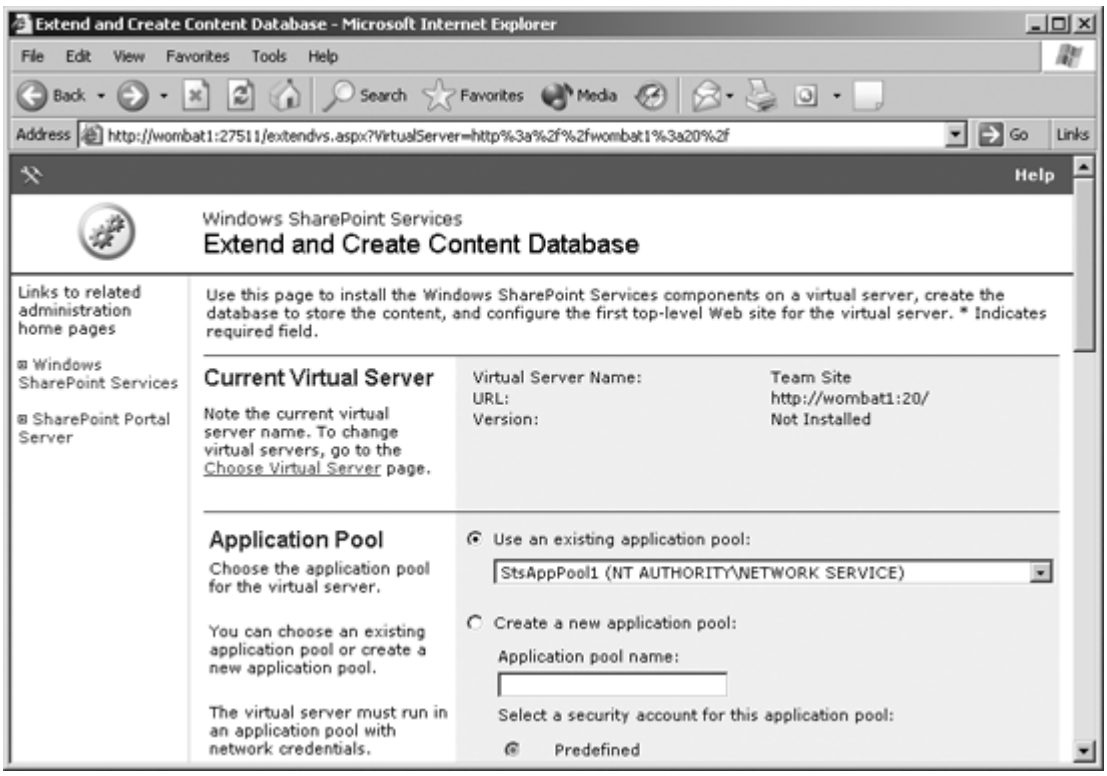
Re-Extending Existing Sites

Installing Portal Server removes the association between existing virtual servers and SharePoint. You must re-extend each virtual server before the server and all of the sites it contains will begin working again. To re-extend a virtual server:

1. From the SharePoint Central Administration page, click Extend or upgrade an existing virtual server → complete list; and then click the name of the server. SharePoint displays the Extend Virtual Server page.

