Join the discussion @ p2p.wrox.com



Wrox Programmer to Programmer™

SharePoint® 2010 Field Guide

Steven Mann, Colin Murphy, Pablo Gazmuri, Chris Caravajal, Christina Wheeler

www.it-ebooks.info

www.it-ebooks.info

PROFESSIONAL SHAREPOINT® 2010 FIELD GUIDE

INTRODUCTION xxi

▶ PART I USING SHAREPOINT TECHNOLOGIES AND EXPLOITING FUNCTIONALITY

CHAPTER 1 Leveraging the SharePoint Object Model 3

CHAPTER 2 Creating Visual Web Parts 35

CHAPTER 3 Customizing Search 55

CHAPTER 4 Leveraging the SharePoint Lists Web Service 73

CHAPTER 5 Custom Field Types, Content Types, and List Definitions 93

CHAPTER 6 Workflow 131

CHAPTER 7 Administrating SharePoint 2010 with PowerShell 175

▶ PART II SHAREPOINT SOLUTIONS

CHAPTER 8 UI Design and Branding 211

CHAPTER 9 Building a Site Provisioning System 233

CHAPTER 10 Building Scalable Applications in SharePoint 297

CHAPTER 11 Building a Custom Global Navigation Solution 343

APPENDIX A URL Shortcuts 399

APPENDIX B CSS Quick Reference 401

APPENDIX C Hiding and Customizing Page Breadcrumbs 405

APPENDIX D PowerShell Quick Reference 407

GLOSSARY 409

INDEX 411

www.it-ebooks.info

www.it-ebooks.info

PROFESSIONAL

SharePoint® 2010 Field Guide

www.it-ebooks.info

www.it-ebooks.info

PROFESSIONAL

SharePoint® 2010 Field Guide

Steve Mann
Colin Murphy
Pablo Gazmuri
Chris Caravajal
Christina Wheeler



John Wiley & Sons, Inc.

www.it-ebooks.info

Professional SharePoint® 2010 Field Guide

Published by
John Wiley & Sons, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2012 by John Wiley & Sons, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-1-118-10505-4
ISBN: 978-1-118-22403-8
ISBN: 978-1-118-23767-0
ISBN: 978-1-118-26224-5

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at http://www.wiley.com/go/permissions.

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Web site is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Web site may provide or recommendations it may make. Further, readers should be aware that Internet Web sites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services please contact our Customer Care Department within the United States at (877) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at http://booksupport.wiley.com. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2011942765

Trademarks: Wiley, the Wiley logo, Wrox, the Wrox logo, Wrox Programmer to Programmer, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. SharePoint is a registered trademark of Microsoft Corporation. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc., is not associated with any product or vendor mentioned in this book.

www.it-ebooks.info

ABOUT THE AUTHORS

STEVE MANN was born and raised in Philadelphia, Pennsylvania, where he still resides today. He is an Enterprise Applications Engineer for Morgan Lewis and has more than 17 years of professional experience. Steve was previously a Principal Architect for RDA Corporation, where he worked for over thirteen years. Steve graduated from Drexel University in 1993. For the past eight years, he has focused primarily on collaboration and business intelligence solutions using Microsoft technologies. Steve managed the internal BI Practice Group at RDA for several years. He was also heavily involved within RDA's Collaboration/Search Practice Group. Steve has authored and coauthored several books related to the subject of SharePoint Server 2010. Steve's blog can be found at: `www.SteveTheManMann.com`. Steve enjoys vacationing with his family along the east coast, including locations such as Orlando, Florida; Williamsburg, Virginia; Ocean City and Sea Isle City, New Jersey; and New York City. They usually hits three or four locations each year. Follow Steve on Twitter @stevethemanmann!

COLIN MURPHY lives in Atlanta, Georgia, where he works as a Principal Architect for RDA Corporation, a Microsoft Gold Partner. He has spent the last fifteen years developing Web sites and Windows applications using a variety of languages including C++, PowerBuilder, Java, PHP, and Python, but has primarily been a .NET/C# developer for the last ten years. Colin has been developing enterprise solutions built on SharePoint since SharePoint 2007 was in beta. Colin holds a number of technical certifications including Microsoft Certified Professional, Microsoft Certified Application Developer, Microsoft Certified Solutions Developer, and is also a Microsoft Certified Technical Specialist for SharePoint. He has been recognized as a K2 Insider and is the coauthor of the book *Professional K2 blackpearl* by Wiley.

PABLO GAZMURI is a Principal Architect with RDA Corporation, an IT Consulting firm. He has led SharePoint efforts for a number of Fortune 500 companies where he conducts planning, mentoring, training, and coding activities. With fifteen years of development experience, Pablo is a web, collaboration, and search expert who has built numerous corporate intranets, extranets, ecommerce sites, and custom applications over the course of his career. Pablo has recently built custom SharePoint applications enabling project management, document generation, SharePoint site provisioning, governance enforcement, and more. You can find him on Twitter as @PGazmuri, or contact him through RDA at `http://www.rdacorp.com`.

CHRIS CARAVAJAL has been working exclusively with SharePoint technologies since 2007. As a consultant, he works with organizations on planning and architecting SharePoint solutions that vary in both size and complexity. Primarily an "out-of-the-box" specialist, Chris's main focus is getting the most out of SharePoint with the native features and tools. He also has experience building InfoPath forms and SharePoint Designer workflow solutions. Chris is a MCTS and is a contributing author to the top-selling SharePoint technical publication, "Professional SharePoint 2010 Administration."

www.it-ebooks.info

CHRISTINA WHEELER is a SharePoint Trainer for Mindsharp and founder of CM Portal Solutions, LLC. She is a highly respected SharePoint consultant who is extremely active in the SharePoint community. With over ten years of experience in the industry, Christina has knowledge and experience in graphic design, web development, and custom development. Her work is targeted primarily toward educational institutions, financial institutions, and telecommunications. As a trainer, Christina brings her real-world experience to the classroom. She enjoys contributing articles to technical Web sites and often speaks at SharePoint community events and conferences. You can follow her on Twitter as @cwheeler76.

ABOUT THE TECHNICAL EDITOR

REZA ALIREZAEI MVP, MCP, MCPD, MCITP & MCTS for SharePoint 2010, is a SharePoint solution architect focused on designing custom applications with SharePoint, Office, and Microsoft Business Intelligence technologies. Reza is the founder and president of Development Horizon, where he has helped many clients architect and build large-scale, mission-critical SharePoint applications. He also speaks in many local and international conferences. Reza achieved the status of Microsoft Most Valuable Professional (MVP) for SharePoint in 2006, which he still is today. He has also co-authored several books, papers, and articles. Reza can be reached at `reza@devhorizon.com`.

www.it-ebooks.info

CREDITS

ACQUISITIONS EDITOR
Paul Reese

PROJECT EDITOR
Christina Haviland

TECHNICAL EDITOR
Reza Alirezaei

PRODUCTION EDITOR
Daniel Scribner

COPY EDITOR
San Dee Phillips

EDITORIAL MANAGER
Mary Beth Wakefield

FREELANCER EDITORIAL MANAGER
Rosemarie Graham

ASSOCIATE DIRECTOR OF MARKETING
David Mayhew

MARKETING MANAGER
Ashley Zurcher

BUSINESS MANAGER
Amy Knies

PRODUCTION MANAGER
Tim Tate

VICE PRESIDENT AND EXECUTIVE GROUP PUBLISHER
Richard Swadley

VICE PRESIDENT AND EXECUTIVE PUBLISHER
Neil Edde

ASSOCIATE PUBLISHER
Jim Minatel

PROJECT COORDINATOR, COVER
Katie Crocker

COMPOSITOR
Craig Woods, Happenstance Type-O-Rama

PROOFREADER
Mark Steven Long

INDEXER
Robert Swanson

COVER DESIGNER
LeAndra Young

COVER IMAGE
© iStock / ulamonge

www.it-ebooks.info

www.it-ebooks.info

ACKNOWLEDGMENTS

THANKS TO CHRISTINA WHEELER for taking over one of the chapters in this book. I am so glad she joined the group of authors. Also, thanks to Chris Caravajal from SharePoint 911 for stepping in and taking a chapter as well. Thanks to all my SharePoint expert friends and Twitter followers. You inspire me to keep going and doing more with technology. Special thanks to Paul Reese, our acquisitions editor, and the entire Wrox project team for helping us create an excellent SharePoint 2010 resource.

www.it-ebooks.info

www.it-ebooks.info

CONTENTS

INTRODUCTION *xxi*

PART I: USING SHAREPOINT TECHNOLOGIES AND EXPLOITING FUNCTIONALITY

CHAPTER 1: LEVERAGING THE SHAREPOINT OBJECT MODEL **3**

Customizing SharePoint **3**
UI Components 4
Nonvisual Components 11
External Access 13
Putting It All Together **24**
A Blog Aggregator 24
Calendar Reminders 28
Scheduled Workflows 31
Summary **33**

CHAPTER 2: CREATING VISUAL WEB PARTS **35**

Creating Visual Web Parts **35**
Visual Web Part Project 35
Adding Controls 38
Adding Code 40
Deployment 41
Connecting Web Parts **43**
Creating an Interface 44
Creating the Provider Web Part 45
Creating the Consumer Web Part 46
Deploying the Solution 48
Connecting the Web Parts 48
Using SharePoint Controls **49**
Leveraging the XSLT Web Part **53**
Summary **54**

www.it-ebooks.info

CHAPTER 3: CUSTOMIZING SEARCH 55

Configuring and Creating Content Sources 55
Indexing Content 55
Planning Content Sources 56
Implementing Content Sources 57
Configuring Search Scopes 57
Defining Search Scopes in the Service Application 58
Defining Site Collection Search Scopes and Scope Display Groups 60
Creating Managed Properties 62
Customizing the Enterprise Search Center 63
Creating a New Search Results Page 65
Modifying the Advanced Search Screen 68
Finding Content with Alerts 70
Summary 71

CHAPTER 4: LEVERAGING THE SHAREPOINT LISTS WEB SERVICE 73

Web Services Overview 73
SharePoint Web Services Introduction 73
Instantiating a Web Service 73
Setting Credentials 76
Querying Lists and List Items 78
Modifying the SharePoint Location 78
Retrieving Lists from a SharePoint Location 79
Retrieving List Items from a SharePoint List 79
Modifying List Items 83
Creating a New Item 83
Updating an Existing Item 84
Deleting an Existing Item 86
Programmatically Uploading Files to SharePoint 89
Adding References 89
Create Upload Method 90
Create Web Request and File Buffer 90
Use Stream and FileStream Objects 90
Perform the Upload 91
Error Handling and Returns 91
Summary 92

www.it-ebooks.info

CHAPTER 5: CUSTOM FIELD TYPES, CONTENT TYPES, AND LIST DEFINITIONS 93

Field Types 94
Field Type Definition Components 95
Building a Restricted Social Security Number Field Type 99
Custom Content Types 114
Creating the Employee Details Content Type Stub 114
Implementing the Employee Details Content Type 116
List Definitions 120
Creating the Employees List Definition 120
Customizing the List Schema 122
Summary 129

CHAPTER 6: WORKFLOW 131

Workflow Introduction 131
Getting Started 132
Defining the Process 132
Types of Workflow Associations 133
Starting Workflows 134
Selecting Your Solution Method 134
SharePoint 2010 Native Tools 136
SharePoint Workflow Tools 136
SharePoint Workflow Templates 138
Microsoft Office Visio 2010 Premium 148
SPD 2010 151
Conditions 151
Actions 154
Workflow Options and Features 160
Visual Studio 2010 171
Summary 173

CHAPTER 7: ADMINISTRATING SHAREPOINT 2010 WITH POWERSHELL 175

Getting Started 176
Starting PowerShell 176
PowerShell Conventions 177
Command Discovery 178
Function Key Shortcuts 180
Command Prompt System Commands 180

www.it-ebooks.info

SharePoint Management with PowerShell **181**
SharePoint Farm Management 181
Web Application Management 184
Site Management 186
Solutions and Features 189
Enterprise Content Management 192
SharePoint Database Management 193
Search Administration with PowerShell **195**
SharePoint Foundation Search 195
Enterprise Search 196
Service Applications **199**
Access Services 200
Excel Services 200
InfoPath Forms Services 201
Metadata Service 202
PerformancePoint Services 202
Secure Store Service 204
State Service 205
User Profile Service 205
Visio Services 206
Word Services 207
Summary **208**

PART II: SHAREPOINT SOLUTIONS

CHAPTER 8: UI DESIGN AND BRANDING **211**

Themes **211**
Creating Themes 212
Using Themes 212
Configuring Themes from the Site User Interface 215
Deploying Themes Through a Feature 217
Branding with Custom CSS **217**
Themes and CSS 217
Building and Including Custom CSS 218
Using a Centered Fixed Width Design 218
Styling Individual Navigation Items 219
Changing SharePoint Search Icon with CSS 220
Master Pages **220**
Browser Support 221
Deploying Master Pages via Visual Studio 2010 221
Customizing the Ribbon 224

www.it-ebooks.info

Working with Page Layouts **225**
Creating Page Layouts Using SharePoint Designer 2010 226
Deploy Custom Page Layouts Through Visual Studio 2010 227
Branding Layouts Pages 228
Customizing Application Pages **229**
Creating Custom Application Pages 229
Registering Custom Application Pages from Receivers 231
Registering Custom Application Pages from Windows PowerShell 232
Summary **232**

CHAPTER 9: BUILDING A SITE PROVISIONING SYSTEM **233**

Sites and Site Collections **234**
Determining Your Site Taxonomy **235**
Example Taxonomy Matrix 235
Additional Factors 236
The Site Request Process **237**
An Example Site Request Process 237
Security Challenges 240
Our Example Solution 241
Alternatives in Implementation 242
Buildout **243**
Creating the Project Framework 244
Creating the Necessary Lists 244
Creating a Custom Feature Receiver 251
Building and Deploying the Solution 255
Creating a Custom List Item Form 256
Creating Form Helper Functions 258
Creating Form Controls 261
Building Out the Custom Form Code 265
Creating the Core Site Provisioning Functionality 276
Deploying the Solution 284
Next Steps **293**
User Experience 293
Site Request and Configuration 293
Additional Uses for Site Data 294
Summary **295**

CHAPTER 10: BUILDING SCALABLE APPLICATIONS IN SHAREPOINT **297**

General Performance Issues **298**
Memory, Disk, and Network Access 298
Algorithmic Efficiency 299

www.it-ebooks.info

Load Balancing and Farm Configuration **300**
Server Configuration Options 300
Load-Balancing Options 302
Database Allocation 304
Data Sharding 305
SharePoint Caching Options **306**
Page Output Caching 306
SharePoint Object Cache 309
The BLOB Cache 312
Caching in Code **316**
Application Object 316
Session Object 317
View State 317
HTTPContext 318
HTTP Cache 319
Static Variables 320
Local Files and the SharePoint Content DB 320
A Reusable Caching Method 321
Managing Large Lists **325**
Query Throttling 325
Indexed Fields 327
Configuring Views 328
Client-Side Optimizations **329**
Content Distribution Networks 329
Reducing Server Round Trips 330
Writing Efficient Code in SharePoint **331**
Adding and Retrieving List Data 331
Batch Operations 333
Supported Batch Operations 334
Multithreaded Operations 336
String Processing 338
Exception Handling 340
SPMonitoredScope 340
General Tips for Solving Hard Problems 341
Summary **342**

CHAPTER 11: BUILDING A CUSTOM GLOBAL NAVIGATION SOLUTION 343

Defining the Requirements **343**
Reviewing the Architecture **344**
Understanding OOTB SharePoint Navigation 345
Pieces of the Solution 346

www.it-ebooks.info

Creating the Solution **348**
Creating the Visual Studio Solution and Projects 348
Building the Navigation Menu UI 350
Building the Processing Layer 363
Building the Global Navigation Consumer 377
Building the Timer Job 394
Summary **398**

APPENDIX A: URL SHORTCUTS **399**

Site Management **399**
Galleries **400**
Page Maintenance **400**

APPENDIX B: CSS QUICK REFERENCE **401**

APPENDIX C: HIDING AND CUSTOMIZING PAGE BREADCRUMBS **405**

APPENDIX D: POWERSHELL QUICK REFERENCE **407**

Backup and Restore **407**
Features **407**
Form Templates **407**
Sites and Webs **408**
Solutions **408**
Users **408**

Glossary **409**

INDEX ***411***

www.it-ebooks.info

www.it-ebooks.info

INTRODUCTION

THIS BOOK WAS WRITTEN TO PROVIDE a wide coverage of complex solutions to common needs and desires from SharePoint 2010 implementations along with great reference material for digging in and learning the new platform; an excellent guide and handbook for people in the field. There are plenty of books to ramp up and learn how to do things with SharePoint; however, this book focuses on the main areas of customization and provides more intricate details as well as specific full-scale solutions that people need to implement. It is not a beginner how-to but a second or third book to further deepen one's understanding and knowledge in SharePoint 2010. It is meant to be a guide for those in the field that suddenly find themselves needing to create custom SharePoint solutions.

WHO THIS BOOK IS FOR

This book is for SharePoint or .NET Developers who now need to implement solutions in SharePoint 2010. These are developers that may have been exposed to some SharePoint development but haven't spent extensive time customizing solutions. Those that consider themselves more than beginner but not quite intermediate or advanced can benefit greatly from this book. If you need to become an expert in SharePoint 2010 solutions, this book takes you down that road.

WHAT THIS BOOK COVERS

This book covers the main aspects of customizing SharePoint from branding and creating web parts, to exploiting the SharePoint API and building workflows. PowerShell administration is discussed and explained. The Search features of SharePoint are demonstrated and customized. Full-scale solutions are dissected to present best practices and understanding of SharePoint development way beyond out-of-the-box.

HOW THIS BOOK IS STRUCTURED

There are two parts to this book. The first part starts out with the building blocks of SharePoint knowledge and customization. The second part gets deeper and more complex while building out applications and custom solutions.

WHAT YOU NEED TO USE THIS BOOK

In order to benefit the most from this book, readers need to have a SharePoint 2010 development environment along with Visual Studio 2010 Professional Edition (or higher). Usually a virtual-machine-based environment is more available than physical servers. SharePoint 2010 requires 64-bit hardware with at least 8GB of memory and the Windows 2008 R2 operating system.

www.it-ebooks.info

CONVENTIONS

To help you get the most from the text and keep track of what's happening, we've used a number of conventions throughout the book.

As for styles in the text:

- We *italicize* new terms and important words when we introduce them.
- We show keyboard strokes like this: Ctrl+A.
- We show file names, URLs, and code within the text like so: `persistence.properties`.
- We present code in two different ways:

```
We use a monofont type with no highlighting for most code examples.
```

```
We use bold to emphasize code that is particularly important in the present
context or to show changes from a previous code snippet.
```

SOURCE CODE

As you work through the examples in this book, you may choose either to type in all the code manually, or to use the source code files that accompany the book. All the source code used in this book is available for download at `www.wrox.com`. When at the site, simply locate the book's title (use the Search box or one of the title lists) and click the Download Code link on the book's detail page to obtain all the source code for the book. Code that is included on the Web site is highlighted by the following icon:



Listings include the filename in the title. If it is just a code snippet, you'll find the filename in a code note such as this:

Code snippet filename



Because many books have similar titles, you may find it easiest to search by ISBN; this book's ISBN is 978-0-470-10505-4.

Once you download the code, just decompress it with your favorite compression tool. Alternately, you can go to the main Wrox code download page at `www.wrox.com/dynamic/books/download.aspx` to see the code available for this book and all other Wrox books.

www.it-ebooks.info

ERRATA

We make every effort to ensure that there are no errors in the text or in the code. However, no one is perfect, and mistakes do occur. If you find an error in one of our books, like a spelling mistake or faulty piece of code, we would be very grateful for your feedback. By sending in errata, you may save another reader hours of frustration, and at the same time, you will be helping us provide even higher-quality information.

To find the errata page for this book, go to `www.wrox.com` and locate the title using the Search box or one of the title lists. Then, on the book details page, click the Book Errata link. On this page, you can view all errata that has been submitted for this book and posted by Wrox editors. A complete book list, including links to each book's errata, is also available at `www.wrox.com/misc-pages/booklist.shtml`.

If you don't spot "your" error on the Book Errata page, go to `www.wrox.com/contact/techsupport.shtml` and complete the form there to send us the error you have found. We'll check the information and, if appropriate, post a message to the book's errata page and fix the problem in subsequent editions of the book.

P2P.WROX.COM

For author and peer discussion, join the P2P forums at `http://p2p.wrox.com`. The forums are a Web-based system for you to post messages relating to Wrox books and related technologies and interact with other readers and technology users. The forums offer a subscription feature to email you topics of interest of your choosing when new posts are made to the forums. Wrox authors, editors, other industry experts, and your fellow readers are present on these forums.

At `http://p2p.wrox.com`, you will find a number of different forums that will help you, not only as you read this book, but also as you develop your own applications. To join the forums, just follow these steps:

1. Go to `p2p.wrox.com` and click the Register link.
2. Read the terms of use and click Agree.
3. Complete the required information to join, as well as any optional information you wish to provide, and click Submit.
4. You will receive an email with information describing how to verify your account and complete the joining process.



You can read messages in the forums without joining P2P, but in order to post your own messages, you must join.

www.it-ebooks.info

Once you join, you can post new messages and respond to messages other users post. You can read messages at any time on the Web. If you would like to have new messages from a particular forum emailed to you, click the Subscribe to this Forum icon by the forum name in the forum listing.

For more information about how to use the Wrox P2P, be sure to read the P2P FAQs for answers to questions about how the forum software works, as well as many common questions specific to P2P and Wrox books. To read the FAQs, click the FAQ link on any P2P page.

www.it-ebooks.info

PART I

Using SharePoint Technologies and Exploiting Functionality

- **CHAPTER 1:** Leveraging the SharePoint Object Model
- **CHAPTER 2:** Creating Visual Web Parts
- **CHAPTER 3:** Customizing Search
- **CHAPTER 4:** Leveraging the SharePoint Lists Web Service
- **CHAPTER 5:** Custom Field Types, Content Types, and List Definitions
- **CHAPTER 6:** Workflow
- **CHAPTER 7:** Administrating SharePoint 2010 with PowerShell

www.it-ebooks.info

www.it-ebooks.info

1

Leveraging the SharePoint Object Model

SharePoint provides a rich and complex object model for working with SharePoint data. Although it is challenging to master the details of the SharePoint object model, an even greater challenge that many developers face is taking their knowledge of the object model and using it to craft a solution that delivers on a set of requirements. By its nature, SharePoint solutions are often made up of loosely coupled components that combine to deliver a full set of functionality, and it is sometime difficult to figure out how to translate knowledge of the object model into the set of loosely coupled components in SharePoint. Ask three SharePoint developers how to solve a particular, nontrivial problem, and you will likely receive three unique solutions that might all be equally valid.

This chapter attempts to first cover, in broad strokes, the various customization mechanisms that SharePoint exposes to developers and then provides a number of sample problems and describes how they can be solved the "SharePoint Way." The chapter provides some basic code snippets illustrating how you can implement various features, but because this chapter is more about how components can be plugged together and combined into solutions, the coverage is not exhaustive.

CUSTOMIZING SHAREPOINT

SharePoint is a powerful platform that offers many points of extension and customization. You can't cover everything that SharePoint encompasses within a single chapter, but this chapter highlights some of the most common bits of functionality that you are likely to need to implement in your role as a SharePoint developer.

www.it-ebooks.info

UI Components

Master Pages and Themes

Master Pages and Themes are the primary mechanism in SharePoint to modify the look and feel of SharePoint. Master Pages can best be thought of as controlling the "edges" of a SharePoint page. Everything from the breadcrumb and up and from the quick launch and left is determined by the Out of the Box (OOTB), v4.master Master Page. This could be extended to control items to the right and below the primary content area of SharePoint pages as well. One common change typically implemented by deploying a custom Master Page is the addition of a standard footer to the bottom of pages in SharePoint.

Themes determine the color palette utilized by SharePoint. You can build themes using the Theme Designer available from Site Settings or also by using PowerPoint 2010 themes. In addition to developing a custom theme, you likely also need a custom .CSS file to provide the granular control most branding efforts require.



Chapter 8, "UI Design and Branding," provides a good overview of Master Pages, themes, and what goes into branding a solution.

Custom Web Parts

Web parts are the primary UI building block within SharePoint. They are modular elements that you can place on almost any page within SharePoint. SharePoint ships with a large number of web parts, but it will often be necessary to write your own.

Within SharePoint 2010, there are two types of web parts: visual web parts and traditional web parts. Back in the days of SharePoint 2007, there was only a single type of web part, the traditional web part. Unlike most other visual controls within Visual Studio, web parts did not have any sort of WYSIWYG designer, and developers had to build the UI via code. SharePoint 2010 introduces the ability to build visual web parts that enable the developer to use a WYSIWYG designer to build the UI.

Following is code for a traditional, "Hello World!" web part:

```
using System;
using System.ComponentModel;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using Microsoft.SharePoint;
using Microsoft.SharePoint.WebControls;

namespace Wrox.ObjectModel.TraditionalWebPart
{
    [ToolboxItemAttribute(false)]
    public class TraditionalWebPart : WebPart
```

www.it-ebooks.info

```
        {
            protected override void CreateChildControls()
            {
                Label label = new Label();
                label.Text = "Hello World!";

                Controls.Add(label);
            }
        }
    }
```

As stated earlier, the "visual" aspect of the new Visual Web Part simply refers to the design time experience. Rather than having to programmatically create the controls and adding them to the web part as in the preceding example, a Visual Web Part enables you to drag and drop controls onto the design surface. Figure 1-1 shows what a visual web part looks like in Visual Studio.

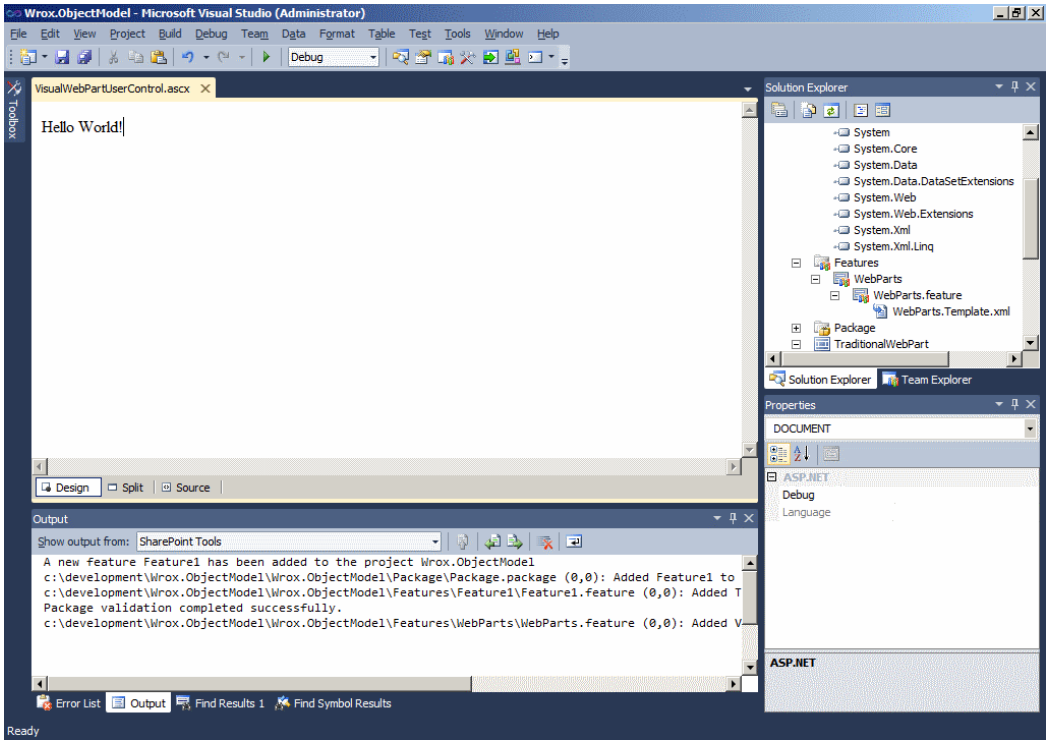


FIGURE 1-1: A Visual Web Part design canvas

When placed on a SharePoint page, the two web parts appear to be nearly identical, as shown in Figure 1-2.

www.it-ebooks.info

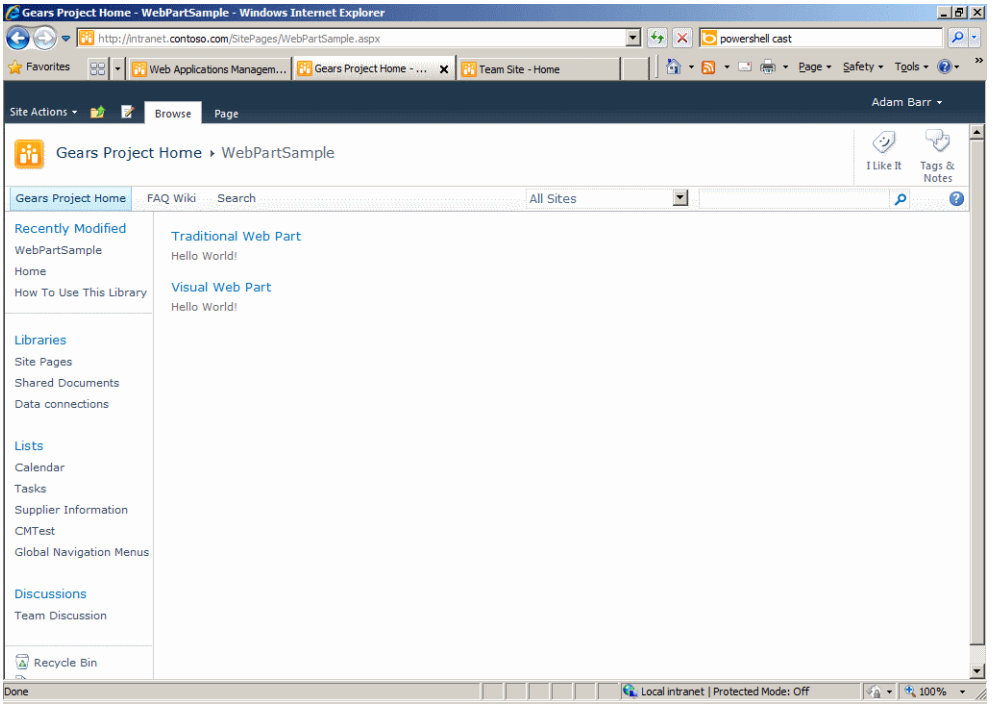


FIGURE 1-2: Web parts

You probably wonder why traditional web parts continue to exist within SharePoint 2010 because Visual Web Parts provide a better design experience. The answer is twofold. The first is to maintain backward compatibility and to allow solutions that can be deployed to both 2007 and 2010 environments. The second is that because of the way Visual Web Parts deploy, they are not supported within Sandboxed Solutions. There are some alternative implementations of Visual Web Parts, such as those found within Microsoft's Visual Studio 2010 SharePoint Power Tools, which do work for sandboxed solutions.

Web parts are primarily applicable when you need to implement functionality that requires a UI, is meant to be end-user configurable, and is meant for the site owners or designers to place on pages of their choosing. This may not always be the case however.



Chapter 2, "Creating Visual Web Parts," delves into Visual Web Parts in much greater depth.

Custom Application Pages

The other primary UI building block within SharePoint is the custom application page. These are standard .ASPX pages that are deployed beneath the `_layouts` directory within your SharePoint environment.

www.it-ebooks.info

Unlike web parts, custom application pages are standalone pages of self-contained functionality. Users cannot edit the contents of the page or add custom web parts. If you don't want the user to determine the placement of the control, or you need to link to the page from another solution element (for example, a custom action, web part, and so on), application pages are generally the right solution to the problem.

A common use for custom application pages is for settings pages, which is probably the scenario in which you see them most used within the OOTB SharePoint screens. For example, go to the site settings section of any site in SharePoint, and every one of the settings links take you to an application page.

Creating a custom application page is as simple as selecting Add ➪ New Item within your project, and selecting the Application Page type. This creates an .ASPX page beneath the Layouts directory of your project, as shown in Figure 1-3.

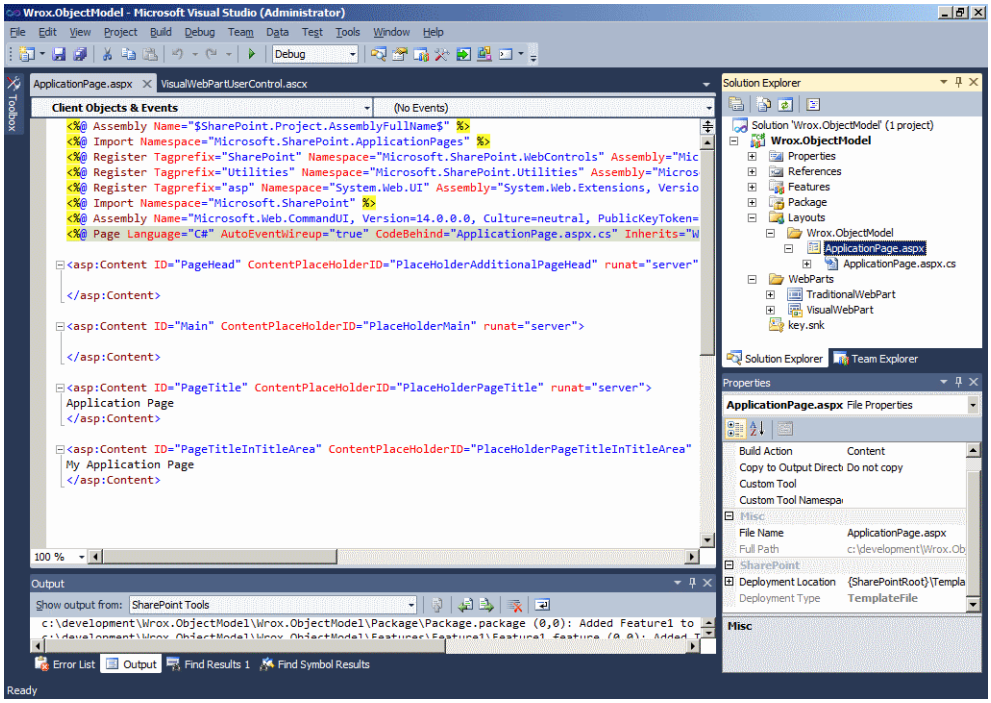


FIGURE 1-3: Application page in Visual Studio

As you can see in the screenshot, application pages are just like any traditional .ASPX page written in .NET with a few references added to SharePoint assemblies. Just like the other pages within SharePoint, application pages make use of a Master Page, and content placeholder regions are exposed for you to add content to. Figure 1-4 shows a sample application page.

Application pages are deployed to the Layouts directory and, as such, are not deployed as part of a feature. This means that application pages are accessible from within any site within your farm just by appending _layouts\<path to page> and SharePoint does not secure them assuming the user has access to the site, so you need to check user access within your own code.

www.it-ebooks.info

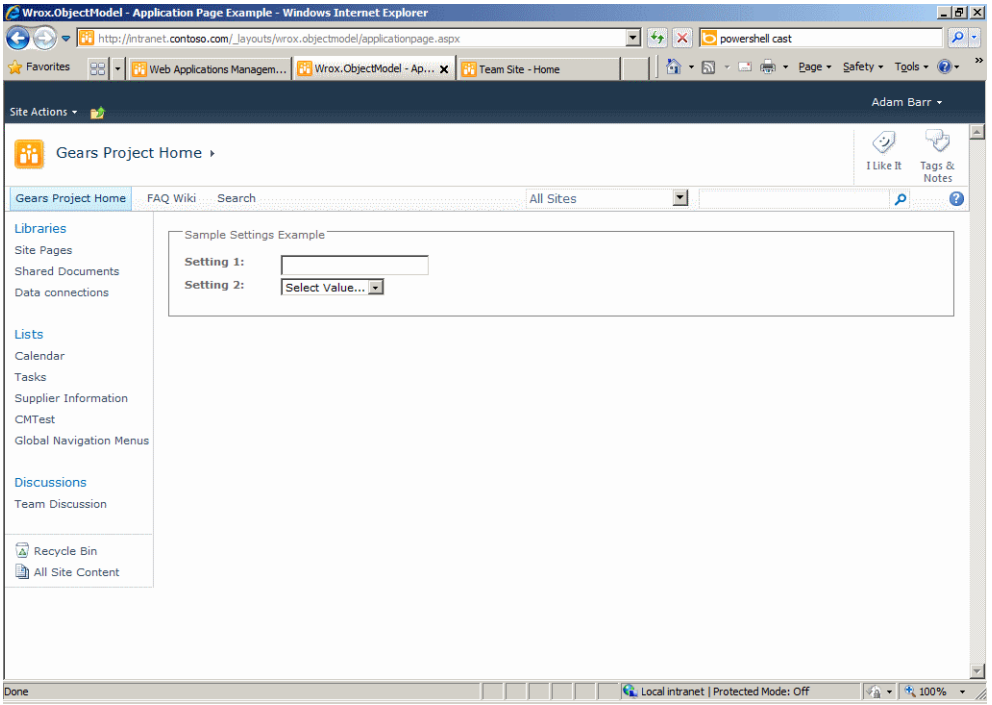


FIGURE 1-4: Application page in SharePoint

Custom Lists

SharePoint Lists provide a flexible and extensible way for users to store data within SharePoint and is frequently be used by developers as a location to store information as well. In many instances, these types of "configuration" lists are hidden from the user, and all interactions with the list data occurs through UI elements such as web parts and application pages, but there are cases, particularly when the user must maintain lists of data that you want to expose the list directly to the user.

In those cases, it may be sufficient to simply link them directly to the list and use the default list UI to allow the users to manage the list items. In more complicated scenarios (consider an example in which you have complex multicolumn validation requirements or in which you have multilevel drop-downs) the OOTB UI provided by lists is insufficient.

Custom List Forms

List forms refer to the add, edit, and display forms accessible from SharePoint when a user adds a new list item or clicks on an existing list item and chooses to view or edit it. Typical scenarios in which you might need to customize these would be if you need to add complex validation logic that spans across multiple columns in the list or in which selections made to one column affect those available in another. However, list forms are only one way in which you can enter list data within a list. Users interacting with a list through the datasheet view, through the office information panel, or through Access cannot enjoy the same experience, so you need to account for these usage scenarios.

www.it-ebooks.info

Chapter 5, "Custom Field Types, Content Types and List Definitions," covers lists and list forms in great detail and illustrates how to create custom list forms, so they will not be covered further here, but 2010 provides a number of mechanisms for customizing these forms: You can use SharePoint Designer, InfoPath, and Visual Studio to create more robust list forms.

Custom Field Types

Field types are the basic building blocks of lists and content types. Whenever you add a new column to a SharePoint list, you select from a variety of predefined column types such as Currency, Number, and Choice. By selecting these types, you can decide what types of data can be entered and how that data displays and can be manipulated. Figure 1-5 shows a standard column creation dialog.

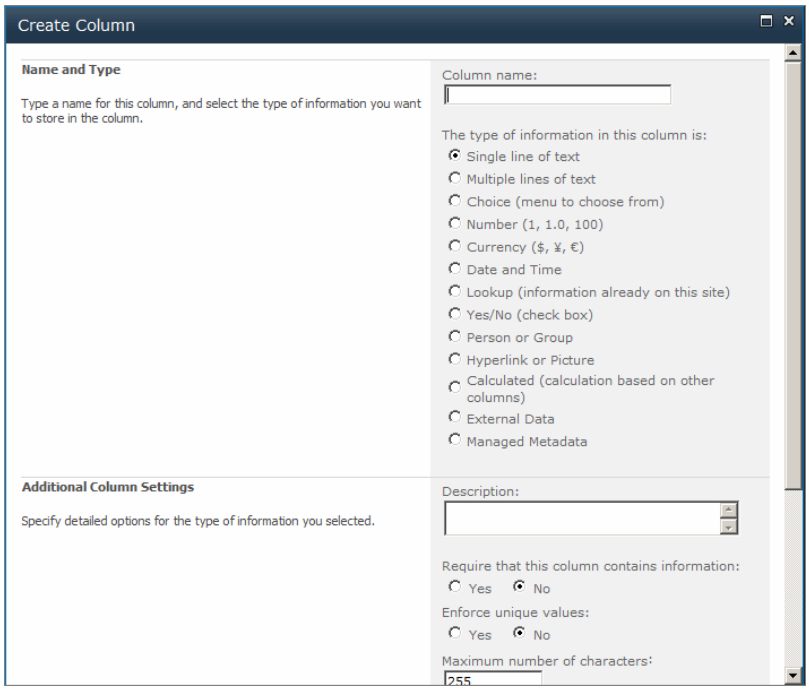


FIGURE 1-5: Creating a column in SharePoint

Creating custom field types adds to the list of standard types of new columns and enables you to dictate how information displays and interacts within the standard SharePoint list UI. They are most appropriate when you need to implement rules about the types of data fields can contain or the formatting of the display of your data (masking a Social Security number, for example).



Chapter 5 discusses creating custom field types in greater detail.

www.it-ebooks.info

Delegate Controls

SharePoint supports delegate controls, which might best be described as functionality placeholders. Developers can register their own implementation of the functionality that overrides the OOTB implementation. You can embed many delegate controls within the default Master Page and include things such as the search box.

As example of how to embed a delegate control in a Master Page is shown in the following code snippet:

```
<SharePoint:DelegateControl ControlId="SmallSearchInputBox"
    AllowMultipleControls="false"/>
```

This snippet basically tells SharePoint that the developer wants to embed a control called `SmallSearchInputBox` into this location, but the developer requires no knowledge of the implementation of the control.

Delegate controls can also have a default implementation specified directly within the body as well. Following is an example in the `TopNavigationDataSource` delegate control from the v4.master:

```
<SharePoint:DelegateControl runat="server"
    ControlId="TopNavigationDataSource" Id="topNavigationDelegate">
    <Template_Controls>
        <asp:SiteMapDataSource
            ShowStartingNode="False"
            SiteMapProvider="SPNavigationProvider"
            id="topSiteMap"
            runat="server"
            StartingNodeUrl="sid:1002"/>
    </Template_Controls>
</SharePoint:DelegateControl>
```

You can override this default implementation of a delegate control by deploying a feature that contains a control definition that uses the same ID specified in the `ControlId` attribute of the delegate control. The elements file would look something like this:

```
<?xml version="1.0" encoding="utf-8" ?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Control
    Id="SmallSearchInputBox"
    Sequence="25"
    ControlClass="WroxSearchBox"
    ControlAssembly="Wrox.ObjectModel"/>
</Elements>
```

The preceding definition basically tells SharePoint that this control (`WroxSearchBox`) should be used anywhere the `SmallSearchInputBox` delegate control is requested. The `"AllowMultipleControls"` attribute of the `DelegateControl` determines what happens if multiple implementations of the control exist. If it is set to `true`, all controls will be included in the order of their sequence number (lowest to highest). If it is `false`, only the control with the lowest sequence number is used.

Delegate controls are most useful when you need to replace a bit of OOTB functionality that exists within multiple locations in SharePoint such as the search box, navigation providers, and so on. Unfortunately, not every control in the OOTB Master Pages uses delegate controls, so their use is limited to specific controls.

www.it-ebooks.info

Following is a list of the delegate controls that exist within the v4.master Master Page:

- `AdditionalPageHead`
- `GlobalNavigation`
- `GlobalSiteLink0`
- `GlobalSiteLink2`
- `GlobalSiteLink3`
- `PublishingConsole`
- `SmallSearchInputBox`
- `TopNavigationDataSource`
- `QuickLaunchDataSource`
- `TreeViewAndDataSource`

Some of these delegate controls, such as the `TopNavigationDataSource` control, are embedded within content placeholders of the Master Page. This means that page layouts can replace the entire contents of this section and remove the delegate control definition. This is an issue covered in more depth when discussing implementing a global navigation solution in Chapter 11, "Building a Custom Global Navigation Solution."

Nonvisual Components

Visual components are only half of the story with most SharePoint customizations. Equally necessary is the ability to implement nonvisual functionality, such as executing code at periodic intervals (a timer job) or executing code after an item is added to a list (event handlers or workflow).

Event Handlers

You can attach event handlers in SharePoint to lists and document libraries and can trigger them before adding, updating, or deleting items or after you add, update, or delete items. The event handlers that occur before the event end in "ing" (for example, *ItemAdding*), whereas the event handlers that occur after an event occurred end in "ed" (for example, *ItemAdded*).

Within the "ing" events, the developer can stop the action from occurring. So for example, if you need to implement a rule that enforces that items can be added to a list, but can never be removed, you could do that with an `ItemDeleting` event handler that cancels the action.

Workflow

SharePoint 2007 supported a single type of workflow called a list workflow. These were workflows associated with a list that you could manually trigger when you created an item and when you updated an item. Unlike event handlers, you can trigger list workflows only after an action occurs and only on item addition or update. You cannot trigger event handlers on a deletion.

www.it-ebooks.info

Because there is considerable functionality overlap, determining when to write a list workflow versus writing an event handler is confusing. Workflows provide some advantages during the creation process because codeless workflows can be authored via SharePoint Designer, and workflow actions exist so that developers don't need to write custom code. But assuming you need to write code with either implementation, event handlers are best used for short-running, atomic processes that do not require user involvement and are tied to the specific list in question. Event handlers are also the only option if you want to intercept an action before it happens. If the requirement being met involves long-running processes that involve waiting for multiple inputs from users or which are triggered from multiple locations, a workflow is more likely to be the correct solution.

SharePoint 2010 also introduces a new type of workflow called a site workflow. Site workflows are tied to sites rather than individual lists and can be only manually triggered (or via code). Site workflows are primarily useful when the workflow in question is not tied to actions around a particular item in a list. Imagine a scenario in which you need to find a single item in a list based on some characteristic and then email the creator of that item. In that case, because you want the workflow to find the item, a site workflow is appropriate. If instead you want to email the creator of an item every time it updates, a list workflow is appropriate.



Chapter 6 covers workflow in greater detail.

Timer Jobs

Timer jobs are SharePoint's mechanism to enable you to run code on a scheduled basis. You can schedule timer jobs to run anywhere from every minute to every month.

Timer jobs are the preferred mechanism any time you must run code on a scheduled basis or where operations are particularly long running. An example of a sample scenario in which a timer job might be appropriate might be if you have a requirement in which every month you want to scan all MySites sites and find any files older than 90 days to confirm if you can delete them.

Chapter 11 contains an example of a timer job used to process links contained within a list to retrieve access information to secure the items.

Feature Receivers

Feature receivers are just an alternative form of an event receiver tied to SharePoint Features rather than SharePoint Lists. Feature receivers enable the developer to run arbitrary code whenever a feature is installed or removed from a farm/web application/site collection/web, whenever a feature is activated or deactivated, or when a feature is upgraded.

One common use for feature receivers is to register timer jobs deployed by a feature. Following is an example of what code to add for a timer job whenever the feature activates:

```
public override void FeatureActivated(
    SPFeatureReceiverProperties properties)
{
    SPSite site = (SPSite)properties.Feature.Parent;
    CustomTimerJob smbJob = new CustomTimerJob("My Job Name", site);
```

www.it-ebooks.info

```
        SPMinuteSchedule schedule = new SPMinuteSchedule();
        schedule.BeginSecond = 0;
        schedule.EndSecond = 59;
        schedule.Interval = 2;
        smbJob.Schedule = schedule;
        smbJob.Update();
    }
```

Another sample use would be if a feature depends on certain data, such as a SharePoint subsite existing, prior to it properly working. Your feature receiver could create the required subsite as part of the activation process.

External Access

Up to this point, this chapter has focused only on solutions that exist within SharePoint and run within the farm, but it is also common to need to access SharePoint content and interact with SharePoint from outside of the farm. One example of such a need would be an application that enables you to scan documents from your desktop and store them directly within SharePoint.

SharePoint provides two mechanisms to support external access. The first, which existed in SharePoint 2007 and continues to exist in 2010, are web services. The second is Client Object Model, which is new to SharePoint 2010, which you can access from .NET, Silverlight, or JavaScript.

The main advantage to using the Client Object Model over the web services is that rather than having to learn a completely new way to access SharePoint content, developers can reuse much of their knowledge of the server-side object model. There definitely are some differences, but in general, the Client Object Model provides a much more familiar framework.

SharePoint Web Services

SharePoint provides a rich set of web services (both .ASMX and WCF/RESTful) to enable external applications to interact with SharePoint. Although not everything is exposed via web services, a great deal of functionality is. For those writing code in .NET, Silverlight, or Javascript, the Client Object Model introduced in 2010 will likely be the preferred mechanism for interacting with SharePoint, but for developers writing in other languages, web services continue to be the primary mechanism used to interact with SharePoint.

The list of web services SharePoint provides follows:

- Admin
- Alerts
- Authentication e
- BDC Admin
- Cell Storage
- Copy
- Diagnostics
- Document Workspace

www.it-ebooks.info

- Forms
- Imaging
- Lists
- Meetings
- People
- Permissions
- Shared Access
- Distribution List
- Site Data
- Sites
- Search
- User/Group
- Versions
- Views
- Web Part Pages
- Webs
- Organization Profile Service
- Published Links
- Social Data
- User Profile

Covering each one of these services is outside of the scope of this chapter, but you can benefit from learning the capabilities of these web services. As mentioned, Microsoft recommends using the managed Client Object Model whenever possible instead of the web services. Chapter 4, "Leveraging the SharePoint Lists Web Service," focuses on just one of these web services, the List Web Service.

Client Object Model

The Client Object Model is a set of APIs that enable you to design custom applications that access SharePoint content. It includes libraries for client applications based on the .NET Framework. This new API is targeted for building things such as console, Windows Forms, and WPF applications. The Client Object Model also includes a library for Silverlight and JavaScript client applications. The Silverlight library is composed of a subset of the object model. These interfaces include the Managed .NET Client Object model, Silverlight Client Object, ECMAScript (JavaScript, Jscript) Client Object Model, and LINQ to SharePoint.

The .NET Client Object Model for SharePoint 2010 is one of the newest APIs for working with SharePoint content. The Client Object Model API enables you to build custom applications in any of

www.it-ebooks.info

the Managed .NET languages using an object-oriented approach. This new API is the ideal method to access and manipulate SharePoint 2010 content from a client application. The new APIs were designed to have counterparts to many of the types of objects that you have been using from the `Microsoft.SharePoint` server API. However, some objects from the server API provided with limited functionality in the Client Object Model APIs.

Using the new object model is fairly straightforward. You begin by creating a client context object like you would if you were using the server API. From there, you can load, create, and manipulate the core components of SharePoint: sites, webs, lists, and libraries. You can, of course, access the children of these objects. Depending on the client that you develop, you can use methods to synchronously or asynchronously perform operations against these objects, which enables you more control of your application's user experience. For Silverlight and JavaScript client, you can only asynchronously perform actions, whereas .NET clients enable only synchronous operations.

As previously mentioned, Client Object Model APIs exist for .NET, Silverlight, and ECMAScript. When you use the API for any of these languages, the syntax looks similar, but there may be some minor differences depending on the language. In the following sections you learn how to use the Client Object Model to load sites and to manipulate lists and libraries.

Working with the ClientContext

The most basic building block of content access from standalone applications to SharePoint is the `ClientContext` class. This class is similar to the `SPContext` server class. The `ClientContext` class is responsible for making connections to sites, executing queries, fetching lists, and performing all other actions in SharePoint 2010.

To create a client context for a SharePoint 2010 site located on the local server, use the follow code:

```
ClientContext clientContext = new ClientContext("http://localhost/")
```

This code assumes that your user logged in with an account that has SharePoint permissions. This new client context allows you to access SharePoint 2010 objects. This newly created client context will use the credentials of the user running the application. In some scenarios, this can be a problem. The user account that you need to use for SharePoint could be different from the account you use to log into the computer. In this case you can specify the credentials that your application needs to use. To replace the default network credentials with custom ones you use code like the following:

```
NetworkCredential credential = new NetworkCredential("Administrator", "Password",
                                                     "TestDomain");
clientContext.Credentials = credential;
```

This creates a new network credential and sets the credentials for the client context to them. If you use forms-based authentication, you also need to change the `AuthenticationMode` property of the `ClientContext` object to `ClientAuthenticationMode.FormsAuthentication` to make this work.

Your next step toward accessing content with the Client Object Model is to load a web or multiple webs into the context you have created. The Client Object Model does not load any content until it is explicitly requested. To request that the client context load a SharePoint 2010 client object, you must write lines of code that specify what objects to load. To add objects to load into the context, you first access the property from the client context that you want loaded. Next, you call the `Load` method on

www.it-ebooks.info

the client context with the property as a parameter. Finally, you must call the `ExecuteQuery` method on the client context to send the request to SharePoint.

The Client OM enables you to load multiple objects by calling `Load` multiple times. When this is done, all requests are batched and performed using specialized WCF services. This need to explicitly load objects or queries into the context serves a vital purpose. The `Load` method consolidates multiple requests together to reduce network traffic and improve performance.

Working with Sites and Webs

Now that you have a client context, look at the things you can do with it. For starters, you can load the root web and its child webs. To do this, use the following code:

```
Web rootWeb = clientContext.Web;
clientContext.Load(rootWeb);
clientContext.Load(rootWeb.Webs);
clientContext.ExecuteQuery();
```

This loads the entire web and its properties except for the `EffectiveBasePermissions`, `HasUniqueRoleAssignments`, and `RoleAssignments` properties. If you need to work with any of these properties, you must explicitly request them. If you are concerned with reducing unnecessary data transfer between your client application and the server, you should request only the specific properties that your application will use. For example, if you know that you want to use only the title of a Web site, you would use the following code to load only the title property:

```
Web oWeb = clientContext.Web;
clientContext.Load(oWeb, web=>web.Title);
clientContext.ExecuteQuery();
```

Using the Client OM, you can also change many properties of an existing web. For instance, you can update a web and change the title and description. To do so, access the web you want to change, modify the properties, and update the web. Because this is the Client OM, you must always call `ExecuteQuery` to send the request to the server. Code to do this would look like the following:

```
Web oWeb = clientContext.Web;
oWeb.Title = "Updated Web Title";
oWeb.Description = "This is a sample of updating a web";
oWeb.Update();
```

You can also create Web site objects with the Client OM. To do this use the `WebCreationInformation` class. You need to set the various properties on the `WebCreationInformation` object and add it to the web's collection. The web's collection you add the new creation object to becomes the parent web for your new one. To add a new blog site use the following code:

```
Web oWeb = clientContext.Web;

WebCreationInformation webBlogCreate = new WebCreationInformation();
webBlogCreate.Description = "This is a new Blog Site";
webBlogCreate.Language = 1033; //English Language code is 1033
webBlogCreate.Title = "New Blog Site";
webBlogCreate.Url = "newblogsite";
webBlogCreate.UseSamePermissionsAsParentSite = true;
```

www.it-ebooks.info

```
webBlogCreate.WebTemplate = "BLOG#0";

Web oNewWeb = oWeb.Webs.Add(webBlogCreate);

clientContext.ExecuteQuery();
```

In the previous example, the value for the `Language` property is `1033`. This is the Microsoft locale code for the English – United States locale. If you create a site for another locale, you need to replace `1033` with the locale code for your locale. A value for the `URL` of the new site is set. You do not need to put in a full URL here; this needs to be only the final part of the URL. Finally, set the `WebTemplate` property. In this case you use the string `BLOG#0`, but you can use any template in your SharePoint 2010 environment. Because this property is just a string, you can even use a custom web template.