2. Click Extend and create a content database. SharePoint displays Figure A-4.

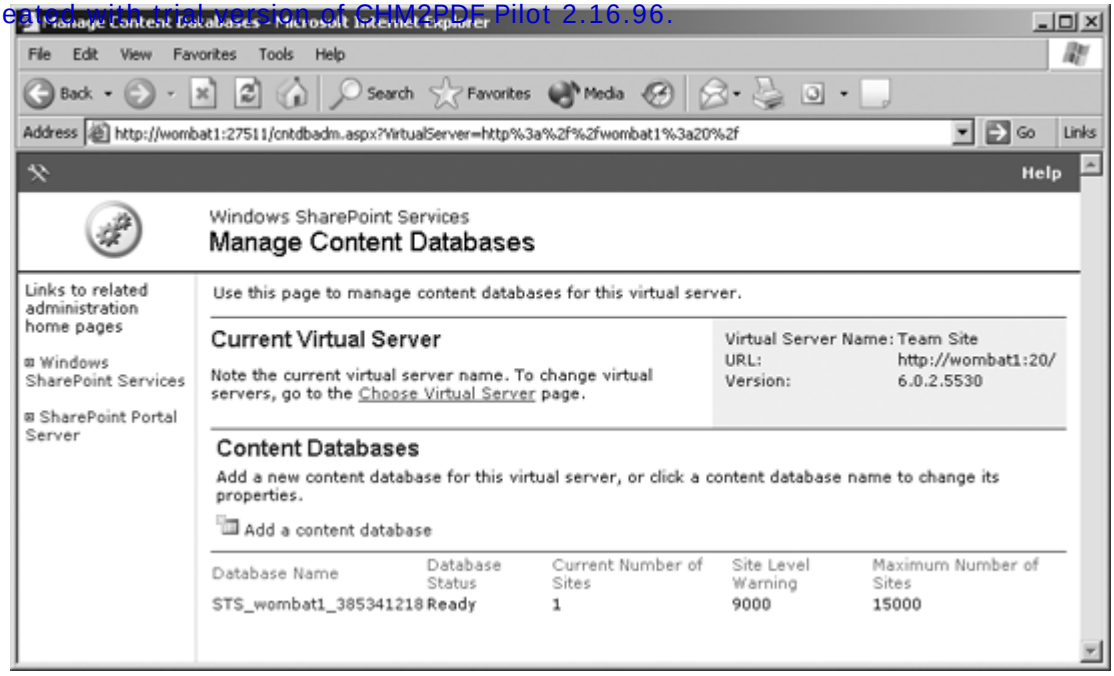
animal A-4. Set the application pool



3. Select the application pool you wrote down earlier in the "Recording Settings" section, but don't enter database information at this pointyou'll re-establish the data connection later. For now, just accept the default content database server, enter a site owner and click OK.

4. After SharePoint extends the site, click Windows SharePoint Services → Configure Virtual Server Settings → select the site name → Manage content databases. SharePoint displays Figure A-5.

animal A-5. Removing the default database



5. Click the database name (STS_*servername_n*) and select select Remove content database. SharePoint displays a warning. Click OK to close the warning; then click OK again to make the change.

6. SharePoint returns you to the Manage Content Databases page. Click Add a content database to display the page shown in Figure A-6.

7. Select Specify database server settings and re-enter the database information you printed in the earlier section "Recording Settings" and then click OK. SharePoint reassociates the virtual server with the content database.

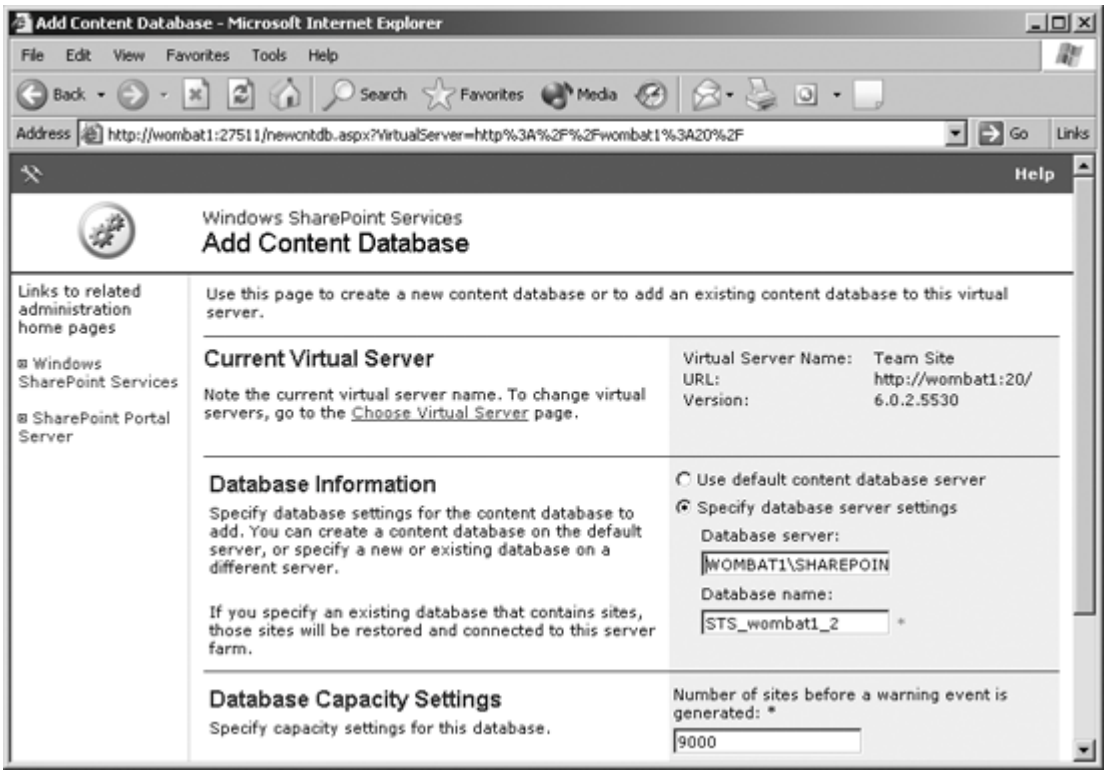
Confirm that the existing SharePoint virtual server is working by navigating to its home page in the browser. If you changed the site's port number in "Installing and Configuring Portal Server," remember to include that number as part of the address.