Now that you have some understanding of the basic building blocks of the Client Object Model, look at using them to access lists.

Working with Lists and Libraries

Lists and document libraries are a key component of the content stored in SharePoint. Lists and libraries are quite similar in design, but they certainly hold different types of information. Lists generally hold data that you can represent in a spreadsheet or tabular manner, whereas libraries hold documents and metadata about those documents. In the following sections you learn how to work with list and library data. You also learn how to create new lists and libraries.

Working with Lists

If you have worked with SharePoint before, lists are probably not new to you. They are one of the main components for storing data in SharePoint. This section discusses how to create, update, and delete lists and list items.

Managing Lists

Before starting to work with list data, you need to create a custom list to work with. This is certainly not required because most SharePoint 2010 site templates come with predefined lists. It is, however, a common development task to create and modify lists. To create a list, use the client context you created in the last section. You also use a new class called `ListCreationInformation`. To create a list, you create a `ListCreationInformation` object and set the `Title` and `TemplateType` properties. The `Title` property is just the string title that you want to use for the name of the list. The `TemplateType` property is an integer representing the template to use for your new list instance. You can retrieve template types from the `ListTemplateType` enumeration and cast to an integer, or you can enter them directly as an integer. The following code creates a list using the announcements template and titled Custom Announcements List:

```
ListCreationInformation listCreationInfo = new ListCreationInformation();
listCreationInfo.Title = "Custom Announcements";
listCreationInfo.TemplateType = (int)ListTemplateType.Announcements;

List oList = oWebsite.Lists.Add(listCreationInfo);

clientContext.ExecuteQuery();
```

www.it-ebooks.info

You can also update properties of the list with the Client OM API. To do this you need to load the list using the client context. You can do this with a LINQ to objects query or CAML query, or by calling the `GetByTitle` method on the `clientContext.Webs.Lists` object and passing the title of the list. The `GetByTitle` approach has the benefit of loading only the list that you modify instead of loading all the lists in your query. After you load the list, you can set the property that you want to update, and call the `Update` method on the list object. Of course, you also must call `ExecuteQuery` on the client context to send the update to SharePoint, as follows:

```
List oList = clientContext.Web.Lists.GetByTitle("Custom Announcements");
oList.Description = "This is the new Custom Announcements List";

oList.Update();
clientContext.ExecuteQuery();
```

Another common list management task is adding a field to an existing list. To perform this task, you need to get a reference to the list object that you want to add the field to. You can add a field with any of the types available from the user SharePoint 2010 interface. You can then use the `AddField` or `AddFieldAsXml` method to add a field. If you need to add attributes to your new field, you can also do that. The following code creates a new number field and sets the minimum and maximum values for the field.

```
List oList = clientContext.Web.Lists.GetByTitle("Custom Announcements");
Field oField = oList.Fields.AddFieldAsXml(
     "<Field DisplayName='Percent Complete' Type='Number'/>",
     true, AddFieldOptions.DefaultValue);

FieldNumber fieldNum = clientContext.CastTo<FieldNumber>(oField);
fieldNum.MaximumValue = 100;
fieldNum.MinimumValue = 0;

fieldNum.Update();

clientContext.ExecuteQuery();
```

The final list management task discussed is deleting a list. Deleting a list with the Client OM is simple. As with adding a field or modifying a list, you start by getting a reference to the list you want to delete. Then you call the `DeleteObject` method on that object. Finally, you call `ExecuteQuery` on the client context to perform the request.

```
List oList = clientContext.Web.Lists.GetByTitle("Custom Announcements");
oList.DeleteObject();

clientContext.ExecuteQuery();
```

Now that you have learned how to create, modify, and delete SharePoint lists, you learn how to work with list data. As you will see in the next section, this is similar to the creating of lists from this section.

Adding List Items

Creating list items with the Client OM is straightforward. To create a list item and add it to the list, you use the `ListItemCreationInformation` class. After you instantiate a `ListItemCreationInformation`

www.it-ebooks.info

object, you need to add it to a list object using the `AddItem` method on the list. Of course, that means that you need to get or load the list that you want to modify. With the list item creation information added to the list, you can set the various field values for the list item and call the `Update` method on the list item. Finally, call the `ExecuteQuery` method on the client context to send your changes to the SharePoint server. The code to perform follows:

```
List oList = clientContext.Web.Lists.GetByTitle("Custom Announcements");
ListItemCreationInformation itemCreateInfo = new ListItemCreationInformation();
ListItem oListItem = oList.AddItem(itemCreateInfo);
oListItem["Title"] = "New List Item";
oListItem["Body"] = "This is my new List Item";

oListItem.Update();
clientContext.ExecuteQuery();
```

Updating List Items

Updating list items is easy with the Client OM. All you need to do is get the list item you want to update, set the fields to update, and call the `Update` method on the list item. The list properties are accessed and modified with the indexer for the list item in the same way as if you were modifying a dictionary object. The following code gets the third list item in the `Custom Announcements` list and updates the `Title` property:

```
List oList = clientContext.Web.Lists.GetByTitle("Custom Announcements");
ListItem listItem = oList.Items.GetById(3);

listItem["Title"] = "My Updated Title.";
listItem.Update();

clientContext.ExecuteQuery();
```

You can also update more than one list item at a time with only a single call to execute the query. To do this, you need to get the list items that you want to update. The easiest way to get multiple items is to use the `GetItems` method. This method takes a `CamlQuery` as a parameter. To illustrate this, you can load all the items that have a completed percentage of 50 and update them to 100 like this:

```
CamlQuery query = new CamlQuery();
query.ViewXml = "<View><Query><Where><Eq>"+
"<FieldRef Name='Percent Complete'/><Value Type='Number'>" +
"50</Value></Eq></Where></Query>" +
"<RowLimit>2</RowLimit></View>";

ListItemCollection collListItem = list.GetItems(query);
context.Load(collListItem);
context.ExecuteQuery();

foreach (SPCL.ListItem item in collListItem)
{
item["Percent Complete"] = 100;
item.Update();
}

context.ExecuteQuery();
```

www.it-ebooks.info

Deleting List Items

Deleting a list item is similar to deleting a list. All you need to do is get the list item you want to delete and call the `DeleteObject` method on the list item. The following code gets the third list item in the `Custom Announcements` list and deletes it:

```
List oList = clientContext.Web.Lists.GetByTitle("Custom Announcements");
ListItem listItem = oList.Items.GetById(3);

listitem.DeleteObject()

clientContext.ExecuteQuery();
```

You can also delete more than one list item at a time with only a single call to execute the query. To do this, you need to get the list items that you want to delete. As you saw in the previous section, the easiest way to get multiple items is to use the `GetItems` method. The following code fetches all list items that have `Percent Complete` of 100 and deletes them:

```
CamlQuery query = new CamlQuery();
query.ViewXml = "<View><Query><Where><Eq>"+
"<FieldRef Name='Percent Complete'/><Value Type='Number'>" +
"100</Value></Eq></Where></Query>" +
"<RowLimit>2</RowLimit></View>";

ListItemCollection collListItem = list.GetItems(query);
context.Load(collListItem);
context.ExecuteQuery();

foreach (SPCL.ListItem item in collListItem)
{
item.DeleteObject();
}

context.ExecuteQuery();
```

Querying Lists

Now that you know how to access lists, consider how to access subsets of data. There are two ways to query lists with the Client OM. You can write a LINQ to objects query, or you can create a CAML query. Because you can create queries for the Client OM in two ways, you probably wonder how to decide when to use one or the other. As a general rule LINQ style queries are much easier to create. If you have used LINQ to SQL, you are already familiar with the basics. So then why use a CAML query?

CAML style queries are not as easy to create as LINQ to objects queries. CAML queries are, however, much faster with the Client OM than LINQ queries are. This performance improvement is because LINQ queries are performed on a frontend web server, whereas CAML queries are performed directly against the database. In addition, you cannot query some objects with LINQ. For instance, you cannot query list items with LINQ directly. In this case you must perform a CAML query.

The LINQ queries are not using the new LINQ to SharePoint provider. LINQ to SharePoint is available only when you program against the server object model, which is discussed later in this chapter.

www.it-ebooks.info

Querying with LINQ

If you are familiar with LINQ syntax, you probably know that LINQ can be expressed as method syntax or query syntax. For those of you new to LINQ, a brief explanation of LINQ syntax is given. Query syntax, also called queryable load, is the style of LINQ that looks similar to SQL expressions. It contains familiar SQL keywords such as *from, in, where,* and *select,* but the syntax does slightly differ. Still, those familiar with writing SQL queries can quickly pick up the new syntax. If you want to write a LINQ expression in query syntax to load the list in the root web with the name `Announcements`, it would look like this:

```
var query = from list
      in clientContext.Web.Lists
      where list.Title == "Announcements"
      select list;

var result = clientContext.LoadQuery(query);
clientContext.ExecuteQuery();
```

Method syntax, however, looks different. A LINQ expression written in method syntax, also called in-place load, looks like multiple method calls. The previous query in method syntax would look like this:

```
clientContext.Load(clientContext.Web,
website => website.Lists.Include(
      list => list.Title).Where(
               list => list.Title == "Announcements"));

clientContext.ExecuteQuery();
```

You have seen the two styles of LINQ that you can use. Why is this important? The style of LINQ that you use determines how the Client OM loads your objects. If you use the query style syntax, the results of your query are stored in an object instead of in the client context. This is called a queryable load. The method syntax query causes the Client OM to perform an in-place load, meaning that the objects load in the client context. In-place loads keep data in the client context through subsequent loads. In a queryable load, you are responsible for keeping the results in the application because they are not stored in the client context. The benefit of the queryable load is that you can perform multiple queries that each return different results and keep the data separate. If you perform multiple queries using the in-place loads and then you looped through the results in the client context, you would loop over records that do not match your latest query.

Querying with CAML

Querying items with LINQ is nice and easy as you have seen. Unfortunately, you cannot query all the objects in the Client OM using LINQ. For example, you cannot query list items with LINQ. If you want to query only list items with LINQ to objects, your only option is to pass an empty query to the `GetItems` method and then work with the result. This is not considered to be a good practice with the Client OM. You should not create queries to return too many records. Microsoft recommends against returning 2,000 or more records. This causes a large amount of traffic to cross the wire, and realistically the user experience of a grid with so many records would not be good. If you need to work with so many records, your user interface is probably going to show them in some kind of paged view.

www.it-ebooks.info

Querying with CAML queries gives you the ability to execute queries and return the results in pages. To use this built-in paging, use a class called `ListItemCollectionPosition`. Initially, you set it to null so that the CAML query starts at the first item. You create a new `CamlQuery` object and set the `ViewXml` property. Now here is the first part of paged results with CAML. You set the `RowLimit` in the `CamlQuery` object to the number of items you want returned at once. To set up the client request, you create a new `ListItemCollection` and set it equal to `list.GetItems(camlQuery)`. In this example, `list` is the `List` Client OM object that you query and `camlQuery` is the `CamlQuery` Client OM object representing your query.

```
ListItemCollectionPosition itmPosition = null;

CamlQuery query = new CamlQuery();
Query.ListItemCollectionPosition = itmPosition;

while(true)
{
query.ViewXml = "<View><ViewFields>" +
"<FieldRef Name='Title'/>"+
"<RowLimit>10</RowLimit></View>";

ListItemCollection collListItem = list.GetItems(query);
context.Load(collListItem);
context.ExecuteQuery();
itmPosition = collListItem.ListItemCollectionPosition;

foreach(ListItem item in collListItem)
         Console.WriteLine("Title: {0}",item["Title"]);
if(itmPosition == null
break;
}
```

Working with Libraries

If you are somewhat new to SharePoint, you might not be aware of the relationship between lists and document libraries. For those new to SharePoint, a document library is a special type of list that, in addition to containing metadata, contains actual files and folders. Because of this relationship accessing lists and accessing libraries is similar.

Managing Libraries

Because libraries are specialized lists, all the list management operations previously described apply to libraries. Some additional management tasks are specific to libraries. For instance, libraries can be versioned; individual documents must be checked in and out. Documents are published and unpublished while this operation doesn't make sense of list items. In this section, you learn how to add and upload documents, work with file versions, and work with publishing.

Adding Documents

Adding items to document libraries is a little more complicated than adding list items to lists. One major item is getting the actual file bytes into the document library. There are two methods to do this, and each have their own issues. In the first method you read the bytes into a

www.it-ebooks.info

`FileCreationInformation` object and add that object into a document library. This returns a SharePoint Client OM File object. With that file object you call the `Load` method on the `ClientContext` that causes the file to upload. After that you call the `ExecuteQuery` method on the `ClientContext`. Finally, you can get the `ListItem` that the file is associated with by accessing the `ListItemAllFields` method on the SharePoint `File` object that you just loaded. At this point, you have a list item, and you can modify the metadata properties and update the list item. The code for this would look like:

```
ClientContext context = new ClientContext("http://localhost/");
Web web = ctx.Web;

FileCreationInformation newFile = new FileCreationInformation();
newFile.Content = System.IO.File.ReadAllBytes(@"C:\TestFile.doc");
newFile.Url = "/" + fileName;

List docs = web.Lists.GetByTitle("Shared Documents");
File uploadFile = docs.RootFolder.Files.Add(newFile);

context.Load(uploadFile);
context.ExecuteQuery();

SPClient.ListItem item = uploadFile.ListItemAllFields;

//Set the metadata

string docTitle = string.Empty;
item["Title"] = docTitle;
item.Update();

context.ExecuteQuery();
```

This approach is relatively straightforward and makes it simple to set the document metadata. It is also good for creating folders in libraries. This approach does, however, come with a major issue. Using the `FileCreationInformation` approach works only for files that are not too large. You get server errors when the file byte size is larger than the size that your site is configured to use. Although there are ways to change this setting, there is actually a better way to add and upload documents.

The second method is to utilize the WebDAV feature of SharePoint by calling the `File.SaveBinaryDirect` method on the Client OM `File` class. This method takes a client context, server relative file path, stream object, and boolean flag indicating if the method should replace an existing file.

```
context.Load(list.RootFolder,item => item.ServerRelativeUrl);
context.ExecuteQuery();

string path = list.RootFolder.ServerRelativeUrl + "/";
using (FileStream fs = new FileStream(txtFilename.Text, FileMode.Open))
{
SPCL.File.SaveBinaryDirect(context, path + filename, fs, true);
}

string file = path + filename.Replace("\\","");

SPCL.CamlQuery query = new SPCL.CamlQuery();
```

www.it-ebooks.info

```
query.ViewXml = "<View><Query><Where><Eq><FieldRef Name='FileRef'/>" +
"<Value Type='Text'>" + file +
"</Value></Eq></Where></Query>" +
"<RowLimit>2</RowLimit></View>";

SPCL.ListItemCollection collListItem = list.GetItems(query);
context.Load(collListItem);
context.ExecuteQuery();

if (collListItem.Count == 1)
{
collListItem[0]["Title"] = txtTitle.Text;
collListItem[0].Update();
context.ExecuteQuery();
}
```

This chapter just touched the surface of the things that you can do with the Client Object Model but has hopefully enabled you to understand the power exposed by it.

PUTTING IT ALL TOGETHER

Knowing all the different options for customizing SharePoint is only half of the battle. The far greater challenge is knowing how and when to use the various pieces of functionality to build a solution to a complex set of requirements. The rest of this chapter focuses on presenting some common problems that many organizations face and provides high-level design solutions.

A Blog Aggregator

Imagine that you have a site collection with a number of blogging subsites beneath it. Your boss comes to you and requests that on the top level site, you must provide a roll-up of the ten most recent blog postings regardless of which blog they appear on. Figure 1-6 shows the scenario. Take a minute to think about how you might solve this problem.

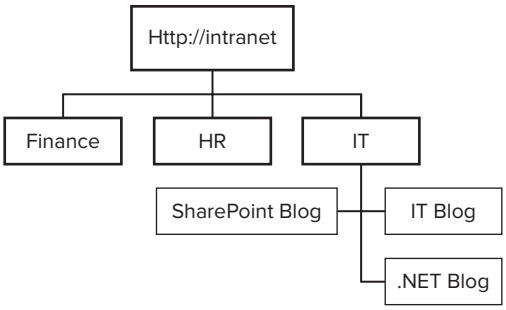


FIGURE 1-6: Blog sites beneath site collection

This was a bit of a trick question because this chapter focuses on the object model, but you can solve this problem without writing any custom code by using a Content Query Web Part. So pat yourself on the back if you correctly identified a Content Query Web Part as the appropriate solution.

www.it-ebooks.info

A solution that you can solve without writing any code isn't exciting, though, so now imagine a similar but different scenario. Instead of rolling up blog postings from subsites of a site, what if you want to display the ten most recent blog postings across the entire farm? Figure 1-7 demonstrates the scenario.

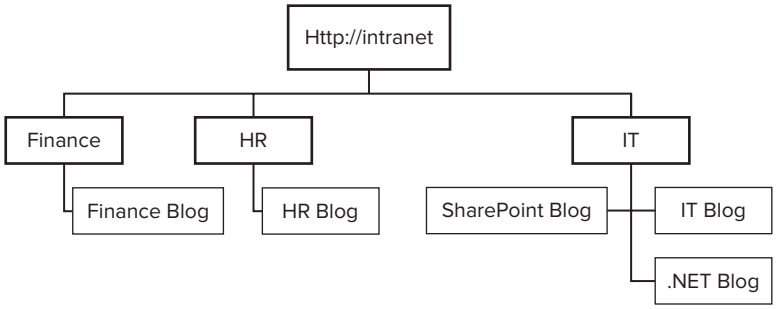


FIGURE 1-7: Blog sites beneath multiple site collections

Can a Content Query Web Part be leveraged here? If not, what might a solution look like?

Custom Web Parts

Web parts are server side controls that you can add to any SharePoint page that accepts web parts. They are the primary way you can create modular UI within SharePoint. SharePoint ships with a number of web parts out-of-the-box, and as a developer you have the ability to create custom web parts. Any solution that requires any sort of visual component likely involves a web part of some sort.

The key differentiator between the first scenario and the second scenario is that the blog postings now cross site collection boundaries, which is something that isn't supported by a Content Query Web Part. Most people's initial thought would be to write a custom web part that queries all sites in the web application to get a list of all blog posts and displays the ten most recent blog posts, as shown in Figure 1-8.

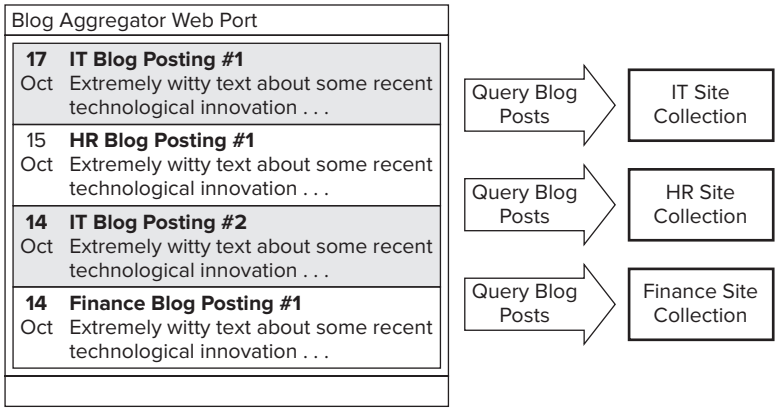


FIGURE 1-8: Simple web part solution

www.it-ebooks.info

If the web application has only a small number of site collections and blog sites, such a solution would likely be acceptable, but how well could it scale in a large web application that has thousands of sites? When you get to a certain scale, iterating through all site collections and querying for blog postings could take many seconds to many minutes, which is likely too long to make a user wait for a page to render.

At this point, you might consider implementing a web part that uses asynchronous calls to not delay the rendering of the page while the user waits for the blog posts display. This would solve the page rendering wait, but the user would still need to wait for the list of blog entries to be returned every time the page refreshed, which is not an ideal scenario.

Caching Data Across Web Frontends in SharePoint

Perhaps you might then consider caching the results after the first render such that only the first user hitting each web frontend in the farm would experience a delay. Users hitting the site after the list has been cached would view the cached list of blog postings in near real time. But where do you store the cached list?

Those coming from an ASP.NET development background might suggest using the `HttpRuntime Cache` object or even Application state, but because SharePoint farms often have multiple web frontends (WFEs), those caching solutions are less than ideal because they aren't shared across WFEs. So every WFE in the farm needs to build up the cached list of blog posts, and depending on when each cache was built, you might run into scenarios in which certain WFEs have different lists of posts cached.

Luckily, SharePoint provides a few different options for caching data across WFEs. The most obvious is simply to use a hidden SharePoint list to store a list of the ten most recent blog postings, but if you don't require a UI to access the data, SharePoint also offers something called a property bag and something else called the hierarchical data store, which can be leveraged.

Your boss is a stickler, though, and doesn't like that the proposed solution requires even the first user to wait for the blog post list to generate. How might you eliminate this initial load time?

Timer Jobs

If you were writing a pure .NET solution, you might consider writing a windows service application that periodically would build up a list of cached blog postings that could be consumed by your web part. SharePoint provides something similar in its Timer Jobs, which enable particular bits of code to run on a scheduled basis. You could write a timer job that handles building up the cache of the ten most recent blog postings and then your web part can just check the cache.

You demonstrate your improvements to your boss, but he notices that your list of the ten most recent blog postings show blogs he shouldn't be able to see. When he clicks them to view them, he gets an access denied error. Your boss now adds a new requirement to security-trim the results. How can you security-trim the results?

One option would be to enhance your timer job to also cache security information when it builds up the cache of blog posts and then have the web part restrict the items displayed to only those the user has permission to view. For those of you who are curious as to how this might work, a similar

www.it-ebooks.info

solution is used in Chapter 11 as part of a global navigation solution. Caching security otherwise known as Access Client Lists (ACLs) would certainly work, but there might be a simpler way.

Assuming search is configured for the web application, you can use the Search API to execute a query to return the ten most recent blog postings' security trimmed to the user. This works because blog posts use their own content type (Posts). Using search greatly simplifies the solution and enables you to not require a timer job. Figure 1-9 demonstrates what this architecture might look like.

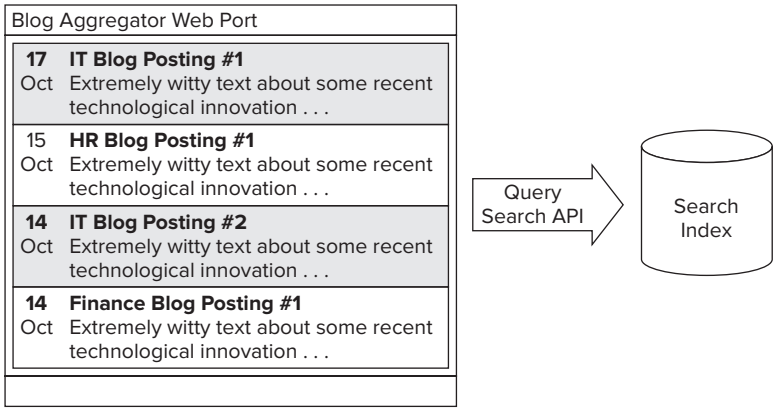


FIGURE 1-9: Web part using search API

Feeling quite confident, you again demonstrate your solution to your boss. Everything works great. Your boss sees only the posts he has access to and you feel confident. But then your boss creates a new post and takes a look at your Recent Posts' web part. His post doesn't display in the list.

The post doesn't display in the list because the search service crawls site content on a scheduled basis. Until the new post is crawled and indexed, it won't be returned by the search API, and the post does not appear in the list. This same issue also existed when the solution used a timer job. In most scenarios, a delay of a few minutes is acceptable, but your boss is adamant that blog posts must show up in the list as soon as they are created. How might you craft a solution to allow near real-time post display when it may take minutes just to iterate through all the site collections to build up a list of posts? Obviously, one option would be to increase the incremental crawl frequency of search, but in large farms even incremental crawls can take tens of minutes to complete. Delivering near real-time results requires a shift in the approach.

Event Handlers

You can attach event handlers to list and document libraries and web sites to intercept common events, such as when items have been added, updated, or removed from a list. In this case, you can attach an event handler to all post lists to implement a custom change log that your recent posts' web part can use to determine the most recent posts in real time.

Hopefully, this example helps illustrate how you can combine the various components of SharePoint to provide a complete solution. Depending on the requirements, any one of the preceding iterations may be an appropriate solution.

www.it-ebooks.info

Calendar Reminders

SharePoint has a calendar list type that functions like a typical calendar and can be used to keep track of events, meetings, and appointments. One piece of functionality that isn't provided out-of-the-box, however, is any sort of reminder capability that would alert users some period of time before a calendar item is scheduled to occur. You have been asked to add this functionality to a particular calendar in SharePoint. How might you go about it?

The first thing you should ask yourself when confronted with a particular requirement is to see if there is anything provided out-of-the-box in SharePoint that can help deliver the functionality requested. One thought might be to try to leverage the alerting capabilities of SharePoint, but a quick examination of alerting can show that alerts are triggered only when items are created or updated. There is no functionality built into alerts to alert based on a particular time period.

Next, is this something that could be delivered via SharePoint Designer? Designer offers the ability to create workflows, so could a SharePoint Designer meet the requirements? Yes, it could. Chapter 6 goes into greater depth about workflow, but Figure 1-10 shows a simple description of what such a workflow might look like.

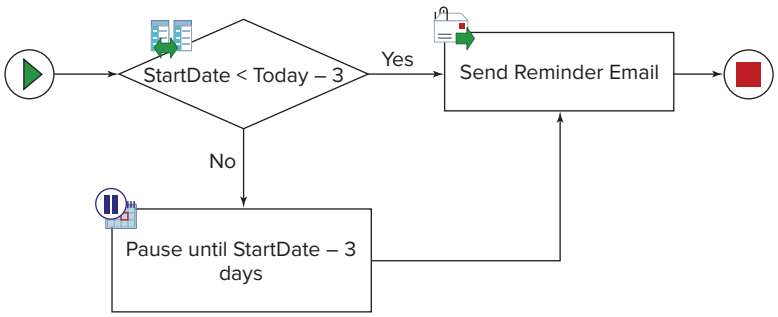


FIGURE 1-10: Calendar reminders via workflow

Now, are there any reasons you might not want to use a SharePoint Designer workflow? Based on the simple requirements given, a SharePoint Designer workflow is actually an excellent fit. Some requirements that might make a Designer workflow less than ideal might be the following:

- **Advanced email templates:** SharePoint Designer workflows have limited ability to format email messages.
- **Bulk/digest reminders:** Workflows would be tied to individual calendar items and would send out individual emails, but you may want to provide a digest capability that sends out a single email with reminders for all events occurring on the same day.
- **Individual configurability:** Different users might have different preferences over when and how they are notified. Some might like to be notified a day before, whereas others might prefer to be notified two days before. Some users might like individual emails per event, whereas others might want to do the digest reminder previously described. Although you might create some combinations of workflows that provide some number of reminder "buckets," it is quite challenging to implement a truly flexible solution using a SharePoint workflow.

www.it-ebooks.info

- **Scalability:** Workflows are tied to individual list items, so when you have thousands of future events in your calendar, you would have thousands of individual workflows "running." SharePoint does a good job to dehydrate and rehydrate workflows, but even so, it is not the most efficient way to process thousands of events within a list.

The next section discusses options for delivering the functionality discussed previously.

To Workflow or Not to Workflow

If SharePoint Designer workflows aren't a great fit, might a Visual Studio Workflow work better? Visual Studio workflows certainly have the capability to deliver advanced email templating because you have the full power of .NET development, but workflow is not an ideal option when you want to operate over a number of list items at the same time to support the bulk digest requirement. Furthermore, although Visual Studio workflows do provide a greater capability to support the complexity individual configurability would entail, it would still pose some challenges that probably make other solutions more appropriate. For these reasons, workflow is probably not the best solution.

Timer Jobs, Again?

Whenever you have a requirement around a bulk operation that must occur on a scheduled basis, a timer job is your most likely solution. Timer jobs can recur, which can handle periodically sending out reminders. And timer jobs are not tied to individual items, which enables you to deal with the bulk notification and scalability requirements, and you have the full power of .NET at your disposal, so custom email templates won't be an issue either.

The bulk of the solution can exist within a timer job. The timer job can periodically execute (probably once a day) and iterate over all calendar items within the list that have a future start date, and then pull back a list of user subscriptions and send out the reminder emails. But how do you allow users to manage their subscription preferences, and how do you allow list owners to handle email templates?

Custom Application Pages

Support subscription preferences and email templates requires both someplace to store the data and a UI that enables users to manage that data. SharePoint lists are generally the go-to option when deciding where to store data within SharePoint and seem to be a good fit in this case, but the UI provided by a list for managing list item data is not the most intuitive, so building a custom UI for managing a user's reminder settings is probably appropriate. Figure 1-11 presents a wireframe of what the UI might look like.

When thinking about custom UI for data entry, three potential options exist:

- Work within the list framework by replacing the standard list item forms with custom forms using InfoPath Services, SharePoint Designer, or custom ASPX pages.
- Create a custom web part.
- Create a custom application page.

Working within the list framework is most appropriate when you intend to leverage other aspects of the list, such as the Ribbon, list administration menus, and list security, and you need to provide

www.it-ebooks.info

a top-level view of all the data. In this case, the list serves only as a backend data store, so users shouldn't even know that the data is stored in a list.

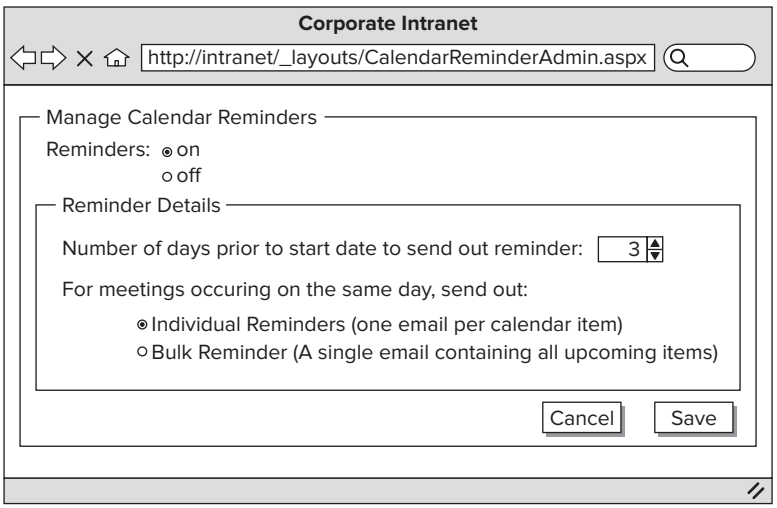


FIGURE 1-11: Calendar reminders mockup

Custom web parts are most appropriate when you want the functionality contained within them to be loosely coupled, such as needing to embed the functionality on existing pages or when you want the end users to determine where such functionality should exist on the site. When deciding if a web part is an appropriate solution, it is helpful to ask how you want your user to access the functionality. Should it be integrated into the calendar's Ribbon menu? Should it just appear on the same page as the list? Should it be up to the site's owners or designers to expose this functionality as they want? In the case of calendar reminders, the functionality seems like it would be most useful when tightly integrated into the calendar UI. Because the solution needs to be tightly integrated, a web part is probably not the best option.

Custom application pages make the most sense when you do not require a modular UI that can be customized by end users and when you require the functionality to be exposed via integration with a SharePoint menu or within the Ribbon. So in this case, a custom application page is likely the best fit.

Exposing the UI

Now that a custom application page has been determined to be the best fit for allowing users to maintain their reminder settings, the next question to ask is: how can the UI be accessed by the end user? As mentioned in the previous section, because reminders are something that are going to be configured for the calendar as a whole, it makes sense to expose the option in the Calendar tab of the Ribbon. If reminders were instead something that were going to be configured individually on a per entry basis, however, it would make sense to expose them on the Events tab of the Ribbon. Another option would be to expose the functionality via a custom list item action on the item drop-down, but because the default calendar view does not provide an item drop-down menu, this is unlikely to be a best option for exposing the functionality.

www.it-ebooks.info

The Complete Solution

To summarize the solution previously discussed, the functionality that is delivered is a mechanism to provide users with the ability to be sent reminders prior to the meeting start date. The reminders can be configured per user, per calendar.

This functionality was delivered using a timer job that runs daily and does the actual work to send out the emails. Behind the scenes, the user's reminder preferences are stored within a list. The UI for configuring a user's preferences is exposed via a custom Ribbon menu item, and the actual reminder setting UI is a custom application page.

Scheduled Workflows

SharePoint 2007 supported a single kind of workflow, the List Workflow. This type of workflow was tied to a particular list and would be triggered for each item within a list either manually, upon creation, or upon update. Although powerful, because the workflow was tied to list items it was not always applicable to many workflow scenarios, so SharePoint 2010 introduces the Site Workflow. The Site Workflow is associated with a SharePoint site or subsite, but one critical limitation of the site workflow is that it must be manually started.

Quite often, you come across the need to schedule code to run at periodic intervals. Consider the scenarios encountered in this chapter. The typical SharePoint solution to this problem is to author a timer job that executes the scheduled code, but this requires the involvement of a developer to write the timer job and the code. In many instances, a site workflow would suffice if only the user could schedule workflows. So consider the scenario in which you need to implement the capability within SharePoint to schedule recurring workflows. What might such a solution look like?

Envisioning the Solution

The simplest way to execute workflows on a scheduled basis would be to use PowerShell scripts and the Windows Task Scheduler to execute the script on a scheduled basis. However, this would require that the users have remote access to the servers in the farm to configure the schedule and would require them to modify the scripts by hand (or at least the parameters passed in to the script) every time they needed to schedule a new workflow. While that may be acceptable in some scenarios, a more robust solution would provide a UI within SharePoint and allow nontechnical users to handle scheduling. Users need to specify the workflow, which site it should run on, and when it should run. In addition, it would be nice to support the ability to schedule workflows to execute on a recurring basis such as daily, weekly, monthly, or yearly. Most scheduling systems enable an end date to be specified for the schedule. And finally, to assist with debugging, it would be nice to log some diagnostic information, such as the last time a particular workflow executed.

To sum up, following is the information that needs to be collected:

- Workflow to execute
- The site the workflow is associated with
- Start date
- End date

www.it-ebooks.info

- Frequency (daily, weekly, monthly, yearly)
- Last run date

In the previous solution, a list was used as the backend store for data that was collected, and a list seems to also be an appropriate choice to store the data about the workflows. The real question is if the UI provided by a standard list interface is acceptable or if a custom UI must be built. Although a custom UI might enhance the user experience, the data collected seems simple enough that the standard list UI should be sufficient.

Beyond the UI and where the data about workflow schedules is stored, the other primary piece of functionality that must be accounted for is how to kick off the workflows according to the schedule specified. In what is becoming a common theme, a timer job is the only option available to developers wanting to execute arbitrary code on a scheduled basis. The timer job needs to read from the workflow schedule list and execute the site workflows according to their schedule.

The UI

As discussed, the UI for this feature request is relatively simple. When determining how much effort should be put into developing a UI, it can be beneficial to consider typical use cases and how frequently the functionality will be accessed. In this case, it is unlikely that new workflows will be added that often to the schedule, so even though some aspects of the data entry piece might benefit from a custom UI, such as selecting the workflow to execute from a drop-down rather than typing the name into a text box, it seems like that should be left to a future version of the solution.

So, the solution requires a list definition that defines a Workflow Schedule list type as well as a list instance that actually creates a list of type "Workflow Schedule" when the solution is activated.

The Backend

The backend consists of a timer job. Because the minimum frequency specified in the requirements is the ability to run daily, the timer job also needs to only run daily. A more advanced implementation might support recurring workflows that run on a schedule of minutes, but such a requirement should be carefully considered because the impact on performance could be considerable depending on what actions the workflow performs.

The timer job needs to query the workflow schedule list and find any workflows that need to run on the current day and then loop through the workflows and individually kick them off. In addition, the timer job will update each workflow schedule to show the most current run date.

The Complete Solution

As you are probably beginning to realize, many of the solutions to these problems share a lot of similar characteristics. In this case, the solution simply involves a standard SharePoint list and a timer job to deliver the complete functionality.

The workflow scheduler is, architecturally, the simplest solution discussed in the chapter, but the code within the timer service needs to be relatively complex to handle determining which workflows are scheduled to run and then appropriately kicking them off. This is just the tip of the iceberg, however, and a much more complicated solution could be crafted that supports more granular intervals,

www.it-ebooks.info

workflow initiation data, and list workflow support. Start thinking about how you might update the proposed solution to support some of those features.

SUMMARY

This chapter provided an overview of many of the mechanisms used to develop custom solutions within SharePoint and also provided an overview of the Client Object Model that was released with SharePoint 2010. It then showed how those building blocks could be plugged together to implement solutions for some common requests within SharePoint.

For those who are ambitious, take some time and attempt to implement and enhance the three solutions discussed in this chapter after you've read through the rest of the book.

www.it-ebooks.info

www.it-ebooks.info

2

Creating Visual Web Parts

One of the greatest enhancements in the 2010 platform for SharePoint development is the notion of visual web parts. These are still the same web parts that you created in MOSS 2007; however, now a project template provides mark-up abilities so that development does not need to rely on creating and instantiating control objects from code alone.

Previously, one of the biggest complaints in building web parts was not seeing what you were building until you thought you were finished and deployed the code. Now with Visual Studio 2010 and SharePoint 2010, development of web parts is easier and more reliable when starting fresh. Essentially, Visual Studio provides the framework to create visual web parts without having to manually hook up all the pieces.

In addition to Visual Studio 2010, you can also leverage SharePoint Designer 2010 to create visual web parts based on an XSLT List View web part. This enables you to easily add web parts to a page, view the results, and stylize the content before publishing the page to your SharePoint site.

CREATING VISUAL WEB PARTS

You can facilitate the creation of a visual web part using Visual Studio 2010. This section describes the steps and tasks necessary to create a visual web part from end to end.

Visual Web Part Project

The first step to creating a new visual web part is to open Visual Studio 2010 and create a new project. Under the SharePoint 2010 templates, Visual Web Part is one of the available project templates for project creation, as shown in Figure 2-1.

As with other SharePoint 2010 project templates, when creating a new project, you are prompted to enter the SharePoint site and the trust level for the solution within the SharePoint Customization Wizard, as shown in Figure 2-2. Because visual web parts need to be deployed as farm solutions, the option to deploy as a sandboxed solution is disabled.

www.it-ebooks.info

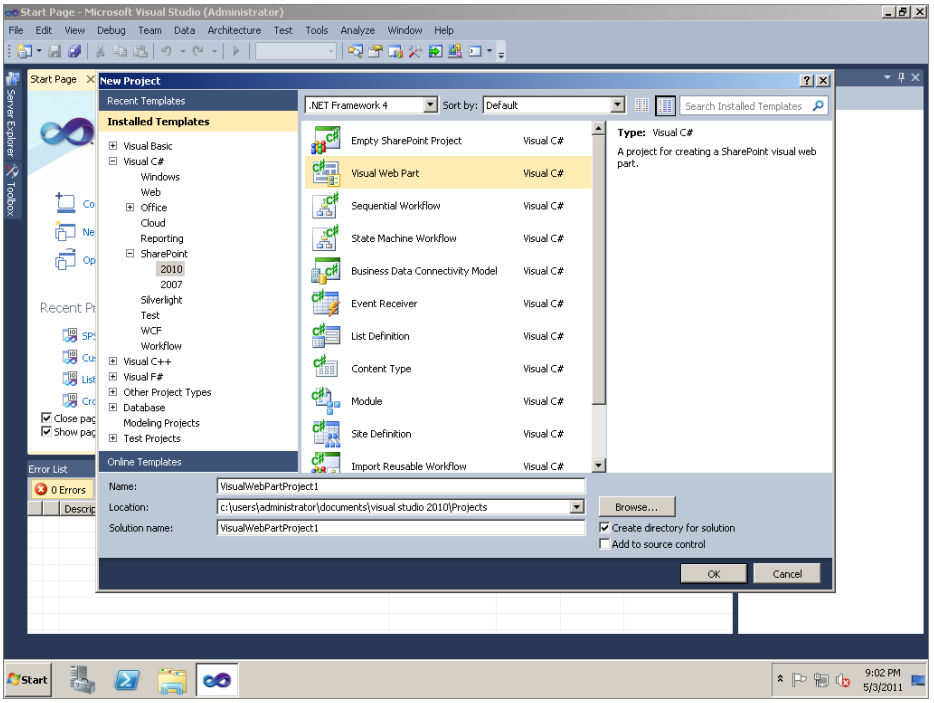


FIGURE 2-1: Visual Web Part project selection

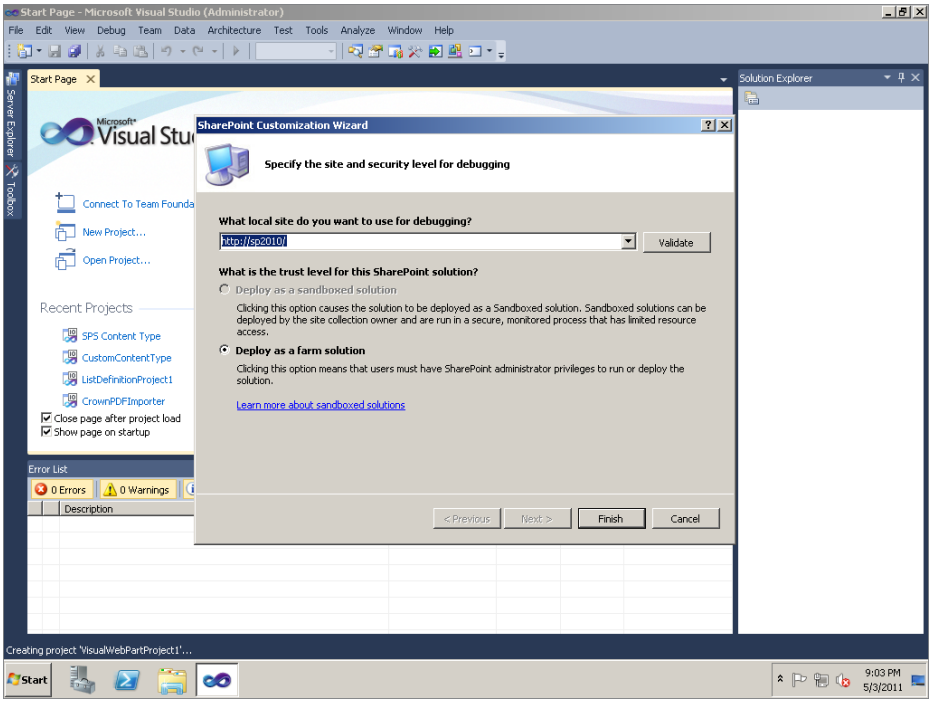


FIGURE 2-2: SharePoint Customization Wizard

www.it-ebooks.info

The generated visual Web Part project contains all the necessary objects and files needed to generate a visual web part. The web part folder contains these files, as shown in Figure 2-3.

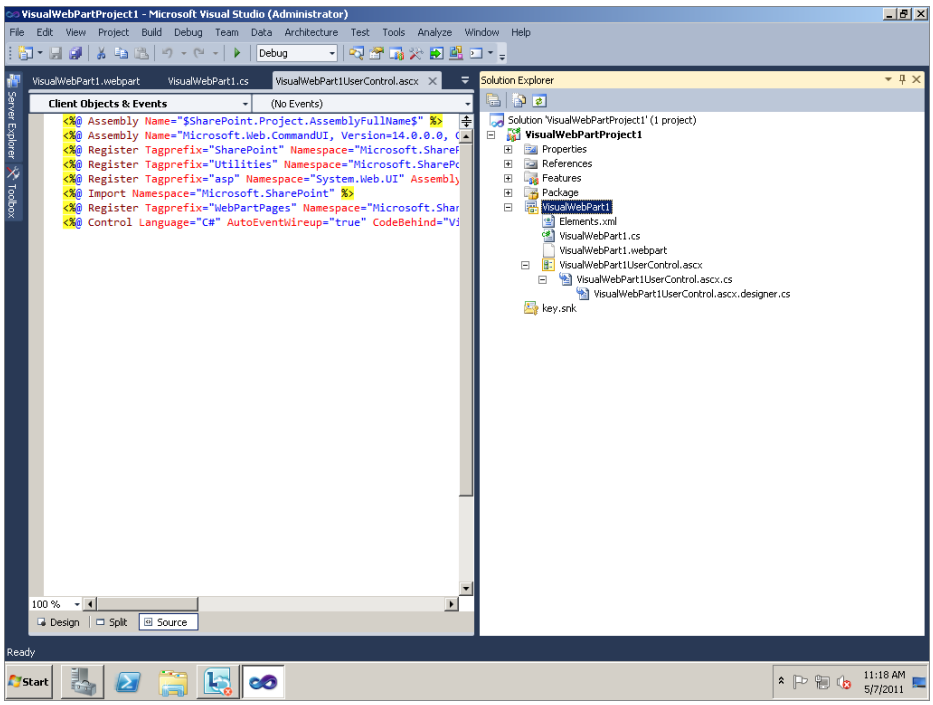


FIGURE 2-3: Visual web part project components

The `Elements.xml` file is used for deployment purposes and contains the appropriate configurations so that the web part file can be installed into the SharePoint Web Part Gallery.

The `VisualWebPart1.cs` is the actual web part code that loads controls, adds them to the page, and then renders them. This is facilitated by the class implementing the `WebPart` object. When creating a code-only type web part in MOSS 2007, this was the class used along with many overridden methods. In this case, only the infamous `CreateChildControls` method is overridden. The code within this method essentially loads an ASP.Net control (included in the project) that is the visual part of the web part.

The `VisualWebPart.webpart` file is the XML-based file that lives in the SharePoint Web Part Gallery. This file explains to SharePoint what assembly needs to be instantiated when the web part is added or rendered on a page.

The `VisualWebPartUserControl.ascx` is the visual ASP.NET user control that may be rendered in SharePoint or any .NET web application. This provides the mark-up for rendering controls and objects. Because this is based on ASP.NET, the `ascx` file also has code-behind contained in the `VisualWebPart.ascx.cs` class. This is where logic for populating or manipulating the controls within the user control can live.

www.it-ebooks.info

If you are a seasoned SharePoint developer, you realize at this point that a visual web part isn't new. The same code and methodology can be applied in MOSS 2007. Therefore, Visual Studio 2010 essentially provides a solid framework for building out web parts without having to manually produce or reproduce the necessary web part, feature, and solution files. In addition, using the ASP.NET user control paradigm, all the web part UI and logic is facilitated by the `.ascx` file, and the web part class simply loads the user control — thus providing a way to visually develop web part functionality.

Adding Controls

Because the visual web part is built out using an ASP.NET user control, you may simply add controls to the `.aspx` file in your project. The whole idea is to design and generate visual user controls that may be visible during design time and not buried within web part code. Therefore, switching to the design view of the user control presents the canvas to add controls from your toolbox, as shown in Figure 2-4.

Use the Toolbox to drag controls onto the user control in the Design view, as shown in Figure 2-5, or modify the mark-up in the Source view, as shown in Figure 2-6.

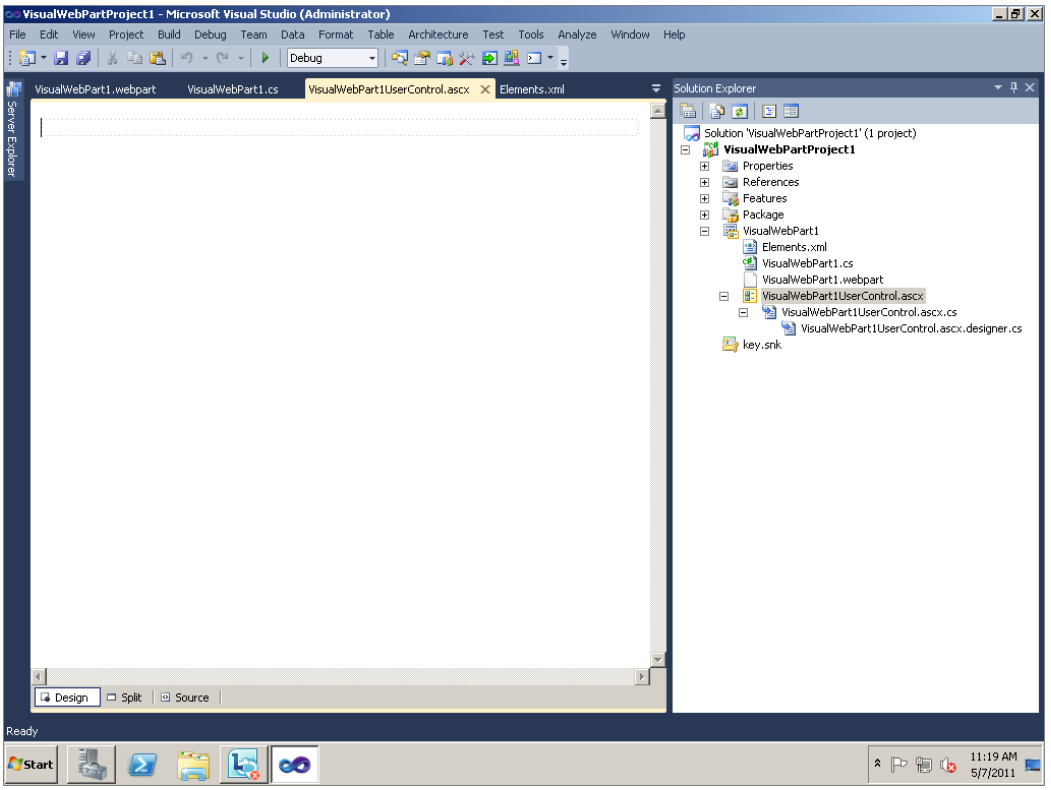


FIGURE 2-4: The Visual Web Part user control in design mode

www.it-ebooks.info

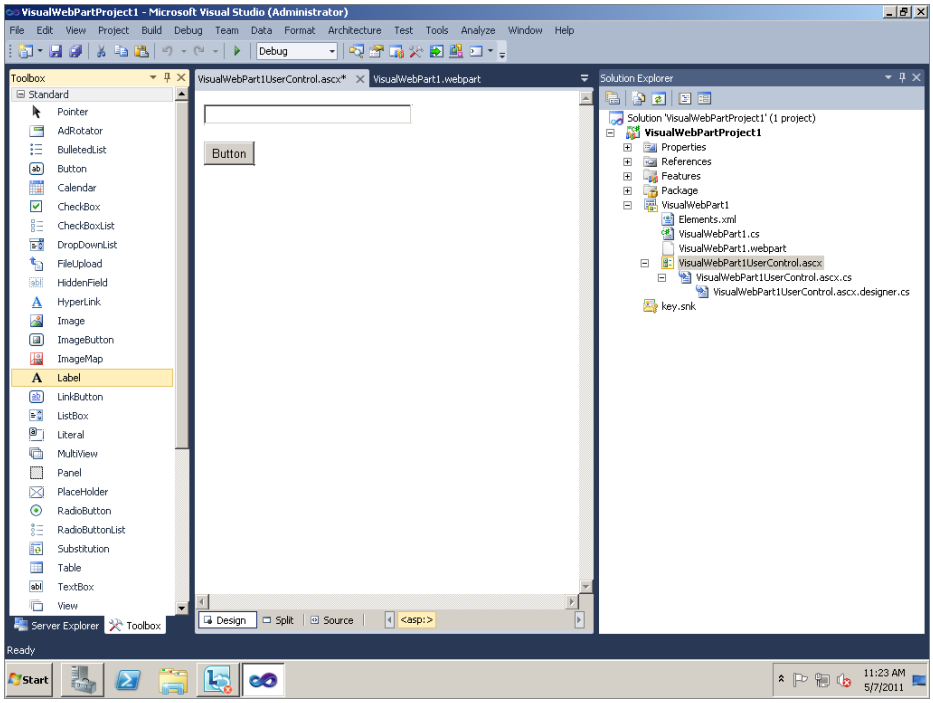


FIGURE 2-5: Adding controls in Design View

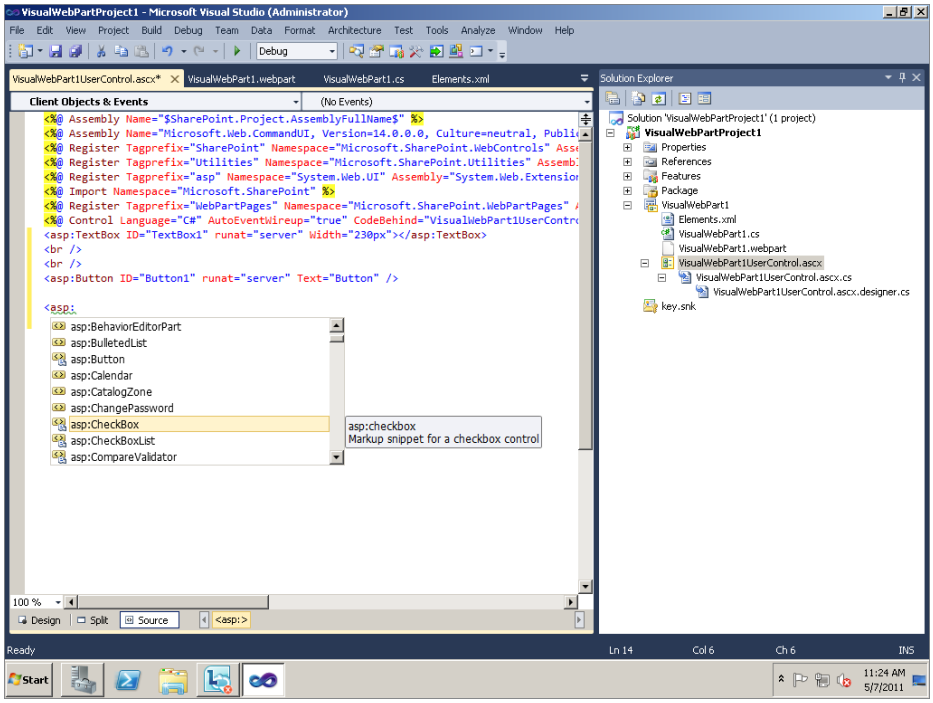


FIGURE 2-6: Adding controls in Source View

www.it-ebooks.info

Adding Code

The main method within the user control code-behind is the `Page_Load` method. You may add code in this method, as shown in Figure 2-7, or call other custom methods to populate the control. A good practice is to implement error handling within this method and make calls to other custom methods to perform the logic within your user control.