Connecting Sites to the Portal

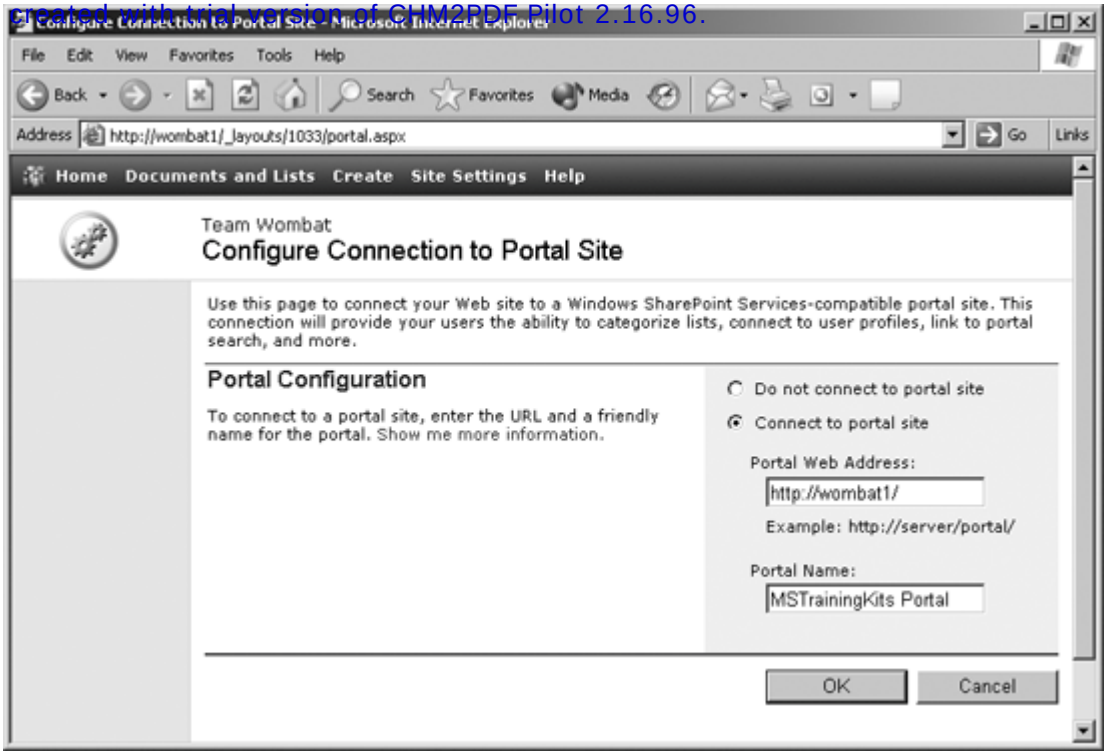
The whole point of Portal Server is to integrate multiple SharePoint sites through a single portal, so you'll want to connect your existing SharePoint virtual servers to the portal site. To connect a virtual server's sites to the portal, go to the home page of the top-level site. Click Site Settings → Go to Site Administration → Configure connection to portal site. SharePoint displays Figure A-7. Enter the address of the portal and the portal name; then click OK.

Connecting a site to a portal adds a link to the site's navigation bar to take users to the portal site. This link appears on the home page, lists, libraries, and administrative pages of the virtual server. The portal link does not appear on subsites or workspace sites, however. To link subsites to the portal, repeat this task for each subsite.

animal A-6. Reassociating the site with the existing original database



animal A-7. Connecting a site to the portal

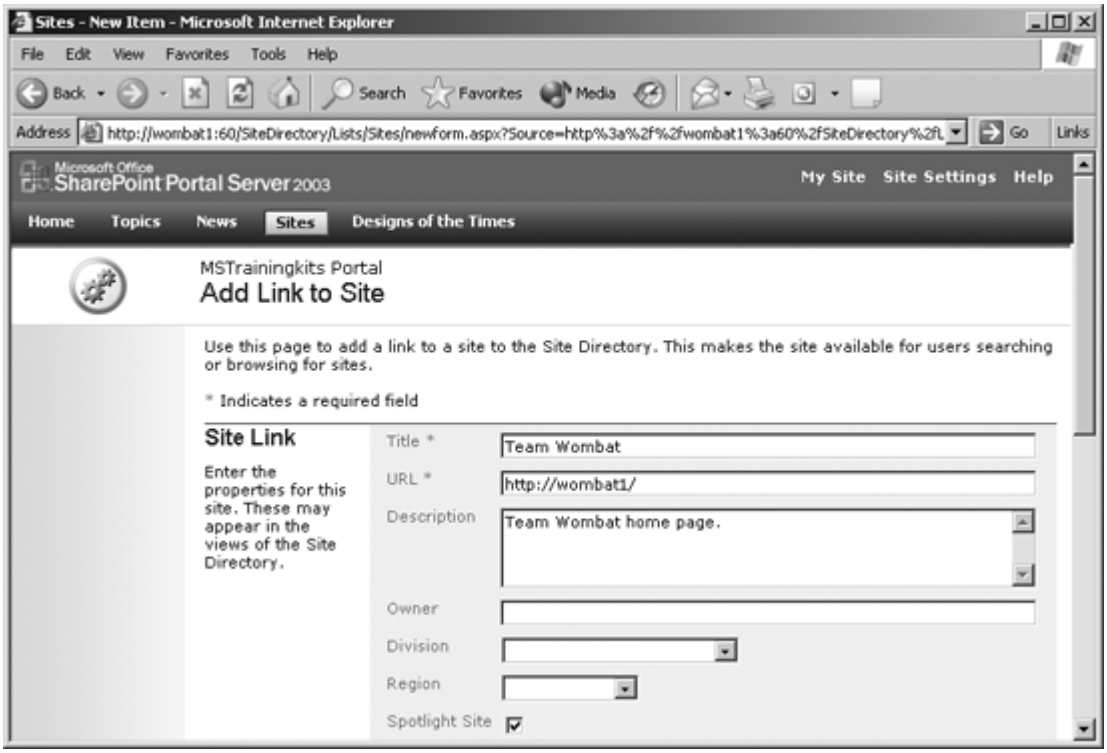


PREV NEXT

Adding Links to the Portal

To provide navigation from the portal site to existing virtual servers, add links to the portal's Sites Directory. From the portal site's home page, click Sites → Add Link to a Site. SharePoint displays Figure A-8. Complete the page and click OK. SharePoint adds the site to the Sites list in the Site Directory subsite. You can view the portal's Sites list by clicking Sites on the home page. The default view of the Sites list features new sites and spotlight sites in web parts on the page.

animal A-8. Adding links for existing sites to the portal



Upgrading from Team Services

SharePoint Team Services is replaced by SharePoint Services. This is more than a typical version upgrade, although their version numbers are sequential (Version 1.0 and version 2.0). Microsoft provides a tool for migrating sites from Team Services to SharePoint Services (*smigrate.exe*).

The migration must be done from one running server to another: there's no way to upgrade a site in place. Because of the differences in technology, this isn't a simple process, and you may want to leave Team Service sites in place and gradually replace them with SharePoint Services sites. Microsoft provides a good deal of guidance on this at http://www.microsoft.com/resources/documentation/wss/2/all/adminguide/en-us/stsb04.mspx.

Resources

To learn about	Look here
Upgrading to Portal Server	http://www.microsoft.com/technet/prodtechnol/office/sps2003/deploy/inwsssps.mspx

Appendix B. Reference Tables

Office Versions

SharePoint works with Office 2000, 2002 (XP), and 2003. However, the feature set varies a great deal among these versions. Basically, all versions allow you to save and open files from SharePoint sites through a browser; 2002 supports some integration (export list to spreadsheet); and 2003 provides full integration through the Shared Documents task pane.

			Office version	
Product	Feature	2000	2002	2003
All	Save and open files from SharePoint sites	X	X	X
	Shared Workspace task pane			X
	Updates document for shared attachments			X
	View and edit a shared attachment	X	X	X
	Create new documents in web browser		X	X
	Collect metadata automatically			X
	Promote and demote file properties and metadata automatically	Limited	X	X
	Track document versions			X
	Check out and check in documents			X
	Manage Microsoft Project documents, risks, and issues			X
	Upload multiple documents			X
	Inline discussions	X	X	X
	Microsoft Office Components for SharePoint			X
	Person Names Smart Tag			X
	Integration with Microsoft Business Solutions			X
	Shared attachment			X
	Create Document workspace sites			X
Outlook	Create Meeting workspace sites automatically			X
	Synchronize calendar and contact list sites			X
	Alerts	X	X	X
	Alert integration with Outlook			X
Excel	Two-way synchronization with SharePoint lists			X
	Export list data to Excel spreadsheet		X	X
	Create custom list from Excel spreadsheet			X
Access	Link table to SharePoint list			X
	Export list data to Access database table			X
	Create custom list from Access database table			X
FrontPage	Edit and customize Windows SharePoint Services web sites			X
	Create and customize data-driven web part pages			X
	Solution packages			X

	Browse and search web part galleries			X
	Manage list views			X
	Design templates			X
	Web part connections			X
	Backup and restore site			X
InfoPath	Business document library	N/A	N/A	X
	Edit documents in InfoPath	N/A	N/A	X
	Aggregate business reports	N/A	N/A	X

PREV NEXT

StsAdm Commands

The *StsAdm.exe* utility allows Administrators to maintain and modify SharePoint sites through a command-line interface. The *StsAdm.exe* command-line has the general form:

```
stsadm.exe -o command -url siteAddress commandParameters
```

This utility is installed in the *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\BIN* folder.

Command	Use to	Parameters
addpath	Add a defined path (inclusion or exclusion) to a virtual server.	-url -type
addtemplate	Add a site template to the template gallery.	-filename -title -description
adduser	Add a user account to the specified site and assign it to the specified site group.	-url -userlogin -useremail -role -username -siteadmin
addwppack	Add a Web Part package to your server Web Part gallery.	-filename or -name -url -globalinstall -force -lcid
backup	Create a backup of the site at the specified URL.	-url -filename -overwrite
binddrservice	Register a data-retrieval service.	-servicename -setting
createadminvs	Create the administration virtual server for Microsoft Windows SharePoint Services.	-admapcreatenew -admapidname -admapidtype -admapidlogin