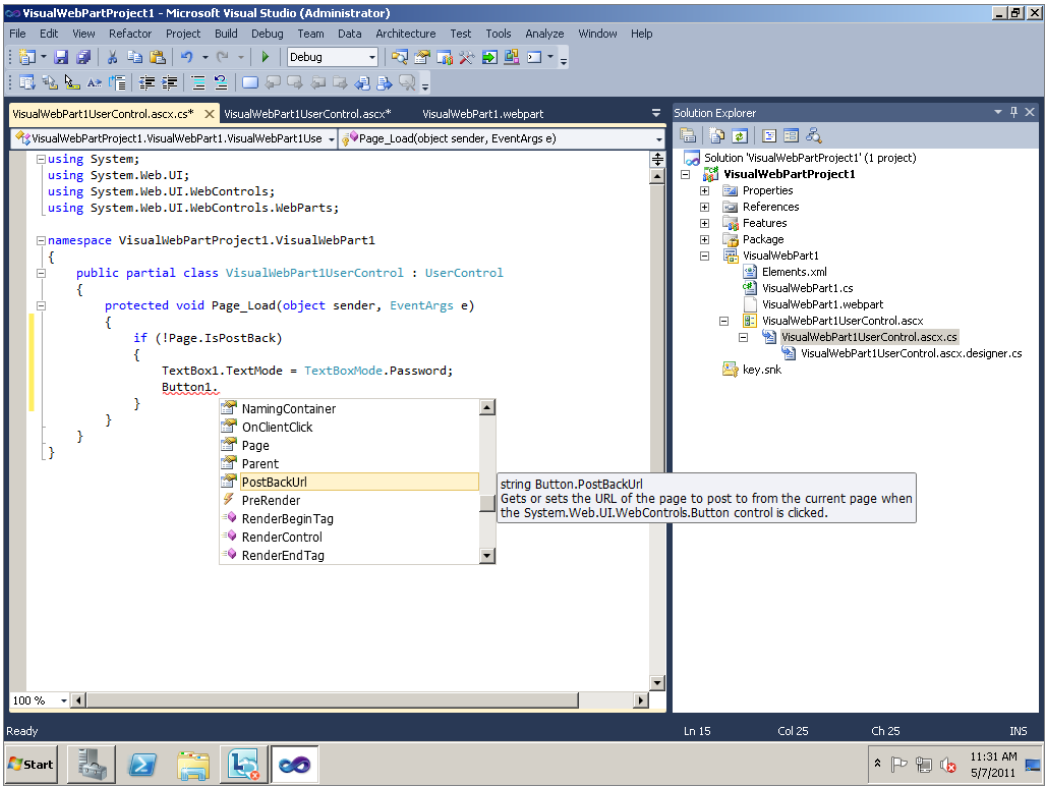


FIGURE 2-7: Adding controls in Source View

A common condition to check is the `Page.IsPostBack`. Depending on the functionality of your user control, you may or may not want to perform actions if there is a post back on the page in which the web part lives. A sample `Page_Load` method may look like the following:

```
protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
                try
                {
                    SetControls();  //custom method
                }
                catch (Exception webPartException)
```

www.it-ebooks.info

```
            {
                HandleError(webPartException);  //centralized error handling
            }

        }
    }
```

Deployment

Visual Studio 2010 is well aware of your SharePoint 2010 environment. (The SharePoint 2010–specific templates won't load if SharePoint 2010 is not detected on your development machine.) This makes deployment easier.

Because the SharePoint 2010 project templates contain all the necessary files and settings, automatically building your Visual Studio solution generates a feature and packages your web part into a SharePoint solution file (`.wsp`). Before deploying, you should build the solution to ensure that everything compiles. After a successful build, right-click your solution in the Solution Explorer pane, and select Deploy Solution, as shown in Figure 2-8. If your solution contains several projects, you can deploy each project individually by right-clicking the project and selecting Deploy.

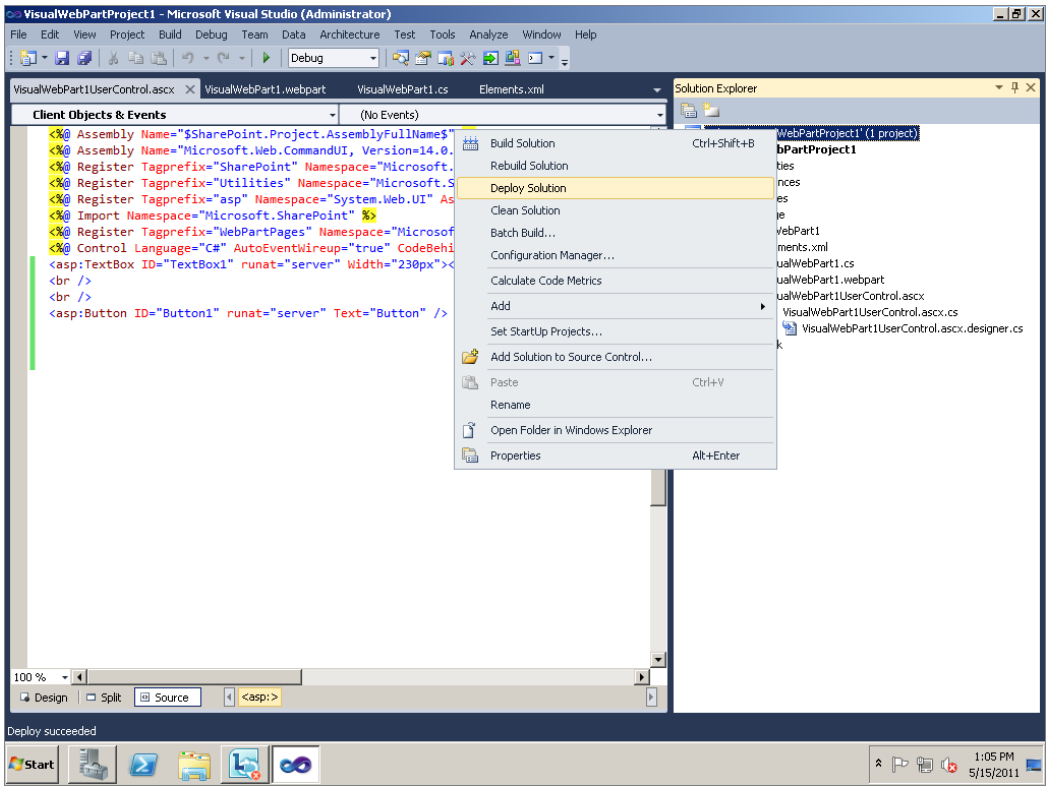


FIGURE 2-8: Deploying a solution from Visual Studio

www.it-ebooks.info

The solution is deployed to the previously configured site entered when the Visual Studio project was created. You can change the site by viewing the project properties and modifying the Site URL setting, as shown in Figure 2-9.

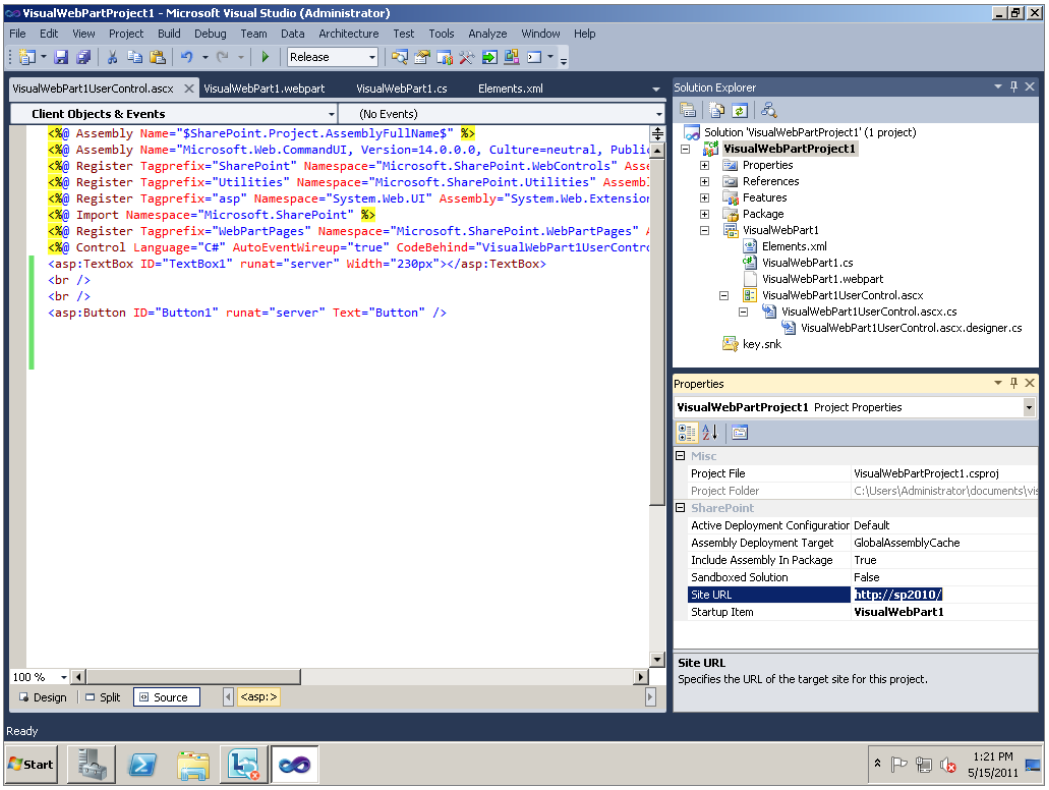


FIGURE 2-9: Project file properties

Each project in your Visual Studio solution contains deployment properties. To view the actual deployment properties, select the project, right-click, and select Properties. The tabbed project properties screen appears. Select the SharePoint tab. The deployment properties and configurations display, as shown in Figure 2-10.

Two deployment configurations are included within the project: Default and No Activation. These configurations may be viewed but not modified. You may, however, create new deployment configurations using the New button within the Edit Configurations section of the SharePoint tab.

The No Activation configuration is almost the same as the default, however, when used, your solution is deployed but any features within that solution are not activated on any site or site collection. Therefore this should be used when you do not wish to have any features activated automatically. Some suggest that this is helpful when debugging feature receivers or other processes that occur during feature activation. One thing to note, however: if you have your solution deployed and features already activated, redeploying your solution under the No Activation configuration will deactivate any features included during the redeployment.

www.it-ebooks.info

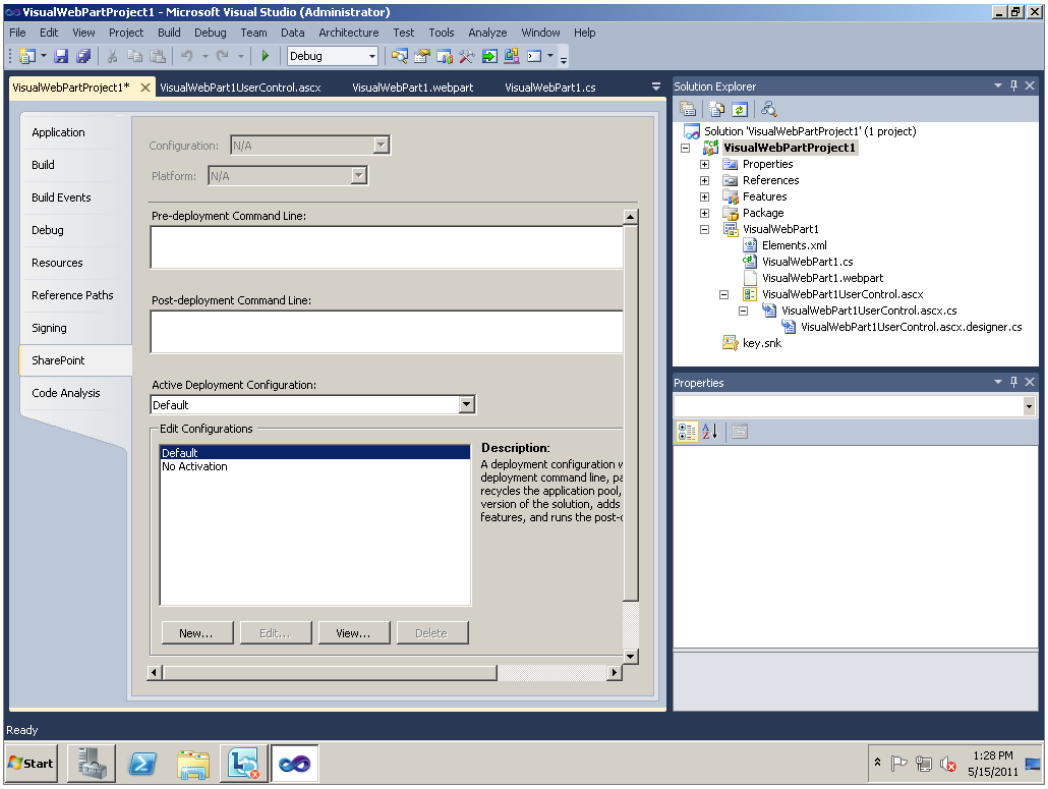


FIGURE 2-10: Project deployment properties

CONNECTING WEB PARTS

This section explains how two or more web parts can be "connected" together so that communication is achieved between them. Web parts communicate with each other by passing values from one to another. For this to happen, there needs to be a common interface so that each web part understands the values or data being transferred.

After an interface is defined, a web part may either be the provider of the value or values, or the consumer of the values. A web part can be both a provider and a consumer; however, the communications between web parts is usually one way.

The example used to connect web parts creates a provider web part that displays items from a SharePoint list. After an item is selected, the values are passed to the consumer web part and displayed accordingly. To prepare for the example, create a generic SharePoint list and populate it with various list items.

www.it-ebooks.info

Creating an Interface

The first step to connect web parts is defining and creating the Connection Interface that each web part will implement. To start, create an Empty SharePoint Project in Visual Studio 2010 by selecting the Empty SharePoint Project template. Right-click the Project in the Solution Explorer, and select Add ➪ New Item. From the Visual C# Code category, select Interface, as shown in Figure 2-11. Give the file a name and click Add. The code window appears.

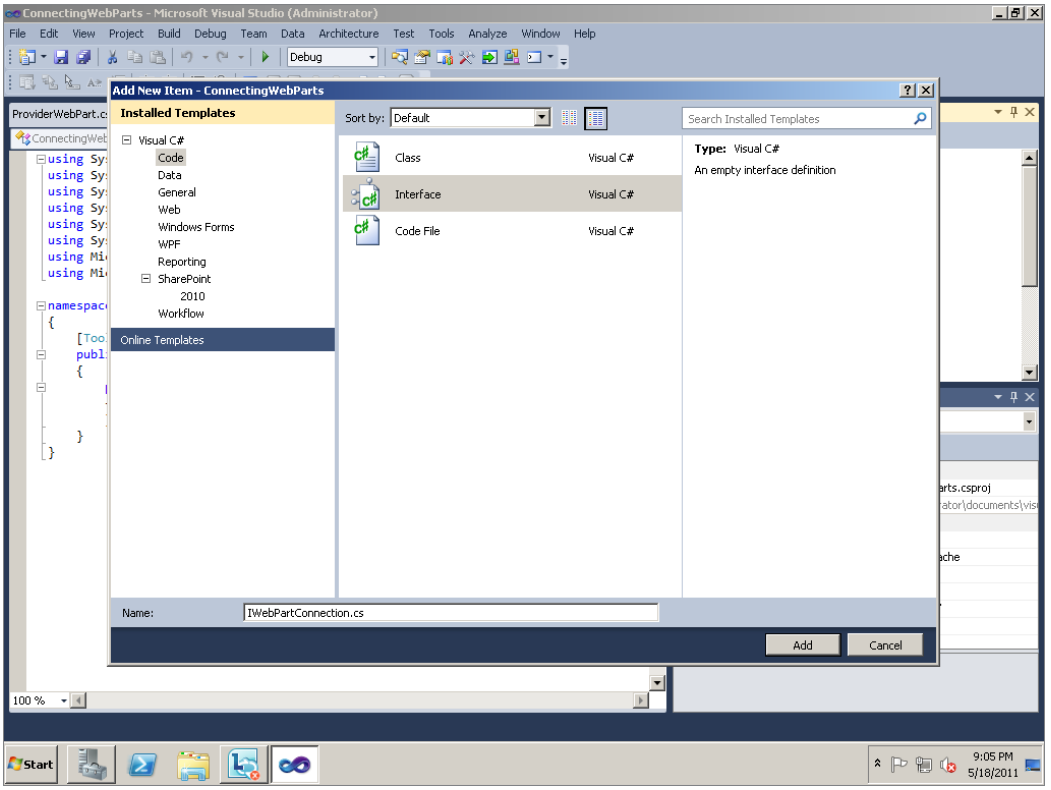


FIGURE 2-11: Adding an interface item

Because this is a public interface that multiple code bases need to access, add `public` in front of the interface declaration. The standard name for an interface begins with an uppercase `I`, so name the interface appropriately (for example, `IWebPartConnection`). The rest of the code defines the information that may be passed between web parts. For example purposes, pass the item Id and the item Title from a list. Therefore, the interface needs to define `ItemId` and `ItemTitle` properties. Enter these two properties within the code so that your interface looks like the following.

```
public interface IWebPartConnection
  {
      int ItemId { get; }
      string ItemTitle { get; }
  }
```

www.it-ebooks.info

Build the solution to ensure it compiles. This is the first step in connecting web parts. The interface is structured, and now the provider web part needs to be created.

Creating the Provider Web Part

Continuing from the interface creation above, add a new item to your project again. This time select Web Part from the SharePoint 2010 category. Modify the web part so that it also implements the web part connection interface created in the previous section, as shown in Figure 2-12. Give the file a name and click Add. The code window appears.

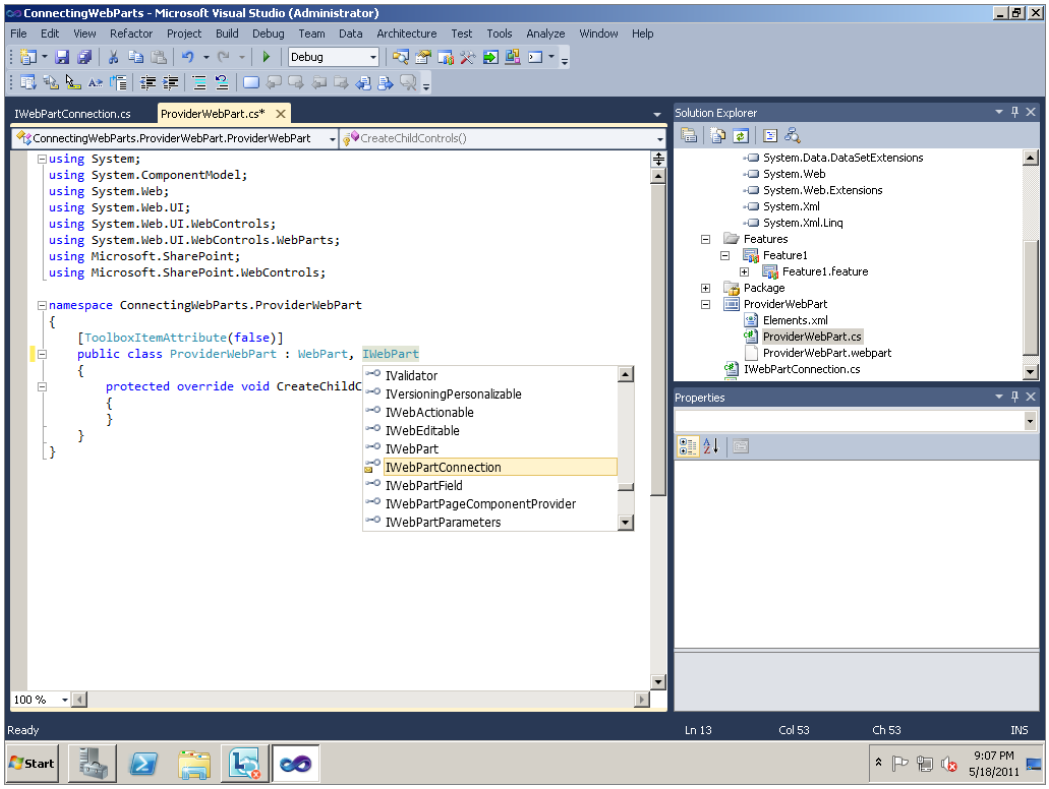


FIGURE 2-12: Implementing the interface

Because the provider web part implements the connection interface, the `get` methods need to be defined within the web part class. In addition, because the values will be based on a drop-down list, the drop-down control needs to be declared. The `ItemId` is the underlying value of the drop-down selection, and the `ItemTitle` is the text of the selected item in the drop-down list. Therefore, these properties should be implemented as follows.

```
private DropDownList _listDropDown;

        public int ItemId
        {
```

www.it-ebooks.info

```
        get { return int.Parse(_listDropDown.SelectedValue);}
    }

    public string ItemTitle
    {
        get { return _listDropDown.SelectedItem.Text; }
    }
```

Next in the `CreateChildControls`, enter code to read from a custom list on your SharePoint site, and populate the drop-down list control accordingly.

```
_listDropDown = new DropDownList();
using (SPSite spSite = new SPSite(SPContext.Current.Web.Url))
using (SPWeb spWeb = spSite.OpenWeb())
{
  SPList customList = spWeb.Lists["Custom List"];
  foreach (SPListItem li in customList.Items)
  {
     _listDropDown.Items.Add(new ListItem(li.Title, li.ID));
  }
}
_listDropDown.AutoPostBack = true;
this.Controls.Add(_listDropDown);
```

To provide the connection interface, you need to have a method that returns the connection object using the interface class. Because the web part implements the interface, the method returns the web part class instance as the interface object.

```
[ConnectionProvider("Item Title and ID")]
public IWebPartConnection ConnectionInterface()
{
     return this;
}
```

The provider web part code should now look similar to the code in Figure 2-13. Build the solution to ensure it correctly compiles.

Creating the Consumer Web Part

Right-click the project and add a new web part item. Name the web part **ConsumerWebPart**. The consumer web part needs to have an object for the web part connection interface. Also, the consumer web part displays information about the selected list item in the provider web part; therefore, you need a control as well. Add global declarations at the top of the web part class to handle these two requirements.

```
IWebPartConnection _provider = null;
Literal _literal = null;
```

In the `CreateChildControls` method, enter code to use the provider data. You must check to see if the connection between the web parts has been made. If the connection is established, the `_provider` object is no longer null.

```
_literal = new Literal();
if (_provider != null)
```

www.it-ebooks.info

```
{
    if (_provider.ItemId > 0)
    {
        _literal.Text = _provider.ItemTitle + " has been selected.";
    }
    else
    {
        _literal.Text = "Please make a selection in the provider web part";
    }
}
else
{
    _literal.Text = "Web Parts have not been connected.";
}
this.Controls.Add(_literal);
```

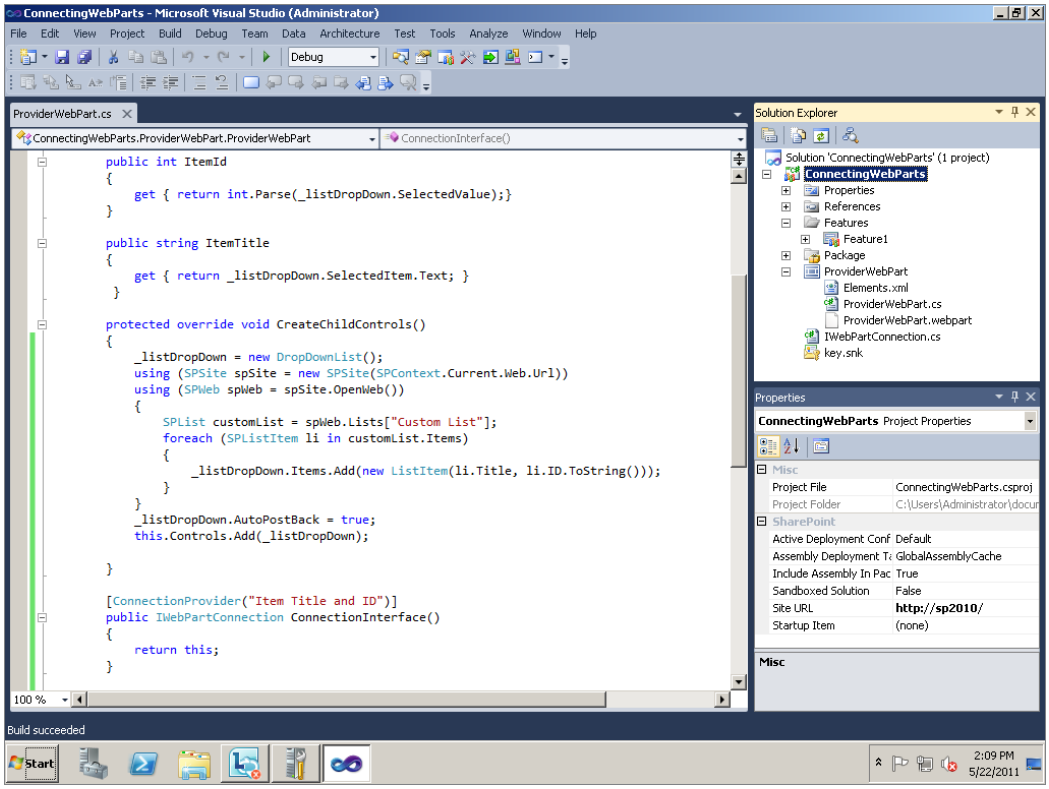


FIGURE 2-13: Provider web part code

Just like the provider web part, the consumer web part needs a connection interface to consume the information and instantiate the local interface object (`_provider` in the example). Therefore, you need to include a connection method within the consumer web part class.

```
[ConnectionConsumer("List Item Title and ID")]
        public void ConnectionInterface(IWebPartConnection connectionInterface)
```

www.it-ebooks.info

```
    {
        _provider = connectionInterface;
    }
```

The consumer web part code should now look similar to the code in Figure 2-14. Build the solution and ensure it correctly compiles.

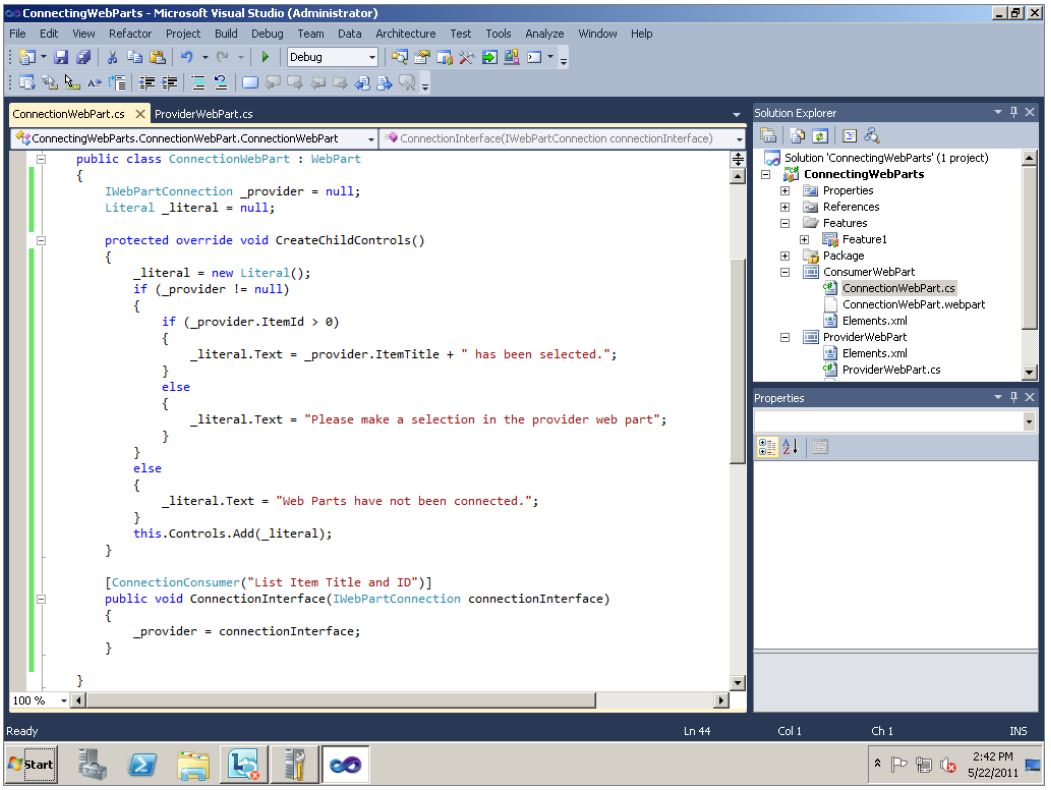


FIGURE 2-14: Provider web part code

Deploying the Solution

After everything compiles and builds successfully, right-click the solution, and select Deploy Solution. The solution is deployed to your SharePoint site. The feature is deployed as a Site Collection feature and is activated accordingly. You may verify the feature installation by viewing the Site Collection features in your SharePoint site.

Connecting the Web Parts

Create a new page within SharePoint, and add the provider and consumer web parts to the page. The web parts appear in the Custom category, as shown in Figure 2-15.

www.it-ebooks.info

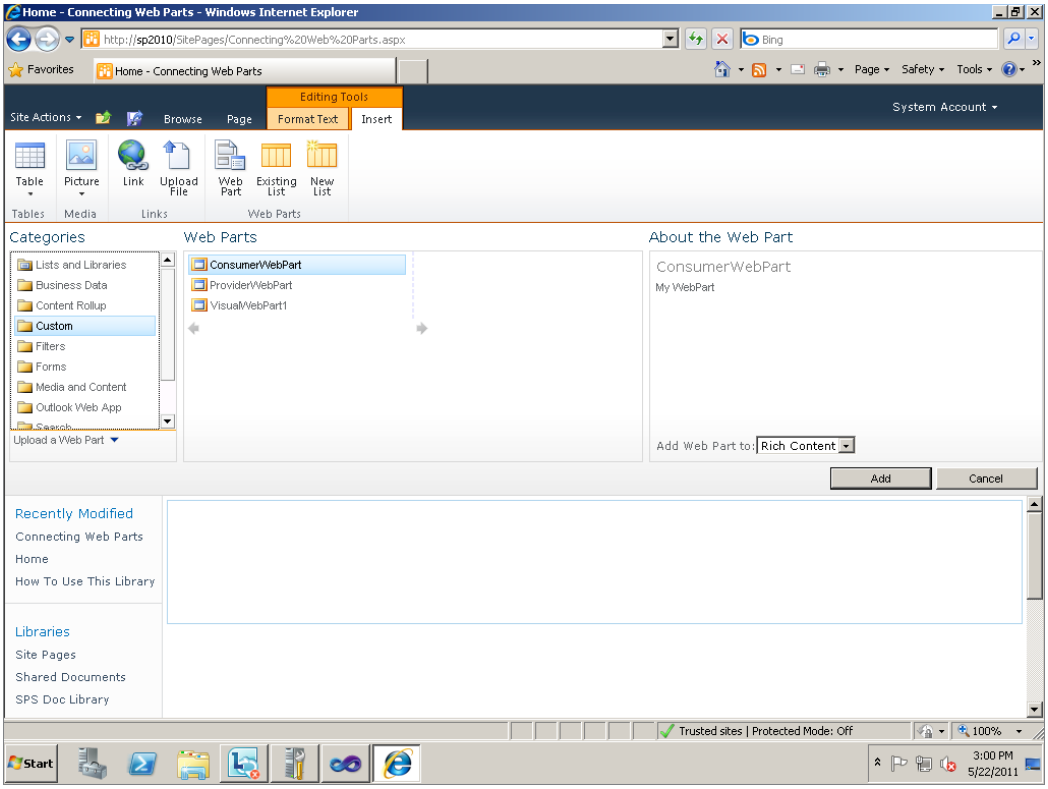


FIGURE 2-15: Adding the web parts to a page

After the web parts are added to the page, they need to be connected. Edit either the provider or consumer web part using the arrow drop-down button. When in edit mode, the Connections menu item appears within the arrow drop-down. On the provider web part, select Connections ➪ Send Item Title and ID To ➪ ConsumerWebPart, as shown in Figure 2-16. Alternatively, you could edit the consumer web part and select Connections ➪ Get Item Title and ID From ➪ ProviderWebPart.

Now that the web parts are connected, based on the example code of this section, when you select an item from the drop-down list in the provider web part, the consumer web part can determine what value was selected. Figure 2-17 shows an example implementation.

USING SHAREPOINT CONTROLS

SharePoint provides a myriad of controls that you may use within a Visual Web Part for SharePoint deployment and usage. Returning to the Visual Web Part project, if you use the SharePoint tag within the user control mark-up (`.ascx`) you can see the available SharePoint controls, as shown in Figure 2-18.

www.it-ebooks.info

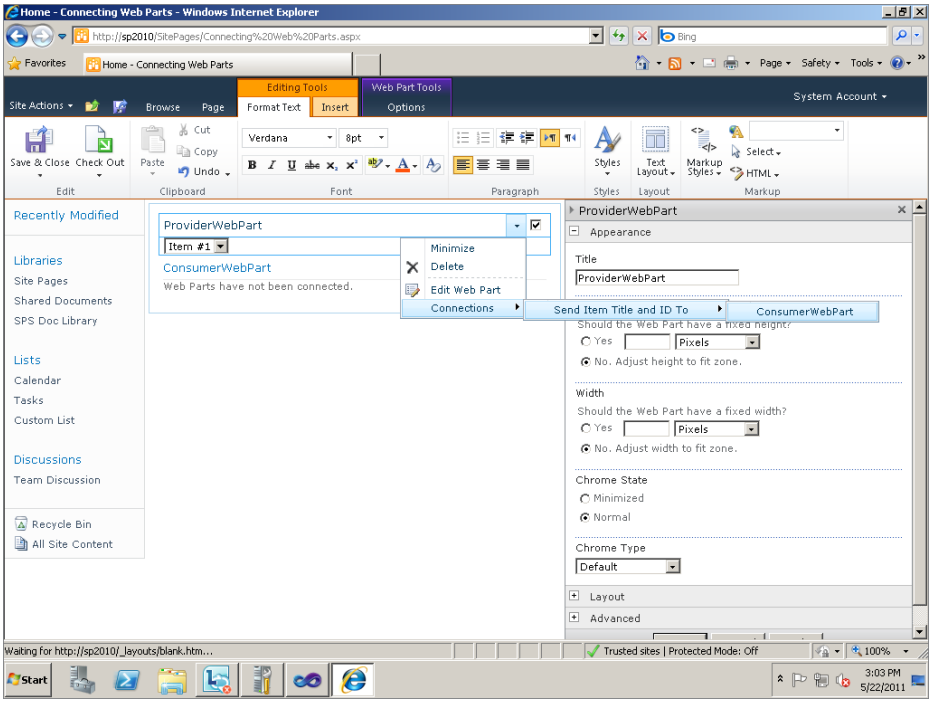


FIGURE 2-16: Connecting the web parts

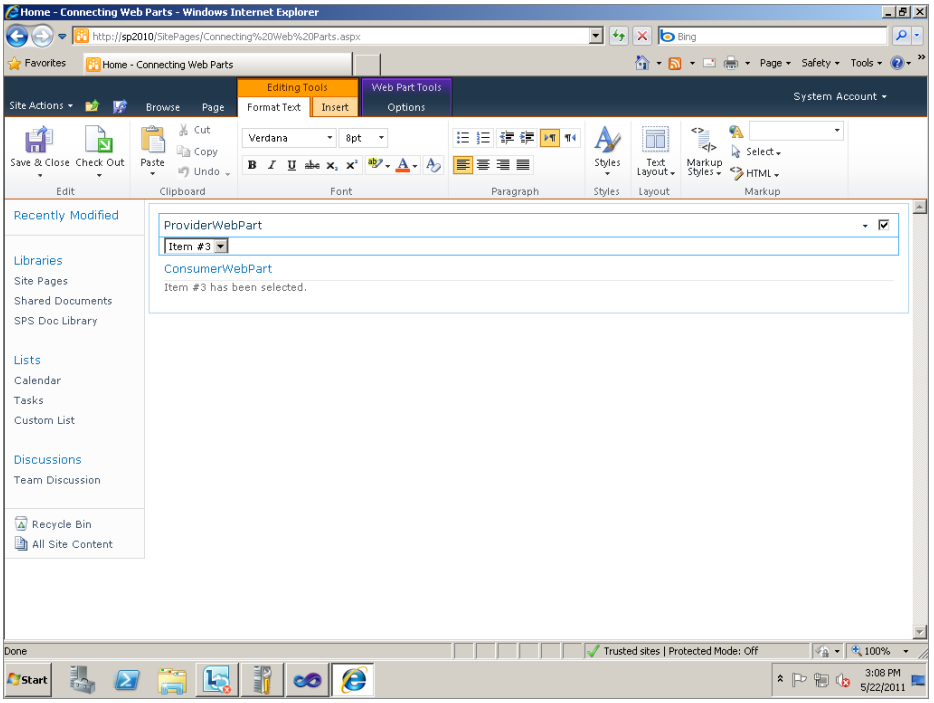


FIGURE 2-17: Selected item info is passed from the provider to the consumer

www.it-ebooks.info

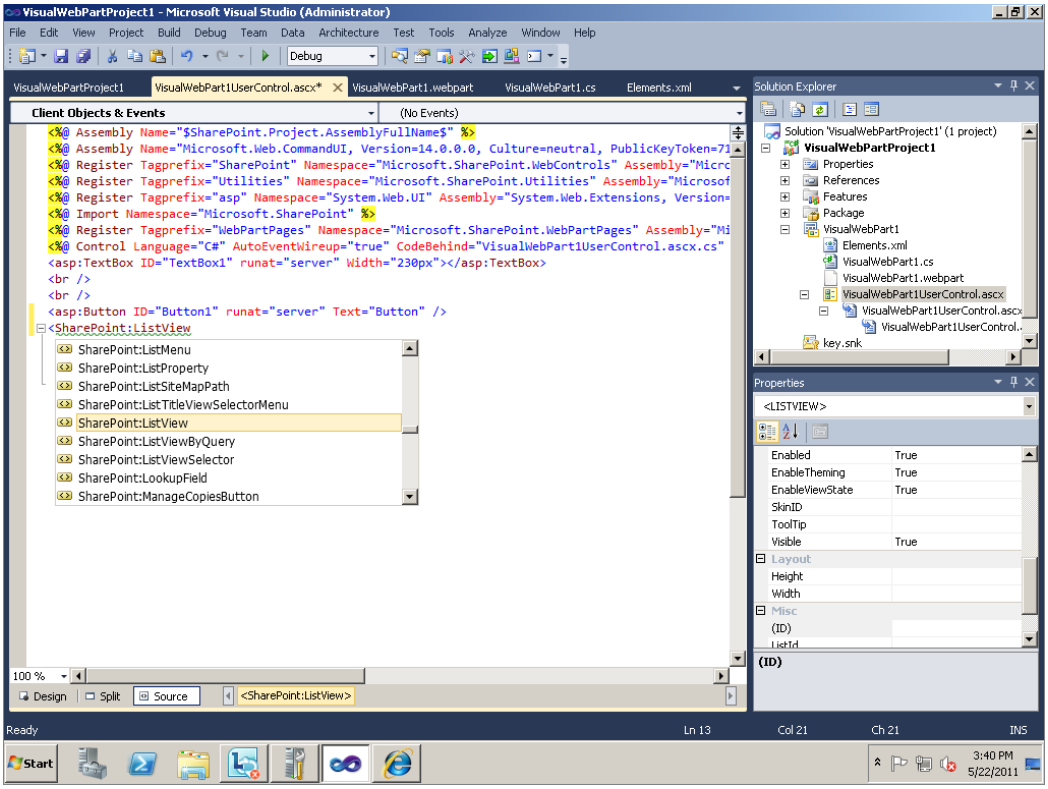


FIGURE 2-18: Using the SharePoint tag within the user control mark-up

For example purposes, use a `ListViewByQuery` control along with a date picker provided by the `DateTimeControl`. The markup looks like the following.

```
<SharePoint:ListViewByQuery runat="server" ID="CustomListView" />
<br />
<asp:Label ID="DateLabel" runat="server" Text="Expected Completion:" />
<SharePoint:DateTimeControl ID="CompletionDate" AutoPostBack="true"
     OnDateChanged="CompletionDate_Changed" DateOnly="true" runat="server" />
```

This provides a way to view a list based on a date selection. Make sure the list you use for this example contains a date column. The label prompts the user for the date. The `DateTimeControl` renders a date picker for the user to select a date. The `OnDateChanged` parameter references a code-behind method that needs to be created.

In the code-behind, you first need to add `using` statements for Microsoft.SharePoint, Microsoft.SharePoint.Utilities, and System.Web. These assemblies should already be referenced within the project. Add the `using` statements at the top along with the default entries. The entire list should look like the following.

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;
```

www.it-ebooks.info

```
using System.Web.UI.WebControls.WebParts;
using Microsoft.SharePoint;
using Microsoft.SharePoint.Utilities;
using System.Web;
```

Within the `Page_Load` method enter code to handle the display of the list based on the entered date by calling a `SetControls` method. The `SetControls` method code produces a list query and enables the `ListViewByQuery` control to render the list contents based on the date conditions written as a CAML query.

```
DateTime completeDate = DateTime.Now;

        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
               SetControls();
            }
        }

        private void SetControls()
        {
            CompletionDate.SelectedDate = completeDate;

            //Set the List on the ListViewByQuery control
            SPWeb webRoot = SPContext.Current.Web;
            CustomListView.List = webRoot.Lists["Custom List"];

            //Create SPQuery object
            SPQuery query = new SPQuery(CustomListView.List.DefaultView);
            query.Query =
            string.Format("<Where><Leq><FieldRef Name=\"Completion_x0020_Date\" />
            <Value Type=\"DateTime\">{0}</Value></Leq></Where>",
            completeDate.ToString("s"));

            //Disable filtering and sorting
            CustomListView.DisableFilter = true;
            CustomListView.DisableSort = true;

            //Set the list view Query to the query object
            CustomListView.Query = query;
        }
```

When comparing dates in CAML, the date value needs to be in a YYYY-MM-DD format. Otherwise the comparison will not work. Therefore the “s” format is used when converting the date variable to a string within the CAML query.

Now all you need is the date change event handler, which retrieves the completion date value from the date picker control and then sets the controls accordingly.