		-admapidpwd
createsite	Create a site at the specified URL.	-url -ownerlogin -owneremail -ownername -lcid -sitetemplate -title -description -quota
createsiteinnewdb	Create a site at the specified URL and create a new content database.	-url -ownerlogin -owneremail -ownername -databaseuser -databasepassword -databaseserver -databasename -lcid -sitetemplate -title -description -secondarylogin -secondaryemail -secondaryname
createweb	Create a subsite at the specified URL.	-url -lcid -sitetemplate -title -description -unique
deleteadminvs	Delete the administration virtual server.	None
deleteconfigdb	Delete the configuration database.	None
deletepath	Remove an included or excluded path from the list of paths.	-url

deletesite	Delete the specified site.	-url -deleteadaccounts
deletetemplate	Delete the specified site template.	-title -lcid
deleteuser	Delete the specified user.	-url -userlogin
deleteweb	Delete the specified subsite.	-url
deletewppack	Remove the web parts in a web part package from a virtual server	-name -url -lcid
disablessc	Disable self-service site creation for the specified virtual server.	-url
disablestsisapis	Disable the Windows SharePoint Services ISAPI extensions.	None
email	Set the e-mail configuration settings for your server, or for a specific virtual server.	-outsmtpserver -fromaddress -replytoaddress -codepage -url
enablessc	Enable self-service site creation for the specified virtual server.	-url -requiresecondarycontact
enablestsisapis	Enable the Windows SharePoint Services ISAPI extensions.	None
enumroles	List the site groups that are available for use in a particular site or subsite.	-url
enumsites	List all of the sites that have been created under a particular virtual server.	-url
enumsubwebs	List the subsites that have been created under a particular site.	-url
enumtemplates	Lists the site templates that are available.	-lcid
enumusers	List the users of a particular site or subsite.	-url
enumwppacks	List the web part packages currently in your server web part gallery.	-name -url
extendvs	Extend a virtual server with SharePoint and create a new content database.	-url -ownerlogin -owneremail -ownername -databaseuser (du) -databaseserver (ds) -databasename (dn)

		-databasepassword (dp) -lcid -sitetemplate -donotcreatesite -apcreatenew -apidname -apidtype -apidlogin -apidpwd
extendvsinwebfarm	Extend a virtual server with SharePoint for use in a server farm.	-url -vsname -apcreatenew -apidname -apidtype -apidlogin -apidpwd
getadminport	Return the administration port for Windows SharePoint Services.	None
getproperty	Return the property value for the specified property name.	-propertyname or -pn -url
removedrservice	Remove a data retrieval service.	-servicename -setting
renameweb	Rename a subsite.	-url -newname
restore	Restore a web site from a backup file.	-url -filename -overwrite
setadminport	Set the port number for the administration virtual server.	-port -ssl -admapcreatenew -admapidname -admapidtype -admapidlogin -admapidpwd
setconfigdb	Create the configuration database or specify the connection to an existing configuration database.	-databaseserver (ds)

		-connect -databaseuser (du) -databasepassword (dp) -databasename (dn) -hh -adcreation -addomain -adou
setproperty	Set a property by name.	-propertyname or -pn -propertyvalue or -pv -url
siteowner	Set the owner or secondary owner of a site collection.	-url -ownerlogin or -secondownerlogin
unextendvs	Removes SharePoint from a virtual server.	-url -deletecontent
uninstall	Uninstall SharePoint from the default virtual server at port 80.	-deletecontent
upgrade	Upgrade the server with SharePoint.	-url
userrole	Specify the site group membership for a user.	-url -userlogin -role -add -delete

SetupSts Commands

The *SetupSts.exe* utility allows Administrators to install SharePoint through a command-line interface. The *SetupSts.exe* command-line has the general form:

```
SetupSts.exe options
```

This utility is part of setup and so is installed by *STSV2.EXE* at a location similar to *C:\Program Files\STS2Setup_1033*. In this case, `1033` is the language ID for English. Other language versions will use a different ID.

Option	Use to
/remotesql=yes/no	Specify whether or not WMSDE is installed. The default value is no. Set this property to yes if you are going to use an existing or remote installation of Microsoft SQL Server with Windows SharePoint Services.
/fulluninstall=yes/no	Specify whether or not to remove Windows SharePoint Services from extended virtual servers when performing an uninstall. The default is yes.
/provision=yes/no	Specify whether or not to provision the administrative virtual server, extend the default virtual server, and create a top-level web site during installation. The default is yes.
/datadir="*path*\\"	Specify where to install WSMDE. Set this property to a path on your local server.
/ l *logfile*	Log setup messages.
/q or /qn	Run in quiet mode (unattended setup with no user intervention).
/qb	Run in basic mode (limited user intervention).
/qf	Run in full mode (user must fill in options during setup). This is the default.
/qr	Run in reduced mode. Displays reduced UI during installation.
+/-	Specify whether or not to display a success dialog box at the end of installation (modifies -q options).
/x	Uninstall SharePoint.

Server Files and Locations

The following folders are found under the install path of SharePoint Services. Typically, that is *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60*.

Folder	Description	Files and purpose
\ADMISAPI	The physical directory addressed by the SharePoint Central Administration virtual directory.	*admin.asmx* (ISAPI filter for administration)
\BIN	Contains the core binary files for Windows SharePoint Services.	**.DLL* (core binary files) *OWSTIMER.EXE* (Microsoft SharePoint Timer service) *SMIGRATE.EXE* (Microsoft SharePoint Migration Tool) *STSADM.EXE* (stsadm utility) *STSCFG.EXE* (configuration utility used by Setup)
\BIN\LCID	Contains the core binary files used by specific languages.	*FPEXT.MSG* '(error messages and text strings) *ONETINTL.DLL* (core international binary file)
\CONFIG	Contains configuration files and default values for the server.	**.xml* (XML files used to map default values) *appwpresweb.config* (configuration file) *adminweb.config* (configuration file for the administrative virtual server) *gacwpresweb.config* (configuration file) *layoutsweb.config* (configuration file for the _layouts directory) *web.config* (configuration file for virtual servers) *wss_mediumtrust.config* (configuration file) *wss_minimaltrust.config* (configuration file)
\ISAPI	The physical directory addressed by the */_vti/_bin* virtual directory.	**.asmx* (SOAP protocol receptors) **.aspx* (form pages) *Global.asax* (ASP.NET namespace definition) **.xml* (XML file for managed code) **.DLL* (core binary files for managed code) *web.config* (configuration file)
\ISAPI\BIN	Contains binary files for the */_vti/_bin* virtual directory.	*STSSOAP.DLL* (binary file used for SOAP code)
\ISAPI\HELP\LCID\STS\HTML	Contains Help files and support files used in the	**.css*

	Help system.	*.htm *.js layout.swf
\ISAPI\HELP\LCID\STS\IMAGES	Contains images used in the Help system.	*.gif
\ISAPI_VTI_ADM	Contains Microsoft Office FrontPage 2003 legacy binary files.	ADMIN.DLL (binary file used for administration from Office FrontPage 2003)
\ISAPI_VTI_AUT	Contains Office FrontPage 2003 legacy binary files.	AUTHOR.DLL (binary file used for authoring from Office FrontPage 2003)
\TEMPLATE	Contains all site templates and core web site files.	
\TEMPLATE\LCID\MPS	Contains files that are copied to the root of the web site upon instantiation with a Meeting Workspace template (for example, default.aspx).	*.aspx (form pages)
\TEMPLATE\LCID\MPS\DOCTEMP\SMARTPGS\	Contains files used for web part pages in Meeting Workspaces.	spstd1.aspx (form page)
\TEMPLATE\LCID\MPS\LISTS	Contains the actual lists used in the Meeting Workspace templates, along with schema definition and default views.	
\TEMPLATE\LCID\MPS\LISTS\AGENDA	Contains files used for the Agenda list.	*.aspx (form pages) SCHEMA.XML (schema file)
\TEMPLATE\LCID\MPS\LISTS\DECISION	Contains files used for the Decisions list.	*.aspx (form pages) SCHEMA.XML (schema file)
\TEMPLATE\LCID\MPS\LISTS\DOCLIB	Contains files used for document libraries in the Meeting Workspace templates.	*.aspx (Form pages) *.HTM (dialog boxes) SCHEMA.XML (schema file)
\TEMPLATE\LCID\MPS\LISTS\MEETINGS	Contains files used for the Meeting Workspace templates.	MoveToDt.ASPX (form page) SCHEMA.XML (schema file)
\TEMPLATE\LCID\MPS\LISTS\OBJECTIV	Contains files used for the Objectives list.	*.aspx (Form pages) SCHEMA.XML (schema file)
\TEMPLATE\LCID\MPS\LISTS\PEOPLE	Contains files used for the Attendees list.	*.aspx (form pages) SCHEMA.XML (schema file)
\TEMPLATE\LCID\MPS\LISTS\TEXTBOX	Contains files used for the Text Box list.	*.aspx (form pages) SCHEMA.XML (schema file)
\TEMPLATE\LCID\MPS\LISTS\THGBRING	Contains files used for the Things to Bring list.	*.aspx (form pages) SCHEMA.XML (schema file)
\TEMPLATE\LCID\MPS\LISTS\WKSPGLIB	Contains files used for lists in the Meeting Workspace templates.	SCHEMA.XML (schema file)
\TEMPLATE\LCID\MPS\XML	Contains the available lists in the Meeting Workspace templates, base types for fields (onet.xml), and the standard view template for new views.	*.XML (XML files for site schema and views)