```
    protected void CompletionDate_Changed(object sender, EventArgs e)
        {
            completeDate = CompletionDate.SelectedDate;
            SetControls();
        }
```

www.it-ebooks.info

Build the solution to ensure everything properly compiles. Then deploy the solution to SharePoint. On a site page, add the new web part onto the page. Changing the date causes a post back and filters the view of the list items.

LEVERAGING THE XSLT WEB PART

SharePoint Designer 2010 enables you to add XSLT List View web parts to pages and configure them accordingly. It is a little different than creating a web part in Visual Studio 2010, which can then be packaged up and deployed to multiple sites or environments. XSLT List View web parts are used more for customizing list content displays on specific pages.

The first step to create an XSLT List View web part on a page is to launch SharePoint Designer 2010 and open your SharePoint site. You can launch SharePoint 2010 Designer from the Start menu (All Programs ➪ SharePoint ➪ Microsoft SharePoint Designer 2010). When SharePoint Designer 2010 loads, the Sites menu page displays. Click the big Open Site button to open your SharePoint site.

Navigate to the Pages or Site Pages library where the page you want to modify resides. When you find the page, you must edit the page in normal mode by using the Edit File menu item. On the Pages top ribbon, select Edit File ➪ Edit File in Normal Mode, as shown in Figure 2-19.

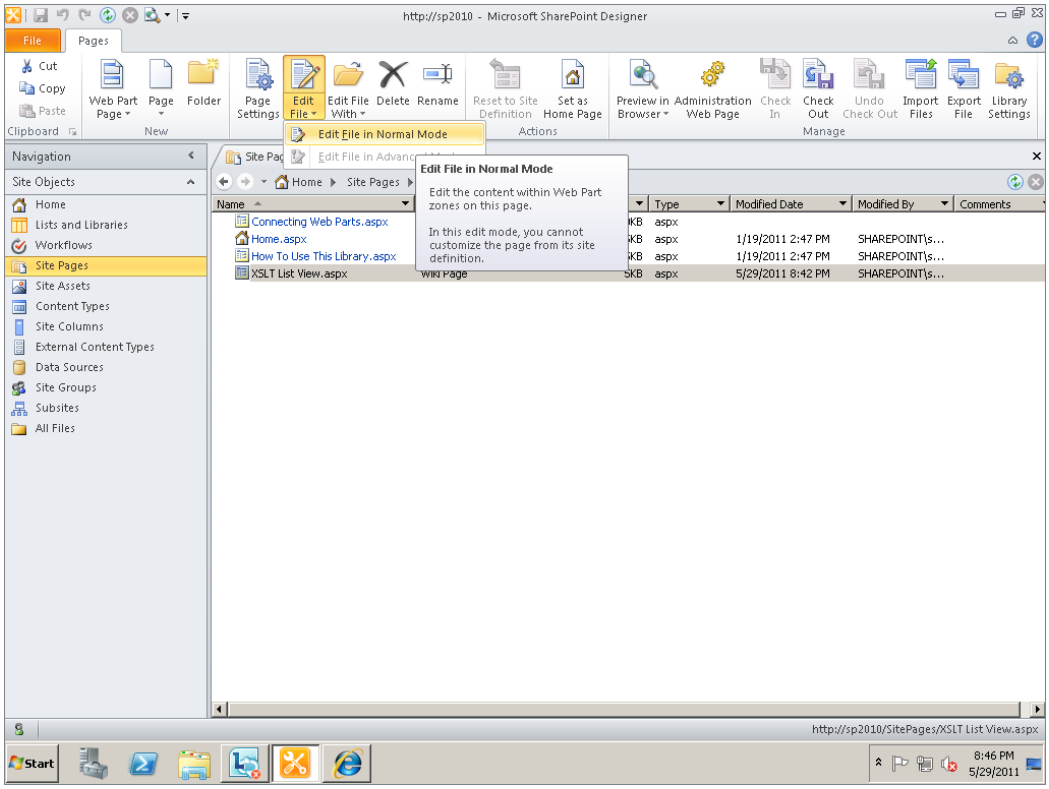


FIGURE 2-19: Editing the page in advanced mode

www.it-ebooks.info

When the page loads in the designer, locate the PlaceHolderMain section of the main page content. From the Insert top ribbon, click the Data View drop-down menu and select a list, as shown in Figure 2-20.

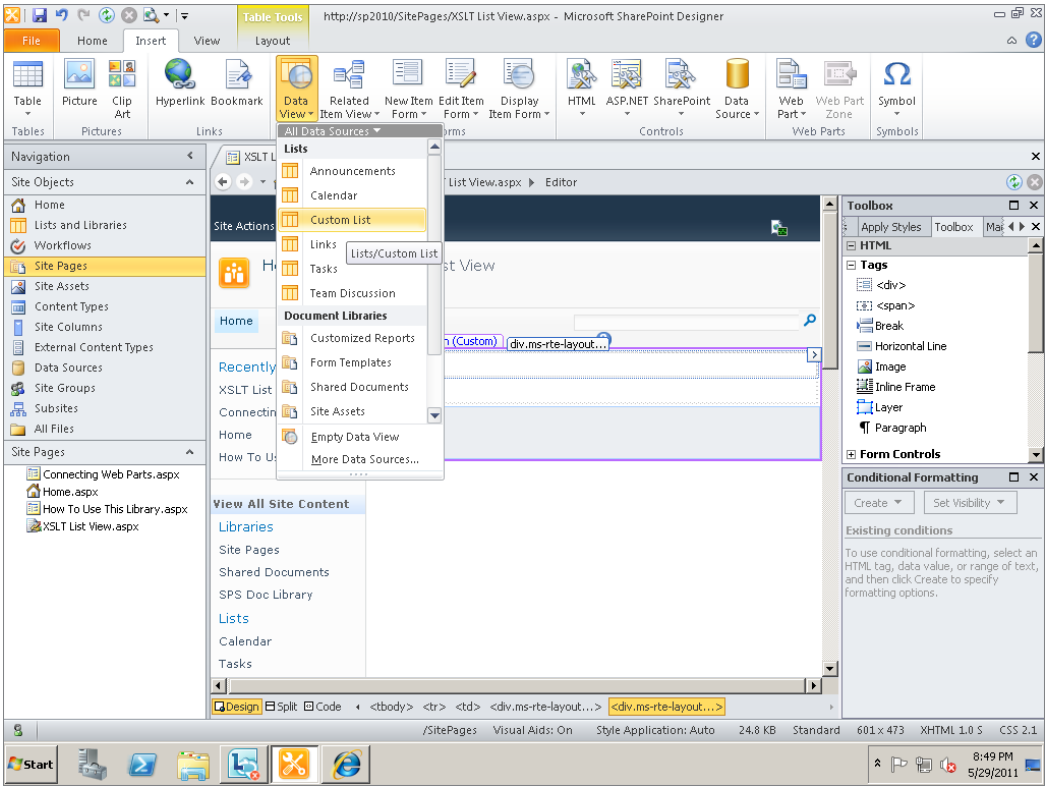


FIGURE 2-20: Inserting the XSLT web part

The XSLT web part is generated on the page and becomes selected. The Options ribbon displays. Many different menu items within the Options ribbon enable you to modify the XSLT web part. You may change the paging by using the Paging button or add and remove columns by using the Add/Remove columns button. Saving the changes to your page saves the file back to SharePoint.

SUMMARY

This chapter explored various methods and options to create visual web parts for SharePoint 2010. Web parts are the building blocks of SharePoint page content and may be constructed, packaged, and deployed from Visual Studio 2010. SharePoint Designer 2010 may also be leveraged and used to create specific data view content using the XSLT List View web part.

The next chapter looks at the Enterprise Search service application in SharePoint 2010 and explores the steps to customize the search experience.

www.it-ebooks.info

3

Customizing Search

This chapter discusses how to use the search features of SharePoint to find content and to proactively notify users of new content; in addition, it covers the steps necessary to plan, configure, and maintain the search components of SharePoint. Search is a key component for easily finding content in a portal or site of any considerable size. When navigation, bookmarks, and guessing let users down, Enterprise Search is the right tool to finding what they need.

CONFIGURING AND CREATING CONTENT SOURCES

Correctly configuring search and indexing is important so that users receive accurate, relevant, and expected results. All the content that you want to make available to users must be defined as content sources and indexed on a regular basis so that the results are available to be returned in searches. Content sources are defined and indexing is scheduled and completed by the Search Service Application (SSA).

The SSA requires configuration to determine how content is crawled and what data should appear in what scopes. Search scopes enable users to target the slice of content they want to search. If configured appropriately, users can pick a scope for their search that returns a reasonable number of relevant results.

Indexing Content

Choosing what content to index and configuring your indexing settings is the primary task for indexing content. SharePoint 2010 can index content from several types of content sources, including these sources for which SharePoint has provided out-of-the-box support:

- **File share content:** SharePoint can index content placed on file shares.
- **Exchange server content:** Exchange public folders content is a good potential source for indexing corporate knowledge.

www.it-ebooks.info

- **Lotus Notes servers:** If your organization uses Lotus Notes, SharePoint can index the Lotus Notes databases. You need to run the Lotus Notes Index Setup Wizard to configure the Lotus Notes protocol handler before configuring a Lotus Notes content source.
- **SharePoint sites:** SharePoint 2010 searches all the local SharePoint content and can be configured to index content for cross-organizational content.
- **Business data:** SharePoint 2010 can also index data that has been defined in the Business Connectivity Service.
- **Web sites:** SharePoint 2010 can index Web site content for cross-platform or cross-product integration. Web sites are defined as any other Web-based services that may be accessed through a URL either internal or external to an organization.

Planning Content Sources

Selecting your content sources from the myriad of available corporate repositories of data is an important step in the indexing configuration process. Indexing content can be a resource-intensive task, both for the indexing server and for the server responding to the crawler requests for the content.

The SSA is automatically configured with a content scope that searches all the local SharePoint sites, which are defined as all the site collections using the SSA. You can choose to index SharePoint content that is external to the SSA, but this most likely results in content being indexed more than once: first by the local SSA and one or more times by external SSAs that have defined it as a content source. The exception to this would be in the SharePoint sites that you index as part of SharePoint Foundation Services farm.

For each set of content that is a potential source for your SSA users, you need to decide what the source is and how often you will do full and incremental updates to the index. For external content, or content that you do not control, requesting too much content or content too often may overload the external source, and that administrator can block you from crawling in the future.

For each content source that you identify, determine the following content source options:

- **How deep would you like to crawl?** For SharePoint sites, you can determine whether you want to search for everything under the start address or just the SharePoint site of the start address. For Web content, you need to decide if you will stay on just the first page of the site or the entire server. You can also set custom hop settings for the indexing to follow to limit the number of server hops and depth of the pages. Setting the server hop limit to 2 or more can be overwhelming to your indexing resources because indexing will not only index your starting address, but also any server linked from the starting address content (one server hop) and any other servers mentioned in the first server hop (second server hop).
- **What is the crawl schedule?** The crawl schedule can be determined by understanding how often the target content changes and how long it takes to index the content source. Try to plan full crawls for times when the content source has low resource usage, and schedule them less frequently if the content does not frequently change. Schedule incremental crawls to update the content between full crawls. Usually running incremental crawls is sufficient to keep the content

www.it-ebooks.info

sources up-to-date. Full crawls are necessary when introducing new content sources, types of content, or business entity data fields which need to be mapped to managed properties.

- **Does this content source need to be accessed by an account other than the default content access account?** The default content access account credentials will be presented to gain read access to content sources unless specified by a crawl rule for a specific site. Managing several content access accounts can be a time-consuming procedure, especially if the accounts require password changes on a regular basis. It is recommend to define only unique content access accounts if necessary because the default content access account cannot be used.

Implementing Content Sources

To implement the content sources that you have identified, follow these steps for each content source:

1. Navigate to the administration page for your Search Service Application, and from the Search section, select Search settings.
2. Select Content Sources.
3. In the top navigation bar, click New Content Source.
4. In the Name field, enter a name for your content source, as shown in Figure 3-1.
5. Select the type of content to be crawled.
6. For Web sites or SharePoint sites, enter the root start address from which the crawler should access and crawl from.
7. Set the schedule for the full crawl and incremental crawls in the Crawl Schedules section. Any schedules that you have previously configured will be available in the pull-down, or you can select the Create Schedule link to define a new schedule.
8. In the Start Full Crawl section, select whether the crawl should immediately start a full crawl of the content source.
9. Click OK.

CONFIGURING SEARCH SCOPES

You can configure search scopes at both the SSA level and the site collection level. SSA search scopes should be broad scopes relevant to all users regardless of the site collection. In the out-of-the-box installation, SharePoint creates two SSA-level search scopes: All Sites and People. Additional SSA search scopes could be configured for organizational information pertinent to everyone, such as a corporate event portal. Site collection search scopes should be specific to the information on that site collection.

Search scopes are defined by one or more rules. The rules can be based on managed properties and location or content sources, and can include rules that exclude content. Your goal in creating search scopes is to create logical divisions of the content so that users understand which scope to pick and get a reasonable number of results returned when they execute their search.

www.it-ebooks.info

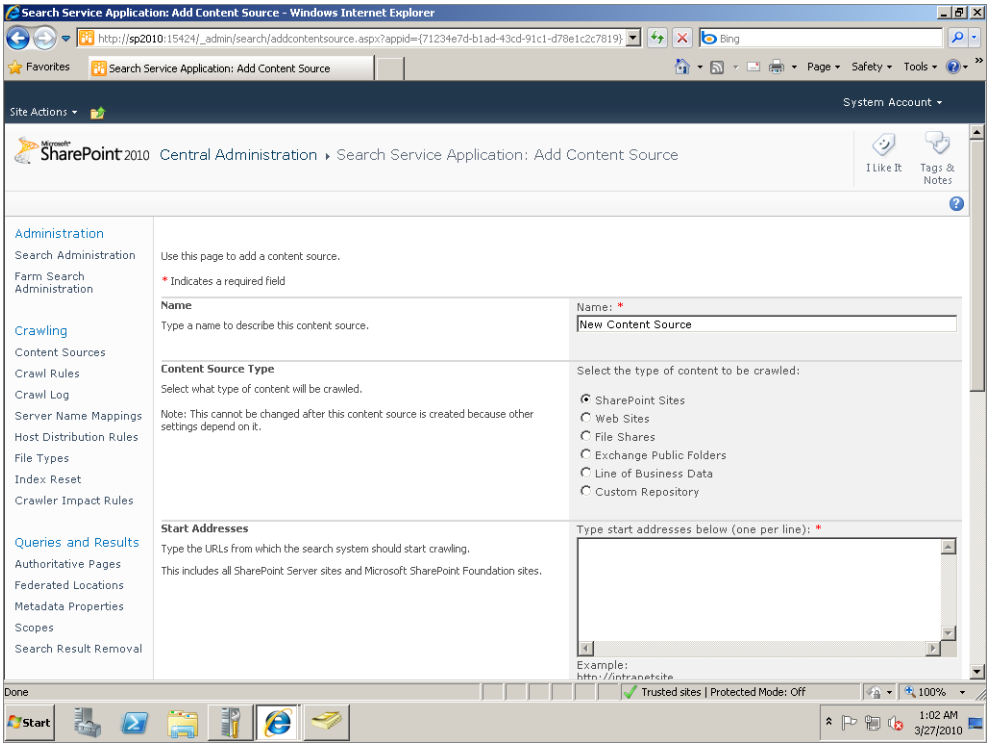


FIGURE 3-1: Adding a new content source

Search scopes are organized in display groups. Search web parts use these groups to identify which scopes to show in the search drop-down menu.

Defining Search Scopes in the Service Application

In addition to defining the SSA scopes, the SSA administrator can also create scope rules. To define a scope at the SSA level, follow these steps:

1. Navigate to the administration page for your Search Service Application.
2. From the Queries and Results Quick Launch section, Select Scopes.
3. From the page's top navigation bar, Select New Scope.
4. In the Title boxes, provide a name for your search scope, as shown in Figure 3-2.
5. Select whether you want to use the default Search results page or enter a different search results page that you would like to use, and click OK.
6. From the View Scopes page, select Edit Properties and Rules from the drop-down menu on the scope title that you just added.
7. Select New Rule.

www.it-ebooks.info

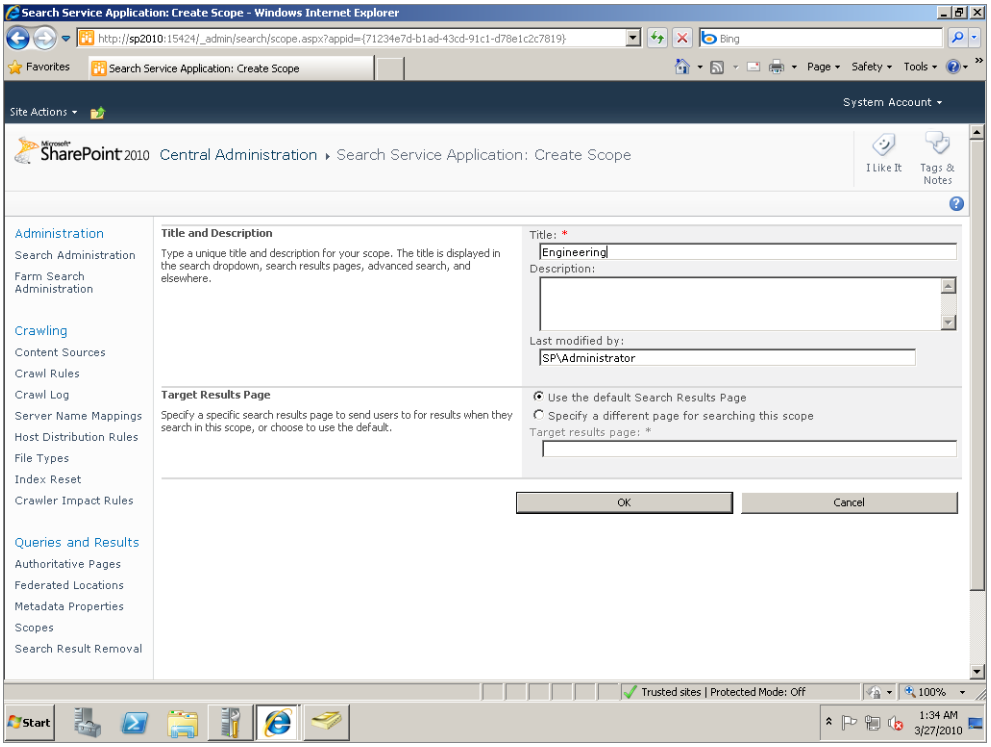


FIGURE 3-2: Creating a new shared scope

8. If you want to create a rule based on the Web address properties of the indexed items, select Web Address in the Scope Rule Type, and select whether the Web address will be limited by folder, hostname, or domain/subdomain.
   - If you select Folder, enter the URL of the folder that you want the rule to be based on in the Folder box, as shown in Figure 3-3, for example, `http://server/site/folder`.
   - If you select Hostname, enter the hostname that you want the rule to be based on in the Hostname box, for example, servername.
   - If you select Domain or subdomain, enter the domain name that you want the rule to be based on in the Domain or subdomain box, for example, `office.microsoft.com`.
9. If you want to create a rule based on the properties of the indexed items, in the Scope Rule Type section, select Property Query. In the Add property restrictions pull-down, select the property that you want the rule to be based on. In the = field, enter the rule value, for example, Author (is the property) = John Doe (rule value).
10. If you want to create a rule based on a specific content source, in the Scope Rule Type section, select Content Source, and from the pull-down menu, select the content source.
11. Select the All Content radio button if you want the scope to return all indexed items.

www.it-ebooks.info

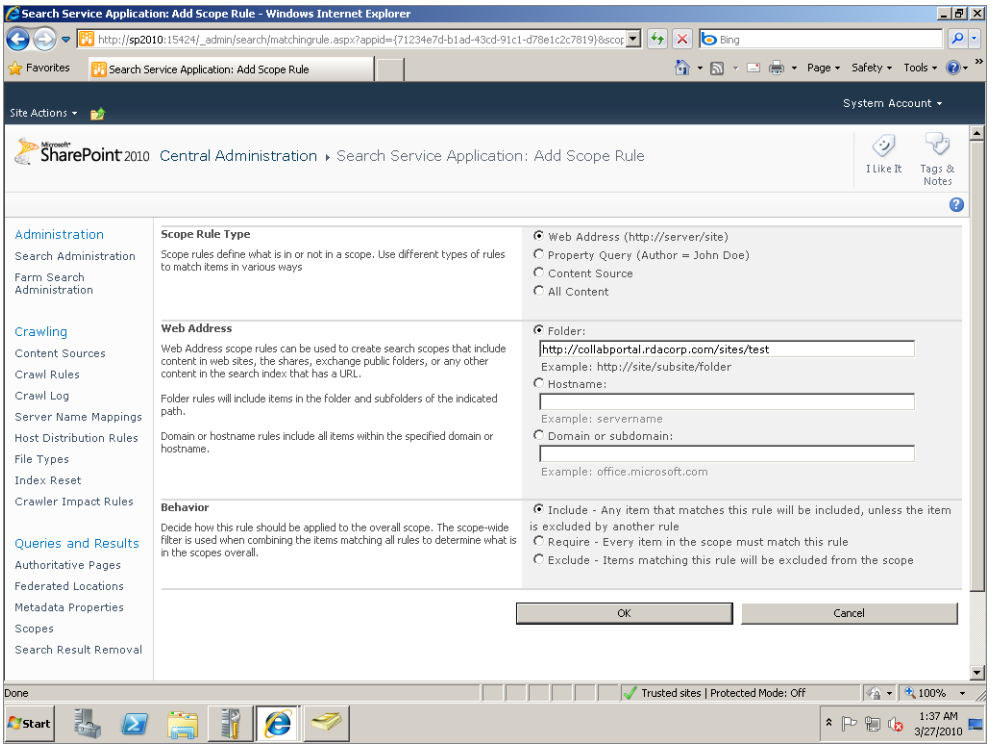


FIGURE 3-3: Creating a scope rule to include content from a Web address

12. Select if you want to include, require, or exclude content based on the rules you enter.
    - Include rules specifying what content will be included unless another rule excludes them.
    - If you choose a required rule, all items returned in the scope must match the rule.
    - Exclude rules specify what content will not be included. This content will not be included even if it matches the other rules.
13. Click OK.
14. Add as many rules as you need to tune your scope to the appropriate content.

Defining Site Collection Search Scopes and Scope Display Groups

SharePoint provides two display groups to organize your site collection search scopes, one for the search drop-down and one for the advanced search page. To create a new display group, follow these steps:

1. Go to the top-level site of the site collection for which you want to add the scope, and from the Site Actions menu in the top-left corner, select Site Settings.
2. From the Site Collection Administration menu, select Search scopes.

www.it-ebooks.info

3. From the top navigation bar, select New Display Group.
4. In the Title field, provide a name for your display group.
5. If you have already created the scope or scopes that you want to include in this display group, select the scope in the Scopes section, as shown in Figure 3-4.
6. From the View Scopes page, select Edit Properties and Rules from the drop-down menu on the scope title that you just added.
7. Click OK.

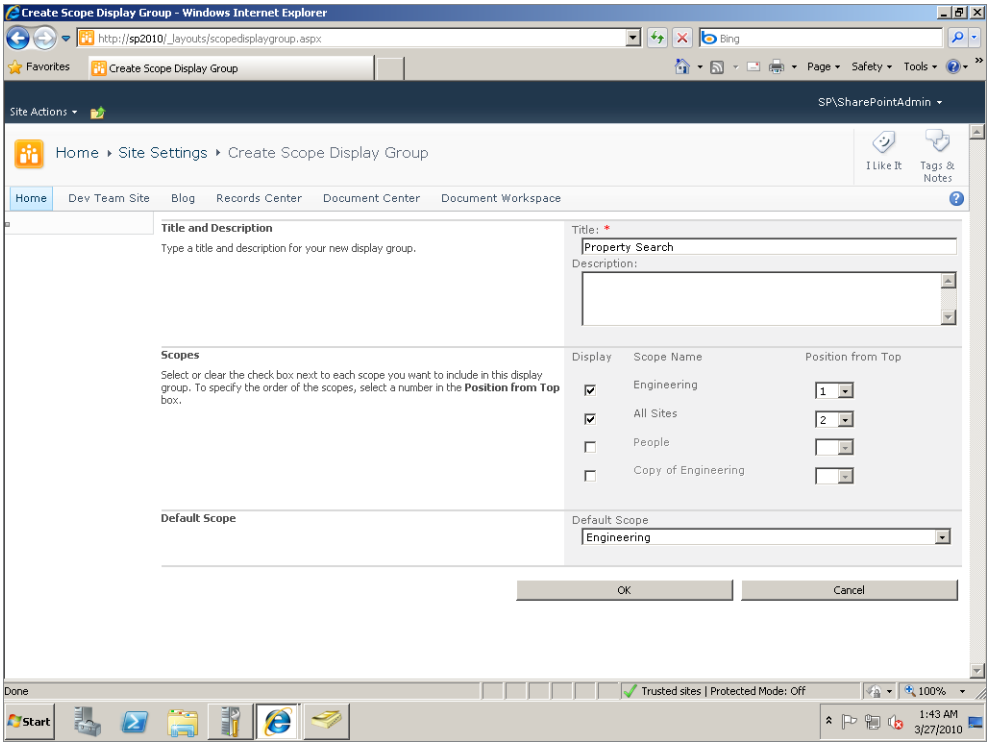


FIGURE 3-4: Creating a new display group

The scopes text colors give the appearance that they are disabled. You can still click the check boxes next to them. You may see them jump out of order depending on the order you click them. This is a result of the defaulting position from the top based on your click order.

To define a scope for your site collection, follow these steps:

1. Go to the top-level site of the site collection for which you want to add the scope, and from the Site Actions menu in the top-left corner, select Site Settings.
2. From the Site Collection Administration menu, select Search scopes.
3. From the page menu bar, select New Scope.

www.it-ebooks.info

4. In the Title field, provide a name for your scope.
5. Select whether you want to use the default Search results page, or enter a different search results page, and click OK.
6. From the View Scopes page, from the drop-down menu on the scope title that you just added, select Edit Properties and Rules.
7. Select New Rule.
8. If you want to create a rule based on the Web address properties of the indexed items, in the Scope Rule Type section, select Web Address, and select whether the Web address will be limited by folder, hostname, or domain/subdomain.
   - If you select Folder, type the URL of the folder that you want the rule to be based on in the Folder box, for example, `http://server/site/folder`.
   - If you select Hostname, type the hostname that you want the rule to be based on in the Hostname box, for example, servername.
   - If you select Domain or subdomain, enter the domain name that you want the rule to be based on in the Domain or subdomain box, for example, `office.microsoft.com`.
9. If you want to create a rule based on the managed properties of the indexed items, in the Scope Rule Type section, select Property Query. Select the property that you want the rule to be based on in the Add property restrictions pull-down. Enter the rule value in the = field. For example, Author (is the property) = John Doe (rule value). This rule operates by finding exact matches.
10. Select the All Content radio button if you want the scope to return all indexed items.
11. Select if you want to include, require, or exclude content based on the rules you enter. Include rules specify what content will be included unless another rule excludes them. If you choose a required rule, all items returned in the scope must match the rule. Exclude rules specify what content will not be included. This content will not be included even if it matches the other rules.
12. Click OK.
13. Add as many rules as you need to tune your scope to the appropriate content.

CREATING MANAGED PROPERTIES

Managed properties are gathered during the indexing process and selected by the SSA administrators to be used to customize and hone a search. Managed properties are part of the formula that determines the relevance ranking of search results and can also be used to configure search scopes.

By defining managed properties, you can make sure that the metadata that is important to your organization and content is consistent and used to order search results. The SSA administrator maps managed properties to crawled properties. SharePoint 2010 creates many or most of the useful managed property mappings when it is installed, including the following:

- Author
- Department

www.it-ebooks.info

- Division
- Location
- Status
- Title

You should add managed properties for key pieces of metadata that the crawler returns and that are important to your organization. These properties can be used both for search scopes and relevancy ranking. If you want a division-level portal to create search scopes based on the division, you must have division as an SSA managed property. SSA managed properties are not available as search criteria unless specified in their managed property settings.



You can map one or more crawled properties to one managed property to eliminate duplication of properties. For example, you may want to map each appearance of the author in the crawled properties to the author managed properties.

To create a new managed property, follow these steps:

1. Navigate to the administration page for your Search Service Application.
2. Under the Queries and Results section of the Quick Launch, select Metadata Properties.
3. From the page menu bar, select New Managed Property.
4. In the Property name field, type the name you want to use. This should be a clear and identifiable name such as "author."
5. Select the type of property, for example, text, integer, or decimal.
6. Enter the crawled property that you want to match to the managed property. If you want to map multiple crawled properties to the managed property, add the multiple crawl property values and order them in descending order, of which the top value is the most authoritative value, as shown in Figure 3-5. If a document has multiple crawled property values, the managed property returns the crawled property at the top of the list.
7. Select whether you want the managed property to be used in search scopes.
8. Click OK.

CUSTOMIZING THE ENTERPRISE SEARCH CENTER

SharePoint 2010 provides the search components as web parts so that you can customize how and where search results are returned. This flexibility enables you to customize the Enterprise Search Center so that results are returned in the most usable format for your organization and users.

www.it-ebooks.info

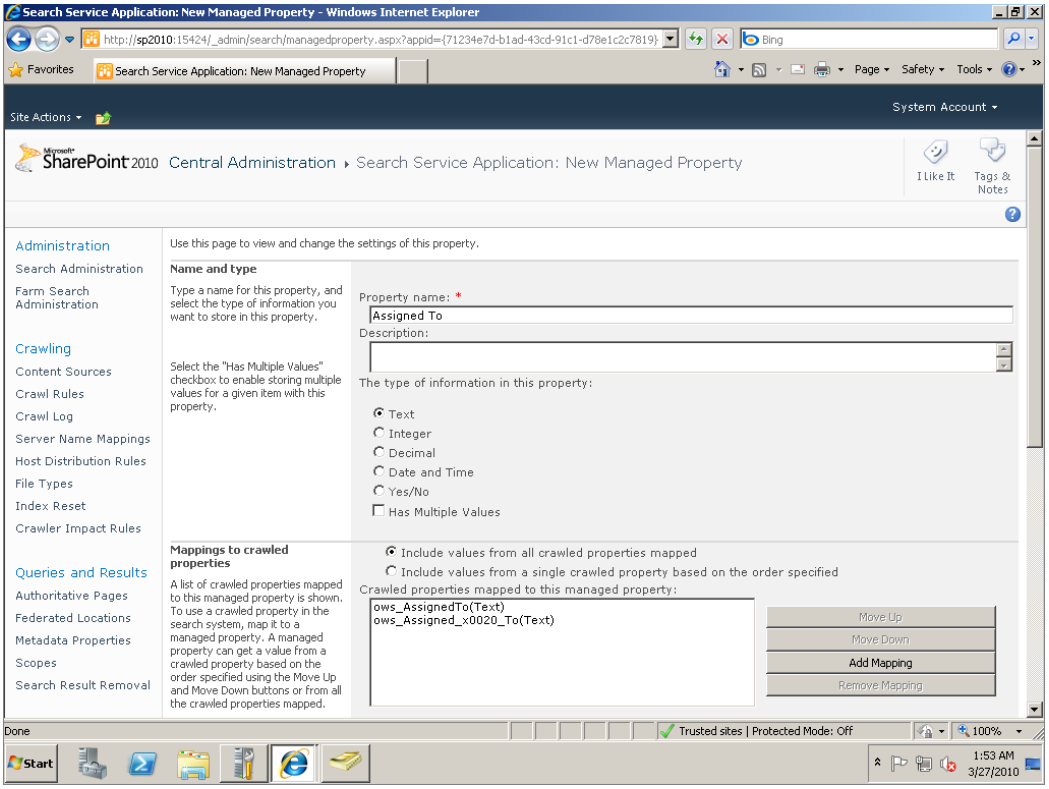


FIGURE 3-5: Creating a managed properties group

The available search web parts follow:

- **Advanced Search Box:** Enables users to find documents using various search criteria including words, phrases, scope, language, and document type.
- **Federated Results:** Displays results from a configured location.
- **People Refinement Panel:** Provides filtering of people search results. Depending on the actual results, refinement is performed by various profile items including Department, Title, and Responsibilities.
- **People Search Box:** Enables users to search for a person by name, department, title, responsibilities, skills, and memberships.
- **People Search Core Results:** Displays the people search results and should typically be on the same page as the People Search Box web part.
- **Related Queries:** Displays similar queries that other users have used to perform the current query.
- **Search Box:** Contains the simple search box found on all portal pages by default.
- **Search Core Results:** Displays the search results generated from the search box search.

www.it-ebooks.info

- **Search High Confidence Results:** Displays the special term and high confidence result sets.
- **Search Paging:** Displays the search paging capability, so a user can scroll through multiple pages of search results.
- **Search Statistics:** Displays search statistics, including number of results and duration of search time.
- **Search Summary:** Displays search summary.
- **Search Visual Best Bets:** Displays best bets for the current search results based on the search query.
- **Top Federated Results:** Displays the top results from a configured location.

Creating a New Search Results Page

As a sample of customizing the Enterprise Search Center, you create a new tab to display search results for a display group. To do this, follow these steps:

1. From the Site Actions menu on any page in your Enterprise Search Center site, select New Page, and type a title for your new page.
2. Click Create.
3. SharePoint creates the page as a Search Results page, as shown in Figure 3-6.

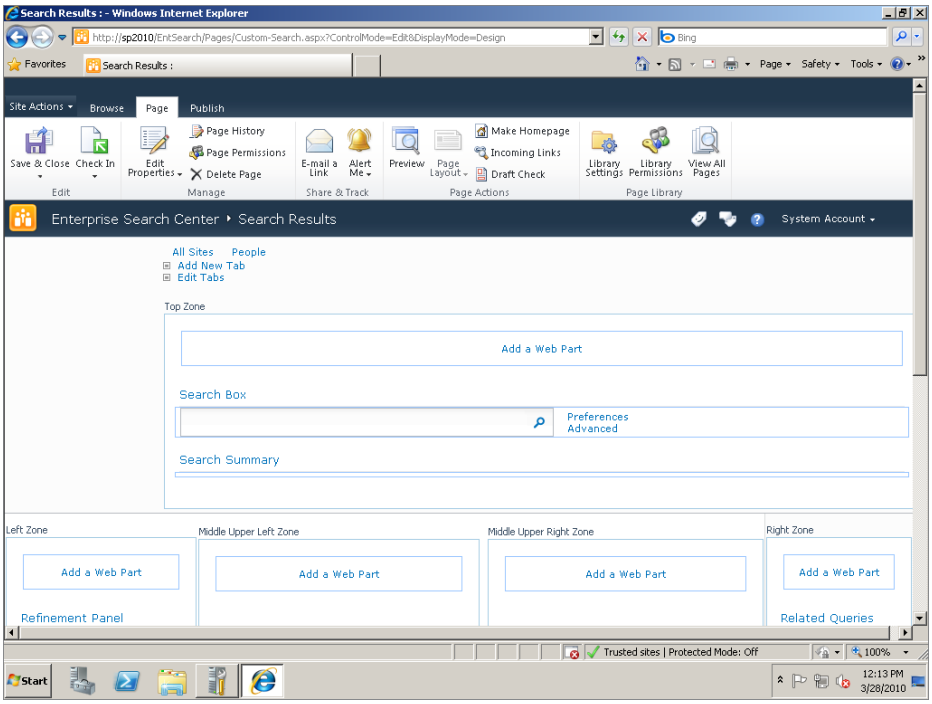


FIGURE 3-6: New Enterprise Search Center Search Results page

www.it-ebooks.info

4. In the top section of your new page, select Add New Tab.
5. Type a tab name in the Tab Name field, your page address in the Page field, and Tooltip (optional), as shown in Figure 3-7.
6. Click OK.

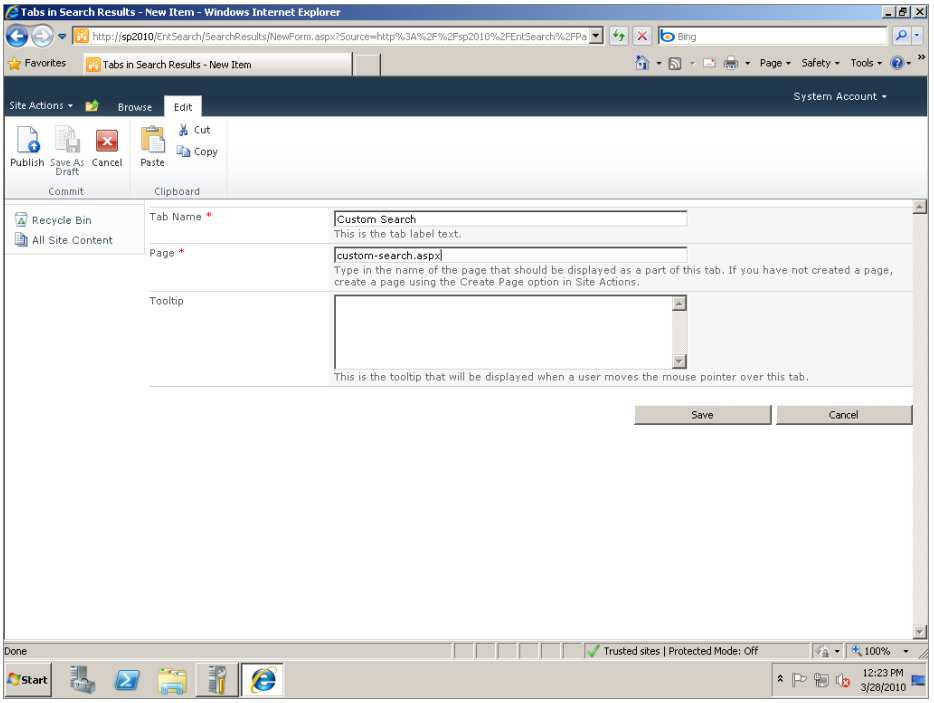


FIGURE 3-7: Creating a New Enterprise Search Center tab

7. From the top-right edit menu on the Search Box web part, select Edit Web Part.
8. Expand the Miscellaneous section of the web part.
9. In the Target search results page URL field, type your page name, as shown in Figure 3-8.
10. In the Scope Display Group field, type the name of your display group, as shown in Figure 3-9.
11. Click OK to close the Search Box Web Part Properties Pane.
12. If you would like to return more than ten results per page, from the edit menu on the Search Core Results Web Part, select Edit Web Part.
13. Expand the Results Display/Views section.
14. In the Results Per Page field, type the number of results that you want returned, as shown in Figure 3-10.
15. Click OK to close the Search Core Results Web Part Properties Pane.

www.it-ebooks.info

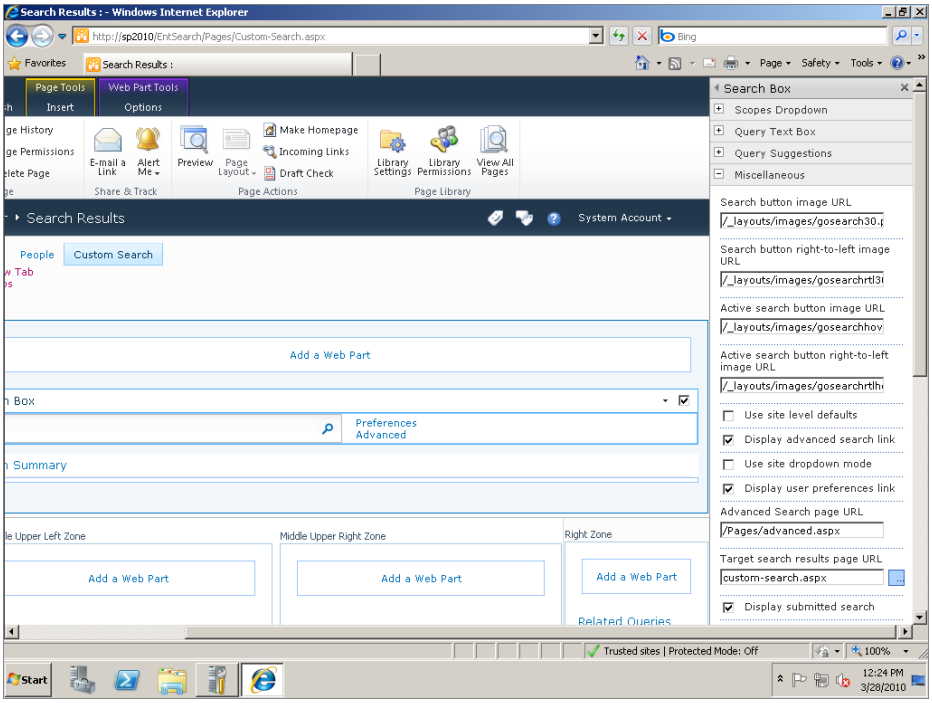


FIGURE 3-8: Modifying the Target Search Results page

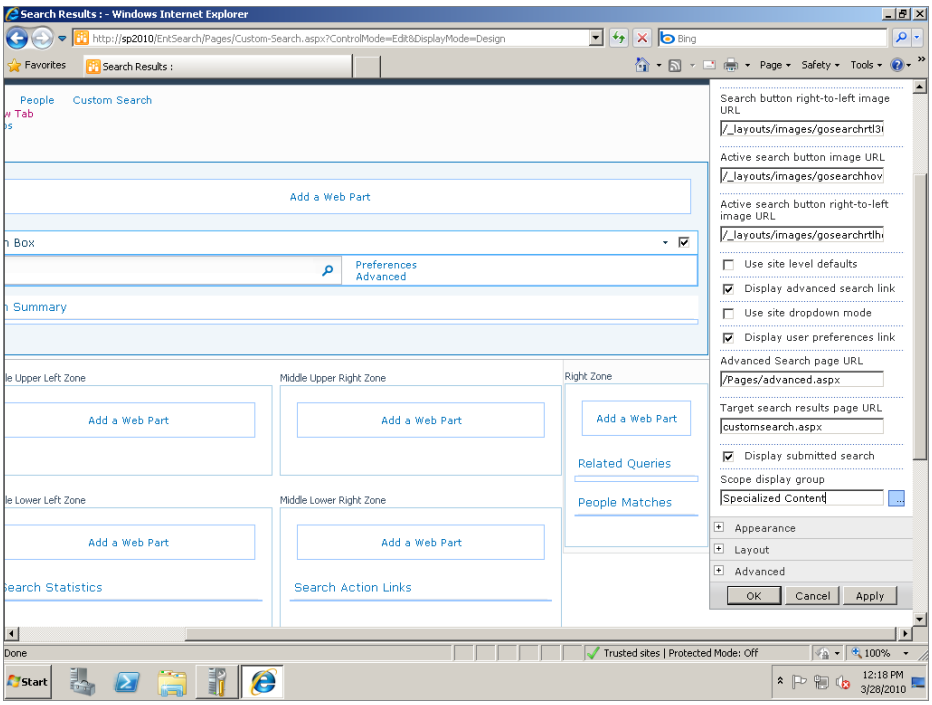


FIGURE 3-9: Modifying the scope display group in the Search Box web part

www.it-ebooks.info

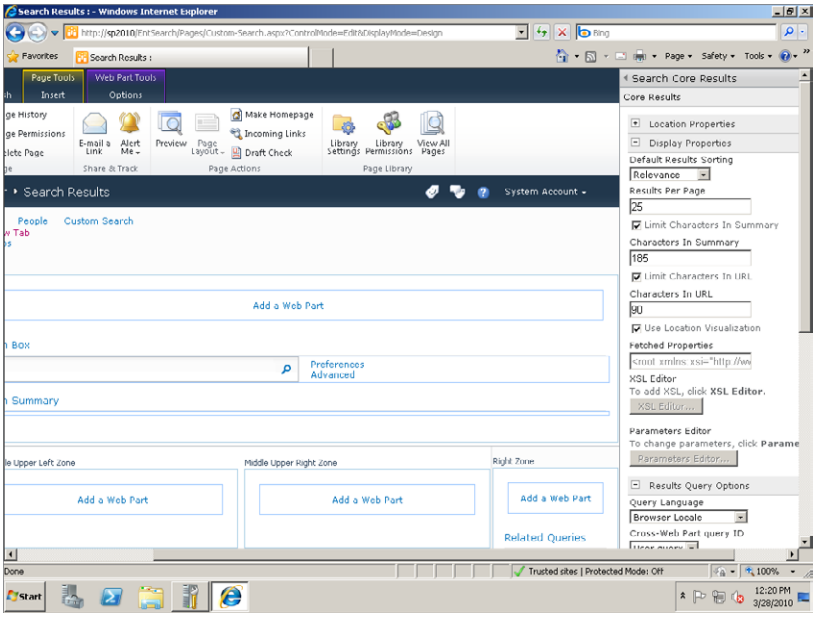


FIGURE 3-10: Modifying the number of results returned in the Search Core Results web part

If you want to use a specific scope but do not want to show the scope drop-down in the search box web part, you can configure the web part to not show the scope drop-down and enter the scope in the Additional Query Terms field in the query text box section of the web part. The display group term should be entered as scope: "scope name," where scope name is the name of your scope in the additional query terms field.

Modifying the Advanced Search Screen

Eventually users will find the Advanced Search screen and want to perform targeted or specific searches. The Advanced Search page is automatically generated within your Search Center and contains the Advanced Search web part. Modifying this web part enables you to customize the user experience. You can access the Advanced Search page by navigating to your Search Center site and clicking the Advanced link next to the Search Box.

Removing the Language Selection

Usually your SharePoint site contains one common language. Therefore, users may not need to select a specific language. To reduce the clutter on the Advanced Search page, you may remove the language selection by following these steps:

1. Edit the Advanced Search page.
2. Edit the Advanced Search web part.
3. In the Scopes section, uncheck the Show the Languages Picker option, as shown in Figure 3-11.
4. Click OK.

www.it-ebooks.info

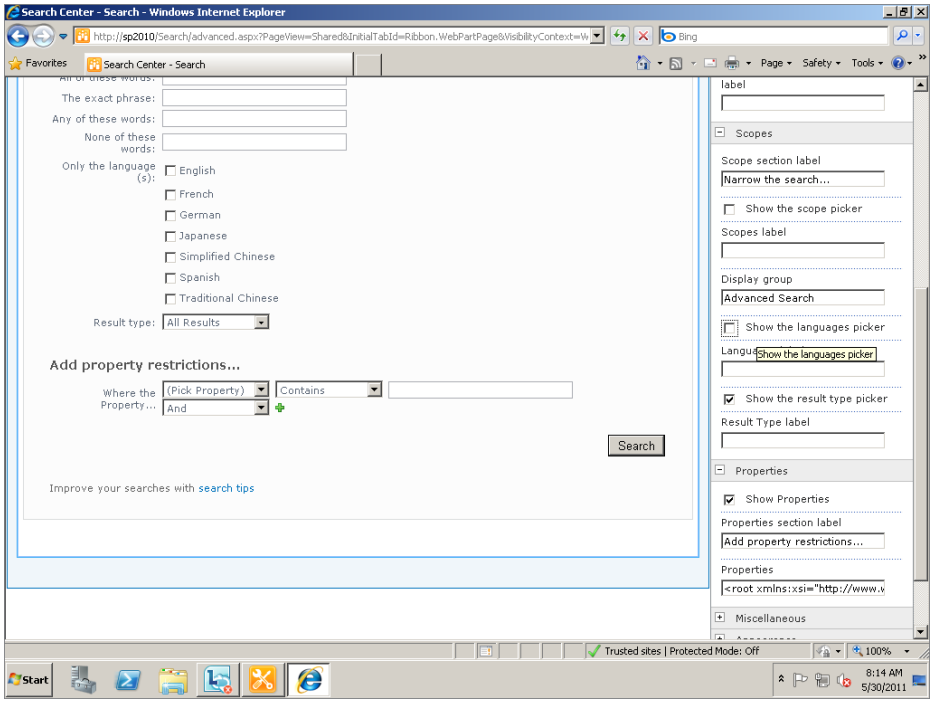


FIGURE 3-11: Removing the Languages Selection

Adding a New Results Type

The Result Types and their corresponding properties are preconfigured with several options including the most common Microsoft Office document types (for example, Word, Excel, and so on). If you have installed the PDF iFilter and are crawling PDF documents, you may want to include these as one of the result types. This requires creating the type definition and the corresponding properties of the documents. To add a new results type, follow these steps:

1. Edit the Advanced Search page.
2. Edit the Advanced Search web part.
3. Under the Properties section, click the Ellipsis button to the right of the Properties text box. The Text Editor dialog appears.
4. Copy and paste the contents of the Text Editor into Visual Studio or some other text editing application.
5. Scroll down to the `PropertyDefs` section. These are all the available properties. Any new document properties need to be defined here.
6. Add new property definitions as needed for your new results type.

```
<PropertyDef Name="CustomerName" DataType="text" DisplayName="Customer Name"/>
<PropertyDef Name="CustomerNumber" DataType="text" DisplayName="Customer Number"/>
```

www.it-ebooks.info

7. Scroll down to the `ResultTypes` section. This is where the result types are defined. Each result type appears in the Result Type drop-down on the Advanced Search page.
8. Create a new result type for the document type.

```
<ResultType DisplayName="PDF Documents" Name="pdfdocuments">
     <KeywordQuery>FileExtension="pdf"</KeywordQuery>
     <PropertyRef Name="Author" />
     <PropertyRef Name="CustomerName"/>
     <PropertyRef Name="CustomerNumber"/>
     <PropertyRef Name="DocComments"/>
     <PropertyRef Name="Description" />
     <PropertyRef Name="DocKeywords"/>
     <PropertyRef Name="FileName" />
     <PropertyRef Name="Size" />
     <PropertyRef Name="DocSubject"/>
     <PropertyRef Name="Path" />
     <PropertyRef Name="Write" />
     <PropertyRef Name="CreatedBy" />
     <PropertyRef Name="ModifiedBy" />
     <PropertyRef Name="Title"/>
</ResultType>
```

9. Copy the contents from the text editor, and paste it back into the Text Editor dialog in SharePoint.
10. Click OK on the Text Editor dialog.
11. Click OK on the web part tool pane. Your new result type appears in the Result Type drop-down, and any custom properties appear in the Property picker drop-down, as shown in Figure 3-12.

FINDING CONTENT WITH ALERTS

When users subscribe to alerts anywhere on the portal, the search service processes those alerts. This interdependence means that if search is not working, alerts also are not working; and in addition, the search server performance can be impacted if your portal has a lot of alert subscriptions.

There is a unique additional scenario in which search and alerts are combined. Users can subscribe to search-based alerts so that they receive notifications when the results of their search query have changed. This ability can be enabled or disabled at the SSA. Search-based alerts increase the load on the search server that processes the search query each time it processes the search-based alert. To enable or disable this feature, follow these steps:

1. Navigate to the administration page for your Search Service Application, and from the Administration Quick Launch section, select Search Administration.
2. Locate the Search Alerts status on the page.
3. Click Enable if you want to activate the feature. If the feature is already activated, it displays this status next to the Enable/Disable link.
4. Click Disable if you want to deactivate the feature. If the feature is already deactivated, it displays this status next to the Enable/Disable link.

www.it-ebooks.info

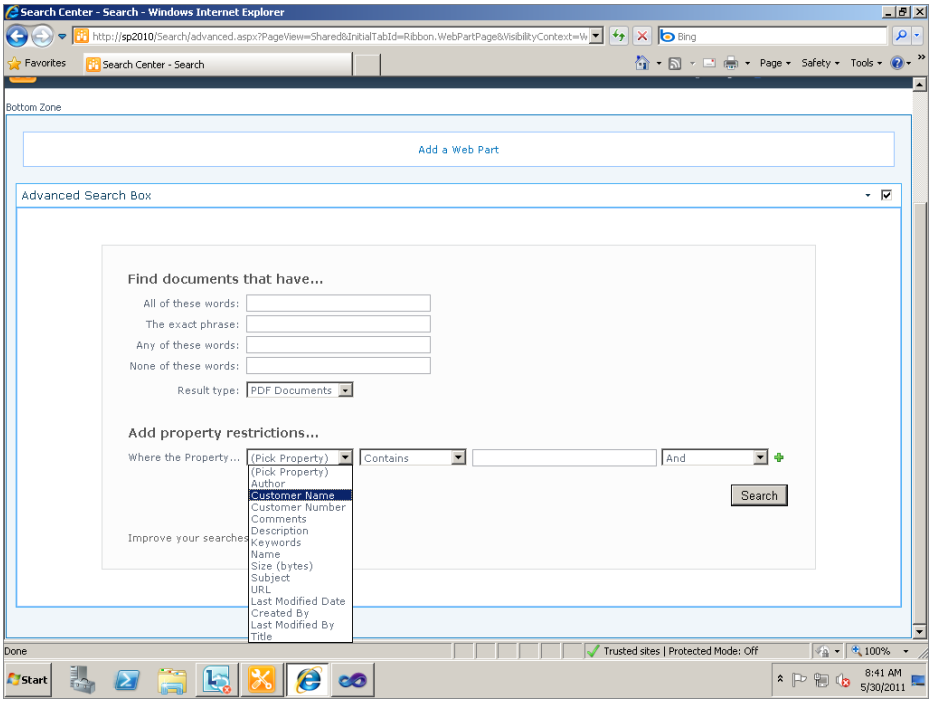


FIGURE 3-12: New Results Type in the Advanced Search

To review and manage what alerts are active on your sites, follow these steps:

1. From the Site Actions menu on the site where you want to review alerts, choose Site Settings.
2. From the Site Administration menu, select User alerts.
3. Select a user from the Display Alerts For drop-down menu, and click Update.
4. If appropriate, select the alerts that you want to delete, and click Delete Selected Alerts.

SUMMARY

Search is a key component in portal deployment. Users search to quickly and effectively find content without using the navigation. In this chapter, the content sources were discussed and configured, managed properties were added, and the Search Center was customized. Modifying the Advanced Search page and web part can improve the user's search experience and incorporate result types and properties.

Alerts and search are closely related because not only does search manage the processing of alert subscription, but also users can subscribe to search results. This chapter reviewed the impact of having these two services tied together.

The next chapter explores the use of the SharePoint list web service and how it may be leveraged within SharePoint solutions.

www.it-ebooks.info

www.it-ebooks.info

4

Leveraging the SharePoint Lists Web Service

This chapter explores the uses of the built-in SharePoint web services, namely the Lists web service. When creating applications that run outside of the SharePoint environment but need access to SharePoint lists, the use of the Lists web service is a must. In this chapter, the web service is explained along with several common operations that you usually need to perform.

WEB SERVICES OVERVIEW

Web services can be considered data sources accessible via http. Using a URL, you may easily retrieve data by performing web method queries. SharePoint has built-in system web services that it uses internally but may also be used for custom programming purposes.

SharePoint Web Services Introduction

The SharePoint web services are implemented as ASP.NET Web Services (`.asmx`). Various web services provide SharePoint functionality and administrative tasks, which are all physically located within the ISAPI folder of the "14" hive. You can access them via IIS by using a virtual directory named `_vti_bin` associated to each SharePoint web application, as shown in Figure 4-1. You may view the available web methods for each web service by using the virtual directory and web service within the browser, as shown in Figure 4-2.

Instantiating a Web Service

You may reference the SharePoint web services within your Visual Studio 2010 project and accordingly instantiate them. This is usually implemented when the application is not running within SharePoint and therefore does not have access to the SharePoint object model via the SharePoint assemblies. For example purposes, a console application is used in this chapter; however, you may follow the same steps for other project/application types.

www.it-ebooks.info

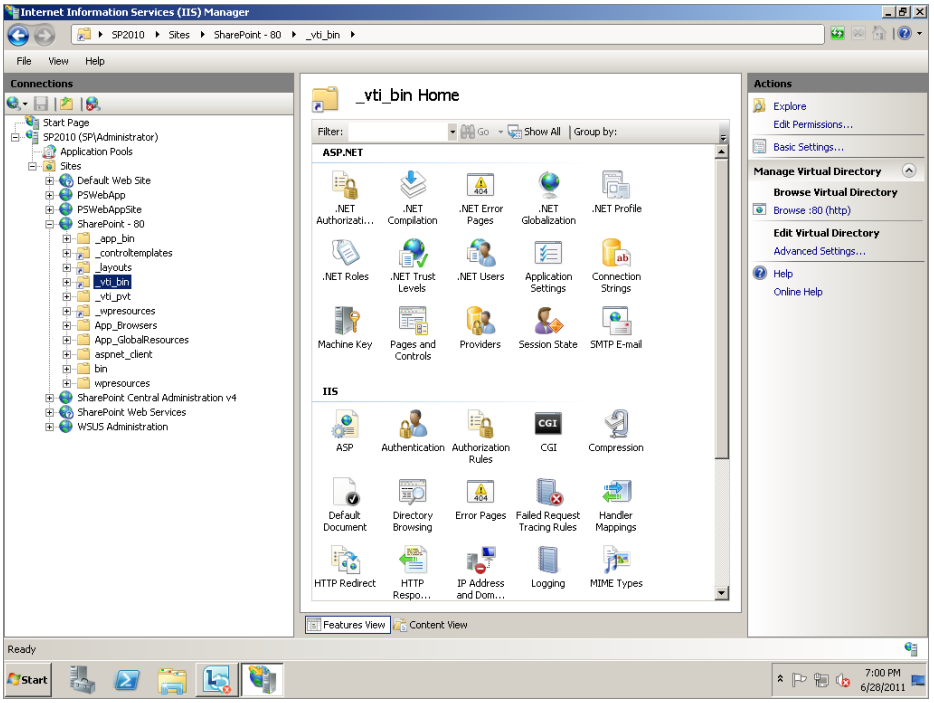


FIGURE 4-1: The _vti_bin Virtual Directory in IIS

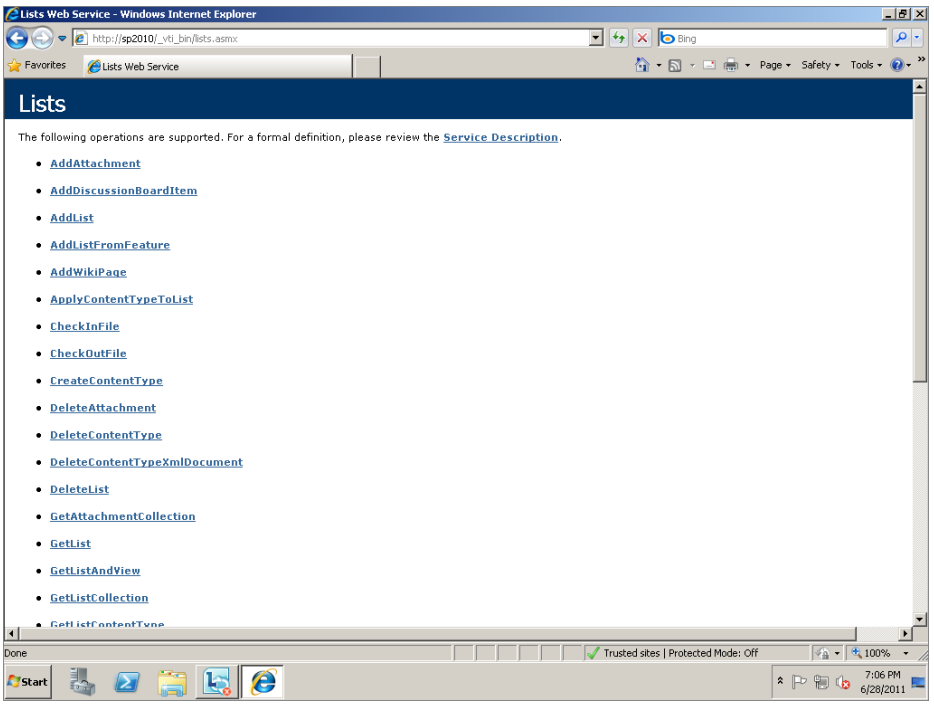


FIGURE 4-2: Viewing the web methods within the browser

www.it-ebooks.info

The first task is creating the reference to the SharePoint web service within your Visual Studio project. To do this, follow these steps:

1. Right-click the project in Visual Studio, and select Add Service Reference, as shown in Figure 4-3. The Add Service Reference dialog appears.

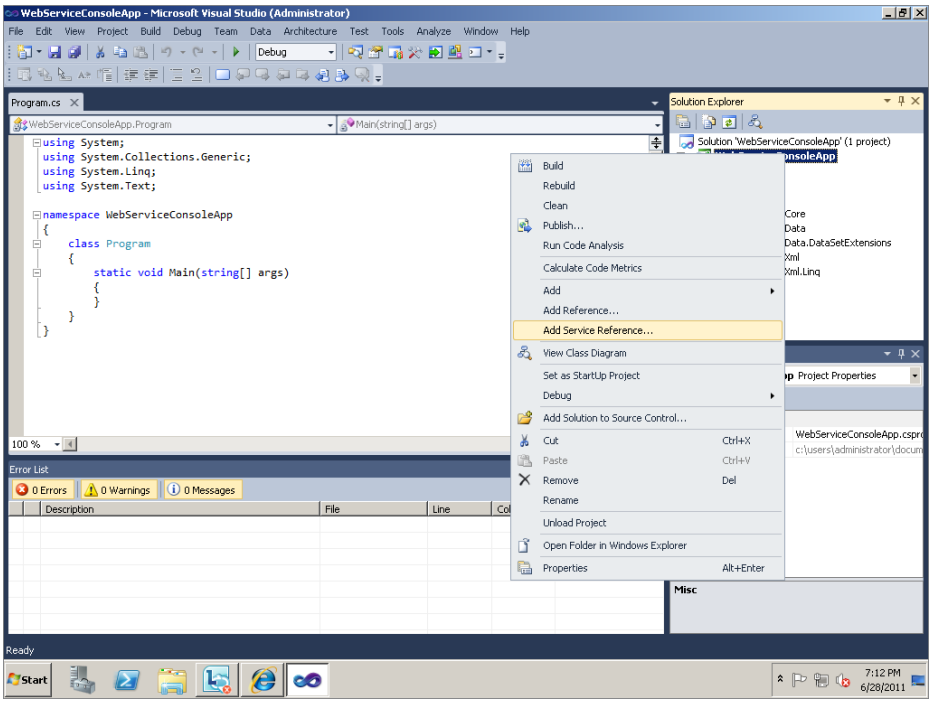


FIGURE 4-3: Add Service Reference menu option

2. Click the Advanced button located in the lower-left corner of the Add Service Reference dialog. The Service Reference dialog appears.
3. Click the Add Web Reference button located in the lower-left corner of the Service Reference dialog. The Add Web Reference dialog appears.
4. Enter the address of the web service, and click the Go button, as shown in Figure 4-4. The web methods appear in the dialog window.
5. Enter a name for the web reference, and click the Add Reference button. The web reference is added to your project and appears within the Solution Explorer window.

Now that the web service is referenced within your project, you need to add code to instantiate the web service. You accomplish this by creating a new object from the web service object-type. In the example, the Lists web service is referenced and therefore the object is lists. The object instantiation syntax is as follows:

```
ListsWS.Lists listsWS = new ListsWS.Lists();
```

www.it-ebooks.info

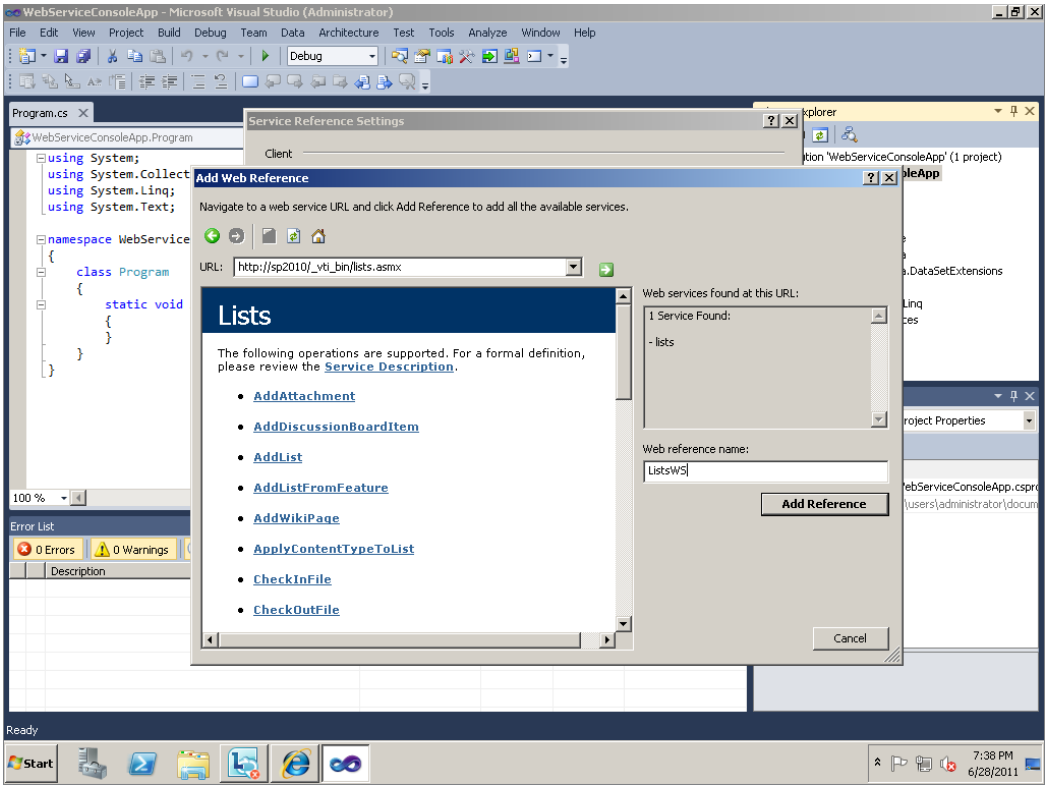


FIGURE 4-4: Add Web Reference

You may now use the variable object to call the web methods of the Lists web service, as shown by the intelli-sense in Figure 4-5. However, it is best to set the appropriate credentials so that your application has access to invoke the methods displayed. This is explained in the next subsection.

Setting Credentials

The web service needs credentials to authenticate against the SharePoint permissions so that the actions performed by the web methods are correctly done. Otherwise, anyone could invoke the web methods and gain unauthorized access. You can provide credentials in two ways: by using the default credentials of the current user or providing specific credentials.

Default Credentials

You may set the credentials of the web service to the current user or account running the application by leveraging the default network credentials from the credential cache. The syntax using the example variable is as follows:

```
listsWS.Credentials = System.Net.CredentialCache.DefaultNetworkCredentials;
```

www.it-ebooks.info

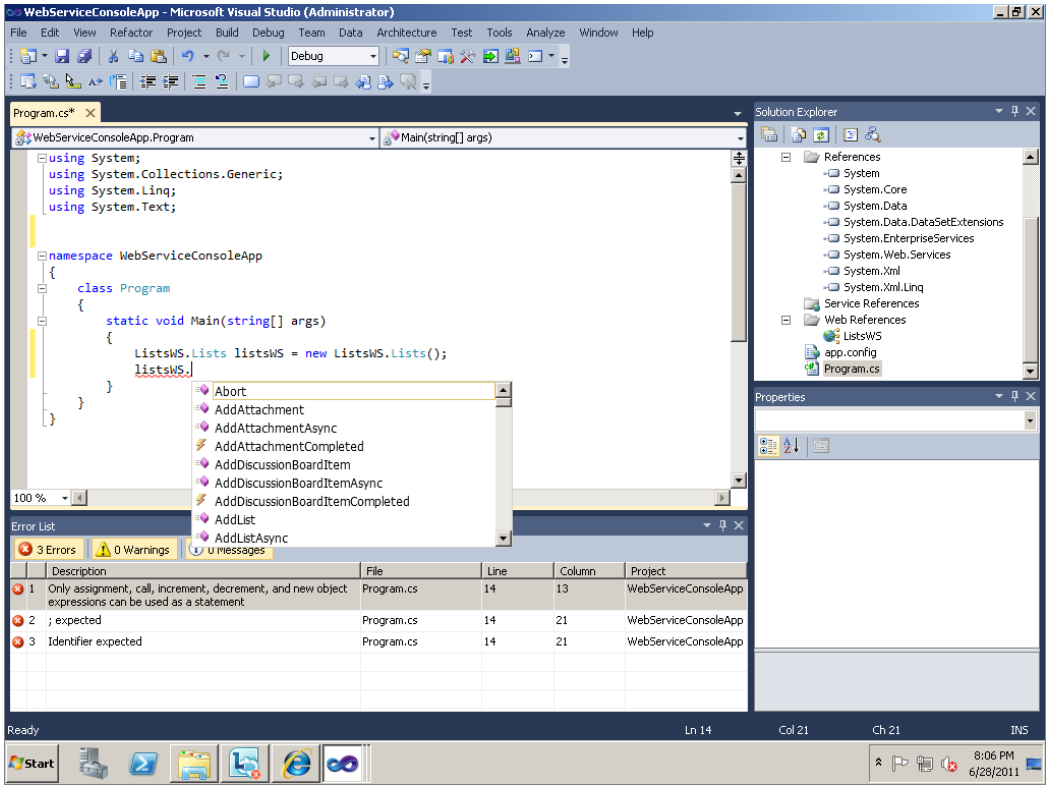


FIGURE 4-5: Available methods from the Lists object

Specific Credentials

Sometimes you need to make sure that certain specific credentials are used when calling the web services and web methods. Usually this occurs in an unattended operation where the service or account running the application is not the same account that has the proper SharePoint permissions.

To provide the web service specific credentials, you need to create a new credentials object and pass it to the web service. You can facilitate this through the use of a static method. The method can be as simple as shown:

```
static ICredentials GetCredential()
{
     return new NetworkCredential("username", "password", "domain");
}
```

To use the `ICredentials` interface, you need to reference System.NET by adding the appropriate `using System.Net` statement at the top of your class. The `NetworkCredential` object creation requires the username, password, and domain. Obviously you do not want to hard-code these values. You may retrieve them from application settings or from command line arguments (if using a console application).

www.it-ebooks.info

After creating the `GetCredential()` method, you may easily set the web service credentials to the results of this method. Figure 4-6 shows the full class code.

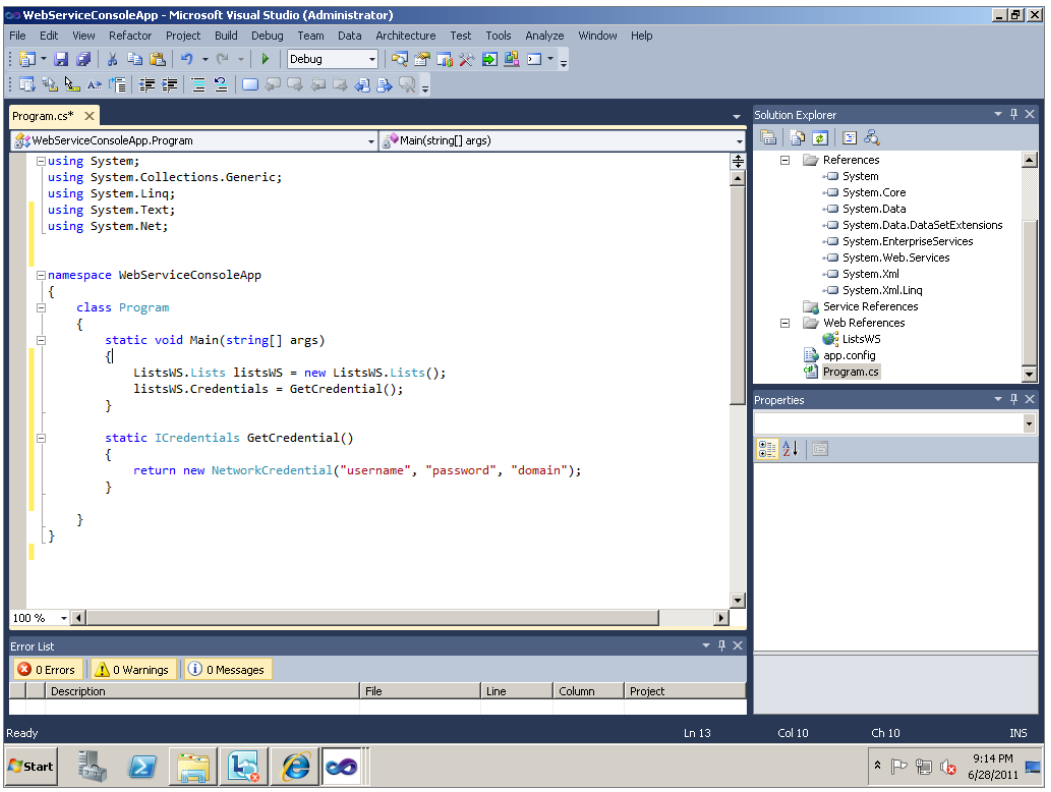


FIGURE 4-6: Using the GetCredential Method

QUERYING LISTS AND LIST ITEMS

The main retrieval processes are querying the web service for available lists and retrieving list items from a particular list. Although this process is similar to the object model methods, XML-based objects and coding is required because the web services use serialized XML to communicate and transfer data.

Modifying the SharePoint Location

When you set up the web service reference, you used a SharePoint location and the web service URL reference (`_vti_bin/Lists.asmx`). If the SharePoint location was the root, the web service works on items at that location. You may easily programmatically change the location by modifying the URL setting of the Lists web service object:

```
listsWS.Url="http://sproot/sites/sitecollection/subsite/_vti_bin/lists.asmx"
```

www.it-ebooks.info

So in other words, whatever site path you are attaching, the `_vti_bin/Lists.asmx` becomes the scope on which the web methods are acting. Whatever lists exist in the site path are the ones that may be accessed and utilized.

Retrieving Lists from a SharePoint Location

The first step in using the Lists web service to retrieve data because the web service uses XML (as explained in the introduction to this section), is to add the `using System.XML` at the top of your class.

To retrieve all the lists from the SharePoint location specified in the web service settings, you need to call the `GetListCollection()` web method of the web service. You may retrieve the results as an `XmlNode`. Therefore calling of this method can simply look like the following:

```
System.Xml.XmlNode nodeLists = listsWS.GetListCollection();
```

To understand what is actually returned, you should examine the `OuterXml` of the object you use to retrieve the XML. This can be done by using a message box within your code (while testing) or in a console application, writing the value out to the screen:

```
Console.WriteLine(nodeLists.OuterXml);
```

In the case of the list collection, the XML returned is a grouping of `<List>` entries within a parent `<Lists>` tag. A simplified view of the XML is as follows:

```
<Lists xmlns=\"http://schemas.microsoft.com/sharepoint/soap/\">
   <List ID=\"{53540725-97A7-4F1E-B096-1828E68C0D34}\" Title=\"Announcements\" />
   <List ID=\"{81A85A67-2BCC-4A8B-A077-8A1FA92B2417}\" Title=\"Calendar\" />
   <List ID=\"{696E0238-D0ED-420B-A1FA-E583B909E0CD}\" Title=\"Custom List\" />
</Lists>
```

When you understand what is returned, you may easily code around the `Xml` object to perform the actions needed. In the case of a list collection, you are interested in the List entries. Viewing the returned XML, the List entries are child nodes of the result; therefore, you need to loop through the child nodes to retrieve all the lists:

```
foreach (XmlNode xn in nodeLists.ChildNodes)
```

Each list has a handful of attributes. For example purposes, use the `Title` attribute that essentially corresponds to the name of the SharePoint list. As you loop through the child nodes, you can output the list titles. Figure 4-7 shows the full code class, and Figure 4-8 shows the resultant output window.



The output of lists includes system lists that aren't shown to the user. Use the Hidden attribute to filter these out.

Retrieving List Items from a SharePoint List

Retrieving actual list items of a SharePoint list is slightly more complicated than retrieving the list collection. Getting list items involves additional XML objects for querying the list. These objects

www.it-ebooks.info

include `query`, `viewFields`, and `queryOptions` as defined within the `GetListItems()` web method parameter list. There are other string-based parameters as well.

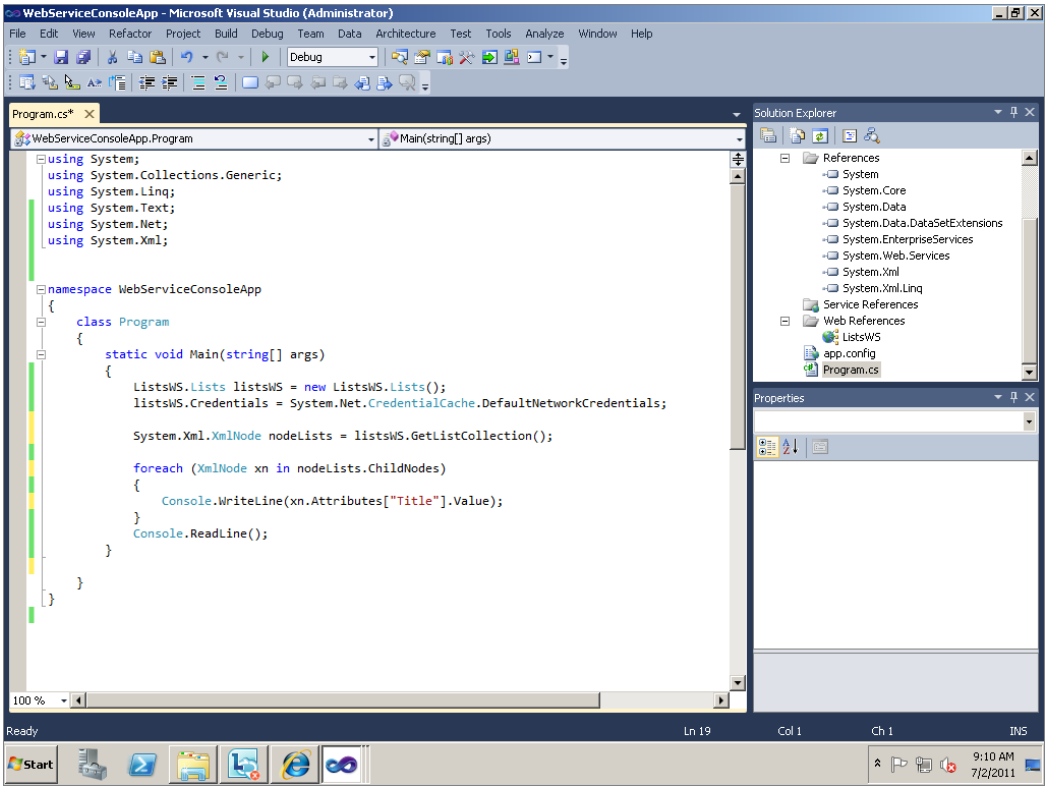


FIGURE 4-7: Using the GetListCollection() Method

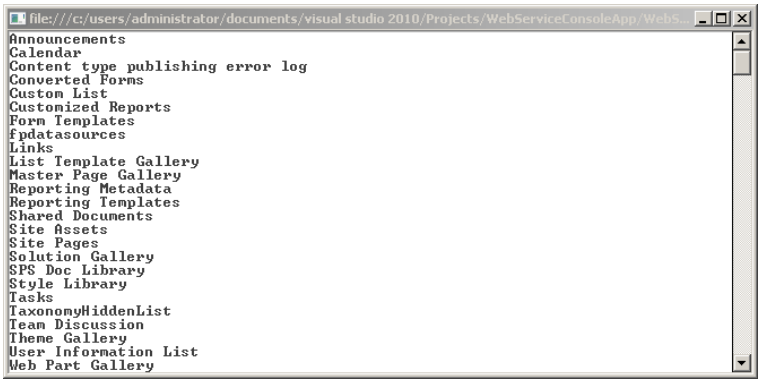


FIGURE 4-8: Result output of the GetListCollection() Code

www.it-ebooks.info

To generate these XML objects for retrieving list items, first create a generic `XmlDocument` object:

```
XmlDocument xmlDoc = new XmlDocument();
```

Next, use the `XmlDocument` object to create the elements needed for the `GetListItems()` method call:

```
System.Xml.XmlElement query = xmlDoc.CreateElement("Query");
System.Xml.XmlElement viewFields = xmlDoc.CreateElement("ViewFields");
System.Xml.XmlElement queryOptions = xmlDoc.CreateElement("QueryOptions");
```

To use these elements, set the `InnerXml` of each element accordingly. The `Query` object is used to filter the list item results using CAML query notation. You can use the `ViewFields` object to limit which list item fields (or columns) are returned from the web service. The `QueryOptions` object enables you to set various options such as looking within a specific folder or including mandatory columns.

```
query.InnerXml = "<Where><Eq><FieldRef Name='Show'/>
                  <Value Type='Text'>Yes</Value></Eq></Where>";
viewFields.InnerXml = "<FieldRef Name=\"ID\"/><FieldRef Name=\"Title\"/>";
queryOptions.InnerXml = "<IncludeMandatoryColumns>TRUE</IncludeMandatoryColumns>";
```

The `GetListItems()` method takes the following parameters:

- `listName`: A string containing the name of the list you want to query.
- `viewName` (Optional): A string containing the name of the list view to use. Passing an empty string uses the default view for the specified list.
- `query`: The `XmlNode` object created using the `XmlDocument` Query element.
- `viewFields`: The `XmlNode` object created using the `XmlDocument` `ViewFields` element.
- `rowLimit`: A string representing the maximum amount of rows to return. If an empty string is passed, the amount of rows returned may be only the page size (default of 100) of the default or specified view.
- `queryOptions`: The `XmlNode` object created using the `XmlDocument` QueryOptions element.
- `webID` (Optional): The unique identifier of the SharePoint Web containing the SharePoint List being queried.

An example call, using the objects created previously, may therefore look like the following:

```
System.Xml.XmlNode nodeListItems = listsWS.GetListItems("Custom List",
String.Empty, query, viewFields, String.Empty, queryOptions, String.Empty);
```

The XML returned from the `GetListItems()` method is much different than the XML returned from the `GetListCollection()` method. Therefore, once again, you need to review the `OuterXml` of the results to understand what is returned. A simplified example of the `GetListItems()` output XML is as follows:

```
<listitems xmlns:s=\"uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882\" >
  <rs:data ItemCount=\"3\">
```

www.it-ebooks.info

```
        <z:row ows_ID=\"1\" ows_Title=\"Item #1\" />
        <z:row ows_ID=\"2\" ows_Title=\"Item #2\" />
        <z:row ows_ID=\"3\" ows_Title=\"Item #3\" />
    </rs:data>
</listitems>"
```

The actual list items are two levels deep instead of one like the list collection. Therefore, to loop through the items, you need to get the grandchildren of the root node:

```
foreach (XmlNode xn in nodeListItems.ChildNodes[1].ChildNodes)
{
   if (xn.Attributes != null)
      Console.WriteLine(xn.Attributes["ows_Title"].Value);
}
```

If you look at the `OuterXml` carefully, you can notice occurrences of `"\n"` sprinkled throughout. This produced XML white space and that is the reason for using the `ChildNodes[1]` within the `foreach` loop as well as ensuring the `xn.Attributes` is not null.

Figure 4-9 shows the full code block in Visual Studio, and Figure 4-10 shows the resultant output window.

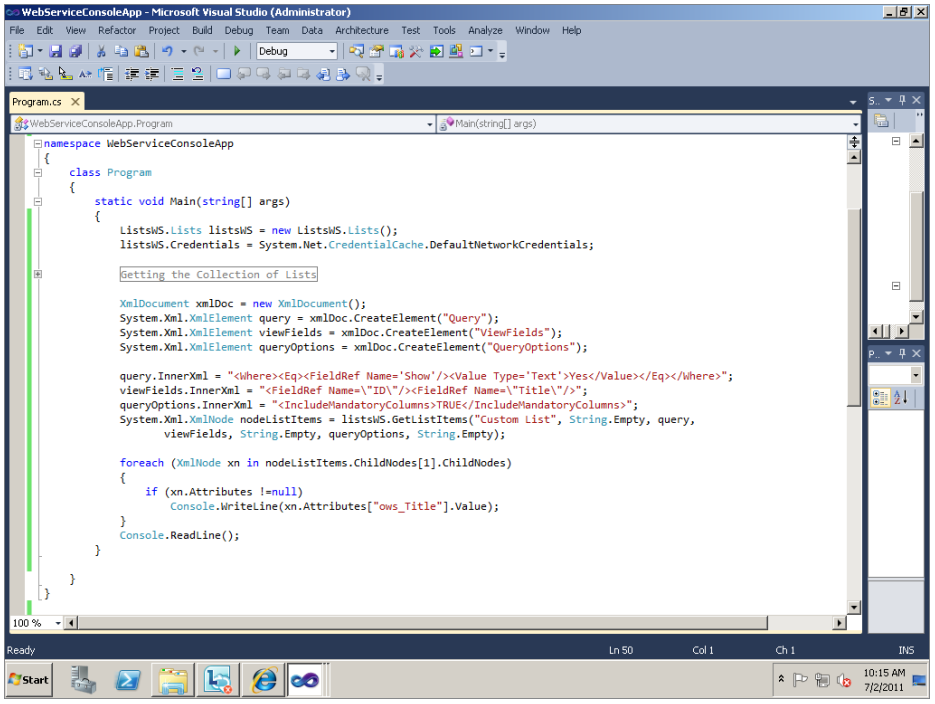


FIGURE 4-9: Using the GetListItems() method

www.it-ebooks.info

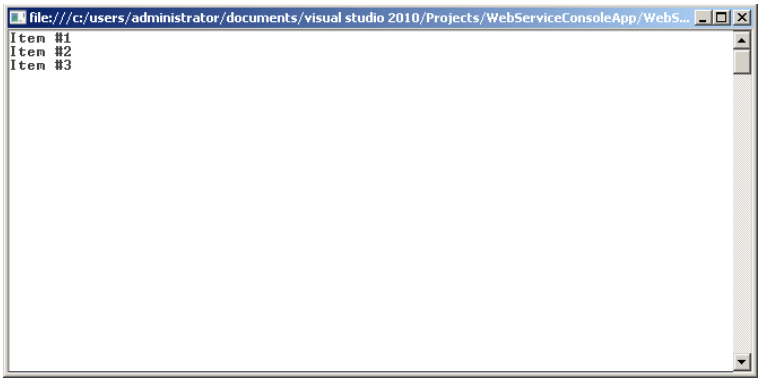


FIGURE 4-10: Result output of the GetListItems() code

MODIFYING LIST ITEMS

It is great to retrieve the lists and their list items from SharePoint; however, usually you want to act on those list items. This section continues the uses of the Lists web service and covers the processes for creating, updating, and deleting list items.

The overall processes of creating, updating, or deleting a list item deals with a "batch" of commands written as XML and passed to the web service (namely the `UpdateListItems()` web method). Therefore, the structure is a batch element with various methods. Each method contains the details of the operation performed. The generic batch element structure is shown as follows:

```
<Batch OnError="Continue">
    <Method ID="1" Cmd="{operation}">
    <Field Name="{column name}">{column value}</Field>
        ...
     </Method>
     <Method ID="2">
     ...
</Batch>
```

Each method is numbered sequentially starting at `"1"` and represents an operation. You may have multiple methods within a batch element to perform multiple operations. However, for the examples in this section, one method is used to demonstrate the particular commands.

Creating a New Item

The first step in creating a new list item is to generate a new `XmlDocument` object and create the batch element:

```
System.Xml.XmlDocument docNewItem = new System.Xml.XmlDocument();
System.Xml.XmlElement batchElement = docNewItem.CreateElement("Batch");
```

www.it-ebooks.info

Next, you need to set the `OnError` attribute to `"Continue"`. This enables all operations to be attempted whether the previous action failed. Although in this example, only one operation is performed, it is good practice to add this attribute whether you want to continue . If there is an error, and you want the processing to halt, set the `OnError` attribute to `"Return"`.

```
batchElement.SetAttribute("OnError", "Continue");
```

Create the methods by setting the `InnerXml` of the batch element. In this case, there is only one method. Use a simple custom list that requires only the Title column, and therefore to create a new list item, you only need to set a value to that field. Use a string variable for the Title value because, in most cases, this operation would be dynamic and the values would not be hardcoded.

```
string title = "New List Item from Web Service";
batchElement.InnerXml = "<Method ID='1' Cmd='New'><Field Name='Title'>" + title +
                        "</Field></Method>";
```

The batch element produced from the code looks like the following:

```
<Batch OnError=\"Continue\">
   <Method ID=\"1\" Cmd=\"New\">
      <Field Name=\"Title\">New List Item from Web Service</Field>
   </Method>
</Batch>
```

This was retrieved by viewing the `OuterXml` of the batch element object within Visual Studio while debugging the console application. Therefore, there are escape characters to appropriately produce the quotes. Use this batch element object along with the list name to insert a new item using the `UpdateListItems()` web method of the Lists web service:




```
System.Xml.XmlNode ndReturn = listsWS.UpdateListItems("Custom List", batchElement);
```

WebServiceConsoleApp\Program.CS

Figure 4-11 shows the full code listing in Visual Studio. Figure 4-12 shows the new list item within the SharePoint list.

Updating an Existing Item

Updating an existing list item is similar to creating a new one except you need to identify the list item you want to update by the ID of the item. For example purposes, the ID is already known and entered; however, usually the ID needs to be obtained from a particular list item that is not known during runtime. Going back to the retrieval of list items, you can obtain the ID of the item by using code similar to the following:




```
itemID = nodeListItems.ChildNodes[1].ChildNodes[1].Attributes["ows_ID"].s
Value.ToString();
```

WebServiceConsoleApp\Program.CS

www.it-ebooks.info

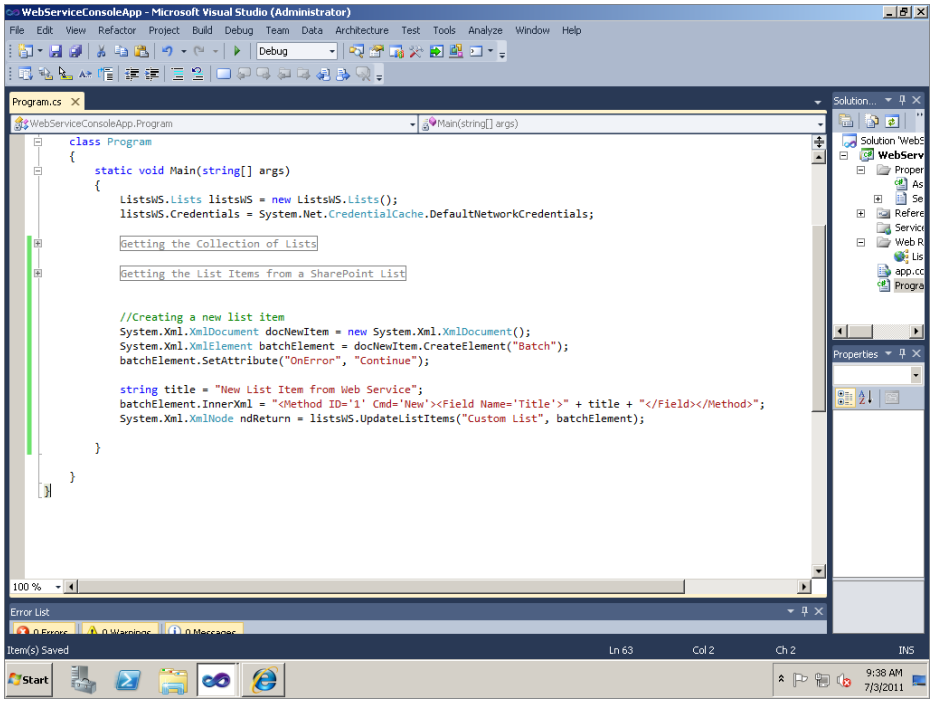


FIGURE 4-11: Creating a New List item code block

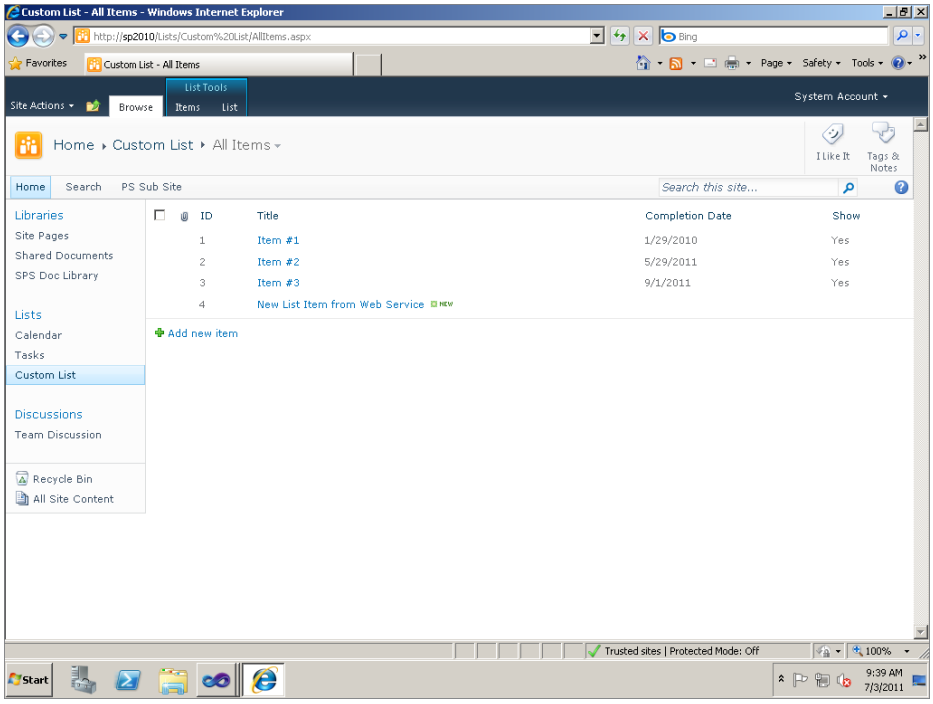


FIGURE 4-12: New List item created in SharePoint

www.it-ebooks.info

Create a new batch element using similar code as in the previous subsection:




```
System.Xml.XmlDocument docUpdateItem = new System.Xml.XmlDocument();
System.Xml.XmlElement batchElementUpdate = docUpdateItem.CreateElement("Batch");
batchElementUpdate.SetAttribute("OnError", "Continue");
```

WebServiceConsoleApp\Program.CS

Create the method using `"Update"` as the command and include the ID column in the field definitions:




```
string updatedTitle = "Updated List Item ID=1";
string itemID = "1";
batchElementUpdate.InnerXml = "<Method ID='1' Cmd='Update'><Field Name='ID'>" +
 itemID + "</Field><Field Name='Title'>" + updatedTitle + "</Field></Method>";
```

WebServiceConsoleApp\Program.CS

Again, variables are used for the ID and the Title to simulate the dynamic use of these values. The batch element for the update now looks like the following:

```
<Batch OnError=\"Continue\">
   <Method ID=\"1\" Cmd=\"Update\">
      <Field Name=\"ID\">1</Field><Field Name=\"Title\">Updated List Item ID=1
      </Field>
   </Method>
</Batch>
```

Use this batch element object along with the list name to update an existing list item using the `UpdateListItems()` web method of the Lists web service:

```
System.Xml.XmlNode ndReturnUpdate = listsWS.UpdateListItems("Custom List",
batchElementUpdate);
```

Figure 4-13 shows the full code listing in Visual Studio. Figure 4-14 shows the updated list item within the SharePoint list.

Deleting an Existing Item

Deleting an existing item is almost identical to updating one except you need to provide only the ID. Create a new batch element using similar code as in the previous subsection:




```
System.Xml.XmlDocument docDeleteItem = new System.Xml.XmlDocument();
System.Xml.XmlElement batchElementDelete = docDeleteItem.CreateElement("Batch");
batchElementDelete.SetAttribute("OnError", "Continue");
```

WebServiceConsoleApp\Program.CS

www.it-ebooks.info

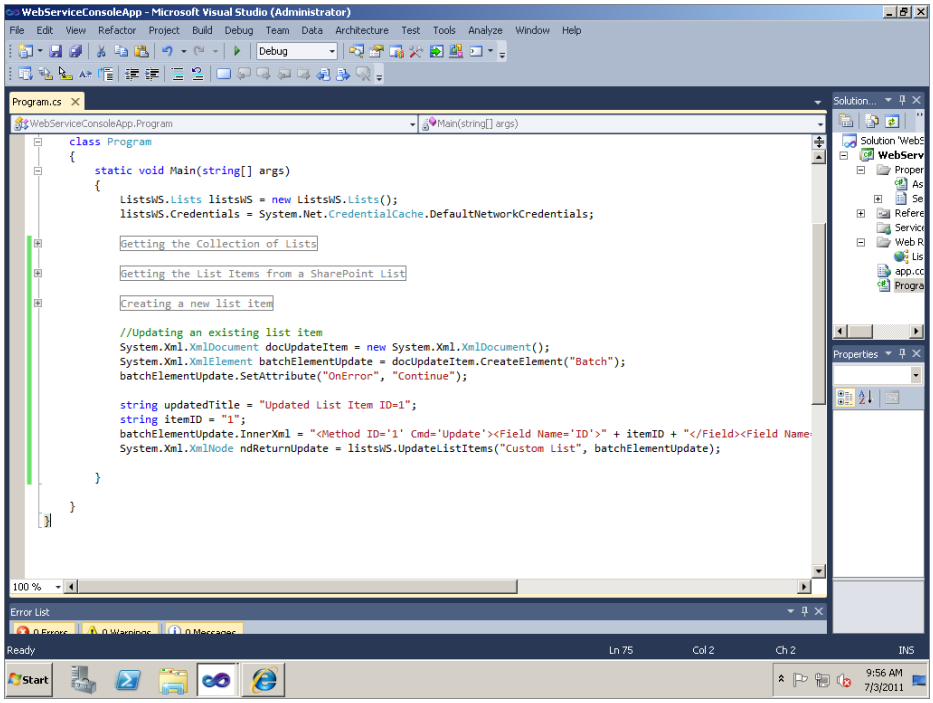


FIGURE 4-13: Updating a List item code block

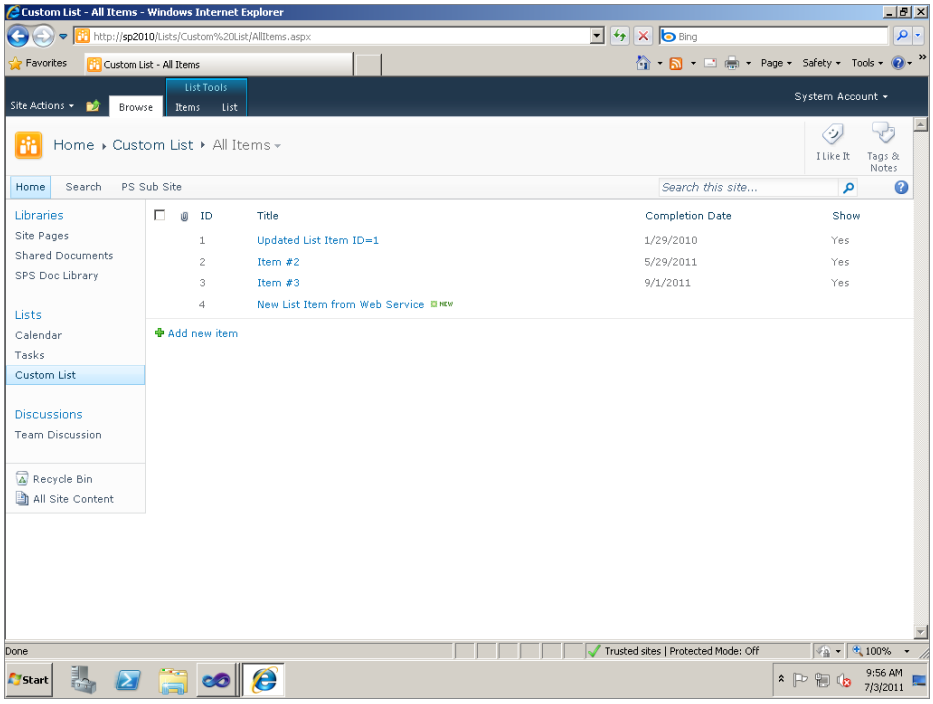


FIGURE 4-14: List item updated in SharePoint

www.it-ebooks.info

Create the method using `"Delete"` as the command and include the ID column in the field definitions:




```
string itemID = "3";
batchElementUpdate.InnerXml = "<Method ID='1' Cmd='Delete'><Field Name='ID'>" +
itemID + "</Field></Method>";
```

WebServiceConsoleApp\Program.CS

The batch element for the delete now looks like the following:

```
<Batch OnError=\"Continue\">
   <Method ID=\"1\" Cmd=\"Delete\">
      <Field Name=\"ID\">3</Field>
   </Method>
</Batch>
```

Use this batch element object along with the list name to update an existing list item using the `UpdateListItems()` web method of the Lists web service:

```
System.Xml.XmlNode ndReturnDelete = listsWS.UpdateListItems("Custom List",
batchElementDelete);
```

Figure 4-15 shows the full code listing in Visual Studio. Figure 4-16 shows the SharePoint list with the deleted item (ID=3).

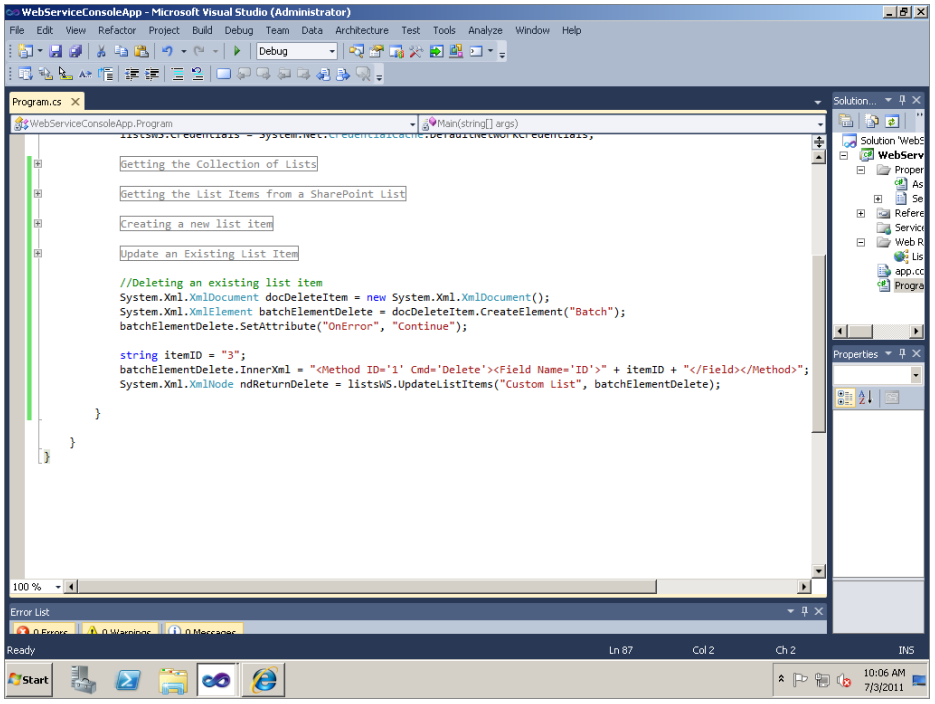


FIGURE 4-15: Deleting a List item code block

www.it-ebooks.info

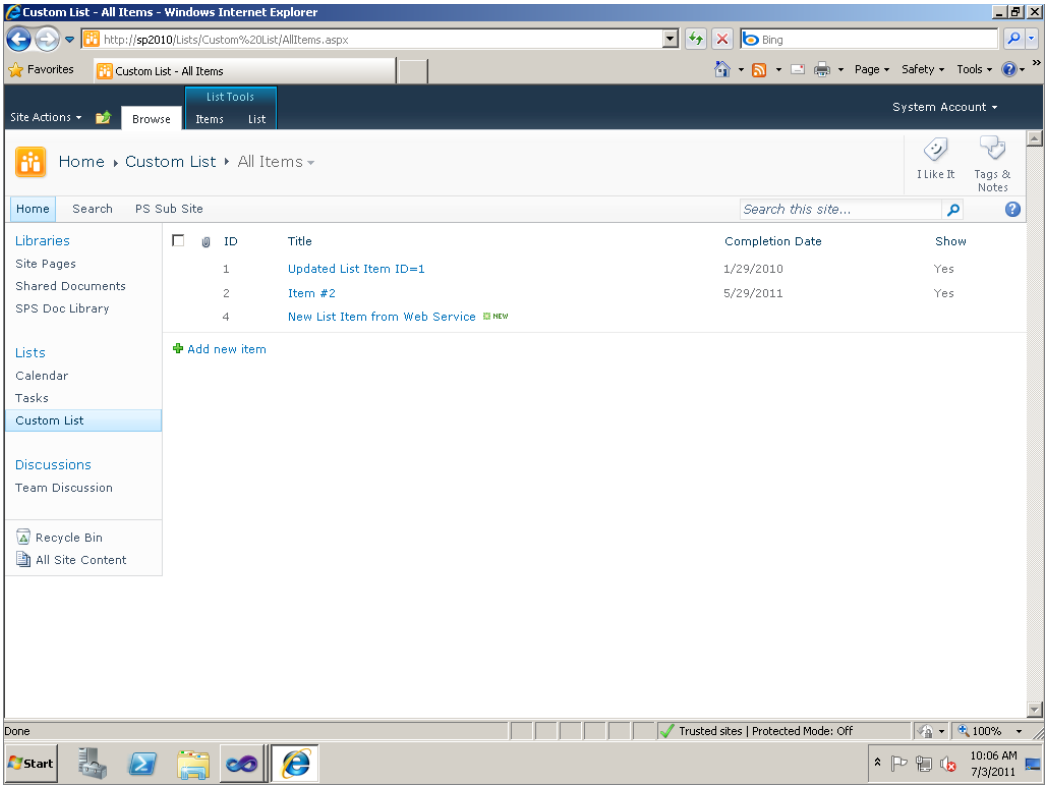


FIGURE 4-16: List item deleted from SharePoint List

PROGRAMMATICALLY UPLOADING FILES TO SHAREPOINT

Programmatically uploading documents to SharePoint often becomes a requirement when organizations decide to implement SharePoint within their enterprise. Author Steve Mann had two occasions in which thousands of documents needed to be imported. In both cases, he used a console application so that the process could be easily invoked from the command line manually or via a scheduled job.

Nonetheless, whether you use a console application or other Visual Studio project type, for these types of supporting methods, you can create a separate Utilities class to house them. Do the same thing to keep your projects, classes, and code maintainable and clean.

Adding References

The upload process uses the `FileStream`, `WebRequest`, and `WebResponse` objects. Therefore, you need to add the following `using` statements to your class:

```
using System.Net;
using System.IO;
```

www.it-ebooks.info

`System.Net` contains the `WebRequest`, which is used for the actual uploading of the file to SharePoint. The System.IO reference enables you to use the `Stream` and `FileStream` objects to read and stream the file from disk.

Create Upload Method

Create a public static method that returns a bool (which represents success or failure). Name the method something like `UploadFileToDocLibrary`. You can call this method from other logic to upload a file. Therefore, essentially you need only two pieces of information: the file you need to upload, and the document library to upload that file. The location of these is included within the full path, so those are the needed values. Your method signature should look like the following:

```
public static bool UploadFileToDocLibrary(string filePath,
     string documentLibraryPath)
```

Create Web Request and File Buffer

Create a `WebRequest` object using the document library path passed into the method. Similar to the SharePoint web services, you need to set the credentials on the `WebRequest` as well. The operation performed is a `"PUT"`. Therefore, the `WebRequest` Method should be set to that value accordingly.




```
WebRequest request = WebRequest.Create(documentLibraryPath);
request.Credentials = CredentialCache.DefaultCredentials;
request.Method = "PUT";
```

WebServiceConsoleApp\Utilities.CS

An array of bytes needs to be initialized to stream the contents of the file being uploaded. Although typically developers use `byte[1024]`, author Mann usually increases this number to retrieve more information from the file at a time and speed up the process a bit.

```
byte[] fileBuffer = new byte[2048];
```

Use Stream and FileStream Objects

Use a `FileStream` object to read the contents of the file and a `Stream` object to write the contents of the file. Loop through the file stream to read chunks of the file (storing them in the `fileBuffer` array) and then write them to the request stream.




```
using (Stream stream = request.GetRequestStream())
{
   using (FileStream fileStream = File.Open(filePath, FileMode.Open,
       FileAccess.Read))
    {
       int startBuffer = fileStream.Read(fileBuffer, 0, fileBuffer.Length);
          for (int i = startBuffer; i > 0; i = fileStream.Read(fileBuffer, 0,
fileBuffer.Length))
           {
```

www.it-ebooks.info

```
                stream.Write(fileBuffer, 0, i);
            }
        }
    }
```

WebServiceConsoleApp\Utilities.CS

Perform the Upload

To actually perform the upload of the file contents, you need to instantiate the request's `GetResponse()` method. Although this returns a `WebResponse` object, you want to perform only the action and do not need to do anything with the response (unless there is an issue and you need to debug). Close the response object to release the web resources.




```
WebResponse response = request.GetResponse();
response.Close();
```

WebServiceConsoleApp\Utilities.CS

Error Handling and Returns

Wrap the entire contents of the method into a `try/catch` block. Return a `true` value at the end of the method but return a `false` value within the `catch` block. The entire code listing is shown:




```
public static bool UploadFileToDocLibrary(string filePath,
      string documentLibraryPath)
{
   try
   {
      WebRequest request = WebRequest.Create(documentLibraryPath);
      request.Credentials = CredentialCache.DefaultCredentials;
      request.Method = "PUT";

      byte[] fileBuffer = new byte[2048];

      using (Stream stream = request.GetRequestStream())
      {
         using (FileStream fileStream = File.Open(filePath, FileMode.Open,
FileAccess.Read))
         {
            int startBuffer = fileStream.Read(fileBuffer, 0, fileBuffer.Length);
            for (int i = startBuffer; i > 0; i = fileStream.Read(fileBuffer, 0,
fileBuffer.Length))
            {
               stream.Write(fileBuffer, 0, i);
            }
         }
      }
      WebResponse response = request.GetResponse();
```

www.it-ebooks.info

```
            response.Close();
        }

        catch (Exception ex)
        {
            //TO DO: Document Exception via Event Logging
            return false;
        }

        return true;

    }
```

WebServiceConsoleApp\Utilities.CS

SUMMARY

This chapter demonstrated how you can leverage the SharePoint Lists web service to perform list operations when an application cannot be running within the SharePoint environment. Using XML objects and structures, you can retrieve, create, update, and delete list items. In addition, you can upload new documents or files to SharePoint using file objects along with the SharePoint web services.

The next chapter explores the creation of custom field types and content types that you can use to generate custom list templates. These make up the foundation of SharePoint lists that form a base for site content.

www.it-ebooks.info

5

Custom Field Types, Content Types, and List Definitions

One of the most used components of SharePoint is the SharePoint List. Lists enable the collection of tabular data and are similar to database tables. SharePoint also contains a specialized version of a list, called a Document Library, which is targeted toward storing a document and metadata about that document. For the purpose of this chapter, unless otherwise specified, "List" refers to both Lists and Document Libraries. SharePoint 2010 provides a number of predefined list "templates," such as the Task List, the Calendar, the Issues List, the Document Library, and the Asset Library.

Although these list templates are quite useful on their own, Lists truly become powerful when they are customized and extended to capture information specific to the lists' particular usage. For instance, if you have a document library used to store purchase order (PO) documents, it might be beneficial to store the PO Number and the Purchase Date as separate columns, or pieces of *metadata,* as they are more commonly referred to.

When adding a column to a list, SharePoint provides a number of *field types*, which determine how the column behaves. For instance, when adding the Purchase Date column to your PO list, you would likely specify the column as a Date and Time column. Doing so enforces certain logic and behavior for the column; for instance, a column specified as a Date and Time column can accept only properly formatted dates and can provide a calendar picker control to enable users to more easily select dates. Field types determine how the data is rendered when displayed (date columns will properly format for users according to their region) and how and what can be entered.

As users and organizations grow more advanced in their usage of SharePoint, they soon realize that they are collecting standard types of metadata throughout the portal. Many companies, for instance, work with external customers, clients, or vendors. It is therefore relatively common that the company wants to capture which customer, client, or vendor every document uploaded into the portal relates to. This information can then be used to refine search results,

www.it-ebooks.info

be used for legal holds, enforce retention policies, and build views. Rather than manually adding a Customer column to every list and library in the portal that contains such information, SharePoint enables the creation of Content Types, which are actually just templates with predefined columns already associated with them, and can also have a custom document template. Using content types, you can create a Customer Document content type, which has a column to specify the customer the document is associated with. Now, when creating a new document library, rather than having to manually add a Customer column and configuring it, your users can now just add the Customer Document content type to the library.

But that's still a manual step the user needs to be trained to perform. Wouldn't it be nice if you could just create a new document library type called Customer Documents that already had the Customer Document content type associated with the library? This is where list definitions come in. List definitions enable you to define lists and library templates that you can use to create preconfigured lists. List definitions can do more than just associate content types with a list, however; they can also be used to set other list settings, preconfigure views, and determine other behavior. For those of you who are already experienced with SharePoint, you may wonder how a list definition differs from a List Template. List Templates can be created using an existing list as a template through the SharePoint UI. List definitions are created through Visual Studio or some other development environment and provide access to capabilities not exposed through the SharePoint UI and which can be packaged up into SharePoint features and solutions for easy deployment across farms. Definitions are usually a small component of a larger solution, whereas templates are often a standalone item.

This chapter walks you through the creation of a custom field type, how to define custom content type that uses that field type, and how to create a list definition that uses that content type. All three of these topics (field types, content types, and list definitions) are extremely complex, and SharePoint supports a myriad of configuration options that are briefly discussed here. Microsoft's MSDN site provides an excellent breakdown of all the various elements and attributes that may be used for more complex scenarios.



The code for the solution created in this chapter is available for download at the book's companion website, `www.wrox.com`.

FIELD TYPES

As discussed, field types are essentially the data type of a SharePoint list column and enable developers to control how the column data are displayed and interacted with. Figure 5-1 shows the standard list of field types that are available from the SharePoint 2010 Column Creation screen.

The out-of-the-box (OOTB) field types provide an extensive list of options but may not be sufficient for certain business requirements. One typical weakness of the existing field types has to do with columns that store sensitive information. For instance, consider a list that stores Social Security numbers (SSN) in addition to other personal information. It is quite common to want to expose the nonsensitive information to certain users, but restrict who can see the Social Security numbers. The rest of this section first discusses the components that make up a field type and then walks you through creating an SSN field type.

www.it-ebooks.info

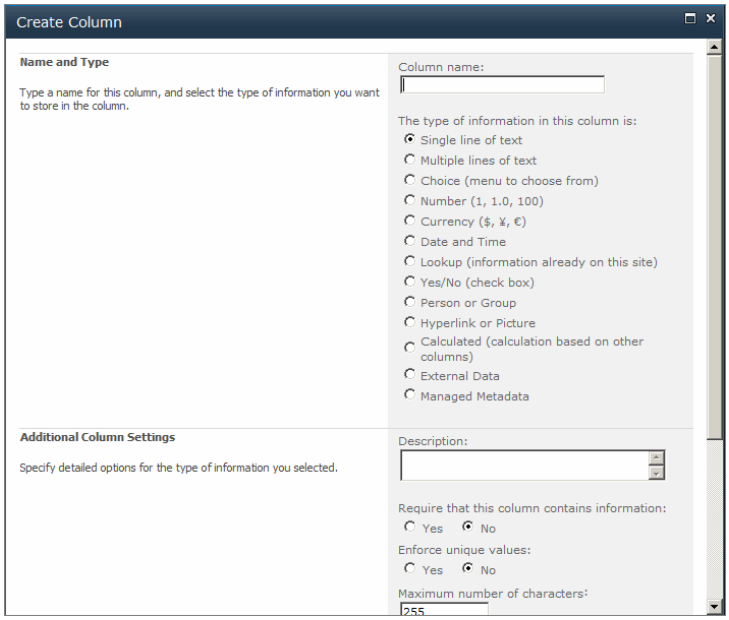


FIGURE 5-1: Standard field types

Field Type Definition Components

All field type definitions consist of four distinct files: the field type definition, the field class, the rendering control class, and the rendering templates. In addition, you can also specify a custom XSLT style sheet, a field value type, and an editing control.

The Field Type Definition

The field type definition is an XML file deployed to the 14 hive beneath the TEMPLATE\XML folder that follows the naming convention of fldtypes<YourFieldName>.xml. To avoid naming collisions, you must follow a standard naming convention for your field type definitions, which includes some type of identifying information beyond the field name, such as your company name. For the SSN field type, the field definition name is `fldtypes_Wrox_SSN.xml`.

The definition file follows the structure, as shown in Table 5-1.

TABLE 5-1: Field Type Definition Elements

ELEMENT NAME	DESCRIPTION
`<FieldTypes>`	Top-level container holding `FieldType` elements.
`<FieldType>`	Represents a single field type.
`<Field>`	Represents a single property of the field. A `FieldType` definition contains multiple field elements.

www.it-ebooks.info

A sample field type definition might look something like the following:

```
<?xml version="1.0" encoding="utf-8" ?>
<FieldTypes>
    <FieldType>
        <Field Name="TypeName">SocialSecurityNumber</Field>
        <Field Name="ParentType">Text</Field>
        <Field Name="TypeDisplayName">Social Security Number</Field>
        <Field Name="TypeShortDescription">
            Masked Social Security Number (XXX-XX-6789)
        </Field>
        <Field Name="AllowBaseTypeRendering">TRUE</Field>
        <Field Name="FieldTypeClass">
            Wrox.Lists.FieldTypes.SSNField, Wrox.Lists,
            Version=1.0.0.0,Culture=neutral,PublicKeyToken=086df048023273e6
        </Field>
    </FieldType>
</FieldTypes>
```

Table 5-2 shows the possible values available to your custom field type definition for the property names.

TABLE 5-2: Field Properties

PROPERTY NAME	DESCRIPTION
`AllowBaseTypeRendering`	Optional, boolean. Specifies if the base type rendering should be used by client applications if it cannot determine how to render the type. Default: *FALSE*.
`CAMLRendering`	Optional, boolean. Specifies that the depreciated CAML rendering be used for rendering the control in list views, rather than XSLT. Default: *FALSE*.
`FieldTypeClass`	Required, string. Specifies the strong name of the field type class library.
`FieldEditorUserControl`	Optional, string. Relative path to the field editor control.
`Filterable`	Required, boolean. Specifies whether the column can be filtered.
`ParentType`	Required, string. Specifies the parent SPField base class.
`ShowOnListAuthoringPages`	Optional, boolean. Determines if the field type shows up as a column type for Lists. Default: *TRUE*
`ShowOnDocumentLibraryAuthoringPages`	Optional, boolean. Determines if the field type shows up as a column type for Document Libraries. Default: *TRUE*

www.it-ebooks.info

PROPERTY NAME	DESCRIPTION
ShowOnSurveyAuthoringPages	Optional, boolean. Determines if the field type shows up as a column type for Surveys. Default: *TRUE*
ShowOnColumnTemplateAuthoringPages	Optional, boolean. Determines if the field type shows up as a column type for Content Types. Default: *TRUE*
Sortable	Required, boolean. Determines if the field type can be used for sorting.
TypeDisplayName	Required, string. The friendly display name for the field type.
TypeName	Required, string. The internal name used for the field type. Must be unique within the farm.
TypeShortDescription	Optional, string. The short description displayed for the field type.
UserCreatable	Optional, boolean. Determines if users can add columns of this type to a list through the UI. Default: *TRUE*.

The Field Class

The field class handles the common operations that occur on the column. A custom field class must inherit from `SPField` or from a class that derives from a child class of `SPField`. Though not required, it is often easiest to inherit the OOTB class, which is the closest in functionality to the field type you implement. Table 5-3 shows the OOTB field classes.

TABLE 5-3: Out-of-the-Box-Field Types

CLASS	DESCRIPTION
SPFieldBoolean	A boolean field type
SPFieldChoice	A choice field type
SPFieldCurrency	A currency field type
SPFieldDateTime	A date time field type
SPFieldLookup	A lookup field that pulls its value from another list in SharePoint
SPFieldMultiChoice	A field type that supports multiple selections
SPFieldMultiColumn	A field type that supports multiple columns of data

continues

www.it-ebooks.info

TABLE 5-3 *(continued)*

CLASS	DESCRIPTION
`SPFieldMultiLineText`	A field type that supports multiple lines of text
`SPFieldNumber`	A field type that supports only numeric data
`SPFieldRatingScale`	A field type that supports rating scales
`SPFieldText`	A field type that supports a single line of data (up to 255 characters)
`SPFieldUrl`	A field type that supports URLs
`SPFieldUser`	A field type that supports users

The field class determines which control is used to render the field value in display, add, and edit modes, how a control is rendered in a mobile view, validation, code to be executed after the column is added, updated, or deleted on a list, default values, and various other details about a field's value.

The Rendering Control Class and the Rendering Template

With the rendering template, the rendering control class is used to render your field in display, new, and edit modes but not in the list view. The rendering template is a user control that has at least one template defined for rendering the field but can have multiple rendering templates defined. It is typical to have at least a display (read-only) template and an edit template.

Custom XSLT Style Sheet

As mentioned, the rendering control and templates handle only rendering the field in the display, add, and edit list forms. Rendering the contents of the field when viewing a list in list view mode is handled by a separate mechanism. If no XSLT style sheet is specified, then the field's base type rendering is used.

Field Value Type

For complex field values that store more complicated data than the base field types such as text, numbers, dates, and so on, it may be helpful to create a custom field value type. This is simply a custom class that knows how to parse the serialized value of a field into a more accessible class.

Editing Control

You can define editing controls to handle adding or updating a column definition to your list. For example, when you add a choice column to a list, you are provided with the ability to enter the values that can be chosen from and how the choice field is rendered, as shown in Figure 5-2.

The contents rendered in this section of the Add Column dialog come from your editing control.

www.it-ebooks.info

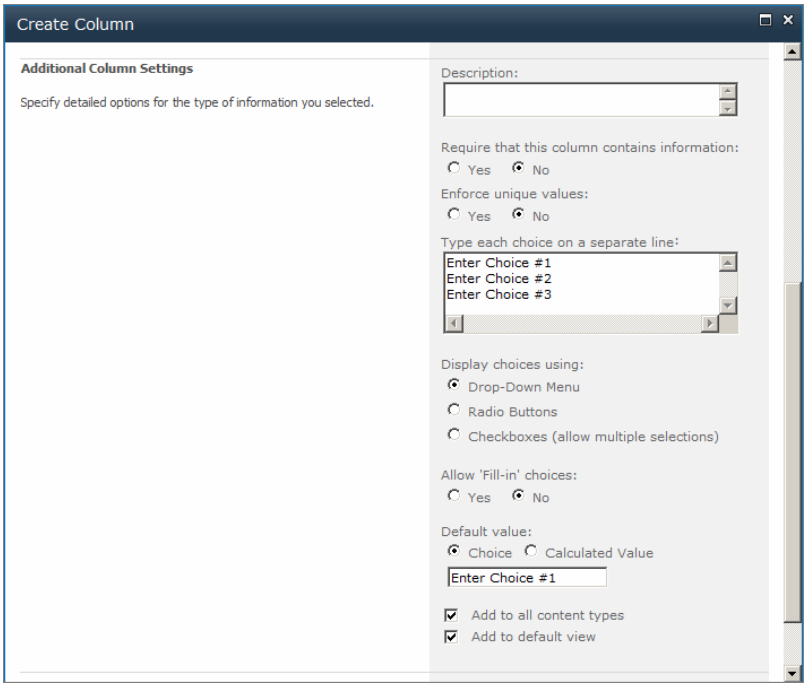


FIGURE 5-2 : Editing control

Building a Restricted Social Security Number Field Type

As mentioned previously, a good example of a scenario where you may want to create a custom field type is when storing sensitive data within a list, such as a person's SSN. The following section will guide you through creating a field type for storing SSN numbers which will prevent displaying the full SSN number within a list view.

Creating the SharePoint Solution

You need to create the Visual Studio solution that holds your SSN Field Definition. Open up Visual Studio, and follow these steps:

1. Select File ➪ New ➪ Project.
2. Select Installed Templates ➪ Visual C# ➪ SharePoint ➪ 2010 ➪ Empty SharePoint Project.
3. Specify the project name as **Wrox.Lists**.
4. Click OK.
5. Specify the URL of your SharePoint portal.
6. Select Deploy as a Farm Solution.
7. Click Finish.

www.it-ebooks.info

8. Right-click on the project file and select properties.
9. Click the SharePoint tab on the left side of the screen.
10. At the bottom of the SharePoint section, uncheck the Auto-retract After Debugging.
11. Save the project file.

You should now have an empty SharePoint 2010 project, as shown in Figure 5-3.

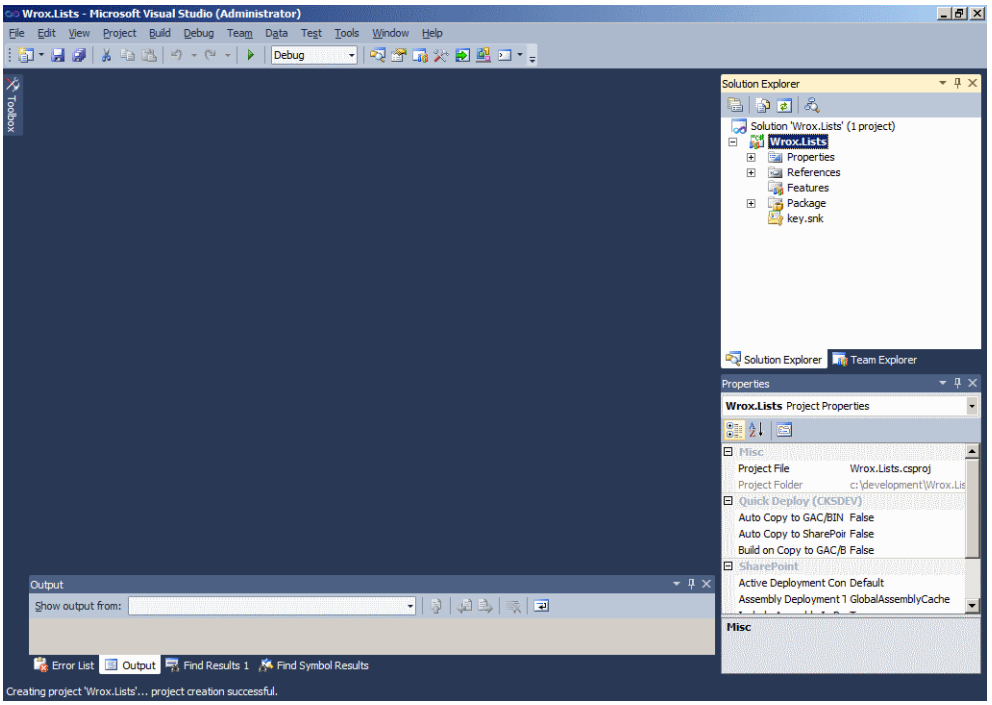


FIGURE 5-3: Empty SharePoint solution

Stubbing Out the Solution Structure and Files

Now that you have an empty solution, you need to create the solution structure and stub out the various files that make up a custom field type. Creating field type definitions is still a manual process, and unfortunately Visual Studio does not have a field type template that you can use to kick off the development.

The field type definition contains a single XML file placed in the Template\XML folder of the 14 hive. The following steps add a SharePoint Mapped Folder to the XML folder:

1. Right-click the project, and select Add ➪ SharePoint Mapped Folder.
2. Expand the TEMPLATE folder and select XML.
3. Click OK.

www.it-ebooks.info

4. Right-click the XML folder in the solution, and select Add ➪ New Item.
5. Select Installed Templates ➪ ata ➪ XML File.
6. Name the file **fldtypes_Wrox_SSNField.xml** and click Add.

Be careful with step 6. Your field definition field must start with **fldtypes**, even if you choose to use your own value instead of **_Wrox_SSNField**.

Next, you need to add the XSLT style sheet to the project to handle rendering the field when viewed in the list view. The XSLT file needs to be placed beneath the `XSL` directory of the `Template` directory in the 14 hive.

1. Right-click the project, and select Add on SharePoint Mapped Folder.
2. Expand the TEMPLATE folder, then expand the LAYOUTS folder and select XSL.
3. Click OK.
4. Right-click the XSL folder in the solution, and select Add ➪ New Item.
5. Select Installed Templates ➪ Data ➪ XSLT File.
6. Name the file **fldtypes_Wrox_SSNField.xsl** and click Add.

Next, you need to add the rendering template file to handle rendering within all the list form views (display, add, and update). The rendering template goes beneath the Control Template folder found within the 14 hive. Follow the steps:

1. Right-click the project, and select Add ➪ SharePoint Mapped Folder.
2. Expand the TEMPLATE folder, and select CONTROLTEMPLATES.
3. Click OK.
4. Right-click the CONTROLTEMPLATES folder, and select Add ➪ New Item.
5. Select Installed Templates ➪ Visual C# ➪ SharePoint ➪ 2010 ➪ User Control.
6. Name the User Control **SSNFieldControl.ascx** and click Add.

You should now have the CONTROLTEMPLATES folder within your solution and a user control called `SSNFieldControl.ascx` within your solution.

The next step is to create the stubs for the class (.cs) files used by the field definition: the field class and the rendering control. Although not required, it is generally a best practice to keep all the field type definitions within a subfolder of the project, such as a folder named "Field Types." Because multiple classes exist for each field type, it also helps to place those in their own folder beneath the Field Type folder. Follow these steps to create the folders and the stub files:

1. Right-click the project, and select Add ➪ New Folder.
2. Name the folder **Field Types**.
3. Right-click the Field Types folder, and select Add ➪ New Folder.
4. Name the folder **SSN**.

www.it-ebooks.info

5. Right-click the SSN folder, and select Add ➪ Class.
6. Name the class **SSNField.cs** and click Add.
7. Right-click the SSN folder, and select Add ➪ Class.
8. Name the class **SSNFieldControl.cs** and click Add.

You now have all the files required to implement the SSN field type; next it is time to start populating the various stub files with actual contents.

Creating the SSN Field Type Definition

The field type definition is what makes SharePoint aware of your custom field type. Before you can create the field type definition, however, you need to know what your assembly's public key token is. To determine your assembly's public key:

1. Build the solution.
2. Open up a Visual Studio command prompt (Start ➪ Visual Studio 2010 ➪ Visual Studio Tools ➪ Visual Studio Command Prompt).
3. Navigate to the bin\Debug directory of your solution.
4. Run the command `"Sn –T Wrox.Lists.dll,"` which spits out the public key token. Ensure you use an uppercase T because case matters.
5. Copy the public key token.

As long as your solution continues to use the same key file (`key.snk`), your public key remains the same. When you have the public key token, you can proceed to creating the field type definition.

1. Double-click the `fldtypes_Wrox_SSNField.xml` file you created in the previous steps.
2. Replace the contents of the file with the following XML snippet:




```
<?xml version="1.0" encoding="utf-8" ?>
<FieldTypes>
    <FieldType>
        <Field Name="TypeName">SocialSecurityNumber</Field>
        <Field Name="ParentType">Text</Field>
        <Field Name="TypeDisplayName">Social Security Number</Field>
        <Field Name="TypeShortDescription">
            Masked Social Security Number (XXX-XX-6789)
        </Field>
        <Field Name="AllowBaseTypeRendering">FALSE</Field>
        <Field Name="FieldTypeClass">
            Wrox.Lists.FieldTypes.SSNField, Wrox.Lists,
            Version=1.0.0.0,Culture=neutral,PublicKeyToken=<your token>
        </Field>
    </FieldType>
</FieldTypes>
```

Wrox.Lists\Wrox.Lists\XML\fldtypes_Wrox_SSNField.xml

www.it-ebooks.info

3. Replace the `<your token>` section of the snippet with the public key token you obtained in step 5 of the previous steps.

Creating the XSL Style Sheet

Now that you created your field definition, you can create your XSL Style Sheet, which handles rendering of the field when it appears in the list view. For this view, you need to mask the value of the SSN so that only the last four digits of the SSN display.

Edit the `fldtypes_Wrox.SSNField.xsl` and replace the contents of the file with the following XML snippet:




```
<xsl:stylesheet xmlns:x="http://www.w3.org/2001/XMLSchema"
    xmlns:d="http://schemas.microsoft.com/sharepoint/dsp"
    version="1.0"
    exclude-result-prefixes="xsl msxsl ddwrt"
    xmlns:ddwrt="http://schemas.microsoft.com/WebParts/v2/DataView/runtime"
    xmlns:asp="http://schemas.microsoft.com/ASPNET/20"
    xmlns:__designer="http://schemas.microsoft.com/WebParts/v2/DataView/designer"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt"
    xmlns:SharePoint="Microsoft.SharePoint.WebControls"
    xmlns:ddwrt2="urn:frontpage:internal">
    <xsl:template match="FieldRef[@FieldType = 'SocialSecurityNumber']"
                  mode="Text_body">
        <xsl:param name="thisNode" select="." />
        <xsl:param name="value" select="$thisNode/@*[name()=current()/@Name]" />
        <xsl:choose>
            <xsl:when test="string-length($value) = 11">
                <span>
                    XXX-XX-<xsl:value-of select=
                     "substring($value, string-length($value) - 3)" />
                </span>
            </xsl:when>
            <xsl:otherwise>
                <span>
                    N/A
                </span>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template >
</xsl:stylesheet>
```

Wrox.Lists\Wrox.Lists\XSL\fldtypes_Wrox_SSNField.xml

The rendering template begins with the `xsl:template` element, and the match attribute tells SharePoint to use this template for all columns in the list that are of type `SocialSecurityNumber` (which is the type you defined in the last section).

The rest of the style sheet uses standard XSLT constructs to display only the last four characters of the field value (assuming it is a well-formed SSN of 11 characters: XXX-XX-XXXX) or nothing if the field is blank. Or if it contains a value but isn't the 11 characters expected in an SSN, N/A displays instead.

www.it-ebooks.info

Creating the SSN Field Class

The SSN Field Class is the class that stored the specifics about a column that has been added to a list, such as the various settings about whether it is required, and also is responsible for telling SharePoint about the field control used by the field type. To add the body to the control, follow these steps:



1. Double-click the `SSNField.cs` file you created in the previous section.
2. Add the following `using` statements to the class:

```
using Microsoft.SharePoint;
using Microsoft.SharePoint.Security;
using Microsoft.SharePoint.WebControls;
using System.Security.Permissions;
using System.Text.RegularExpressions;
```

3. Change the namespace declaration from `Wrox.Lists.FieldTypes.SSN` to remove the `SSN` reference:

```
namespace Wrox.Lists.FieldTypes
```

4. Change the `SSNField` class definition to public and to inherit from `SPFieldText`:

```
public class SSNField : SPFieldText
```

5. Add the following required constructors to the class:

```
public SSNField(SPFieldCollection fields, string fieldName)
    : base(fields, fieldName)
{
}

public SSNField(SPFieldCollection fields, string typeName,
    string displayName) : base(fields, typeName, displayName)
{
}
```

6. Next, add some validation logic to the field to enforce the SSN structure the control expects. Override the `GetValidatedString` method and implement to following code:

```
public override string GetValidatedString(object value)
{
    String val = value as string;

    if (this.Required && String.IsNullOrEmpty(val))
    {
        throw new SPFieldValidationException(
            Title + " is required.");
    }
    else if (!String.IsNullOrEmpty(val))
    {
        Regex regex = new Regex(@"\d{3}-\d{2}-\d{4}");
        if (!regex.IsMatch(val))
        {
```

www.it-ebooks.info

```
                throw new SPFieldValidationException(
                    Title + " must be in the " +
                    "format of XXX-XX-XXXX. ex: 123-45-6789");
            }
        }

        return base.GetValidatedString(value);
    }
```

Wrox.Lists\Wrox.Lists\FieldTypes\SSN\SSNField.cs

The `GetValidatedString` method is called by SharePoint whenever a user attempts to save a column, and if a `SPFieldValidationException` error is thrown, it displays the message to the user and prevents saving of the file. In the previous implementation, you validate for two possible cases. The first is if the field is required but blank, and the second is if the column has a value but isn't in the format of three numeric numbers, followed by a dash, two numeric numbers, a dash, and four numeric numbers.

7. Save all files.

This provides a good point to deploy your solution and make sure everything has been created properly so far. Press F5 to debug your solution.

8. Navigate to a list within your target site, and select the List tab in the ribbon; choose Create Column.

9. If everything goes correctly, you should see a `Masked Social Security Number` field type in the list of column types, as shown in Figure 5-4:

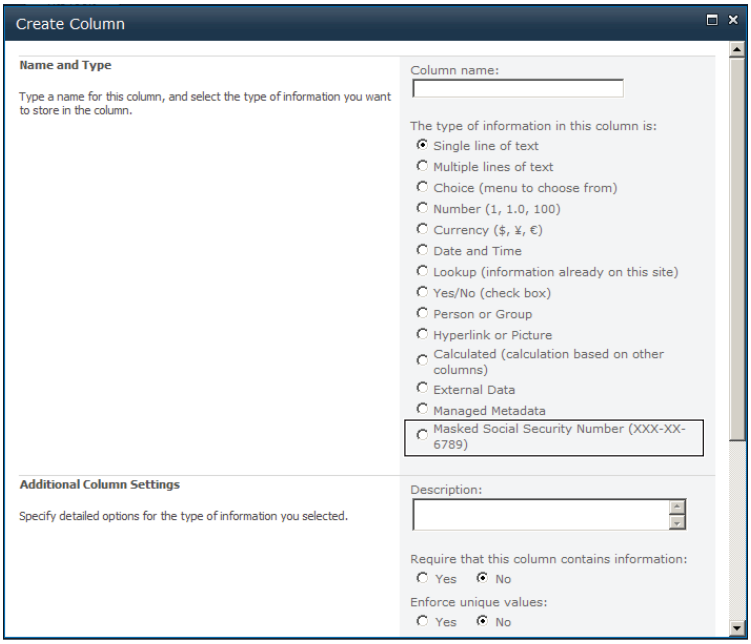


FIGURE 5-4: SSN field type

www.it-ebooks.info

10. Name the column: SSN.
11. Click OK.
12. You should now see an SSN column in your list.
13. Add a list item, and fill in a value for the SSN column that is improperly formatted, such as **123456789** and save the item.
14. You should see an error message stating that the format is incorrect. Change the SSN to a valid format such as **123-45-6789** and save the item.
15. If everything works as expected, you should see something like what displays in Figure 5-5.

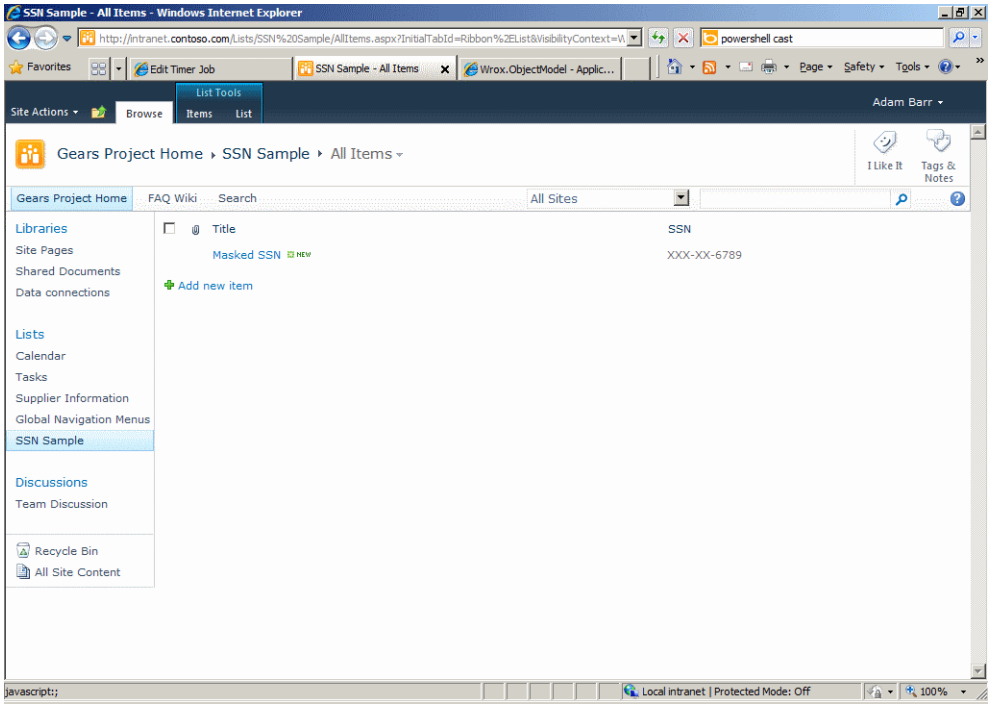


FIGURE 5-5: SSN with masked value

16. However, if you click the list item and view or edit the details, you see that the value is not masked, as shown in Figure 5-6.

As mentioned earlier, the display, add, and edit forms render using a different mechanism from the list view display. To fix the display, add, and edit forms, you need to modify the SSN Field Control (`SSNFieldControl.cs`) and the Rendering Template (`SSNFieldControl.ascx`).

www.it-ebooks.info

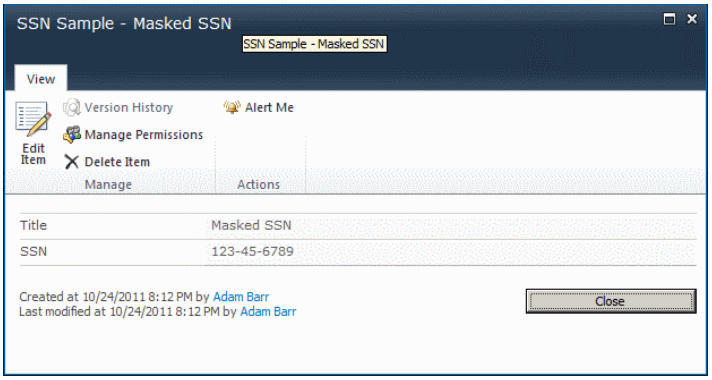


FIGURE 5-6: Item details

Creating the SSN Field Rendering Template

The field rendering template is simply a user control with no code-behind which contains the controls and layout that are used to render the value of a field type. To modify the rendering template, follow these steps:



1. Expand `SSNFieldControl.ascx` to expose `SSNFieldControl.ascx.cs` and `SSNFieldControl.ascx.designer.cs` files.
2. Delete the `SSNFieldControl.ascx.cs` and `SSNFieldControl.ascx.designer.cs` files. (They aren't needed.)
3. Open `SSNFieldControl.ascx` in Visual Studio.
4. Change the last line contained in the file, the Control directive, to read only the following:

```
<%@ Control Language="C#" %>
```

5. Add the following rendering template to the end of the file:

```
<SharePoint:RenderingTemplate ID="SSNDisplay" runat="server">
  <Template>
    <asp:Label ID="SSNLabel" runat="server" />
  </Template>
</SharePoint:RenderingTemplate>
```

Wrox.Lists\Wrox.Lists\CONTROLTEMPLATES\SSNFieldControl.ascx

The rendering template has only a single control, `SSNLabel`. This control displays the read-only view of the field. You may wonder how the editable view of the field renders because no textbox or other control exists that enables editing in the template. In this case, because the `SSNFieldControl` inherits from `TextField` and the behavior of editing an SSN is identical to editing a standard text

www.it-ebooks.info

field, you can leverage the base classes rendering template for editing. In other instances in which you require a customized edit experience and display experience, you would likely define a second `RenderTemplate` in this file for editing.

Creating the SSN Field Control

The SSN Field Control handles the display of the field type in all forms and also handles validation of the control. To modify the SSN Field Control, follow these steps:



1. Open `SSNFieldControl.cs` in Visual Studio.
2. Add the following `using` statements to the top of the file:

```
using Microsoft.SharePoint;
using Microsoft.SharePoint.WebControls;
using System.Web.UI.WebControls;
```

3. Change the namespace declaration from `Wrox.Lists.FieldTypes.SSN to Wrox.Lists .FieldTypes` to remove the SSN reference
4. Change the class definition to be public and to inherit from `TextField`:

```
public class SSNFieldControl : TextField
```

5. Override the `DisplayTemplateName` property with the following implementation:

```
public override string DisplayTemplateName
{
    get
    {
        return "SSNDisplay";
    }
    set
    {
        base.DisplayTemplateName = value;
    }
}
```

The `DisplayTemplateName` property returns the ID of the rendering template responsible for rendering the display form. This is the ID value (`SSNDisplay`) specified in step 5 of the previous section.

6. Override the `DefaultTemplateName` property with the following implementation:

```
protected override string DefaultTemplateName
{
    get
    {
        if (this.ControlMode == SPControlMode.Display)
        {
            return this.DisplayTemplateName;
        }
        else
```

www.it-ebooks.info

```
        {
            return base.DefaultTemplateName;
        }
    }
}
```

Similar to the `DisplayTemplateName` property, the `DefaultTemplateName` property is meant to return the ID of the rendering template responsible for rendering the field based on its current state (`ControlMode`). In this case, if the field displays, the rendering template created earlier is used; otherwise, the base `TextField`'s rendering template is used.

7. Override the `CreateChildControls` method and use the following implementation:

```
protected override void CreateChildControls()
{
    base.CreateChildControls();

    if (this.ControlMode == SPControlMode.Display)
    {
        Label lblSSN = (Label)TemplateContainer.FindControl("SSNLabel");
        SPBasePermissions perms =
            this.ListItem.GetUserEffectivePermissions(
                SPContext.Current.Web.CurrentUser.LoginName);
        string value = this.ItemFieldValue as string;
        if ((perms & SPBasePermissions.EditListItems) ==
             SPBasePermissions.EditListItems)
        {
            lblSSN.Text = value;
        }
        else
        {
            lblSSN.Text = "XXX-XX-" +
                value.Substring(value.Length - 4);
        }
    }
}
```

The `CreateChildControls` method is responsible for setting the value of the controls found within the Rendering Template (`SSNFieldControl.ascx`). Because SharePoint can prevent users from accessing the add and edit forms of a list item if the user does not have the appropriate permissions, you need to change only the display form behavior.

In the preceding method, you first call the base implementation of the `CreateChildControls` method, which allows the base `TextField` control to initialize its behavior. Then, the code checks to see if the form being rendered is the display form. If it is, it then finds the `SSNLabel` control that exists in the `RenderingTemplate` so that this control can be populated with the column data. Finally, it checks the user's permission on the list, and if they can edit the item, it displays the unmasked value, and if they can't, it uses a masked value.

www.it-ebooks.info

8. The `Value` property must also be overridden. This property handles the storing of the column value within the field. Override the property with this implementation:

```
public override object Value
{
    get
    {
        EnsureChildControls();
        return base.Value;
    }
    set
    {
        EnsureChildControls();
        base.Value = (String)value;
    }
}
```

As the name implies, the `Value` property stores the value of the field. It is extremely important to always call `EnsureChildControls()` before accessing the `base.Value` property because the `Value` is typically stored by most field controls in a control on the page.

9. Save all the files and build the project.

Wrox.Lists\Wrox.Lists\FieldTypes\SSN\SSNFieldControl.cs

Associating the SSN Field Control with the SSN Control

There is one final step in the process, which is to let SharePoint know to use the SSN Field Control when rendering the SSN field type. Do this by overriding the `FieldRenderingControl` property in the `SSNControl` class. Follow these steps:



1. Double-click the `SSNField.cs` file.
2. Override the `FieldRenderingControl` and replace it with this implementation:

```
public override BaseFieldControl FieldRenderingControl
{
    [SharePointPermission(SecurityAction.LinkDemand, ObjectModel = true)]
    get
    {
        BaseFieldControl fieldControl = new SSNFieldControl();
        fieldControl.FieldName = this.InternalName;

        return fieldControl;
    }
}
```

The `FieldRenderingControl` property returns the field control that should be used when rendering the field in the list forms.

3. Save the file.
4. Right-click the project, and deploy the solution.

Wrox.Lists\Wrox.Lists\FieldTypes\SSN\SSNField.cs

www.it-ebooks.info

Testing the Solution

You have done some test deployments while developing the solution, but here are complete steps to test the solution. Fully testing the solution requires two accounts (one with edit permissions and one with read permissions).

1. Deploy the solution if you haven't already.
2. Open your browser and navigate to your portal.
3. Add one test user as a site owner (referred to as Owner) and one test user as a site visitor (referred to as Visitor).
4. Log in as your Owner account.
5. Click Site Actions in the upper-left portion of your portal home page.
6. Select More Options.
7. Click the List link on the left side of the dialog window.
8. Click Custom List.
9. Fill in the name of SSN Sample, and click Create.
10. The list will be created, and your browser window should now display the empty list with a single Title column.
11. Click the List tab of the SharePoint Ribbon.
12. Select the Create Column button.
13. Name the Column **SSN**.
14. Specify the type as a Masked Social Security Number.
15. Click OK.
16. You should now see the list again with two columns: Title and SSN.
17. Click the Add New Item link below the list.
18. Enter in a title of **SSN Test**.
19. Enter in a social security number of **123456789**.
20. Click Save.
21. You should see an error message that states the SSN is in an invalid format, as shown in Figure 5-7.
22. Enter in a valid SSN number (**123-45-6789**).
23. Click Save.
24. The list item should be added to the list, and the SSN should be masked, as shown in Figure 5-8.
25. Click SSN Test, and you should see an unmasked SSN value, as shown in Figure 5-9.
26. Click the Close button on the dialog.

www.it-ebooks.info

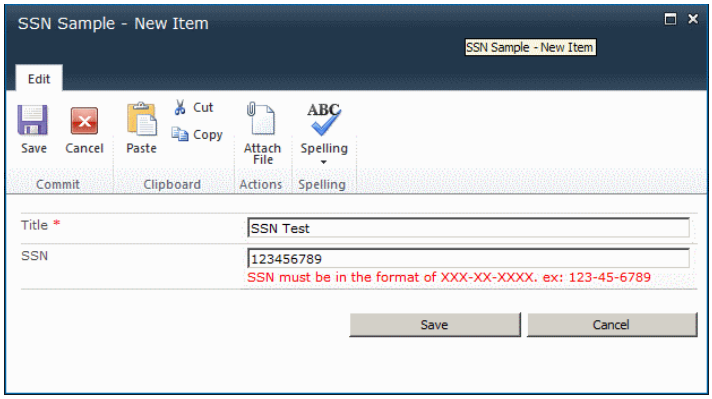


FIGURE 5-7: Invalid format error

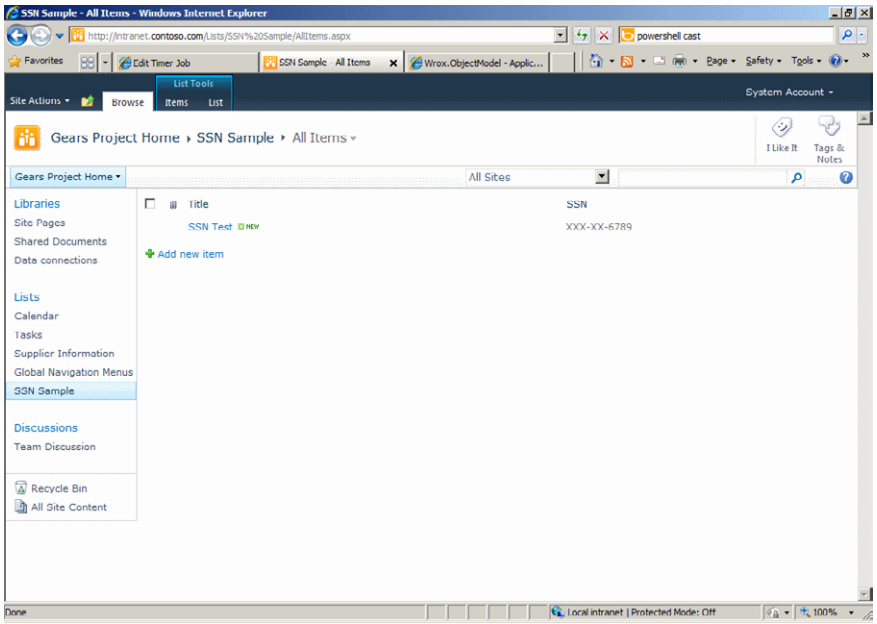


FIGURE 5-8: List view

27. Log in as your Visitor account.

28. Click SSN Test again, and you should see a masked SSN value, as shown in Figure 5-10.

Congratulations! You've created a masked Social Security Number field type, which hopefully serves as a good example of how you can leverage custom lists.

www.it-ebooks.info

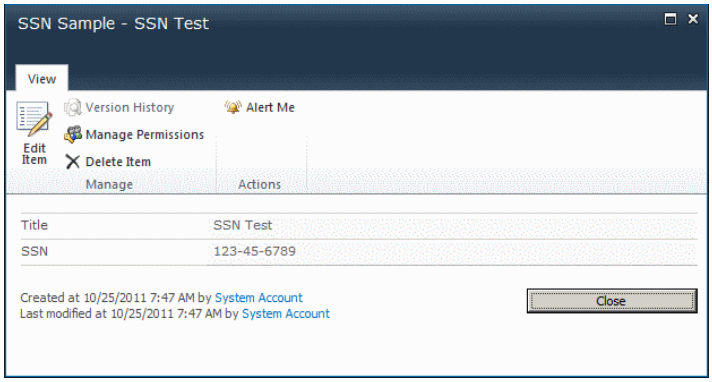


FIGURE 5-9: List display form—Owner

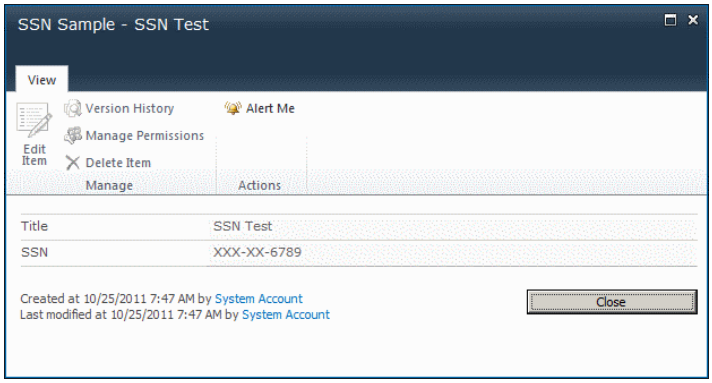


FIGURE 5-10: List display form—Visitor

The preceding modification secures SSNs only within the standard list view and the list forms. If the list can be opened in datasheet view, Access, or exported to Excel, the full users can see the SSN. To prevent this behavior, you need to disable the datasheet view for the list and also create a new permission level for the site, which prevents client integration features.

Field Editing Controls

One aspect of custom field types not required by this solution, which are likely to be used with more advanced field types, are custom field editing controls. These controls are used when adding or editing a field/column and allow the user to specify configuration values or other details relevant to the operation of the field.

Although outside of the scope of this chapter, one enhancement that could be made to the SSN Control relates to who can see the full Social Security number. As currently written, any user who can edit the list has access to the Social Security number information. On the face of it, that may seem sufficiently secure, but whole classes of users (farm administrators, site collection administrators) always have edit

www.it-ebooks.info

access to a list but who probably shouldn't view SSNs. Additionally, one use case for the SSN might be to allow existing employees to validate their SSN. That scenario is possible with the existing SSN control's implementation by breaking inheritance on every list item and assigning edit permissions to the employee for only his record, but that would open up the list to being modified by the individual employees, which is probably not wanted.

One solution is to create a field editing control that allows more granular specification over who can and can't view Social Security numbers. Figure 5-11 shows a possible display of what the edit control would look like. It is left as an exercise for you to implement such a solution.

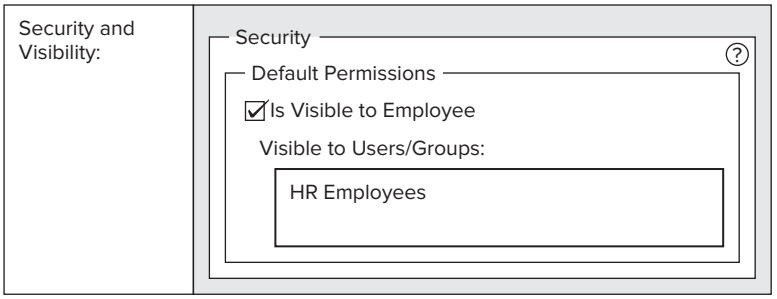


FIGURE 5-11: Field editing control mockup

CUSTOM CONTENT TYPES

Content types are reusable collections of columns (metadata) that you can manage in a centralized manner. Rather than having to constantly configure lists to collect the same columns of information, you can create content types once and then associate them with a list or lists. Content types also support inheritance, so you can define a common base that contains certain metadata and then extend it to collect additional data. For example, this chapter walks you through creating an Employee content type that stores information about employees of your company and uses the SSN field type created in the previous section.

Creating the Employee Details Content Type Stub

Unlike field definitions, there is a Visual Studio template for content types (which consist of a single XML definition file) so that the process is a bit more straightforward than the process to create a field definition. Follow these steps to create a content type.

1. Open the solution (`Wrox.Lists.sln`) created in the previous step in Visual Studio.
2. Right-click the project, and select Add ➪ New Folder.
3. Name the folder **ContentTypes**.
4. Right-click the ContentTypes folder, and select Add ➪ New Item.
5. Select Visual C# ➪ SharePoint ➪ 2010 ➪ Content Type.

www.it-ebooks.info

6. Name the content type **EmployeeContentType** and click Add.

   This brings up the Content Type Wizard dialog. Every custom content type in SharePoint must inherit from an existing content type. In general, select the existing content type that is the most similar to the content type you want to create. For content types associated with document libraries, the base content type is Document. For content types associated with Lists, the Item content type is the most common. Because you create a list to store employee information, it makes sense to use the Item type as your parent type.

7. Select Item from the drop-down list, and click Finish.

   You should now see a file called `EmployeeContentType`; beneath that file, you see an `Elements.xml` file (which is also open in Visual Studio). In addition, a feature named `Feature1` has been created. The `Elements.xml` file is where the content type is defined, and the `Feature1` feature is what deploys the content type to the site collection. Because `Feature1` is not particularly descriptive, take a moment to rename the feature to `Wrox.Lists.ContentTypes`.

8. Right-click the top `Feature1` item in the Solution Explorer, and select Rename.

9. Rename `Feature1` to **Wrox.Lists.ContentTypes**.

10. Double-click the newly named feature to open it in the designer.

11. Change the Title to **Wrox.Lists - Custom Content Types**.

12. Change the Scope of the feature from Web to Site.

13. Save the file.

14. Verify that your solution looks similar to Figure 5-12.

15. Open the `Elements.xml` file beneath `EmployeeContentType`.

16. Change the name attribute to **Employee Details**.

17. Change the Group to **Wrox Content Types.**”

18. Change the Description to read, **A content type that stores employee information**.

19. Save the file.

20. Press F5 to deploy and debug the solution.

21. In your web browser, click Site Actions ➪ Site Settings in the upper left of the page.

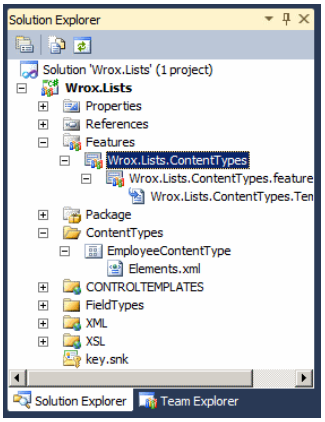


FIGURE 5-12: Sample Solution

22. Beneath the Galleries section, click Site Content Types.

23. Scroll down to the bottom of the list, and verify that a Wrox Content Type group includes the Employee Details content type, as shown in Figure 5-13.

www.it-ebooks.info

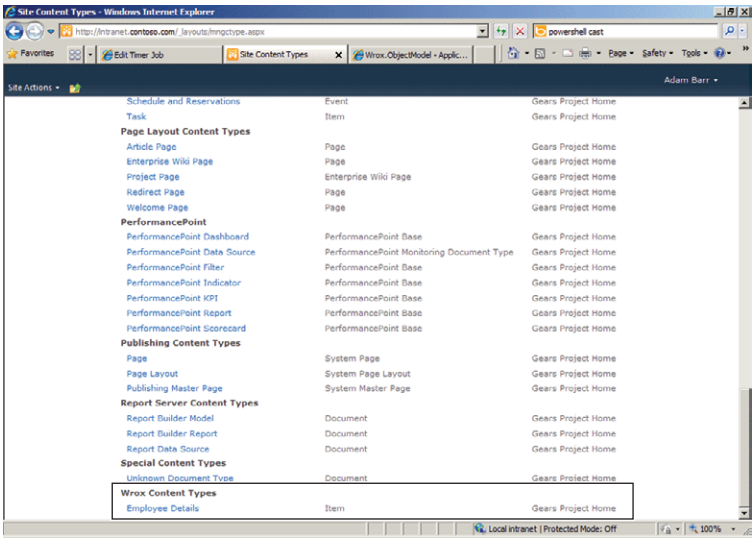


FIGURE 5-13: Content type gallery

Implementing the Employee Details Content Type

Because you didn't specify anything in the body of the content type definition, if you were to add the Employee Details content type to an existing list, you would find that it acts exactly like an Item content type. It is now time to update the content type definition so that it can store information about employees.

For the purpose of this example, the following information will be part of the content type:

- Display Name
- First Name
- Last Name
- Email Address
- Phone Number
- Hire Date
- Social Security Number

For Display Name, it makes sense to repurpose the existing Title column that is standard to SharePoint lists. This enables you to continue to offer the common functionality that is standard to lists such as the drop-down menu and the link that takes you to the display form of the list. The other fields will be new fields, which will be added to the content type.

Any field that you want to use within a content type must be defined outside of the content type and then it can be referenced inside of the content type definition. You can define the fields within the

www.it-ebooks.info

same elements file or as a separate module. In this case, defining them directly within the `Elements.xml` file should be sufficient.

Define the new fields and the content type field references by following these steps:



1. Beneath EmployeeContentType, open the `Elements.xml` file.
2. Add the following Field elements above the content type element:

```
<Field ID="{19500D43-A35C-449D-AE96-C194E5848BEE}"
            Name="WroxFirstName"
            DisplayName="First Name"
            Type="Text"
            Group="Wrox Columns"
            Overwrite="TRUE"
            OverwriteInChildScopes="TRUE"
            />
    <Field ID="{01C6F4BC-2169-457F-9E78-6D4FF5121C8C}"
            Name="WroxLastName"
            DisplayName="Last Name"
            Type="Text"
            Group="Wrox Columns"
            Overwrite="TRUE"
            OverwriteInChildScopes="TRUE"
            />
    <Field ID="{B9E99EA6-B5A8-4DC3-BA1D-0EACA2E4CCF4}"
            Name="WroxEmailAddress"
            DisplayName="Email Address"
            Type="Text"
            Group="Wrox Columns"
            Overwrite="TRUE"
            OverwriteInChildScopes="TRUE"
            />
    <Field ID="{038792BF-E0CC-4A39-815B-F98C32C41C1E}"
            Name="WroxPhoneNumber"
            DisplayName="Phone Number"
            Type="Text"
            Group="Wrox Columns"
            Overwrite="TRUE"
            OverwriteInChildScopes="TRUE"
            />
    <Field ID="{74D38560-8A44-4767-A566-CC3D156173EC}"
            Name="WroxHireDate"
            DisplayName="Hire Date"
            Type="DateTime"
            Format="DateOnly"
            Group="Wrox Columns"
            Overwrite="TRUE"
            OverwriteInChildScopes="TRUE"
            />
    <Field ID="{3A521AE5-6D33-479C-AB39-EEAA66DD7DEB}"
            Name="WroxSSN"
            DisplayName="SSN"
            Type="SocialSecurityNumber"
```

www.it-ebooks.info

```
Group="Wrox Columns"
Overwrite="TRUE"
OverwriteInChildScopes="TRUE"
/>
```

The required attributes of a Field follow:

- **ID:** A GUID that is unique to the farm.
- **Name:** A name that is unique to the web the field is deployed to. To avoid naming collisions, preface the name with an additional identifier.
- **Display Name:** The friendly name seen by users when they see the field in a list.
- **Type:** The data type of the field. Some common types follow:
  - Boolean
  - Calculated
  - Choice
  - Computed
  - Counter
  - Currency
  - DateTime
  - Guid
  - Integer
  - Lookup
  - LookupMulti
  - MultiChoice
  - MultiColumn
  - Note
  - Number
  - Text
  - URL
  - User
  - UserMulti
- **Group:** Appears in the SharePoint UI.

3. Change the Inherits attribute of the ContentType element to FALSE. This is required to change the display name of the Title column.

www.it-ebooks.info

4. Add the following `FieldRef` elements within the `FieldRefs` element of the `ContentType` element:

```
<FieldRef ID="{fa564e0f-0c70-4ab9-b863-0177e6ddd247}"
          Name="Title" DisplayName="Display Name"
          Required="FALSE" Sealed="TRUE" />
<FieldRef ID="{82642ec8-ef9b-478f-acf9-31f7d45fbc31}"
          Name="LinkTitle" DisplayName="Display Name"
          Required="FALSE" Sealed="TRUE" />
<FieldRef ID="{bc91a437-52e7-49e1-8c4e-4698904b2b6d}"
          Name="LinkTitleNoMenu" DisplayName="Display Name"
          Required="FALSE" Sealed="TRUE" />
<FieldRef ID="{19500D43-A35C-449D-AE96-C194E5848BEE}"
          Name="WroxFirstName" DisplayName="First Name"
          Required="FALSE"/>
<FieldRef ID="{01C6F4BC-2169-457F-9E78-6D4FF5121C8C}"
          Name="WroxLastName" DisplayName="Last Name"
          Required="FALSE" />
<FieldRef ID="{B9E99EA6-B5A8-4DC3-BA1D-0EACA2E4CCF4}"
          Name="WroxEmailAddress" DisplayName="Email Address"
          Required="FALSE" />
<FieldRef ID="{038792BF-E0CC-4A39-815B-F98C32C41C1E}"
          Name="WroxPhoneNumber" DisplayName="Phone Number"
          Required="FALSE" />
<FieldRef ID="{74D38560-8A44-4767-A566-CC3D156173EC}"
          Name="WroxHireDate" DisplayName="Hire Date"
          Required="FALSE" />
<FieldRef ID="{3A521AE5-6D33-479C-AB39-EEAA66DD7DEB}"
          Name="WroxSSN" DisplayName="SSN" Required="FALSE"/>
```

All `FieldRef` elements must have attributes specified for the following:

- ID
- Name
- DisplayName
- Required

Wrox.Lists\Wrox.Lists\ContentTypes\EmployeeContentType\Elements.xml

5. Save the file and deploy the solution.
6. Open a browser and navigate to your portal.
7. Click Site Actions in the upper left corner.
8. Click Site Settings.
9. Beneath the Galleries section, click Site Content Types..
10. Scroll to the bottom of the content type list, and click the Employee Details content type.
11. Verify that the details look similar to Figure 5-14.

www.it-ebooks.info

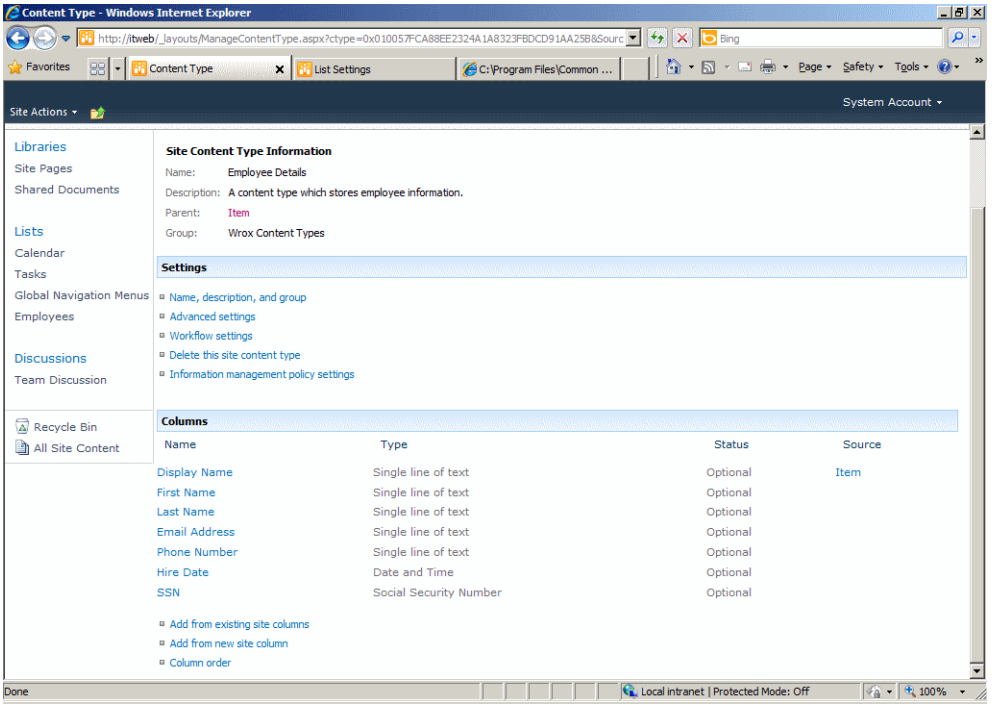


FIGURE 5-14: Content type details

You now have an Employee Details content type that can be associated with lists, but using this content type still involves a lot of steps. You'd have to create a list; modify its advanced properties so that you can manage content types; remove the existing content type and add the Employee Details content type; and then configure the All Items view to display the columns added by the content type. Wouldn't it be nice if there were another way?

LIST DEFINITIONS

Like content type definitions, list definitions enable you to define the details of a list such as which content types associate with the list, which columns exist for the list, what views exist, and what columns appear on every view. After a list definition has been deployed, the list definition appears as one of the available list types that you can use when creating a new list.

Creating the Employees List Definition

You now create an Employees list definition that automates all the manual steps you must take to associate the Employee Details content type with a custom list template. Follow these steps:

1. Open Visual Studio and the `Wrox.Lists` solution you have been using throughout the project.
2. Right-click the project, and select Add ➪ New Folder.

www.it-ebooks.info

3. Name the folder **Lists**.
4. Right-click the Lists folder, and select Add ➪ New Item.
5. Select Visual C# ➪ SharePoint ➪ 2010 ➪ List Definition from Content Type. You need to deploy the Employee Details content type previously for it to be selectable from the wizard in the next step.
6. Name the definition **EmployeesListDefinition**.
7. Click Add.
8. The List Definition Settings Wizard displays.
9. Specify the name of the list definition as **Employee Details**.
10. Specify the content type to use for the list definition as Employee Details (Wrox.Lists\EmployeeContentType).
11. Keep the Add a List Instance for This Definition checked. This can save you the manual step of adding the list to the site but may not be appropriate for all list definitions depending on their intended usage.
12. The values should look like those shown in Figure 5-15.

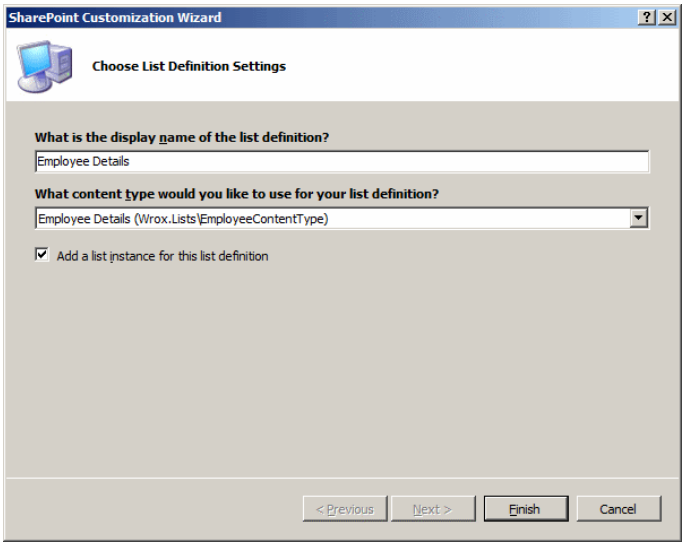


FIGURE 5-15: List Definition Settings Wizard

13. Click Finish.

Visual Studio creates an EmployeeListDefinition file and beneath that creates a file named `ListInstance1`, a `Schema.xml` file, and an `Elements.xml` file. As the name implies, the `ListInstance1` file handles the details of the list instance that will be created when the feature is activated, such as the name of the list, whether it appears on the quick launch, its description, and so on. The `Elements.xml`

www.it-ebooks.info

file contains similar details about the list definition, such as its display name, whether it appears on the quick launch, its description, and so forth. Finally, the `Schema.xml` file contains all the details about the actual contents of the list definition, such as what fields are available, which fields show up in the default view, and more.

You may have noticed that Visual Studio had other list definition templates available than the one you chose (List Definition from Content Type). You could create list definitions without associating them with a content type, but in many ways, even if you use the content type only for a single list, it is simpler to create the content type first as you did and then create a list definition from the content type. If you instead create a list definition without using a content type, you need to write a lot of XML, which is similar to that in a content type definition.

Customizing the List Schema

As mentioned previously, the list schema is where most of the "magic" of a list definition happens. The `Schema.xml` file determines what content types associate with a list, which fields appear in the list, which views exist for the list, and which columns exist in the view. Following are the `Schema.xml` file contents that Visual Studio generates. Other than formatting to fit within the confines of these pages, this should look extremely similar to your own `Schema.xml` file.



Available for download on Wrox.com

LISTING 5-1: List Schema

```
<?xml version="1.0" encoding="utf-8"?>
<List xmlns:ows="Microsoft SharePoint" Title="Employee Details"
      FolderCreation="FALSE" Direction="$Resources:Direction;"
      Url="Lists/Wrox.Lists-EmployeesListDefinition" BaseType="0"
      xmlns="http://schemas.microsoft.com/sharepoint/">
  <MetaData>
    <ContentTypes>
      <ContentType ID="0x010057fca88ee2324a1a8323fbdcd91aa25b"
        Name="Employee Details" Group="Wrox Content Types"
        Description="A content type which stores employee information."
        Inherits="FALSE" Version="0">
        <FieldRefs>
          <FieldRef ID="{fa564e0f-0c70-4ab9-b863-0177e6ddd247}"
                    Name="Title" DisplayName="Display Name"
                    Required="FALSE" Sealed="TRUE" />
          <FieldRef ID="{82642ec8-ef9b-478f-acf9-31f7d45fbc31}"
                    Name="LinkTitle" DisplayName="Display Name"
                    Required="FALSE" Sealed="TRUE" />
          <FieldRef ID="{bc91a437-52e7-49e1-8c4e-4698904b2b6d}"
                    Name="LinkTitleNoMenu" DisplayName="Display Name"
                    Required="FALSE" Sealed="TRUE" />
          <FieldRef ID="{19500D43-A35C-449D-AE96-C194E5848BEE}"
                    Name="WroxFirstName" DisplayName="First Name"
                    Required="FALSE" />
          <FieldRef ID="{01C6F4BC-2169-457F-9E78-6D4FF5121C8C}"
                    Name="WroxLastName" DisplayName="Last Name"
                    Required="FALSE" />
          <FieldRef ID="{B9E99EA6-B5A8-4DC3-BA1D-0EACA2E4CCF4}"
```

www.it-ebooks.info

```
                Name="WroxEmailAddress" DisplayName="Email Address"
                Required="FALSE" />
      <FieldRef ID="{038792BF-E0CC-4A39-815B-F98C32C41C1E}"
                Name="WroxPhoneNumber" DisplayName="Phone Number"
                Required="FALSE" />
      <FieldRef ID="{74D38560-8A44-4767-A566-CC3D156173EC}"
                Name="WroxHireDate" DisplayName="Hire Date"
                Required="FALSE" />
      <FieldRef ID="{3A521AE5-6D33-479C-AB39-EEAA66DD7DEB}"
                Name="WroxSSN" DisplayName="SSN"
                Required="FALSE" />
    </FieldRefs>
  </ContentType>
</ContentTypes>
<Fields>
  <Field ID="{19500d43-a35c-449d-ae96-c194e5848bee}"
    Name="WroxFirstName" DisplayName="First Name"
    Type="Text" Group="Wrox Columns" Overwrite="TRUE"
    OverwriteInChildScopes="TRUE" />
  <Field ID="{01c6f4bc-2169-457f-9e78-6d4ff5121c8c}"
    Name="WroxLastName" DisplayName="Last Name"
    Type="Text" Group="Wrox Columns" Overwrite="TRUE"
    OverwriteInChildScopes="TRUE" />
  <Field ID="{b9e99ea6-b5a8-4dc3-ba1d-0eaca2e4ccf4}"
    Name="WroxEmailAddress" DisplayName="Email Address"
    Type="Text" Group="Wrox Columns" Overwrite="TRUE"
    OverwriteInChildScopes="TRUE" />
  <Field ID="{038792bf-e0cc-4a39-815b-f98c32c41c1e}"
    Name="WroxPhoneNumber" DisplayName="Phone Number"
    Type="Text" Group="Wrox Columns" Overwrite="TRUE"
    OverwriteInChildScopes="TRUE" />
  <Field ID="{74d38560-8a44-4767-a566-cc3d156173ec}"
    Name="WroxHireDate" DisplayName="Hire Date"
    Type="DateTime" Format="DateOnly" Group="Wrox Columns"
    Overwrite="TRUE" OverwriteInChildScopes="TRUE" />
  <Field ID="{3a521ae5-6d33-479c-ab39-eeaa66dd7deb}"
    Name="WroxSSN" DisplayName="SSN"
    Type="SocialSecurityNumber" Group="Wrox Columns"
    Overwrite="TRUE" OverwriteInChildScopes="TRUE" />
  <Field ID="{82642ec8-ef9b-478f-acf9-31f7d45fbc31}"
    Name="LinkTitle"
    SourceID="http://schemas.microsoft.com/sharepoint/v3"
    StaticName="LinkTitle" Group="$Resources:core,Base_Columns;"
    ReadOnly="TRUE" Type="Computed" DisplayName="$Resources:core,Title;"
    DisplayNameSrcField="Title" ClassInfo="Menu"
    AuthoringInfo="$Resources:core,Linked_Item_With_Menu;">
    <FieldRefs>
      <FieldRef ID="{fa564e0f-0c70-4ab9-b863-0177e6ddd247}"
                Name="Title" />
      <FieldRef ID="{bc91a437-52e7-49e1-8c4e-4698904b2b6d}"
                Name="LinkTitleNoMenu" />
      <FieldRef ID="{1344423c-c7f9-4134-88e4-ad842e2d723c}"
                Name="_EditMenuTableStart2" />
      <FieldRef ID="{2ea78cef-1bf9-4019-960a-02c41636cb47}"
```

continues

www.it-ebooks.info

LISTING 5-1 *(continued)*

```
                Name="_EditMenuTableEnd" />
    </FieldRefs>
    <DisplayPattern>
      <FieldSwitch>
        <Expr>
          <GetVar Name="FreeForm" />
        </Expr>
        <Case Value="TRUE">
          <Field Name="LinkTitleNoMenu" />
        </Case>
        <Default>
          <Field Name="_EditMenuTableStart2" />
          <Field Name="LinkTitleNoMenu" />
          <Field Name="_EditMenuTableEnd" />
        </Default>
      </FieldSwitch>
    </DisplayPattern>
  </Field>
  <Field ID="{bc91a437-52e7-49e1-8c4e-4698904b2b6d}"
    Name="LinkTitleNoMenu"
    SourceID="http://schemas.microsoft.com/sharepoint/v3"
    StaticName="LinkTitleNoMenu"
    Group="$Resources:core,Base_Columns;"
    ReadOnly="TRUE" Type="Computed"
    DisplayName="$Resources:core,Title;" Dir=""
    DisplayNameSrcField="Title"
    AuthoringInfo="$Resources:core,Linked_Item;">
    <FieldRefs>
      <FieldRef ID="{fa564e0f-0c70-4ab9-b863-0177e6ddd247}"
                Name="Title" />
      <FieldRef ID="{9d30f126-ba48-446b-b8f9-83745f322ebe}"
                Name="LinkFileNameNoMenu" />
    </FieldRefs>
    <DisplayPattern>
      <IfEqual>
        <Expr1>
          <LookupColumn Name="FSObjType" />
        </Expr1>
        <Expr2>1</Expr2>
        <Then>
          <Field Name="LinkFilenameNoMenu" />
        </Then>
        <Else>
          <HTML><![CDATA[<a onfocus="OnLink(this)" href="]]></HTML>
          <URL />
          <HTML><![CDATA[" onclick="GoToLink(this);
            return false;" target="_self">]]></HTML>
          <Column HTMLEncode="TRUE" Name="Title"
                  Default="$Resources:core,NoTitle;" />
          <HTML><![CDATA[</a>]]></HTML>
          <IfNew>
            <HTML>
```

www.it-ebooks.info