\TEMPLATE\LCID\STS	Contains files that are copied to the root of the web site upon instantiation with the Team Site template (for example, *default.aspx*).	*default.aspx* (default home page for sites based on Team Site templates)
\TEMPLATE\LCID\STS\DOCTEMP\BLANKPGS	Contains the default document templates.	*bpstd.aspx* *_blankpage.htm*
\TEMPLATE\LCID\STS\DOCTEMP\FP	Contains document templates for Office FrontPage 2003.	*FPTMPL.HTM* (default document templates for FrontPage documents)
\TEMPLATE\LCID\STS\DOCTEMP\PPT	Contains document templates for Microsoft Office PowerPoint 2003.	*FILELIST.XML* *MASTER03.CSS* *MASTER03.HTM* *MASTER03.XML* *PPTMPL.HTM* *PPTMPL.POT* *PRES.XML* *PREVIEW.WMF* *SLIDE001.HTM*
\TEMPLATE\LCID\STS\DOCTEMP\SMARTPGS	Contains document templates for web part pages.	**.aspx* *_smartpage.htm* *_webpartpage.htm*
\TEMPLATE\LCID\STS\DOCTEMP\WORD	Contains document templates for Microsoft Office Word 2003.	*WDTMPL.DOC* *WDTMPL.HTM*
\TEMPLATE\LCID\STS\DOCTEMP\XL	Contains document templates for Microsoft Office Excel 2003.	*FILELIST.XML* *SHEET001.HTM* *SHEET002.HTM* *SHEET003.HTM* *STYLE.CSS* *TABSTRIP.HTM* *XLTMPL.HTM* *XLTMPL.XLS*
\TEMPLATE\LCID\STS\DOCTEMP\XMLFORMS\BLANK	Contains document templates for Microsoft Office InfoPath 2003.	*TEMPLATE.XML* (default document templates for XML documents)
\TEMPLATE\LCID\STS\DWS	Contains files that are copied to the root of the web site upon instantiation with a Document Workspace template.	*default.aspx* (default home page for Document Workspaces)
\TEMPLATE\LCID\STS\LISTS	Contains the actual lists along with schema definition and default views.	
\TEMPLATE\LCID\STS\LISTS\ANNOUNCE	Contains files used for the Announcements list.	**.aspx* (form pages) *SCHEMA.XML* (schema file)
\TEMPLATE\LCID\STS\LISTS\CONTACTS	Contains files used for the	**.aspx* (form pages)

	Contacts list.	*SCHEMA.XML* (schema file) *VCARD.VCF* (contacts form)
TEMPLATE\LCID\STS\LISTS\CUSTLIST	Contains files used for custom lists.	**.aspx* (form pages) *SCHEMA.XML* (schema file)
TEMPLATE\LCID\STS\LISTS\DATASRCS	Contains files used for data sources for lists.	*SCHEMA.XML* (Schema file)
TEMPLATE\LCID\STS\LISTS\DISCUSS	Contains files used for the Discussion Board list.	**.aspx* (form pages) *SCHEMA.XML* (schema file)
TEMPLATE\LCID\STS\LISTS\DOCLIB	Contains files used for document libraries.	**.aspx* (form pages) **.HTM* (dialog boxes) *SCHEMA.XML* (schema file)
TEMPLATE\LCID\STS\LISTS\EVENTS	Contains files used for the Events list.	**.aspx* (form pages) *EVENT.ICS* (event form) *SCHEMA.XML* (schema file)
TEMPLATE\LCID\STS\LISTS\FAVORITE	Contains files used for the Favorites list.	**.aspx* (form pages) *SCHEMA.XML* (schema file)
TEMPLATE\LCID\STS\LISTS\GRIDLIST	Contains files used for the Datasheet view of lists.	**.aspx* (form pages) *SCHEMA.XML* (schema file)
TEMPLATE\LCID\STS\LISTS\IMGLIB	Contains files used for picture libraries.	**.aspx* (form pages) *SCHEMA.XML* (schema file)
TEMPLATE\LCID\STS\LISTS\ISSUE	Contains files used for the Issues list.	**.aspx* (form pages) *SCHEMA.XML* (schema file)
TEMPLATE\LCID\STS\LISTS\LISTTEMP	Contains files used for the List Template gallery.	**.aspx* (form pages) *SCHEMA.XML* (schema file)
TEMPLATE\LCID\STS\LISTS\TASKS	Contains files used for the Tasks list.	**.aspx* (form pages) *SCHEMA.XML* (schema file) *TASK.ICS* (task form)
TEMPLATE\LCID\STS\LISTS\VOTING	Contains files used for surveys.	**.aspx* (form pages) *SCHEMA.XML* (schema file)
TEMPLATE\LCID\STS\LISTS\WEBTEMP	Contains files used for the site template gallery.	**.aspx* (form pages) **.HTM* (dialog boxes) *SCHEMA.XML* (schema file)
TEMPLATE\LCID\STS\LISTS\WPLIB	Contains files used for the Web Part gallery.	**.aspx* (form pages) *SCHEMA.XML* (schema file)
TEMPLATE\LCID\STS\LISTS\WPLIB\DWP	Contains web part files.	**.dwp* (default web parts in the site collection web part gallery)
TEMPLATE\LCID\STS\LISTS\XMLFORM	Contains files used for form libraries.	**.aspx* (form pages) **.HTM* (dialog boxes) *SCHEMA.XML* (schema file)
TEMPLATE\LCID\STS\XML	Contains the available lists in the site template, base types for fields (onet.xml), and the standard view template for new views.	**.XML* (XML files for site schema and views)

\TEMPLATE\LCID\XML	Contains the XML files with base list and field types defined for all site templates.	*.XML (XML templates used in all site templates for a particular language)
\TEMPLATE\ADMIN\LCID	Contains files used for the site administration pages.	*.aspx, *.css, *.js (administration pages, styles, and JavaScript files)
\TEMPLATE\ADMIN\LCID\aspnet_client\system_web\Version	Contains ASP.NET files.	SmartNav.htm SmartNav.js WebUIValidation.js ASP.NET files
\TEMPLATE\ADMIN\LCID\BIN	Contains binary files used for the site administration pages.	*.DLL - Core binaries
\TEMPLATE\ADMIN\LCID\XML	Contains XML files used for the site administration pages.	adminleftnavbar.sts.xml (XML file for the left link bar (action menu and views list)) setuperror.htm (setup messages)
\TEMPLATE\IMAGES	Contains images shared by all pages on the server, addressed by the virtual directory /_layouts/images.	*.gif, *.jpg, *.png
\TEMPLATE\LAYOUTS	Addressed by the virtual directory /_layouts, this directory contains language subdirectories that contain the forms for creating lists, site administration pages, and so on. These directories are shared by all sites.	Global.asax (ASP.NET namespace definition) *.aspx (form pages) web.config (configuration file)
\TEMPLATE\LAYOUTS\LCID	Contains forms for creating lists, site administration pages, and so on, for a specific language.	*.aspx (form pages) *.css (style sheets) *.htm (dialog boxes) *.htc (menu control) *.js (JavaScript files) *.xml (XML templates) *.xsd (XML definitions)
\TEMPLATE\LAYOUTS\LCID\IMAGES	Contains images used in the default site pages for a specific language.	*.gif, *.jpg
\TEMPLATE\LAYOUTS\LCID\MPS	Contains form pages and scripting files for Meeting workspaces. Note This folder is added only if you have used a Meeting Workspace template.	*.aspx (form pages) MEETINGS.JS (JavaScript file)
\TEMPLATE\LAYOUTS\LCID\STYLES	Contains style sheets shared by all site templates for a particular language. Addressable by the virtual directory /_layouts/styles.	*.CSS (style sheets)
\TEMPLATE\LAYOUTS\BIN	Contains core binary files.	Microsoft.SharePoint.ApplicationPages.dll (core binary)
\TEMPLATE\SQL	Contains stored procedures for Microsoft SQL Server.	*.SQL (stored procedures for SQL Server)
\TEMPLATE\THEMES	Contains the list of themes.	THEMES.INF (themes list)

\TEMPLATE\THEMES\Theme	Contains files used by a specific theme.	**.gif* (images) **.css* (style sheets) *theme.INF* (theme definition file) *theme.utf8* (theme file for UTF8 encoding)
\TEMPLATE\XML	Contains XML files used by all site templates in all languages.	**.XML* (templates used across all languages and site types)
\TEMPLATE\XML\HELP	Contains XML files used by the Help system.	*STS.XML* (context-sensitive Help mapping file)

PREV NEXT

Content Not Stored in Database

Folder	Description	File
*inetpub**folder*	Configuration file	*web.config*
*inetpub**folder**_vti_pvt*	SpeedDial shortcuts	service
*inetpub**folder**wpresources*\	Configuration file for web parts and other resources.	*web.config*

PREV NEXT

Colophon

About the Author

Jeff Webb has written about computers and technology for 20 years. His books include *SharePoint Office Pocket Guide*; *Excel 2003 Programming: A Developer's Notebook*; *Using Excel Visual Basic for Applications*; *Visual Basic Developer's Workshop*; and *Developing Web Applications with Visual Basic .NET*. He has also written programming guides, articles, and sample applications for Microsoft and Digital Equipment Corporation.

Colophon

Our look is the result of reader comments, our own experimentation, and feedback from distribution channels. Distinctive covers complement our distinctive approach to technical topics, breathing personality and life into potentially dry subjects.

The animal on the cover of *Essential SharePoint* is a wombat. Averaging about 40 inches in length and weighing about 55 pounds, this Australian marsupial is the largest of the burrowing animals; as such, it is compared to the badger, but it is most closely related to the koala. Because wombats walk with an awkward waddle, they appear to be docile and slow, but are actually quite alert and agilewhen necessary, they can move over short distances with the speed of an Olympic sprinter. Although wombats are not particularly territorial, they do prefer solitude, so are known to mark their often-overlapping feeding grounds by rubbing trees, sometimes to a polished appearance, and to leave their distinctive cube-shaped dung atop elevated items such as rocks, mushrooms, and even upright sticks.

Jamie Peppard was the production editor and copyeditor for *Essential SharePoint*. Chris Downey proofread the book. Sanders Kleinfeld, Sarah Sherman, and Claire Cloutier provided quality control. Ellen Troutman Zaig wrote the index.

Emma Colby designed the cover of this book, based on a series design by Edie Freedman. The cover image is an original illustration from *Animate Creation*. Karen Montgomery produced the cover layout with Adobe InDesign CS using Adobe's ITC Garamond font.

David Futato designed the interior layout. This book was converted by Judy Hoer to FrameMaker 5.5.6 with a format conversion tool created by Erik Ray, Jason McIntosh, Neil Walls, and Mike Sierra that uses Perl and XML technologies. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSans Mono Condensed. The illustrations that appear in the book were produced by Robert Romano, Jessamyn Read, and Lesley Borash using Macromedia FreeHand MX and Adobe Photoshop CS. The tip and warning icons were drawn by Christopher Bing. This colophon was written by Jamie Peppard.

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

PREV NEXT

Index

Index

Index

Index

Index

Index

Index

PREV NEXT

Index

Index

Index

Index

Index

Index

PREV