```
<![CDATA[<img src=
"/_layouts/[%=
System.Threading.Thread.CurrentThread.CurrentUICulture.LCID
%]/images/new.gif" alt="]]>
                </HTML>
                <HTML>$Resources:core,new_gif_alttext;</HTML>
                <HTML><![CDATA[" class="ms-newgif" />]]></HTML>
              </IfNew>
            </Else>
          </IfEqual>
        </DisplayPattern>
      </Field>
    </Fields>
    <Views>
      <View BaseViewID="0" Type="HTML" MobileView="TRUE"
            TabularView="FALSE">
        <Toolbar Type="Standard" />
        <XslLink Default="TRUE">main.xsl</XslLink>
        <RowLimit Paged="TRUE">30</RowLimit>
        <ViewFields>
          <FieldRef Name="LinkTitleNoMenu">
          </FieldRef>
        </ViewFields>
        <Query>
          <OrderBy>
            <FieldRef Name="Modified" Ascending="FALSE">
            </FieldRef>
          </OrderBy>
        </Query>
        <ParameterBindings>
          <ParameterBinding Name="AddNewAnnouncement"
            Location="Resource(wss,addnewitem)" />
          <ParameterBinding Name="NoAnnouncements"
            Location="Resource(wss,noXinviewofY_LIST)" />
          <ParameterBinding Name="NoAnnouncementsHowTo"
            Location="Resource(wss,noXinviewofY_ONET_HOME)" />
        </ParameterBindings>
      </View>
      <View BaseViewID="1" Type="HTML" WebPartZoneID="Main"
        DisplayName="$Resources:core,objectiv_schema_mwsidcamlidC24;"
        DefaultView="TRUE" MobileView="TRUE" MobileDefaultView="TRUE"
        SetupPath="pages\viewpage.aspx" ImageUrl="/_layouts/images/generic.png"
        Url="AllItems.aspx">
        <Toolbar Type="Standard" />
        <XslLink Default="TRUE">main.xsl</XslLink>
        <RowLimit Paged="TRUE">30</RowLimit>
        <ViewFields>
          <FieldRef Name="Attachments">
          </FieldRef>
          <FieldRef Name="LinkTitle">
          </FieldRef>
        </ViewFields>
        <Query>
          <OrderBy>
```

continues

www.it-ebooks.info

LISTING 5-1 *(continued)*

```
            <FieldRef Name="ID">
            </FieldRef>
          </OrderBy>
        </Query>
        <ParameterBindings>
          <ParameterBinding Name="NoAnnouncements"
            Location="Resource(wss,noXinviewofY_LIST)" />
          <ParameterBinding Name="NoAnnouncementsHowTo"
            Location="Resource(wss,noXinviewofY_DEFAULT)" />
        </ParameterBindings>
      </View>
    </Views>
    <Forms>
      <Form Type="DisplayForm" Url="DispForm.aspx"
            SetupPath="pages\form.aspx" WebPartZoneID="Main" />
      <Form Type="EditForm" Url="EditForm.aspx"
            SetupPath="pages\form.aspx" WebPartZoneID="Main" />
      <Form Type="NewForm" Url="NewForm.aspx"
            SetupPath="pages\form.aspx" WebPartZoneID="Main" />
    </Forms>
  </MetaData>
</List>
```

Wrox.Lists\Wrox.Lists\Lists\EmployeesListDefinition

The first thing you probably notice about the list schema is that it is extremely long; the second thing you might notice is that parts of it with the `Field` and `FieldRef` elements look strangely identical to the content type definition you created earlier. One of the slightly frustrating aspects of list definitions is how much duplication exists. Why is the duplication required?

SharePoint supports content type inheritance but also supports the breaking of inheritance. Whenever you associate a content type or a column with a SharePoint list or library, a copy of that content type or column is made. This enables you to do things such as renaming or deleting columns without affecting other lists and libraries that use the same parent content type. So essentially, all this duplication specifies the copy of the content type and fields that you want to be made. It is completely possible to radically change the local version of the content type by modifying the schema file.

Luckily, Visual Studio has done most of the heavy lifting for you and already pulled in the details about the content type and inserted them into the `schema` file for you. Otherwise, you would need to manually track down field IDs and attributes, which is no fun. When you need to create a list definition without the assistance of Visual Studio or an existing content type, it is often helpful to use an existing list definition as a starting point. You can find the OOTB list definitions within the 14 hive `TEMPLATE\FEATURES` directory; most have friendly names such as `CustomList` or `EventsList`. These list definitions are also excellent sources of information when you try to figure out how to do something within your own list definition.

www.it-ebooks.info

Because you took advantage of Visual Studio's wizard, the changes you need to make to the `schema` file are relatively minor and consist only of adding the Employee Details content type's fields to the default view used by the list. Follow these steps to customize the view:

1. Navigate down through the `schema.xml` file, and find the Views element toward the bottom of the file.
2. You see two `View` elements within the `Views` element. The first is just a template for the list but is not actually a named view. The second view is the `All Items` view and is the view that you need to modify.

   Within the second `View` element (`BaseViewId 1`), find the `ViewFields` element, which should look something like this xml snippet:

```
<ViewFields>
  <FieldRef Name="Attachments">
  </FieldRef>
  <FieldRef Name="LinkTitle">
  </FieldRef>
</ViewFields>
```

3. Add the following `FieldRefs` after the last `FieldRef` element:

```
  <FieldRef Name="WroxFirstName" />
  <FieldRef Name="WroxLastName" />
  <FieldRef Name="WroxEmailAddress" />
  <FieldRef Name="WroxPhoneNumber" />
  <FieldRef Name="WroxHireDate" />
  <FieldRef Name="WroxSSN" />
```

4. Save the file.

Before testing the list definition, you should modify the `ListInstance` that Visual Studio created as part of the List Definition Wizard.

1. Right-click `ListInstance1` in the Solution Explorer, and select Rename.
2. Name the file: **EmployeesListInstance**.
3. Expand the EmployeesListInstance file, and open the `Elements.xml` file beneath it.
4. Change the title to **Employees**.
5. Change the URL to **Lists/Employees**.
6. Change the description to something more appropriate.
7. Save the file.

Now that the list instance has a decent name, deploy the solution and navigate to your portal using a web browser. If everything works as expected, you should see an Employees list on the left side of the screen in the Quick Launch, as shown in Figure 5-16.

www.it-ebooks.info

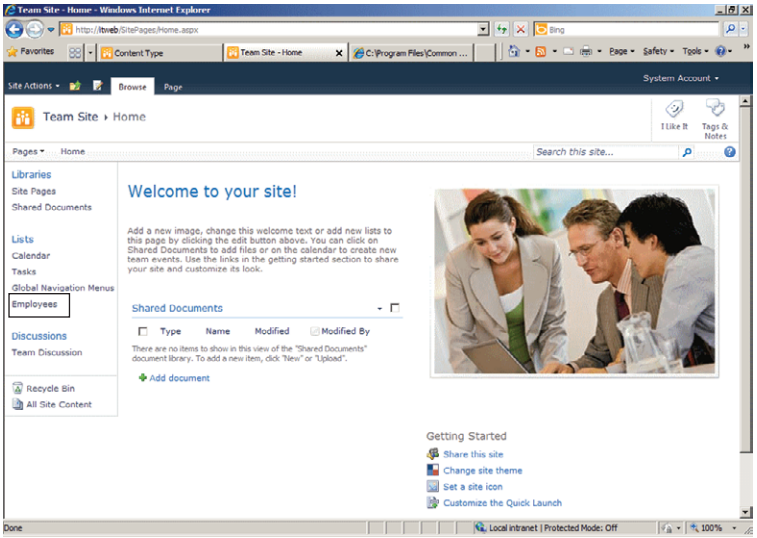


FIGURE 5-16: Employees list in Quick Launch

When you navigate to the list, it should be empty but show all the columns specified in the `schema.xml` file. Add an Employee to the list, and you should see something similar to Figure 5-17.

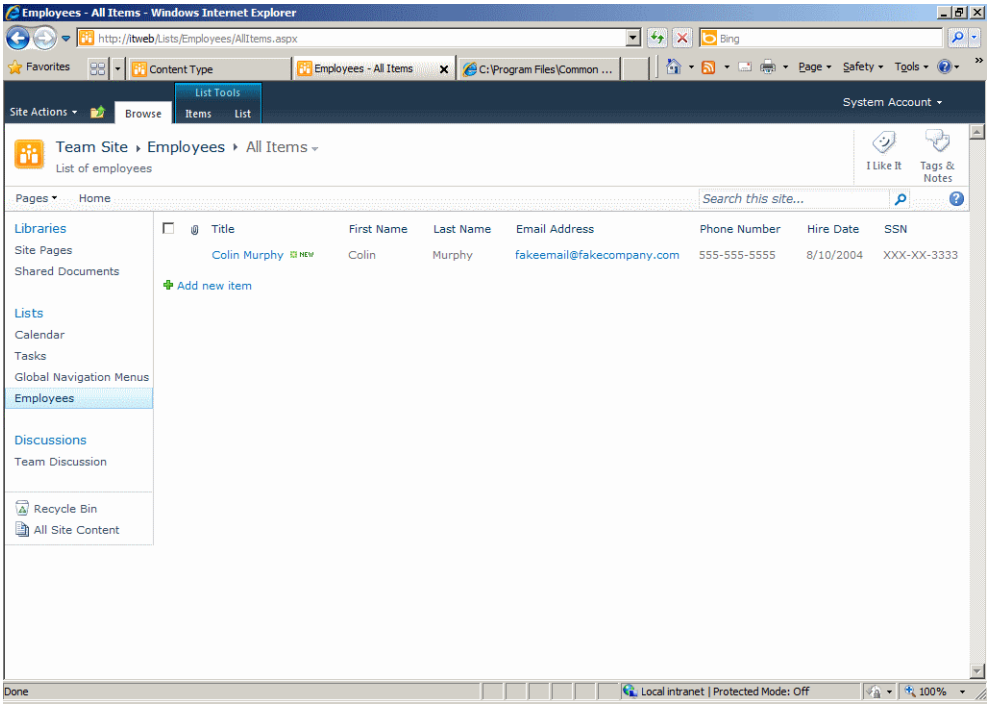


FIGURE 5-17: Employees list view

www.it-ebooks.info

SUMMARY

This chapter walked you through the creation of a custom field type to meet a rather common requirement: restricting the visibility of a column in a list. You created a Social Security number field type, which masks the SSN to prevent users who cannot edit the form from seeing the full SSN. It then showed how you can use content types to preconfigure collections of columns and then demonstrated how to automate the association of a content type with a SharePoint list.

This is just scratching the surface of what you can achieve with custom field types. More advanced solutions, which support item level security, complex UIs, and advanced business logic, are just waiting for you to implement. Even in the simple employees list created for this example, two columns (Email Address and Phone Number) could benefit from having a custom field type, which enforces valid data and formatting.

Another common scenario for the use of list definitions is when you use lists to store data used by other components of a solution. Chapter 11, "Building a Custom Global Navigation Solution," for example, uses a fairly complex list definition to create the repository where users maintain the navigation links used by the global navigation menu. If you want a more advanced example of a list definition, Chapter 11 is a good place to start.

www.it-ebooks.info

www.it-ebooks.info

6

Workflow

Every organization has business processes that must be adhered to in order to efficiently and effectively operate. Unfortunately, in many cases these processes involve a great deal of human interaction and responsibility. With this human involvement come inconsistencies and errors. To combat these issues and improve overall user collaboration, SharePoint 2010 provides organizations with the ability to create workflow solutions. By leveraging automated business processes, users no longer need to get caught up with understanding the process, but rather can simply respond to notifications and tasks assigned to them. With SharePoint handling the steps of the process, workflows run with unparalleled consistency. In this chapter you will learn about the various workflow concepts and tools and how these can be used to build SharePoint 2010 workflow solutions.

WORKFLOW INTRODUCTION

Workflows are a fancy name for automated business processes. By having a defined automated process, work and productivity can run with unfailing reliability. Users, no matter how well trained and attentive, make mistakes. Many of these mistakes might not seem that important, but in some cases they can be disastrous. Imagine your organization working on a multi-million-dollar deal, but somebody forgot to sign a document or get something turned in, and your team has already submitted the proposal. Now all the time and effort you put into a proposal has all been for nothing as the due date has come and gone, and your proposal is incomplete. Heads will roll…. In situations like this it would be nice if there were a way to track and monitor items for a given activity rather than depending on email and Outlook tasks. With workflows, the server does all the heavy lifting. With SharePoint administering the process your users can go back to doing their assignments. From a management perspective, each step of the process can be monitored, and solutions can be created to present workflow items in a way that quickly provides the overseeing users the necessary information to identify which, if any, activities are off track. This, combined with the variety of solution options, makes working with workflow solutions a flexible and valuable feature of SharePoint 2010.

www.it-ebooks.info

GETTING STARTED

Before you start building workflow solutions, it is important to not get ahead of yourself. All too often, users see how easy it is to build solutions in SharePoint 2010. The problem with that is there are many ways to do the same thing, but there is typically a recommended way. If you haven't built workflow solutions before, then you won't have the experience to know which method to choose. Be sure to follow the information in the next few sections. The goal is to provide you with some initial concepts to define, prior to creating a workflow solution. Remember, the more time you spend during the planning and design of your solution, the better off you will be. Cutting corners during this critical piece of the solution may cost you more time and resources in the end.

Defining the Process

The first, and arguably the most important step, in creating a SharePoint 2010 workflow solution is to map out the requirements. No matter how simple or complex the desired solution may be, a common mistake that most users make is not accounting for every possible scenario that might occur during the course of the workflow. This is why it is absolutely paramount that the options and paths are all clearly defined. You can do this in a number of ways: verbal conversations, emailed specifications, documents that display basic diagrams, or a tool such as Microsoft Office Visio 2010 that can help create detailed flowcharts. Best practices recommend that everything should be documented, so make sure that you have a hard copy of everything required. This not only helps during the creation of the solution, but it also acts as a reference by distinctly outlining the details so that everyone involved is on the same page.

Microsoft Office Visio 2010 Premium

Visio 2010 is an option for creating detailed reference material, but a couple more distinct advantages are worth mentioning. The first is improved user-to-user communication. One of the main areas in which solutions such as this falter is that the requesting party (typically business users) and the developing party (typically IT users) fail to communicate effectively. This can create numerous problems during the solution's creation. Visio 2010 provides a medium in which business users can create the process details/diagram file; then the more technical workflow architects can take that file and easily produce a working solution. This is possible because of the next advantage: that Visio 2010 comes with a template that specifically incorporates workflow actions or conditions. When business users use these actions/conditions in the workflow flowchart, these specific items are recognized by SharePoint Designer (SPD) 2010 and Visual Studio 2010 when the Visio 2010 file is imported. By having this level of integration, it drastically improves the ability of the users creating the solution to build the requested process. This integration is available only with Microsoft Office Visio 2010 Premium. You can still create diagrams with other versions of Visio, but they will not have the built-in template, actions, or conditions. Only simple diagrams are possible.

Sample Solution

Contoso, a fictitious company, wants to streamline one of its business processes and has decided to build a solution to use with its SharePoint implementation. The workflow that needs to be built is

www.it-ebooks.info

for a multistep approval process. A user will submit a document to a library in SharePoint; then it will go through a series of approvals. At any point in the process should the assigned reviewer reject the document, it should notify the submitting user and then restart the process after the user makes changes and resubmits. The reviewing process runs in a series. Figure 6-1 shows a basic Microsoft Office Word 2010 diagram for this process.

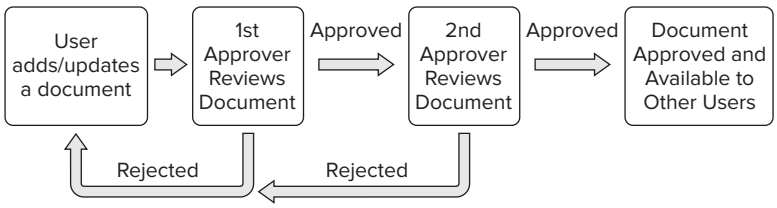


FIGURE 6-1: Sample flowchart for approval process

Obviously, you could add much more detail, but Figure 6-1 shows the minimum amount of information you need to include. For additional details, you could include tasks and notifications that you want to be created and any information associated with those items. You could also include the specific users that have been selected as the 1st and 2nd approvers. The list could go on and on.

Types of Workflow Associations

Part of laying out the details for a workflow request is knowing where to set up the workflow. Within SharePoint 2010, you have three options:

- **Lists and Libraries:** When creating a workflow association with a list or library, the workflow is available to run on every item in the given list or library. These associations are made from SharePoint sites via the Workflow Settings page or from SharePoint Designer 2010. This type of association has specific restrictions on the starting trigger for the workflow.
- **Content Types:** When you create a workflow association for a content type, the workflow is available to run on every item within a list or library that has enabled the corresponding content type. This type of association can be made from the SharePoint sites via the Workflow Settings page of the content type or from SharePoint Designer 2010. This type of association has specific restrictions on the starting trigger for the workflow.
- **Sites:** When you create a workflow association for a site, it is a workflow that must be manually started, and it is not dependent upon list or library items. These types of associations can be made only from SharePoint Designer 2010 and Visual Studio 2010.

The majority of workflow associations fit into the first two options. The main thing to consider here is whether you reuse the solution. If a high probability exists that the solution is needed in another site in your site collection, associate a workflow with a given content type. You don't have to do this, but it can make your life easier in the long run.

www.it-ebooks.info

Starting Workflows

No matter what the solution entails, you can trigger the start of a list workflow in three ways: Manually, upon the list item creation, and upon the list item being edited. It may seem limiting, but these three options cover almost any situation in SharePoint 2010 in which a list workflow needs to be used.

The options to manually trigger the start a workflow vary depending upon the type of workflow associated that exists. For list and library items, which include content types, there is an option in the ribbon and on the Edit Control Block (ECB) menu for each item for Workflows. By selecting this option, you go to a page where you can start any available workflows. Select one of the available workflows, and click Start. For Site Workflows this is the only available option, but for list and library items, manually starting a workflow process isn't ideal because the whole idea is to eliminate as much human involvement as possible. However, in some situations this option is necessary.

For example, for many of the SharePoint workflow templates, you are required to predefine an assigned user for the tasks that will be created. But under some circumstances, this cannot be predefined. Instead, the submitter must set the assigned user. For this situation you can create a workflow that is manually started and leave the assigned user fields blank. When submitters create or upload an item and manually start the workflow process, they are prompted to specify an assigned user.

The next option for starting a workflow is to have the workflow run when an item is created. This option is perfect for continuing with the automated thought process. When an item is created or uploaded by a user, the workflow automatically runs. This guarantees that every item has a workflow instance associated with it. It also requires that the process is carefully defined so that if exceptions occur and if some items won't require approval, those items are taken into account when creating the solution.

To get a better understanding of this, consider the following example. You have check requests submitted by users. For items requesting a check for under $25, no approval is needed. If the item is requesting more than $25, the solution requires the workflow to run. In this instance, using a SharePoint template does not suffice, but you can build this solution with SPD 2010. What you must do is have a column that contains the requesting amount and then have the workflow target that column's value. Because the workflow runs on every item, set up two branches: One with a condition that targets items where the value is under $25, and another that targets items equal to or greater than $25. For the actions, the under $25 would simply stop the workflow, and the other set of actions would run the approval process.

Selecting Your Solution Method

After you gather all the solution requirements and the process has been properly documented, it is time to create your solution. For SharePoint workflows, you have a few options to select from: SharePoint native options, SharePoint Designer 2010, Visual Studio 2010, and third-party tools. For selecting a tool to create your solution, consider several things and weigh several pros and cons. Although the building of solutions won't be covered now, the next few sections provide a high-level overview of some of the possibilities.

www.it-ebooks.info

SharePoint 2010 Workflow Templates

The native workflow features that come with SharePoint 2010 consist of various workflow templates that you can configure to run according to your business's requirements. Capable of running in serial or parallel, these templates are task-based solutions that enable users to track the status of assigned items. In addition to the created tasks, a workflow history list stores logged events and can be used to trace steps the workflow completed. These out-of-the-box templates are a great starting point for organizations; however, the lack of incorporating conditional requirements is a huge drawback.

SharePoint Designer 2010

SharePoint Designer (SPD) 2010 is the preferred web editing tool for SharePoint 2010 sites. One of its more popular uses is to create declarative workflows. A declarative workflow is based on specified conditions and actions and involves no code writing. This defining characteristic is where you start to see the additional value when compared with the SharePoint out-of-the-box template counterparts. The available conditions enable workflow architects to compare a wide range of information related to target items. These conditions specify when certain actions should run. One of the limitations previously mentioned regarding SharePoint 2010 out-of-the-box templates is that they cannot dynamically change their settings based on metadata values, but with SPD 2010 workflows, you can easily configure a condition so that a subsequent action runs only when a metadata field or workflow data has a specific value. If you have several metadata field value options, you just need to create a workflow branch (a single set of conditions followed by actions that run as a result of the conditions) for each option. When you look at the available list of actions, it is quickly apparent that you can use SPD 2010 to create a variety of complex workflow solutions. An added bonus is that the SPD 2010 application is a free download from Microsoft.

With such a competitive price tag, more features, and no code needed, SPD 2010 is an appealing tool for building workflow solutions. It isn't all perfect, though. Similar to the SharePoint out-of-the-box templates, SPD 2010 workflows can be associated with specific lists and libraries and content types, but there is one key difference in their reusable functionality. SPD 2010 reusable workflow associations can target only content types; they are not templates that can be deployed and enabled wherever you choose. This one drawback is sometimes the reason organizations choose to go with the next option, custom development solutions with VS 2010. The other disadvantage is that SPD 2010 workflows are only as good as the available conditions and actions. This has two connotations: A desired functionality may be missing, and even though there is an available condition or action, it may not be the best option. SPD 2010 should be used only with SharePoint 2010 sites; do not use it with SharePoint 2007 sites.

Visual Studio 2010

The third available option for creating workflow solutions is to have a developer create the solution using Visual Studio (VS) 2010. The most flexible and powerful tool, custom-developed solutions can create the most complex workflow solutions a business user can dream up. The two main drawbacks of this option are cost and time. Developers' time is sought after, so to get your organization to dedicate a developer resource for a business user's solution is rare. But if you go with a contractor, you can find a hefty price tag associated with any custom development solution. In the end, the one thing to think about is, "Is this cost-effective?" If the need outweighs the cost, then it is a safe commitment to pursue; but if it isn't, tread softly. Few things are as detrimental to a career as wasteful spending.

www.it-ebooks.info

Third-Party Solutions

With SharePoint gaining popularity and market share, and with the latest release exploding onto the scene, vendors and certified Microsoft partners have been doing their best to keep up with the demand for custom solutions. The area of workflow is no different. Like with any custom solution, the functionality can greatly outweigh the SharePoint out-of-the-box and SPD 2010 capabilities, but be prepared for the cost. Workflow solutions tend to be more complex, so their higher price point is fitting. The same advice for other products applies to workflow tools: Test, test, and test some more. Make sure that you get the product that fits your needs the best and be sure to get some support time. These products, if cost-beneficial, can be a great addition to a SharePoint implementation.

SHAREPOINT 2010 NATIVE TOOLS

Templates are a common theme in SharePoint 2010. Microsoft has done a great job to get people started with some useful options, and the native workflow tools are no different. The following sections cover the various available templates and how you can use them to create some amazingly simple solutions.

SharePoint Workflow Tools

Before covering templates, you need to view how and where workflow options are available to users in SharePoint 2010 sites. These options can vary based on the type of workflow associated with a list, library, or site. These focus more on general workflow information and web pages where you can create workflow associations. Sections that cover the workflow templates and options appear later in the chapter.

Site Workflows

To keep things simple, you can access site workflows in one of two ways. The first option is from the View All Site Content page. You can access this page from the Site Actions menu by clicking Site Actions ➪ View All Site Content. When the View All Site Content page displays, the Site Workflows option is available close to the top of the page, as shown in Figure 6-2.

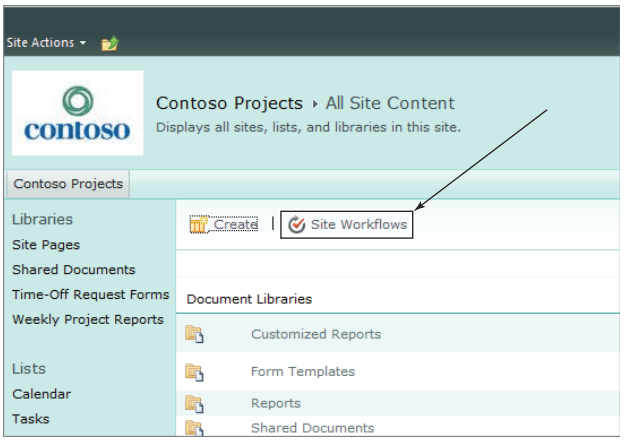


FIGURE 6-2: The Site Workflows button is available from the All Site Content page

www.it-ebooks.info

On the subsequent page, as shown in Figure 6-3, the options for starting an existing site workflow are available. The page also displays any currently running workflow instances.

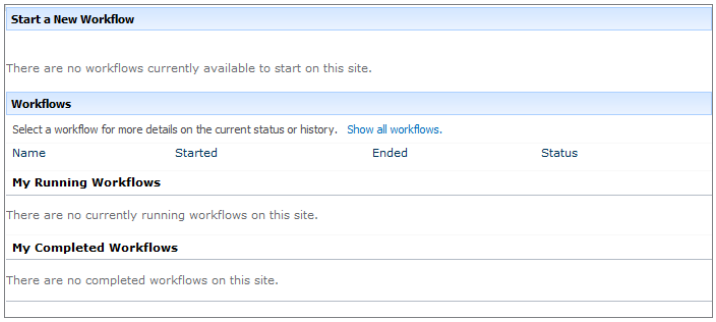


FIGURE 6-3: Site workflow options

The other option for accessing site workflows is on the Site Settings page. This page is also accessible from the Site Actions menu by clicking Site Actions ➪ Site Settings. On the Site Settings page, under the Site Administration header, there is a link for Workflow Settings. The page displayed shows any currently running instances and also provides a link for adding a new workflow.

Viewing All Workflows for a Site

A new feature that provides a lot of value is the ability to view all existing workflow associations and all available workflow options for a given site. As a site owner, a user can click Site Actions ➪ Site Settings and on the Site Settings page, click Workflows. This takes the site owner to a page to view every workflow.

Figure 6-4 shows several instances currently running, but also several other workflows display that do not have any values in the Associations column. These are available workflows for the site but have not been associated with the site, list, or library in the site.

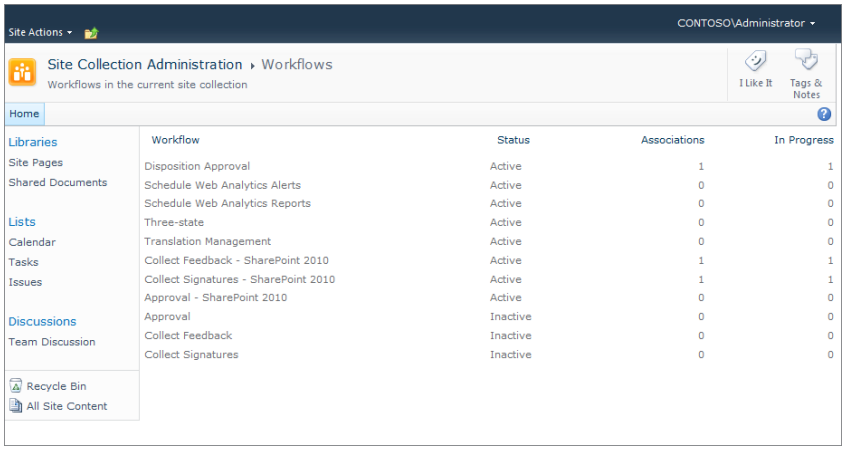


FIGURE 6-4: Workflow templates that are available via the browser for the current site collection

www.it-ebooks.info

List and Library Options

When working with workflows in lists and libraries, the initial options available to users are available via the Ribbon and the ECB menu. The Ribbon section that contains the workflow options can vary with the list and library that is being worked with, but the Settings section should always contain four options: Workflow Settings, Add a Workflow, Create a Workflow in SPD, and Create a Reusable Workflow in SPD. These options are all related to workflow settings that are set for the entire list or library. The same Workflow Settings link available in the Settings section of the Ribbon is also accessible from the List Settings or Library Settings page. The ECB menu has one option for Workflows. This option results in a display of workflow templates that can be manually started for the item and all workflow instances currently running on the item.

SharePoint Workflow Templates

The out-of-the-box tools that you can use via the browser consist of workflow templates enabled as features. This enables administrators to limit their availability if they so choose. The available template options users see depends on the installed license version of SharePoint 2010. The following sections discuss the available templates for the specified installation type, as well as a brief description of the workflow.

SharePoint Foundation 2010 Templates

For SharePoint Foundation 2010 users, there is only one out-of-the-box workflow template that will be available to you: Three State Workflow. Ideally for Issue Tracking lists, this template is used to track a given list item through a series of states or phases. To first set up a three-state workflow, you must create a choice column that contains three state values.

For example: You have a list to track collaborative issues. You have a column to track the status of each item, and it needs to have three values: Initiated, Pending, and Complete. You would set up this column because it has three states and then use it as the basis for your workflow.

A task is created for the first state, and when this task has been completed, the workflow progresses to the next state. This repeats for the second state, and when the task for the second state has completed, the third and final state of the process has been reached, and the workflow is completed. This is a basic workflow and has few configuration options. This native template is not editable via SPD 2010.

SharePoint Server 2010 Templates

The following six templates are available for both the Standard and Enterprise license versions of SharePoint Server 2010.

1. **Three State Workflow:** This is the same template available with SharePoint Foundation 2010.
2. **Approval:** This template is used for serial, parallel, and combined serial/parallel approval processes that have an assigned approver for each stage. This template also provides the author a wide variety of configuration options that can affect the approval process. The user that triggered the start of the workflow is notified when the process begins, and if the item is ever rejected, a change is requested, and then the process completes. For the assigned approvers,

www.it-ebooks.info

approval tasks are created for each user and have the following options: Approve, Reject, Cancel, Request Change, and Reassign Task.

- **Approve:** Approves this stage and the workflow continues on to the next step. If there is only one approver, this approves the item, completes the workflow, and notifies the user that the item has been approved.
- **Reject:** Rejects the item, and depending upon the configuration of the workflow, this could have various implications: The workflow could stop (configuration option); if the workflow is in parallel, the process waits until all users have responded; or the workflow could continue on to the next user, if necessary, for completion.
- **Cancel:** Exits out of the task without making any change to the task.
- **Request Change:** Creates a new task for the submitting user to make a change typically specified in the Comments box of the item filled out by the approver. The submitting user makes the wanted changes to the item and then completes the task. The workflow then reroutes the workflow back to the approver that requested the change. The same options are presented, and the approver can Approve, Reject, Cancel, Request Change, or Reassign Task. This option is a nice way to give a submitting user the opportunity to make a quick change that can make the submitted item approvable.
- **Reassign Task:** Enables the approver to reassign the task ownership to another user. This might be necessary depending upon the submitted item. Another time that this might be a usable option is if an approver is on vacation or unavailable; another user, with the appropriate access, could reassign the task to a different available user.

  For example: Your organization has users upload expense report Word documents to a SharePoint library, but you want to have these items approved by a user that is a supervisor and then also by a member in accounting. You could create a document library for users to upload their documents. When the library is created, set up an approval workflow to run when items are created, and create two stages to run in series. Enter the user values for the assigned approvers.

  You can see that this template seems to be a parent template that some of the other native workflow templates are based on. This provides additional functionalities and configuration options that won't be available in subsequent templates.

3. **Collect Signatures:** Similar to the Approval workflow, but by far the easiest to configure, this template enables users to digitally "sign" a document. This is not to be mistaken with Digital Signatures that require a user to enter a pin or an identification number; instead this workflow simply creates a task in which a user can select from the following options:

   - **Sign:** By selecting the Sign option, SharePoint captures the user information for the user that clicks this button (it does this for all workflow templates) and then stores it in a list for reference. This option is meant for approval/signature options that do not require a physical signature or a Digital Signature. Although this workflow can be configured in series, parallel, and/or both and can also have multiple stages of assigned tasks, the configuration options are more limited than that of the Approval template.
   - **Other options:** The Cancel, Request Change, and Reassign Task options are discussed in the previous Approval template section.

www.it-ebooks.info

4. **Collect Feedback:** A bit less complex than the Approval template, this option enables users to run a workflow that assigns a task to a specified user. When the assigned users open the task item, they are prompted to provide comments and click one of the following options: Send Feedback, Cancel, Request Change, or Reassign Task.
   - **Send Feedback:** This option submits the comments specified in the Comments section of the workflow task item and then completes the task.
   - **Other options:** The Cancel, Request Change, and Reassign Task options are discussed in the previous Approval template section.
5. **Disposition Approval:** Used primarily with, but not exclusively with, records management, the Disposition Approval template is a workflow process that helps users control when documents expire and should be deleted. By default, the workflow task that is created requests the assigned user to approve or reject the deletion request. However, retention settings can be configured. The settings that determine this retention rate are related to the Information Management policies. These can be configured at the document library level and can also be configured for an entire site collection via the Site Collection Administration options. (You must be a site collection administrator to view these options.)
6. **Publishing Approval:** This specific approval process is set up for the Publishing Site with Workflow template. Simply put, it is an approval workflow automatically set up on the default Pages library and runs the approval workflow process for pages created in this library. From an end user standpoint, it runs the exact same way the Approval template does. The main thing to remember here is that the workflow will be automatically set up for you. For this to run, the corresponding site collection feature must be enabled.
7. **Translation Management:** Available only in a Translation Management library, this workflow helps manage the manual translation of documents by assigning tasks to users when they have been assigned a translation task.

These templates have been created as simple solutions to commonly used processes. They are meant to be basic out-of-the-box solution options. Each template has configurations and settings that can be set to fit the necessary requirements for a wanted solution, but there are some limitations. For example, an Approval workflow can start upon the creation of the item, but it can have only prespecified, static values for the approving users. If the workflow needs to be dynamic in that the approving users change based on metadata field values, the native Approval template does not work. As a general rule of thumb, users should not expect the out-of-the-box templates to provide the full functionality required from a requested solution, but they can get you started. After that, it becomes a cost/benefit decision as to whether the additional functionality is worth having someone create the solution using one of the custom workflow applications: SPD 2010 or VS 2010. Although the drawbacks may seem substantial, a few advantages exist. The obvious advantage that the native application tools have is that they are built in to SharePoint 2010 sites. As such these ready-to-go templates can be easily and quickly deployed. Another benefit is that these solutions, because they are templates deployed as features, are reusable solutions. As long as the necessary features are activated, they can be deployed to lists or libraries, and they can be associated with content types. By associating them with content types, the workflow is available to run on items where the content type is used. The last plus mentioned is that if a template does not provide all the requested functionality, the template can be edited within SPD 2010.

www.it-ebooks.info

Tasks and Notifications

Two key components of the out-of-the-box workflow template process are available: Tasks and Notifications. A basic outline of the way these workflows progress is detailed here:

1. When the workflow starts, a notification is created to inform the triggering user that the process has begun.
2. A task generates for specified users. The workflow remains paused until this task has a status of Completed.
3. If the workflow tasks list has been configured to send an email upon task assignment, a notification is sent out to the user that has been assigned the task from step 2.
4. Based on the template and the number of steps, an action is requested for the assigned user. When this action completes, the workflow continues based on the configuration. This continuation might be to complete the workflow, or an additional action may be required.

It may seem simple, but this overall notification ➪ task ➪ notification ➪ user action is the main driving force behind the out-of-the-box templates. What this concept enables users to do is monitor the tasks for a given workflow instance so that progress can be tracked. However, one noticeable drawback exists. Having a list or library where the original item resides and then having tasks created in another list forces users to navigate between two different lists. This can be an undesirable solution for many organizations. Unfortunately, this is the way that SharePoint 2010 workflow templates run. To get processes to run in just one list, without having to have tasks based in a separate list, requires architects to leverage another solution option.

Set Up a Workflow Using a Default Template

The example exhibited in this section is based on the Approval template. Because this template utilizes the most configuration options available for the out-of-the-box templates, this is the ideal example to walk through. The site used for all procedures is a site collection based on the Team Site template. The Workflows and SharePoint Server Standard Site Collection Features site collection features have been activated. An Outlook account has been created to view the email notifications sent by SharePoint to assigned users.

1. Navigate to a document library.
2. At the top of the page in the Library Tools section, click the Library tab.
3. In the Settings section of the Ribbon, click the Workflow Settings drop-down menu, and then click Add a Workflow.
4. One the next page, you will be prompted to configure the first set of workflow options, as seen in Figure 6-5. In the Select a workflow template field, select Approval - SharePoint 2010.
5. Enter **Approval WF** for the name of the workflow.
6. In the Task List section, leave the default value of Tasks. This places task items in the default tasks list for the site. If the architects want workflow tasks to have their own list, they could create a new tasks list and use that one.

www.it-ebooks.info

7. In the History List section, again, leave the default value of Workflow History.
8. This chapter previously discussed the starting options for workflows. For this workflow, check the boxes for the Allow This Workflow to Be Manually Started by an Authenticated User with Edit Item Permissions and Start This Workflow When a New Item Is Created.

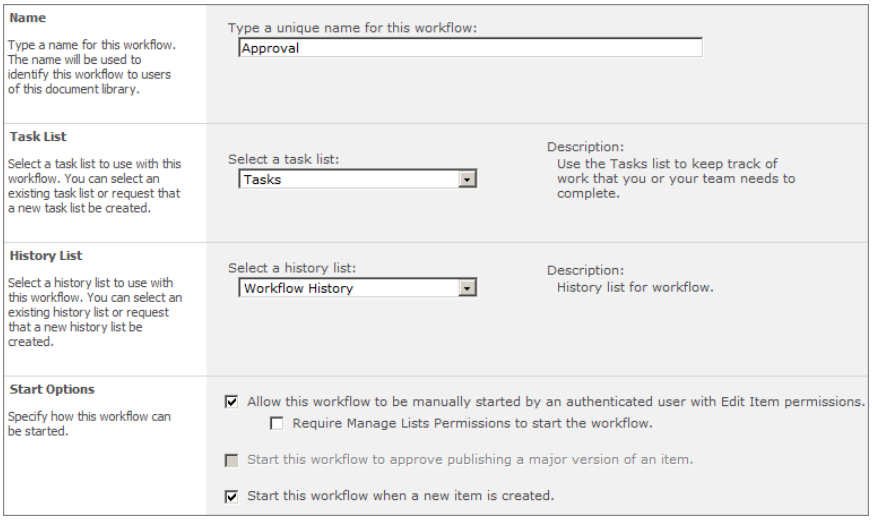


FIGURE 6-5: First set of workflow configurations

9. Click Next.
10. Before any configurations are made in the Approvers section, view all the available options:
    a. Multiple Stages Can Be Leveraged
    b. Each Stage Can Contain Multiple Users
    c. Stages Can Be Run in Parallel or Serial

    This may seem simple, but these options are extremely powerful. For this example, use the people picker field to enter a user value.
11. The next section is related to using groups. It is advised if you are going to use this option that you use a SharePoint security group and that it is set up to enable all users to see the group's membership. If you want to assign the task to a group of users, but have only one user "claim" the task and then complete it, then uncheck this option. If you want a task to be created for every user, leave this box checked. It runs just like a parallel step; all users must complete their task before the workflow moves to the next stage. As no groups should be specified according to step 10, this option has no bearing. Leave this option checked.
12. The request section can be valuable. This is the architect's opportunity to specify some information to assist the assigned user in identifying why the task was created and assigned. This makes completing the task easier for the assigned user. For this example, enter **You have been assigned a task to review the designated item and Approve or Reject it.**

www.it-ebooks.info

13. Do not enter a value for the next section. The obvious issue is that if you have a workflow that will exist for a while, you may not know or want to have all tasks be due by a specific date. However, there are some situations in which a short-term workflow solution can be used and this option specified.
14. The next two sections are more widely used. These options enable you to specify a duration value for tasks. Specify "2" and then "Day(s)." This takes the date the task is created, adds two days, and sets the task item Due Date field to the calculated value.
15. For this instance, no users need to be copied on any emails, so you can leave this blank.
16. The End on First Rejection section is important. When there are multiple users involved in the approval process, the majority of the time they do not want tasks to be assigned to them if the item has already been rejected, so make sure that this option is checked. If you assign tasks to a group, it may make more sense to not have the workflow stop on rejection. If you have five users in a group that review the item in parallel, a sixth reviewer may review the five initial tasks and then provide a final determination based on what option had the most support. For this example, check the box for this section.
17. The End on Document Change is another important option. If a user submits an item and it is routed for approval, reviewers don't want the submitting user to make changes. If this is allowed, how can they know what they are approving? Most of the time, this section is checked. For this example, check the box for this section.
18. The last configuration option enables the workflow to work with the Content Approval feature. If you would like the workflow to work with this feature and update the Approval Status field so that two fields are not needed, check this option. Because the library used in this example does not have Content Approval enabled, leave this section blank.
19. Click Save.

Running Through a Workflow Instance

An approval workflow was created on the Shared Documents library; now the process of running through a workflow instance can be discussed.

1. Add a document to the Shared Documents library.
2. Refresh your browser until you see a new column has been added to the library, Approval WF. This column has been added by SharePoint to display the workflow status. The column takes the name of the workflow, so keep that in mind when naming a workflow. A long name takes up a significant amount of real estate on the list or library.
3. You should see that two emails have been created: one to the user that submitted the item (see Figure 6-6), and one to the user that has been assigned the review task (see Figure 6-7).
4. Navigate to the Tasks list.
5. Open the task item that has been created. This item form should look like the item in Figure 6-8.

www.it-ebooks.info

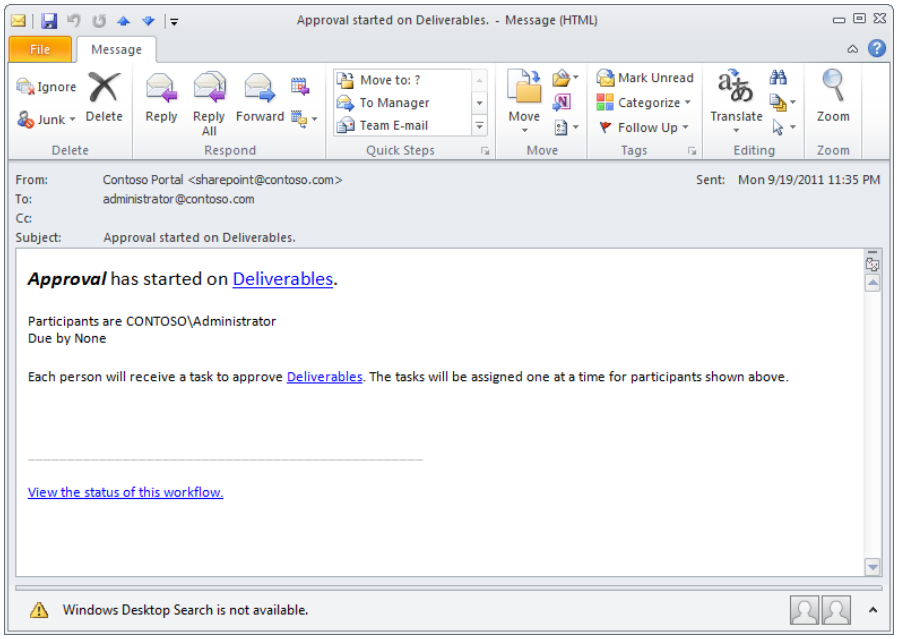


FIGURE 6-6: Workflow initialization notification sent to the user that created the item

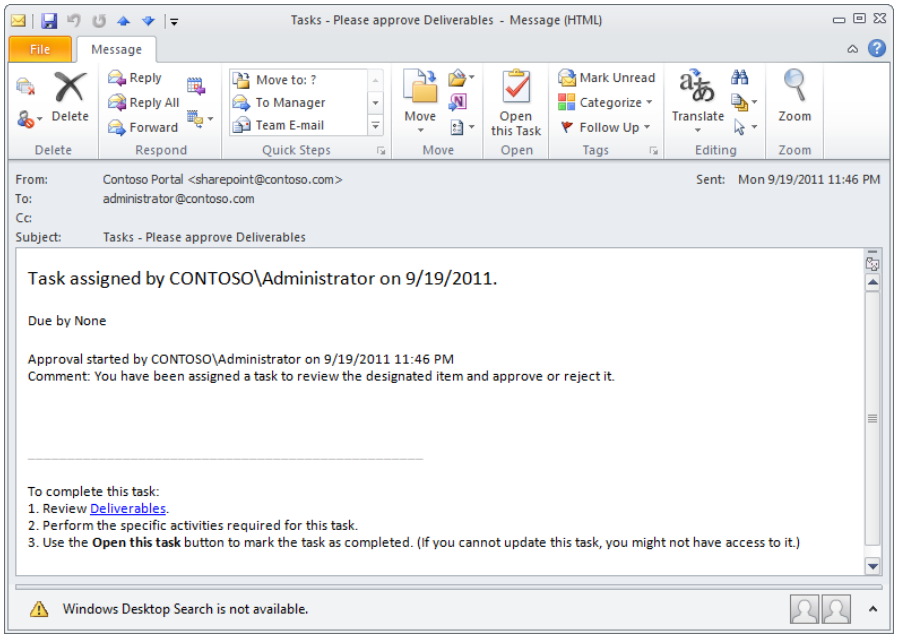


FIGURE 6-7: Workflow notification sent to reviewer of item

www.it-ebooks.info

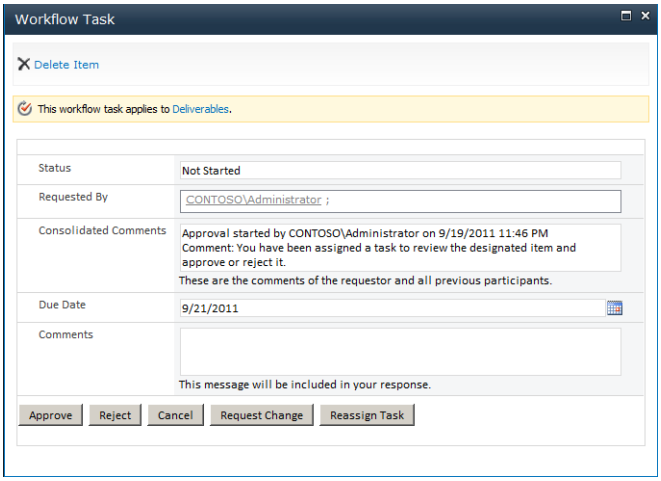


FIGURE 6-8: Approval workflow task item

6. Click Request Change.
7. In the New Request field, enter **Update paragraph 1 to account for updated numbers**.
8. Click Send. This creates a new task for the user that originally submitted the item. A new email has also been generated for the user that submitted the original item.
9. Open the new task item.
10. In the Comments field, enter **Change made, please review again**.
11. Click Send Response. Now that the request change task has been completed, a new approval task request has been created. A new email has also been generated to the reviewer.
12. Open the new task item.
13. Click Approve. A final email is sent out to the user that created the original item informing that the approval process has completed (see Figure 6-9).
14. Navigate back to the Shared Documents library. You see that the Approval WF column has a value of Approved.
15. Click the Approved link. This takes you to the workflow status page.

Workflow Status Page

The workflow status page is available to all users that have access to the content in the list or library. This page provides a great deal of information for users that need to review the workflow instance.

- **Terminate the workflow now:** If the workflow instance has not completed, this option will be available.
- **Initiator:** The user that triggered the workflow. If this were a manually started workflow, it records the user that started it. If the workflow is set to run when an item is created or edited, it specifies the value to be the user that created or edited the item.

www.it-ebooks.info

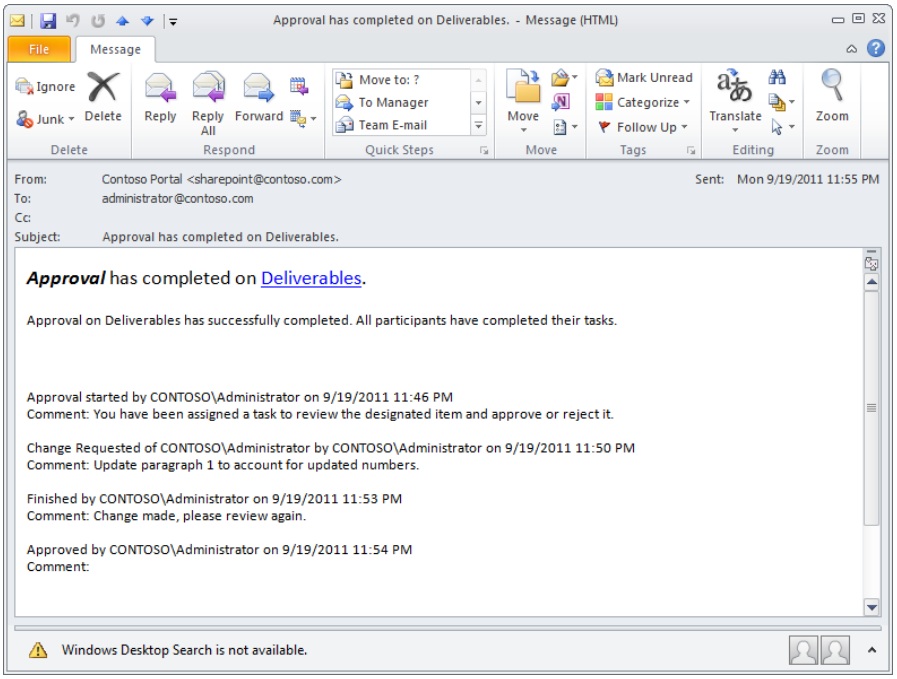


FIGURE 6-9: Workflow completion notification sent to the user that created the item

- **Started:** This one is self-explanatory.
- **Last run:** Depending upon the starting settings, the same workflow can run multiple instances on an item. Multiple instances of the same workflow can't simultaneously run on an item. If an instance runs, it must be stopped before another can run.
- **Document:** This shows the associated Title field for the associated document and serves as a link back to the document.
- **Status:** This should provide the same value as the workflow column.
- **Workflow Visualization:** If you have Silverlight installed on your client machine, you can view a flowchart of the workflow and the progress that has been made.
- **Tasks:** Here you can view all the tasks that have been generated as a result of the workflow.
- **Workflow History:** This is an actual list on the site that contains all logged entries made regarding workflows. Even though it is not accessible via the View All Site Content page, you can access it by navigating directly to the URL. For the Workflow History list, that URL will be `http:/…/lists/workflow%20history/`. The important feature to remember about the Workflow History list is that entries for completed workflows will be retained only for 60 days. After this, a timer job goes through and removes all items exceeding that retention policy. For some instances, it may be a requirement for the workflow history logged entries to be kept for a longer period of time. If this is the case, be sure to specify a new Workflow History list at the time the workflow is created.

www.it-ebooks.info

View Workflow Reports

The View Workflow Reports option is available from the workflow instance for a given item as well as on the Workflow Settings page for a given list. SharePoint can export Excel reports that contain information related to workflow duration and cancellation and errors. These reports are also visible from the Workflow Settings page for the list or library.

Editing an Existing Workflow

Despite best efforts, sometimes workflow configurations need to be adjusted. When this is the case, a few things need to be clarified. When a workflow is edited, it will not retroactively edit any existing workflow instances that are running. It affects only new workflow instances that are created after the change has been made and saved. Follow these same guidelines for editing custom SPD 2010 workflows.

1. Navigate to the Shared Documents library.
2. At the top of the page in the Library Tools section, click the Library tab.
3. In the Settings section of the Ribbon, click the Workflow Settings button. You see that your Approval WF workflow displays. Click the link for the workflow.
4. Change the starting options so that the workflow can be started only manually.
5. Click Next.
6. Click Save.
7. Navigate back to the Shared Documents home page.
8. Add a new document. Because the workflow was changed to only start manually, the workflow does not automatically start on item creation.
9. Highlight the new item so that it is selected. The Documents Ribbon should display.
10. In the Workflows section of the Ribbon, click Workflows.
11. Click Approval WF.
12. You notice that a familiar configuration page appears and that when a workflow starts manually, the triggering user can make configuration settings for how the workflow runs. Scroll down and click Start.

Removing an Existing Workflow

When a workflow association is no longer needed, you can remove it via the Workflow Settings page for the list or library. Whenever a workflow needs to be removed, make sure that there are no active instances of the workflow. Doing so can keep an open instance of the workflow on the server. Follow these same guidelines for editing custom SPD 2010 workflows.

1. Navigate to the Workflow Settings page of the Shared Documents library.
2. Click Remove.
3. You see that there is still an active instance, as shown in Figure 6-10.

www.it-ebooks.info

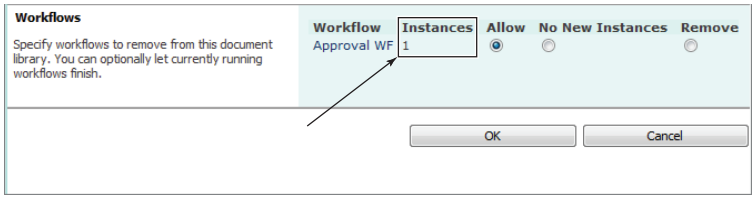


FIGURE 6-10: Existing workflows associated with the correct list

4. Click Cancel.
5. Use one of the navigation options to return to the default Shared Documents page.
6. Highlight the item created at the end of the previous procedure. The Documents Ribbon should display.
7. In the Workflows section of the Ribbon, click Workflows.
8. Click Approval WF. This is the same workflow status page viewed earlier, but this time it is for an item that currently has an active instance running. Scroll down past the Workflow Visualization section. You see three links that were not visible before.
9. Click Terminate This Workflow.
10. When a pop-up appears, click OK. This pop-up informs you that all associated tasks will be deleted. This is exactly what you want.
11. Navigate back to the Workflow Settings page.
12. Click Remove a workflow.
13. You see that there are no active instances. Click the radio button for Remove.
14. Click OK.

Customizing SharePoint Templates

Although several configuration options are available for the out-of-the-box templates, users are still limited by the inability to edit the structure of the processes. Fortunately, the latest release of SharePoint provides architects with the ability to edit these templates via SPD 2010. Details for editing the templates are discussed later in this chapter.

MICROSOFT OFFICE VISIO 2010 PREMIUM

Visio 2010 provides users with the tools needed to get a workflow solution started. In addition to creating detailed flowchart diagrams, Visio 2010 Premium has a Microsoft SharePoint Workflow template that incorporates many of the conditions and actions used in custom-developed workflows. These flowcharts can then be imported into SPD 2010. For the example solution, the process describes

www.it-ebooks.info

the creation of a flowchart that maps a process so that the workflow starts and then, based on a field value, takes one of two paths. For one option, an approval process is needed; for the other, no approval process is needed and the workflow terminates.

1. Open Microsoft Office Visio 2010 Premium.
2. The first thing needed is to pick the necessary template. In the Template Categories section, click Flowchart. You may need to scroll down to see this section.
3. After you have opened the Flowchart options, click the Microsoft SharePoint Workflow template. Your screen should look like the image in Figure 6-11.

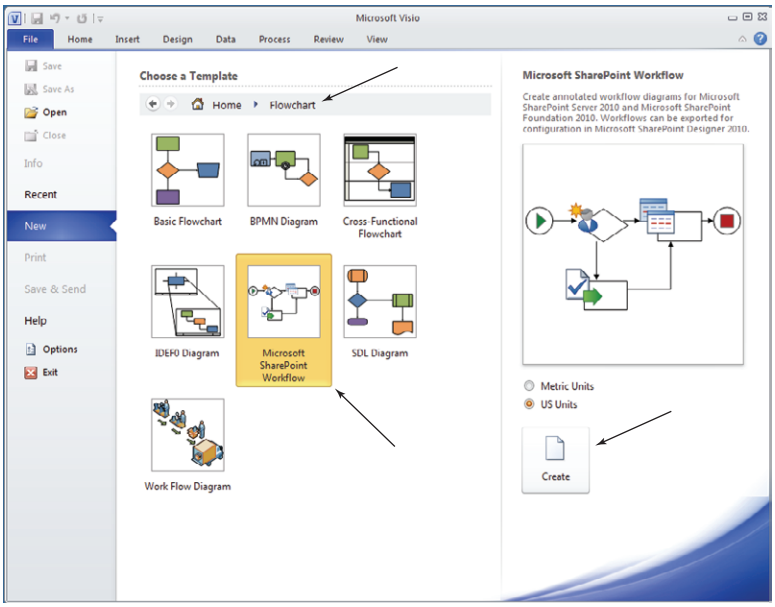


FIGURE 6-11: Selecting the Microsoft SharePoint Workflow template

4. Click Create.
5. On the left, click SharePoint Workflow Terminators.
6. Drag both a Start and Terminate shape onto the page. Separate them so that there is space between them for more shapes. Your screen should look similar to Figure 6-12.
7. On the left, click SharePoint Workflow Conditions.
8. Drag the Compare date source shape onto the page, and place it to the right of the Start shape.
9. Hover over the Start shape, and click the triangle that appears to the right. This connects the two shapes with the flow going from the Start shape to the Compare data source shape. See Figure 6-13 for an example of what this should look like.

www.it-ebooks.info

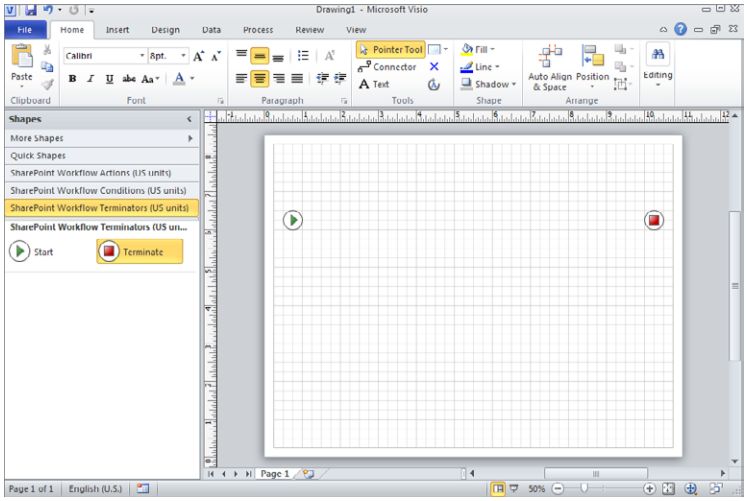


FIGURE 6-12: Add your terminator shapes to the page

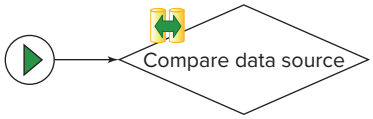


FIGURE 6-13: Connect your shapes to start the flow

10. On the left, click SharePoint Workflow Actions.
11. Drag a Start approval process shape onto the page between the Compare data source shape and the Stop shape, but place it slightly higher up on the page.
12. Hover over the Compare data source shape, and hover over the triangle to the right until it connects to the Start approval process shape, as shown in Figure 6-14.

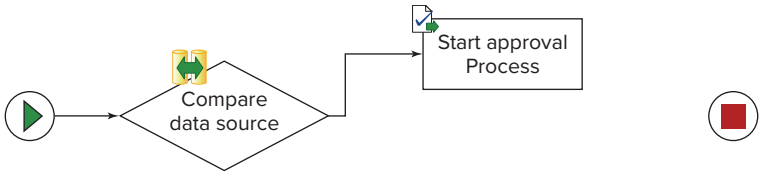


FIGURE 6-14: Connect to Start approval process shape

13. Right-click the connection that was just made, and select Yes from the menu.
14. Drag a Set workflow status shape onto the page below the Start approval process shape.
15. Connect the shapes so that the flow goes from the Compare data source shape to the Set workflow status shape.

www.it-ebooks.info

16. Right-click the connection that was just made, and select No from the menu.
17. Connect the shapes so that the two actions that have been added flow toward the Stop shape, as shown in Figure 6-15. You may need to bring the Stop shape closer for the connections to be made.

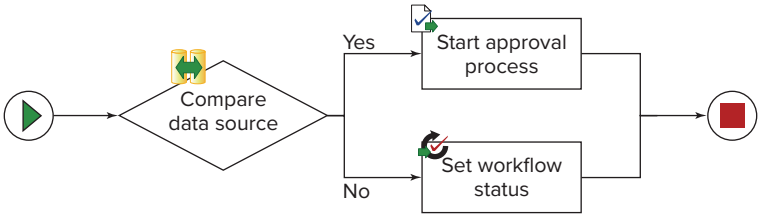


FIGURE 6-15: Connect shapes to the Stop shape

18. Click the Process tab at the top of the application.
19. In the Diagram Validation section of the Ribbon, click the Check Diagram button.
20. In the SharePoint Workflow section of the Ribbon, click Export.
21. Name the document Visio WF and save the document to your desktop.
22. Close Visio. You import this file later in this chapter.

SPD 2010

For nondevelopers, SPD 2010 is a powerful tool. The ability to create semicomplex solutions without having to write custom code is a huge win for organizations because developers tend to be overutilized resources with little availability, whereas consultants have a hefty price tag. Not without its own limitations, SPD 2010 is not a tool that should be used lightly. It provides users with the ability to make a variety of changes to SharePoint sites, whereas in the hands of an untrained and curious user it can be the root cause for catastrophic mouse clicks. It is paramount that any user assigned the task to build solutions in SPD 2010 be adequately trained.

For workflow solutions, SPD 2010 is primarily focused on condition and action-based workflows. With the ability to analyze SharePoint data and adjust the process accordingly, organizations can achieve solutions unobtainable via the out-of-the-box options alone.

Before the various types of solutions are discussed, first take a look at the conditions and actions that are the building blocks for SPD 2010 Workflows. Like most functionalities in SharePoint 2010, the conditions and actions available can vary with the available versions of SharePoint: SharePoint Foundation 2010 and SharePoint Server 2010.

Conditions

The next few sections discuss the conditions that can be configured as part of an SPD 2010 workflow. To help identify where these are used, they have been grouped into logical subsections.

www.it-ebooks.info

Common Conditions

The two most commonly used conditions are If Any Value Equals Value, and If Current Item Field Equals Value. These are used to compare various values that exist in lists or in stored variables.

1. **If Any Value Equals Value:** This condition enables the workflow author to compare two values. These values do not necessarily need to be SharePoint fields. They can also be dynamic strings, workflow variables, context information, and text. The key to remember is that when comparing values, the author can compare only values that exist in the current web. Furthermore, the operator "Equals" is not static. Depending on the type of fields or values compared, the options can vary. The following are the available operators a workflow author sees.

   - Equals
   - Not equals
   - Less than
   - Less than or equal to
   - Greater than
   - Greater than or equal to
   - Contains
   - Does not contain
   - Is empty
   - Is not empty
   - Begins with
   - Does not begin with
   - Ends with
   - Does not end with
   - Matches regular expression
   - Equals (ignoring case)
   - Contains (ignoring case)
   - Equals (ignoring time)

   These operator variances can vary with the selected conditions, but how they can be used with the varying field and value types is consistent. For example, the less than operator is available only for number or date and time fields. Having this operator available for text fields or values would not make any sense.

2. **If Current Item Field Equals Value:** Use this condition to compare a current item SharePoint field to any value. These values are restricted to SharePoint fields, text, dynamic strings, workflow variables, and contextual information that exist in the current site.

www.it-ebooks.info

Other Conditions

The following other conditions target a very specific value with regards to the list item. They are not as frequently used as the previously mentioned conditions, but more complex workflows can sometimes demand these types of value-specific conditions.

- **Created by a Specific Person:** This condition targets the Created By field and enables the author to configure actions based on the user that created a list item. The user specified can be entered as an email address, selected by using the Select Users dialog. The Select Users dialog pulls in user and group values from SharePoint, Microsoft Exchange, and Active Directory.
- **Created in a Specific Date Span:** This condition enables the workflow author to target the Created field value for an item and compare it against a specific date range.
- **Modified by a Specific Person:** Comparing against the Modified By field value for a SharePoint item, this condition enables the author to compare the aforementioned field to a specific user. The specified user can be entered as an email address, selected by using the Select Users dialog. The Select Users dialog pulls in user and group values from SharePoint, Microsoft Exchange, and Active Directory.
- **Modified in a Specific Date Span:** This condition enables the workflow author to target the Modified field value for an item and compare it against a specific date range.
- **Person Is a Valid SharePoint User:** This condition validates that the specified user is a member of the current SharePoint site. The specified user can be entered as an email address, selected by using the Select Users dialog. The Select Users dialog pulls in user and group values from SharePoint, Microsoft Exchange, and Active Directory. This condition is one of the two available workflow conditions when you create a site workflow.
- **The File Size in a Specific Range Kilobytes:** Use this condition to target the File Size field value for the current document and compare it against a range of kilobytes. This can become useful if you try to notify a site owner that an abnormally large file has been added to the site.
- **The File Type Is a Specific Type:** This condition targets the File Type field value for the current document and compares it to the specified file type.
- **Title Field Contains Keywords:** This condition checks to see if the Title field value for the current item contains a specified keyword. Only one keyword can be stated, but the use of logical operators (||, &&, (or), (and)) can be used.

Impersonation Step Conditions

Impersonation is a new feature with SPD 2010 and is discussed in detail in a later section. For now, this section simply lists the conditions available only when using an Impersonation step.

- **Check List Item Permissions:** Unfortunately, the name suggests that this condition checks the exact user permissions that are stated, but this is incorrect. This condition checks to make sure that the specified user or group has at least the specified permissions for the given list. So if a group has Contribute permissions and the condition states Read Only, the condition will be found to be true because Read Only permissions are a subset of Contribute.

www.it-ebooks.info

- **Check List Item Permission Levels:** For this condition, the specified user must have the exact permission level assignment that is specified. So if a user has Contribute permissions and the condition states Read Only, the condition will be found to be false because the permission levels do not exactly match up. Additionally, only users can be specified in this condition. Group values cannot be used.

Actions

Capable of performing a wide range of functionalities, SPD 2010 actions can be used and combined to create basic to complex workflows. Each action is an individual element of the workflow. When the defined conditions are met, the action will complete. Once an action has completed, the workflow will move on to the next component: action, branch, or step.

Core Actions

The most commonly used workflow actions include the following:

- **Add a Comment:** This action enables the author of the workflow to leave a comment regarding the details of the workflow. This is especially beneficial when there is more than one author. When working with more complex processes that have multiple steps, workflow variables, extraction strings, and so on, it is helpful to leave comments for other users to follow.
- **Add Time to Date:** As the name suggests, this action enables the author to add time to a date field or value and store the new date value as a workflow variable. This variable can then be used at a later point in the workflow.
- **Do Calculation:** Use this action to perform a calculation on two values. After the calculation has been performed, the new calculated value will be saved as a workflow variable. This action is similar to the Calculated Column, but rather than have the value visible to users accessing content in SharePoint sites, the workflow can run these calculations behind the scenes and store the values so that the end users can't see them.
- **Log to History List:** This workflow action is underutilized and underrated. The ability to log text to the History List as a workflow action provides an invaluable source of information during the creation and testing of workflow solutions. By logging information after every step or key steps, the author creates a breadcrumb trail that can be retraced if the workflow needs to be debugged.
- **Pause for Duration:** This action enables the author to specify the number of days, hours, and minutes that the workflow should pause before continuing on to the next action.
- **Pause Until Date:** Use this action to pause the workflow until a specified date. This could be a manually entered value, a SharePoint field value, or a workflow variable.
- **Send an Email:** One of the most powerful actions in the SPD 2010 repertoire is the capability to send emails. These notifications can be addressed to any SharePoint user or group, Exchange value, or Active Directory value available in the Select Users dialog.
- **Set Time Portion of Date/Time Field:** Use this action to update the time value of a date and time field or variable value. You can set minutes and hours.

www.it-ebooks.info

- **Set Workflow Status:** Authors can use this action to change the value of the workflow status field to one of the default options: Canceled, Approved, and Rejected. However, a better use of this action is to customize the workflow status options. The default values don't provide a lot of information to users. So rather than use these values, you can use this action to specify a new value of Time-Off Request Denied. There are a few drawbacks to using this action: Customizing status values is not available for Impersonation steps; the custom values are available only for the existing workflow and cannot be transferred to another workflow; and lastly, after a status value has been created, it can't be edited or deleted.
- **Set Workflow Variable:** Simply put, this action sets a workflow variable to a specific value.
- **Stop Workflow:** Based on certain conditions, it is necessary to terminate the workflow. This action is designed to do just that. When the workflow is no longer needed to run, there is no use eating up performance. Use the Stop Workflow action to stop the workflow when the requirements aren't met.

List Actions

Actions that can be performed on list items include the following:



There are also some actions that need to meet additional requirements. Such actions are covered later in this chapter.

- **Check In Item:** This action checks in the current item and enables the author to enter comments.
- **Check Out Item:** This action checks out the current item.
- **Copy List Item:** Use this action to copy the current list item over to a new list that exists in the same site. For this action to run successfully, the two lists involved must share a common column.
- **Create List Item:** Similar to copying an item, this action enables the author to create a list item in another list that exists in the same site. Along with creating the list item, the author can configure metadata values for the new item.
- **Delete Item:** This action deletes the specified item. This item must reside in the current site.
- **Discard Check Out:** This action discards the checkout of a list item. Not as frequently used as many of the other actions, this action can be valuable when conditions require the workflow to discard any changes it may have made. An Advanced Settings feature of lists and libraries is to make Check In and Check Out compulsory. If an initial step of a workflow needs to edit something on the item, but the next step has a condition that isn't met and calls for the preceding changes to be discarded, this action is one of the few ways to automate that request.
- **Set Content Approval Status:** This action will be available only if the current list has Content Approval enabled. Use this action to update the Approval Status field value.
- **Set Field in Current Item:** This action is used to change or set a field value in the current item.

www.it-ebooks.info

- **Update List Item:** Similar to the previous action, but this one can update any list item that exists in the current site.
- **Wait for Field Change in Current Item:** This action is a much-needed improvement to the out-of-the-box templates because this can pause the workflow until a specific field value changes. So if you have an item and that item has a field that needs to be updated by another user before a process can run, this action has your name all over it!
- **Wait for Change in Document Check-Out Status:** The previous action was a bit more broad in that the author could specify the field value that needed to change. This action simply targets the Checked Out To field for a change.

The following List Actions are available only when you use SharePoint Server 2010.

- **Declare Record:** Used with Record Declaration Settings, this action enables the author to declare an item to be a record. For organizations with strict retention policies, this action enables the creating of a solution to automate the process to identify documents that need to be retained.
- **Undeclare Record:** On the flip side, this action will undo any record declaration settings that have been configured for the given item.

Relational Actions

Some actions are available only when you use SharePoint Server 2010. For example, **Lookup Manager of a User** is a welcome addition to the SPD 2010 actions toolset that enables the author to pull in the user value for a specified user. For many approval workflows, the reviewer is the manager of the submitting users. Unfortunately, in previous versions it has been quite troublesome to get this managerial information. The manager value is saved as a workflow variable and can be used in a late action, such as the Start Approval Process action. The manager information is pulled in from the user's profile. If the profile does not contain the manager information, this action will error out when used.

Task Actions

The following are all actions that can be performed on task list items.

- **Assign a Form to a Group:** Rather than simply using the Create List Item in a tasks list, a workflow author can use this action to assign a to-do item, but also customize the fields that need to be filled out. The assigned users fill out the fields in the form, click Complete Task, and the workflow moves to the next action.
- **Assign a To-Do Item:** This action simply creates a new task item to be completed by the specified users or group. The workflow remains paused until the task item has been marked complete.
- **Collect Data from a User:** This action is used by creating a task item for a specified user. This form can be customized by the workflow author, and the responses that are given by the assigned user are stored in the action as a workflow variable. This content can then be used at a later step in the process.

www.it-ebooks.info

The following Task Actions are available only when you use SharePoint Server 2010.

- **Start Approval Process:** This action behaves like the Approval workflow template, with most of the logic already defined. The task behaviors, logged entries, and processes are all set and ready to go. All that is needed is for the workflow author to define the reviewers.
- **Start Feedback Process:** This action behaves like the Collect Feedback workflow template, with most of the logic already defined. The task behaviors, logged entries, and processes are all set and ready to go. All that is needed is for the workflow author to define the reviewers.
- **Start Custom Task Process:** This action behaves like the Approval workflow template; however, most of the logic and process information still needs to be configured. This action enables the ability to incorporate additional conditions, actions, and customizations. If the workflow requirements specify additional functionality on top of the out-of-the-box Approval template, this is the option that is next in line.

Document Set Actions

The following actions are available only in a SharePoint Server 2010 environment. They are related to Document Sets, a feature available only with the specified version of SharePoint 2010. They are not available with SharePoint Foundation 2010.

- **Start Document Set Approval Process:** Similar to the Start Approval Process action, this action targets Document Sets. So when the process runs and the Document Set approval status is defined, so too are the documents contained in the Document Set.
- **Capture a Version of the Document Set:** This action is used to capture the last major versions or the latest minor versions contained in the Document Set. These captures are available in the Ribbon via the Document Set page by clicking the Capture Version button.
- **Send Document Set to Repository:** This action enables the author to move a Document Set to another repository. This could be another document library on the same site or a separate site entirely. To move the Document Set to another site, the Content Organizer feature must be activated for the SharePoint site.
- **Set Content Approval Status of the Document Set:** Similar to the Set Content Approval Status action, this action targets the entire Document Set and updates the Approval Status column.

Utility Actions

The following actions are specifically used to extract data from text strings.

- **Extract Substring from End of String:** Specify a numeric value for the number of characters you want to extract from the string, starting with the last character.
- **Extract Substring from Index of String:** Specify a numeric value for character placement to be used as a starting point for extraction. Then specify the number of characters to extract.
- **Extract Substring from Start of String:** Specify a numeric value for the number of characters you want to extract from the string, starting with the first character.

www.it-ebooks.info

- **Extract Substring from Index with Length:** Specify two numeric values that correspond to character placement in the string. The action extracts the characters starting with the first value and continuing to that last value.
- **Find Interval Between Dates:** This action is used to extract the exact number of minutes, hours, or days between two date values. This extracted value is then stored as a workflow variable.

Actions Available for Task Actions

When working with the Start Approval Process, Start Feedback Process, and Start Custom Task Process actions, a lot of the groundwork has been done. But if things aren't configured to your liking, you can customize the task actions by leveraging the actions listed in this section. Each of these actions is directly related to the original task action.

- **End Task Process:** Use this action to end the task process that was started as a result of a previous action.
- **Set Content Approval Status (as Author):** As SPD 2010 workflows are declarative and run with the permissions of the user that initiates the workflow, there may be an instance in which the initiating users do not have access to update the Approval Status. If that situation is possible, the author can use this action so that the workflow runs the Set Content Approval Status action as the author, rather than the initiator.
- **Wait for Change in Task Process Item:** Use this action to pause the workflow until a change has been made to a specified value.
- **Wait for Deletion in Task Process Item:** Use this action to pause the workflow until a deletion has been made on the task item currently running the approval process.
- **Set Task Field:** Use this action to set a field value for the current task item.
- **Rescind Task:** Use this action to complete the task. It does so without any sort of outcome.
- **Append Task:** Use this action to create a new task that runs at the end of the last assigned task in the current workflow stage. By doing so you can add additional users to the review chain based on specific conditions.
- **Delegate Task:** By using this action, the workflow author can rescind the existing task and assign a new task to a new user. This process can run whether the tasks are assigned to a group or to an individual user.
- **Escalate Task:** This action escalates the task so that it is now assigned to the manager of the user that was originally assigned.
- **Forward Task:** Similar to the Delegate Task action, this action rescinds the existing task and assigns a new task to a new user. If the new user is a group of users, a task item is created for every member of the group.
- **Insert Task:** Rather than interjecting a task into the existing stage, this action creates a new workflow stage that can run after the current stage. The user specified is the only user that reviews the new stage.

www.it-ebooks.info

- **Reassign Task:** This action simply reassigns the task to another user.
- **Request a Change:** Use this action to request a change from a specified user, and then a new task will be created for the current user after the change completes.
- **Send Task Email:** Similar to the Send an Email action, this action emails a notification, but this notification includes an Edit Task button in the email.

Impersonation Step Actions

The following actions are available only in Impersonation steps.

- **Add List Item Permissions:** Use this action to add the specified permissions to the designated user or group.
- **Inherit List Item Parent Permissions:** Use this action to set the item permissions to inherit from the parent.
- **Remove List Item Permissions:** Use this action to remove item permissions from a user or group.
- **Replace List Item Permissions:** Use this action to replace the existing item permissions for a user or group with a new set of permissions.

Document Content Type Actions

The following actions are available when working specifically with a library using the Document content type.

- **Send Document to Repository:** Use this action to move or copy the current item to another document repository. This can be another document library in the same SharePoint site, or it can be a separate site. To move or copy the item to a separate site, the Content Organizer feature must be enabled on the SharePoint site.
- **Delete Drafts:** This action deletes all draft versions for the current item.
- **Delete Previous Versions:** This action deletes all previous versions of the current item. A case in which this could be used is when setting a document to be a record. When keeping official records it is typically necessary to keep only the final copy. When this is the case and you want to automate this process by using the Declare Record action, you may want to first delete all previous versions. If there is no reason to keep previous versions, why bog down your content database with unnecessary files?

It may seem like a lot to handle, but after you start working with SPD 2010 workflows, understanding how each of the conditions and actions can be used together becomes easier. After this last section, it is easy to see why many users choose to jump right to SPD for creating workflow solutions. The available options, although overwhelming at first, can be used by regular users to create some amazing solutions. Now that you have a solid base for the basic workflow components, the features and options for building solutions can be discussed. The following sections cover each of the main focal points followed up with some real-world examples of SPD 2010 workflows.

www.it-ebooks.info

Custom Actions

If the SPD 2010 actions do not fulfill the workflow requirements, options are available to get additional actions. Several third-party vendors have add-ons for SPD 2010, and also a free resource is available at `www.codeplex.com`. Be careful when using CodePlex solutions because they are use-at-your-own-risk solutions. Don't let that scare you off, though. A bunch of tools have much-needed value. The last option would be to have a developer create custom actions and then deploy them for use. It might not be ideal, but for a consultant that works on various projects, the development team has definitely been tasked with creating functionality that was not natively included.

Workflow Options and Features

Not only is SPD 2010 stocked with condition and action options, but the solution features and options are also just as numerous. Now take a deep dive into each of the major components.

Impersonation

Impersonation is the ability for a workflow author to create a step that can run through the conditions and actions with the permissions of the workflow author, as opposed to the user that initiated the workflow. Because most SPD 2010 users have elevated access, it enables the author to incorporate steps that require a higher level of permissions.

For example, if you have a list capturing time-off requests, you want all users to submit request items; this requires them to have Contribute access. After those items have been submitted, they go through an approval process. All approved items are moved over to a calendar list that displays the approved items. Users that submit requests should not have Contribute access to the calendar list that displays the approved items. A workflow author could create a workflow that leveraged an approval process and upon approval copied the item over to the approved calendar, but when the process would run, it would error out. That is because the initiator of the workflow is the submitting user, which does not have Contribute access to the calendar, and the workflow History List would log an access denied error message. Unfortunately, to move the approved item over to the calendar list, a user needs Contribute or higher access. However, you don't want the submitting user to have Contribute access. Here lies the problem. But if the workflow author uses an Impersonation step to run the action to copy the approved item, the item can be copied successfully because the workflow can run with the author's access, not the submitting user.

Application Integration

Another nice feature available in SPD 2010 is the various ways it can integrate with other applications. The compatibility with Visio has been discussed, but only from the side of Visio. It's true that SPD 2010 can import Visio files, but on the flip side, it can also export a workflow to a Visio file. This is beneficial in two ways. First, an earlier section mentioned communication being an issue between business users and workflow authors, and that Visio was the application that could bridge the gap. Yes, business users can create a workflow diagram and save it so that it can be imported by a workflow architect, but a workflow architect could also take a written description of a workflow request, do some initial building, and then export the workflow to a Visio diagram so that it could be reviewed by business users. The flow of information between business users and SharePoint

www.it-ebooks.info

architects is two-way. The second benefit is that it provides a way for solutions to be completed if SPD 2010 cannot successfully fulfill the requirements.

For example, if you are a workflow author and you have been assigned a workflow project, you can take the requirements however you have received them and begin building a SPD 2010 workflow. Unfortunately, the conditions, actions, and features cannot fulfill every possible workflow request. As such, the only option left would be to build the workflow in VS 2010. But rather than start a new VS 2010 project, if the original workflow author exported the unfinished workflow to Visio, the VS 2010 developer could import that same file back into VS 2010 for use.

The same benefit mentioned in the preceding example is also available another way. With Reusable Workflows, an author can save the workflow as a template. This saves the file as a .WSP. This file can be opened in VS 2010 and worked on. So not only do the integration points provide an improved communication mechanism for business and IT users, they also provide an easy way to transfer information so that a project can be added on to or continued rather than scrapping and starting a new project from the beginning.

Office InfoPath 2010 Forms

Unbeknown to some users, the majority of forms used throughout SharePoint 2010 sites are InfoPath 2010 forms: New Item form, View form, List Item forms, and workflow forms. Because SPD 2010 uses these types of forms for its task and initiation forms, they are customizable the same way other InfoPath forms are. So rather than have the standard look and feel, even your workflow forms can contain branding more in-line with your organization's image.

Working with the Out-of-the-Box Templates

Although you can set up and associate out-of-the-box workflows from the browser, some users prefer to work in SPD 2010. For those users, the SPD 2010 interface can perform the same workflow related tasks as one is capable of in the browser, and then some. By selecting a workflow, options become available for performing a number of tasks:

- **View the Workflow Settings:** View the workflow details and configuration options. From SPD 2010 you can find that there are many more options to view.
- **Edit the Workflow:** From both the Workflows page and the Workflow Settings page, a user can edit an out-of-the-box workflow. This is a huge leap forward. Previously all users could do was use an out-of-the-box template or build a workflow from scratch. This new ability to edit the out-of-the-box workflows is a huge benefit and allows users to take a good starting point and customize it to fit their organization's needs. One critical thing to remember when editing the templates is that when editing the template at the top level site, the edit can affect the template for the entire site collection. If you want to update only the template for a web, be sure to edit the workflow template at that level. Customizing the out-of-the-box templates can create only a new template available for the existing site collection. This template cannot be available for other site collections.
- **Copy and Modify:** Rather than editing the workflow file directly, you can copy and modify the copied version. As with most files in SharePoint and in the IT world, always work on the copy.

www.it-ebooks.info

- **Export the Workflow to Visio:** Just as you can import a file into SPD 2010, you can export a workflow to Visio. This exported file can then be opened and edited by business users who can use Visio to state their requirements. After their updates have been made, the file can be imported back into SPD 2010 and finalized.
- **Associate the Workflow to a List:** Just as in the browser, this option creates a workflow association with the specified list. The advantage to working in SPD 2010 is that you can quickly associate the workflow with multiple lists. In the browser you would have to navigate to each list and individually configure the workflow.
- **Associate the Workflow to a Content Type:** Similar to creating a workflow association to a list, this option enables a user to quickly associate the workflow with a content type. The same advantage exists here. In SPD 2010 a user can quickly make these associations.

To copy and edit the existing Approval – 2010 workflow template, use the following steps. The wanted edit is to make approved items read only to site members. These users can have access to create and submit items, but after they are approved, they should not have access to edit the items.

1. Open SharePoint Designer 2010.
2. Select the Sites tab on the left. Here you need to specify the site you want to open. If you have a site in your Recent Sites section that you want to work with, click that option. If not, click Open Site.
3. In the Open Site Dialog enter the URL for the site. Do not add any unnecessary characters to the end of the URL. If your URL is something like `http://server/sites/ContosoPortal/pages/home.aspx` you need to drop the `/pages/home.aspx` for the site to be recognized.
4. Click Open. Your site should now open in SPD 2010.
5. On the left, under Site Objects, click Workflows.
6. Click, not on the name link, but on Approval – SharePoint 2010 workflow. You see options light up in the Ribbon.
7. Click Copy & Modify. Figure 6-16 shows you the options that need to be selected.

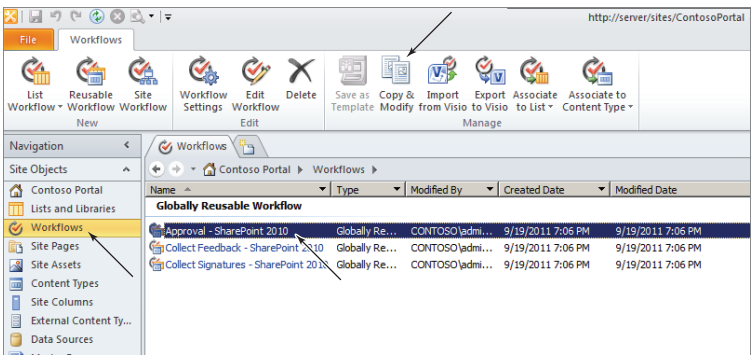


FIGURE 6-16: Copy and modify an existing workflow template

www.it-ebooks.info

8. Enter **Customized Approval** for the workflow name.
9. Click OK. When the workflow opens, notice the flashing orange bar.
10. Click the cursor below Step 1. This places the flashing orange bar below this step, as seen in Figure 6-17.

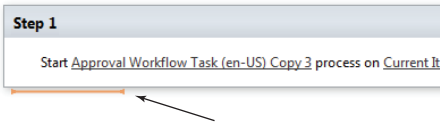


FIGURE 6-17: Place your cursor in the right position

11. In the ribbon, click Impersonation Step.
12. The cursor and orange bar will be inside this step. On the Ribbon, click Condition.
13. Scroll down and click If Any Value Equals Value. This adds the condition to the workflow Impersonation Step and it adds it in a new branch.
14. Click the first value link.
15. Click the function button (fx).
16. In the Data source field, select Workflow Variables and Parameters.
17. In the Field from source field, select Variable: `IsItemApproved`.
18. Click OK.
19. Click the second value link, and select Yes.
20. Place the cursor below the newly configured condition, as shown in Figure 6-18.

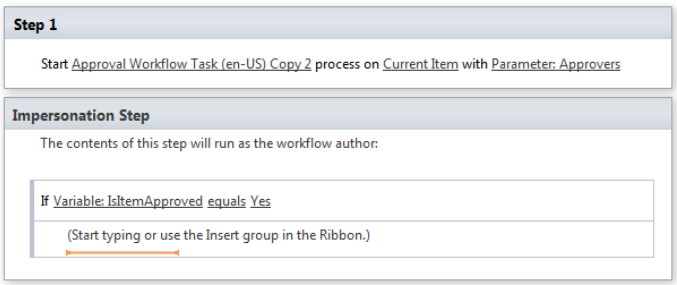


FIGURE 6-18: Place your cursor below the recently created step

www.it-ebooks.info

21. You can notice that in the section below the condition, there is some text that states, Start Typing or Use the Insert Group in Ribbon. You've used the Ribbon to add the condition. Now use the type-ahead feature to add the action. The action you want to add is Replace List Item Permissions.

22. Start typing **Replace List Item Permissions**. After typing **repl**, SPD 2010 should automatically pick up on the action you want to add, as shown in Figure 6-19.

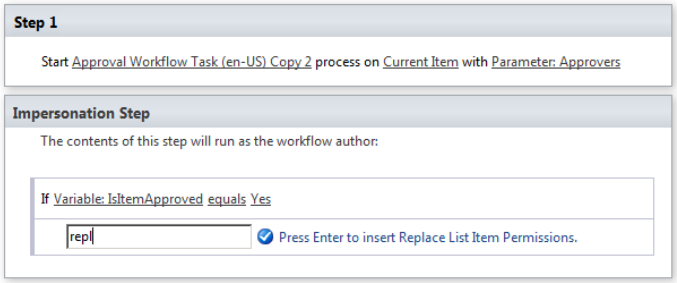


FIGURE 6-19: Using the type-ahead feature

23. Click Enter on your keyboard.

24. Click these permissions.

25. Click Add.

26. In the Select Users dialog, select People/Groups from SharePoint site (see Figure 6-20).

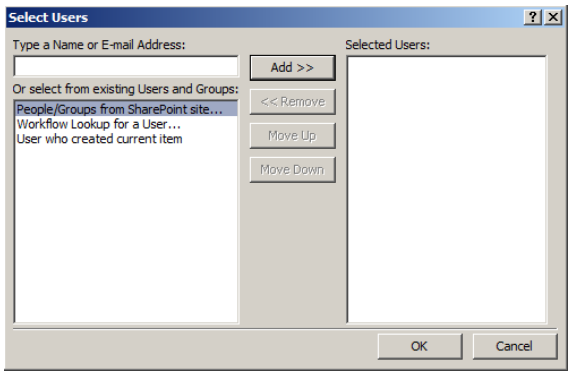


FIGURE 6-20: Using the Select Users dialog to select the desired user value

27. Click Add>>.

28. Type [your site name] Members, and press Enter on your keyboard.

www.it-ebooks.info

29. The members group for your site will be returned as a result. Select this group, and click Add>>. Once you click on Add>>, your window should look like Figure 6-21.

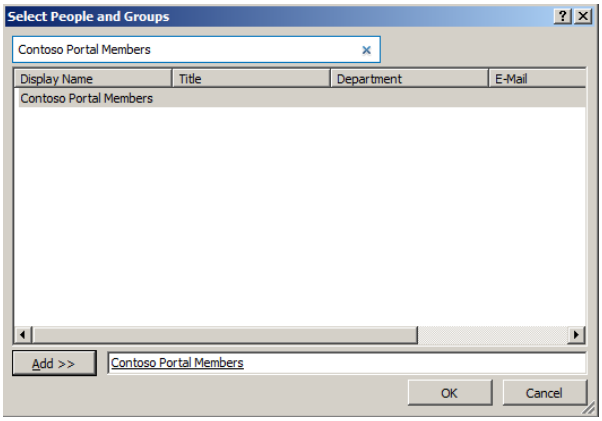


FIGURE 6-21: Selecting the target user/group for the workflow action

30. Click OK.
31. Click OK.
32. Check the box for Read.
33. Click OK.
34. Click OK.
35. Click this list.
36. Click OK. When you do this, your workflow steps should look like Figure 6-22.

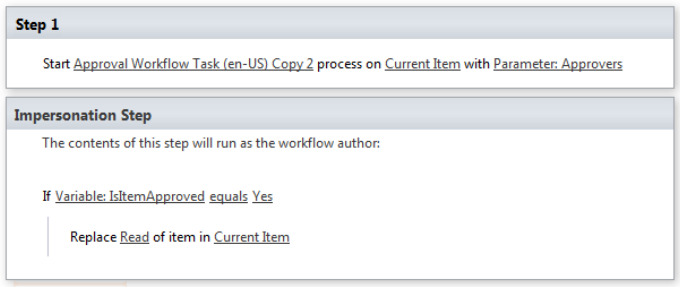


FIGURE 6-22: Finished product

37. In the Ribbon, click Check for Errors. You should get a message saying, The Workflow Contains No Errors.

www.it-ebooks.info

38. Click OK.
39. In the Ribbon, click Publish Globally. Publish this globally because you want this to be a reusable template for the entire site collection.
40. Click OK.
41. Click OK.
42. When the workflow has published successfully, right-click the Customized Approval tab, and click Close.
43. Your new globally available template is ready to go, as shown in Figure 6-23! Keep SPD 2010 open and on this page. The next exercise uses this location as a starting point.

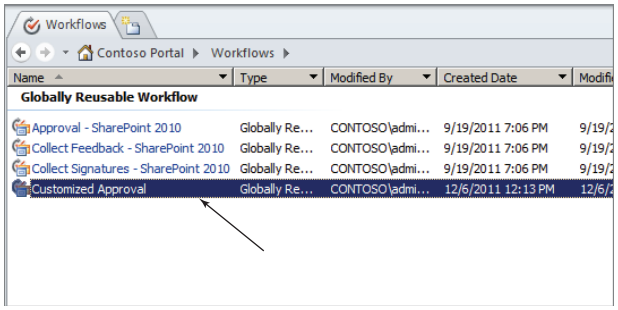


FIGURE 6-23: Your workflow is now visible from the Workflows tab

Import from Visio

The integration with Visio has been mentioned several times. To import a Visio file into SPD 2010 follow these steps:

1. On the Ribbon, click Import from Visio.
2. Click Browse.
3. Browse to the Visio flowchart you previously saved.
4. Click Open.
5. Click Next >.
6. Leave the default name. Select Calendar from the List Workflow drop-down menu, as shown in Figure 6-24.
7. Click Finish. This opens the workflow.
8. Click the first value.
9. Click the function button.
10. In Data source, select Current Item.

www.it-ebooks.info

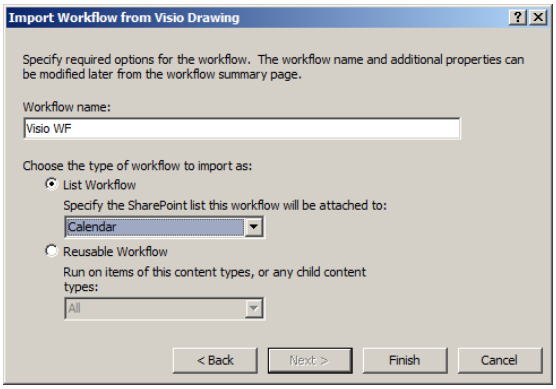


FIGURE 6-24: Import workflow from Visio drawing

11. In Field from Source, select Category.
12. Click this test and select equals.
13. Click value and select Meeting.
14. Click this item and click OK.
15. Click these users.
16. A Select Task Process Participants window opens up. To the right of the Participants field, click the Address Book button.
17. Select a user and click Add>>.
18. Click OK.
19. On the Select Task Process Participants window, in the Title field, enter **Review Meeting Event Calendar Item**.
20. In the Instructions box type **Item**, and add a space.
21. Click the Add or Change Lookup button.
22. In Field from Source, select Title. This adds a lookup to the text of the task message. This automatically pulls in the name of the task item that is created.
23. Add a space and type **has been submitted and requires your approval. Please review the item and Approve or Reject**.
24. Press Enter on your keyboard twice.
25. Click the Add or Change Lookup button.
26. In Field from Source, select Encoded Absolute URL.
27. Click OK. Your task process window should look like Figure 6-25.
28. Click OK. Your steps should now look like Figure 6-26.

www.it-ebooks.info

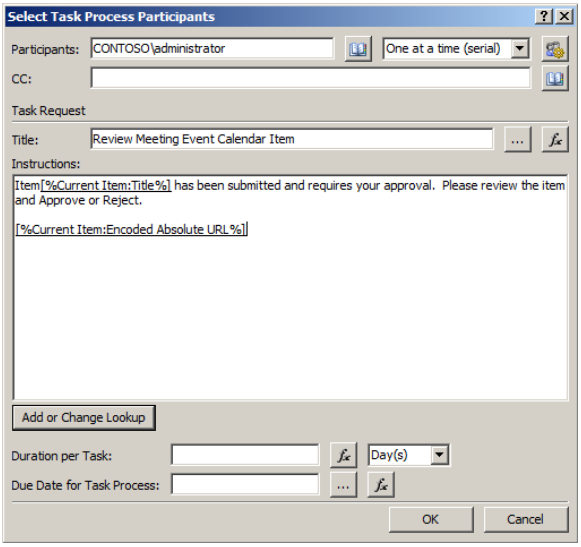


FIGURE 6-25: Configured task process

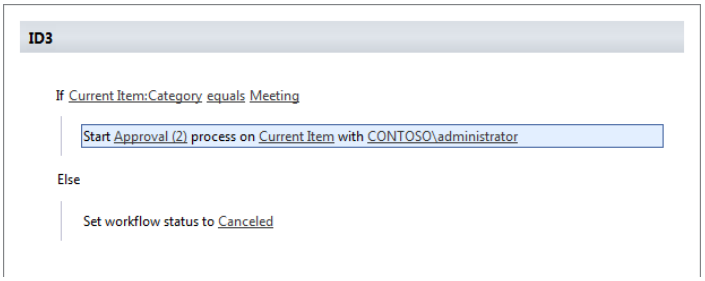


FIGURE 6-26: Finished workflow conditions and actions

29. On the Ribbon, click Workflow Settings.
30. In the Start options, check the box for Start Workflow Automatically When an Item Is Created.
31. On the ribbon, click Publish.
32. Now you have a workflow that was pulled in from Visio and configured to run so that when a calendar list item is created, if that item has a Category column value of Meeting, the approval process will run. If it does not have a Category column value of Meeting, the process will be canceled. Right-click the Visio WF tab, and click Close. Keep SPD 2010 open and on this page. The next example uses this location as a starting point.
33. Open your browser to the calendar list that the workflow is associated with.
34. Navigate to the Workflow Settings page for the calendar list. You see the workflow listed. Here the workflow can be managed just like the out-of-the-box workflow templates when they are associated to a list or library.

www.it-ebooks.info

List Workflows

The most basic of the SPD 2010 workflow options is creating a list workflow. Unlike the out-of-the-box workflow templates, these solutions are not templates made available via a feature. These are direct associations directly written to a SharePoint list or library. These associations can quickly be created and published, but the downside is that they are not reusable. For these to be available for other lists and libraries, you must re-create them from scratch. In the previous version of SPD, this was the only option. As such, this was a deal-breaker for most organizations because reusability is vital for almost every solution. Having heard all the ramblings, Microsoft made it a point to provide additional workflow options. But this solution option isn't completely useless; numerous examples of solutions need not be replicated. In those instances, a direct workflow solution is the perfect choice based on the speed and ease in which the solution can be created and deployed.

Reusable Workflow

A key advancement in SPD 2010 is the ability to create reusable solutions. By selecting the Reusable Workflow option, users create workflow associations directly to a content type. This association works similarly to the way content types work with out-of-the-box workflow associations. After the workflow has been published, it can also be saved as a template .WSP file. This file can then be used as a sandbox solution in other SharePoint 2010 sites or imported into VS 2010.

The following procedure guides you through the process to create a reusable workflow that is associated to the Document content type. You can pull in the submitting user's manager and assign that manager an approval task.

1. If SPD 2010 isn't open based on the last exercise, open the application, and use the steps from previous exercises to get to the workflows page.
2. On the ribbon, click Reusable Workflow.
3. For the name, enter **Manager Approval**.
4. For Content Type, select Document.
5. Click OK.
6. When the workflow opens, add the Lookup Manager of a User action to step 1.
7. Click this user.
8. Select User who Created Current Item, and click Add>>.
9. Click OK.
10. Add the Start Approval Process action.
11. Click these users.
12. To the right of the Participants field, click the address book.
13. Select Workflow Lookup for a User, and click Add>>.
14. In Data Source, select Workflow Variables and Parameters.
15. In Field from Source, select Variable: manager.

www.it-ebooks.info

16. In Return Field As, select Email Address.
17. Click OK.
18. Click OK.
19. In the Title field, enter **Document Approval Request**.
20. In the Instructions field, enter **A subordinate has submitted a document that requires your approval. Please review the item.** Your task process window should look like Figure 6-27.

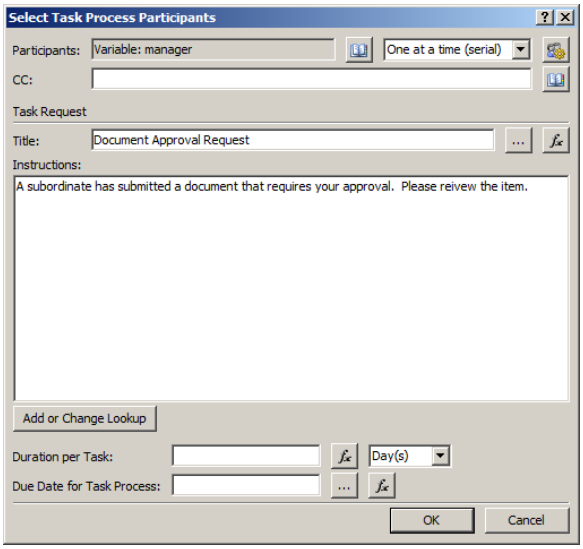


FIGURE 6-27: Configured task process

21. Click OK.
22. On the ribbon, click Publish.
23. When the workflow is successfully published, keep SPD 2010 open to this page.
24. Navigate to the SharePoint site where your reusable workflow was associated.
25. Click Site Actions, and then click Site Settings.
26. In the Galleries section, click Site Content Types.
27. Scroll down and click Document.
28. Click Workflow Settings.
29. Click Add a Workflow.
30. You can see that your Manager Approval workflow is among the workflow options. So even though you created the workflow association to the content type, before the workflow can be used, you must add the workflow and configure the settings.

www.it-ebooks.info

31. Enter **Manager Approval** for the name of the workflow.
32. Scroll down and specify that the workflow will start only when an item is created.
33. Notice the last section. Click OK.
34. Navigate to the Shared Documents library.
35. Add a document.
36. When you see that the workflow status column has been added, navigate to the Shared Documents Workflow Settings page. You can notice that the workflow is not displayed. Workflows displayed here are ones that are associated to the list. Workflows associated with the content type are managed only from that particular content type's page.

Site Workflows

Site workflows are a new concept. These are workflows that do not rely on list or library items or content types. Instead they are associated directly to a SharePoint site. Without an out-of-the-box template for creating a site workflow, users must rely on SPD 2010 or VS 2010 to build these types of solutions. Restricted to only be started manually, these solutions tend to be less prevalent as most organizations try to remove as much of the human element as possible to automate business processes. The majority of the time these types of solution are used to generate reports of some sort.

Always the Best Tool?

The last thing discussed is when to use SPD 2010 versus other options; just because you can do something one way doesn't mean that it the best way. Yes, SPD 2010 has a bunch of useful actions, but unfortunately, real-world experience shows some kinks in the armor. A specific example of this is using the actions that pause the workflow for an extended period of time. Brief pauses aren't an issue, but when you see these actions, it's natural to think that you can pause the workflow for days and months. This is not advisable. When SPD 2010 workflows run, they are constantly running until they complete their steps, branches, conditions, and actions. But if the workflow is paused, it can eat a chunk of performance until the pause specifications complete. One workflow by itself isn't a performance drain, but for organizations that leverage workflow solutions, the reality is that workflows are set up in multiple places with multiple instances running all the time. When the grand scheme is considered, it is dangerously possible for things to grow exponentially when users see how effective workflows can be. Along the same lines and concerns, but different reasoning, extended pauses pose another problem. As they are constantly running, if the server were to crash, or a large update is made, chances are that a long paused process could be affected and error out. What this means for workflow authors is that you must select the designing tool based on not only requirements, but also based on best practices. For SPD 2010, best practices recommend that processes run without extended pauses and complex calculation.

Visual Studio 2010

The last workflow option discussed is VS 2010. This option not only enables the most flexibility, but VS 2010 is the only non-third-party solution that can create highly complex SharePoint 2010 workflow solutions. By writing custom code, an experienced SharePoint developer can create any

www.it-ebooks.info

number of solutions. To aide in this endeavor, Visual Studio 2010 provides developers with a host of improvements. Based on Windows Workflow Foundation (WF) 4.0 in .NET 4, the underlying technology has made vast improvements to the tools and features available for creating VS 2010 SharePoint workflow solutions. By cleaning up the user interface and introducing several new components, developers can find that it is easier than ever to build intricate solutions.

Pluggable Workflow Services

Previously locked down, pluggable workflow services are designed to improve communication between external applications and components and workflow instances. This feature helps to cut back on the performance drain that is experienced when workflows have long pauses or lengthy calculations. This dehydration takes place as a result of the workflow constantly running in an active state. By improving overall communication, the workflow instance is more aware of when tasks are complete and when it is no longer needed and becomes inactive so that it is not consuming resources.

Workflow Designer

This new feature provides developers with a visual tool that can be used to lay out the workflow process. This enables developers to graphically define the various workflow activities and the path in which those activities are performed. With new templates such as the Flow Chart, it is easier than ever to visualize the overall process.

ASP.Net Workflow Form Templates

Building custom forms can be a gruesome process, but fortunately for VS 2010 developers, new templates are available to decrease the pain and burdens associated with creating new forms. Two templates that have received a noticeable amount of notoriety are the Initiation and Association templates.

Workflow Event Receivers

To improve communication with SharePoint processes, several new workflow event receivers have been created. These include Starting, Started, Completed, and Postponed. These new events enable improved messaging and are a massive relief for SharePoint 2010 developers.

Import Reusable Workflows

Although it has been mentioned a few times already, the importance of pulling in SPD 2010 reusable workflows is extremely beneficial. After users have published a reusable workflow in SPD 2010, they can choose to click the option to Save as Template. If this is done, a .WSP file is created and stored in the Site Assets library for the SharePoint site. A developer can download this file and, using the Import Reusable SharePoint Project item, open the .WSP. This requires a new project to be created because workflows cannot be imported into an existing project.

Sandbox Custom Workflow Activities

Custom workflow activities can be created and deployed to extend SPD 2010 and VS 2010's capability to build and implement complex workflow solutions. Using the solution gallery, custom activities

www.it-ebooks.info

can be deployed as sandbox solutions and can be made available to the entire site collection. Unfortunately, the out-of-the-box activities are in an assembly blocked from sandbox solutions, and as such, sandbox custom workflow activities cannot be derived from existing activities.

Site Workflows

Like SPD 2010, VS 2010 can create site workflows. Although this topic has come up quite a bit, it is worth mentioning that due to the lack of out-of-the-box options, and the minimal amount of SPD 2010 actions that can be associated with site workflows, VS 2010 is the ideal choice for creating these types of solutions. In most cases a custom initiation form needs to be created.

Deploying Custom Workflows

Custom workflows should always be deployed as features. For this to happen, the solution needs to be packaged up in a .WSP file. When this is done, a PowerShell cmdlet can be used to add and install the solution package. When installed, the solution can be deployed via PowerShell or through the Central Administration options. After the solution has been deployed, it should be available at the site collection level as a feature. Simply activate the feature, and the workflow should be available to run.

SUMMARY

Overall, SharePoint sites enable various workflow solutions to be implemented, but before starting to build, you should clearly define the requirements for the needed solution. Only then can an accurate assessment of the required tool be made and the solution built.

As with many of the features in SharePoint, the out-of-the-box capabilities provide users with a nice selection of simple solutions, but users are not limited to just these options. With varying difficulty and complexity, more real-world solutions can be created by using approved applications that integrate with SharePoint sites. After these solutions are created, the user interface enables users to manage workflow instances with a high level of ease. This ability to create and manage automated business processes makes SharePoint 2010 a powerful collaboration tool.

www.it-ebooks.info

www.it-ebooks.info

7

Administrating SharePoint 2010 with PowerShell

Relying on humans to correctly perform repetitive tasks is dangerous. Most of us are an undependable bunch and, as a rule, should not be trusted to get things right. Even if you learn nothing about PowerShell from this chapter, remember not to trust humans (including yourself) to correctly perform any task. If you manage SharePoint exclusively via Central Administration, not only are you missing out on half the administration story, but you're also putting too much trust in a human. Even if you manage to get something right once, that's not enough. There are staging environments, developer systems, and disasters. If you set up a SharePoint farm or deploy a customization once, chances are good that you will have to do it again. System administrators and developers alike must have a dependable, consistent, repeatable method for spinning up SharePoint farms and deploying customizations. This means scripting. The investment of effort spent up front in writing a good set of scripts pays large dividends in the future. It's tempting to jump into Central Administration to get things going quickly, but this approach is short-sighted at best. Stop, plan, and spend the time to automate the process.

This chapter looks at how SharePoint 2010 is integrated with Windows PowerShell 2.0. Hopefully you can see that after the learning curve is conquered, a solid working knowledge of PowerShell saves you time and more than a few headaches. Start with script from the beginning, and you won't regret it.

For the scripting-inclined, SharePoint 2007 offered two options: the object model and the venerable command-line utility known as `stsadm`. The object model is all-powerful but has a downside; it requires building a custom application. Building a custom application requires a developer. Many developers have built many quality administration tools on top of the SharePoint 2007 object model, but the problem is one of consistency. Every tool is different, and not every tool offers all the necessary features. In addition, it can be difficult to extend the functionality of custom tools. `stsadm`, on the other hand, ships with SharePoint 2007 (no developer required), uses a

www.it-ebooks.info

(mostly) consistent paradigm, and is extensible (developer required). What it lacks is access to the entire object model.

Developers and administrators alike needed a tool that was consistent, extensible, offered access to the entire SharePoint object model, and was available out of the box. Enter Windows PowerShell 2.0. Windows PowerShell is a command-line-driven environment built directly on the Microsoft .NET Framework. What this means is that PowerShell is a first-class citizen in the .NET world and can easily access any of the objects, methods, and properties available in a custom application.

Before you dive into the world of PowerShell, it's worth explaining a few concepts. The first is the cmdlet, which is simply a command that you can use in the PowerShell environment. Cmdlets are typically implemented in managed code by extending the `System.Management.Automation.PSCmdlet` class. An example of a cmdlet is `Set-Location`, which sets the current location of the prompt. Cmdlets tend to follow a verb-noun naming convention.

Aliases are alternative names assigned to cmdlets. For example, `CD` is an alias for the previously mentioned `Set-Location` cmdlet. Using either `CD` or `Set-Location` provides the same result.

Providers are custom components that enable PowerShell users to access structured data from the command line. For example, the PowerShell registry provider enables the data in the registry to be directly browsed from within PowerShell as if it were a file system.

Scripts are files with a ps1 extension that contain calls to other scripts, cmdlets, or aliases. These scripts can be executed directly from within PowerShell.

Snap-ins and modules are packaging mechanisms for groups of related functionality. Essentially, a snap-in is a set of cmdlets compiled into a DLL (developer required). Modules come in two types: script and binary. Script modules are simply files with a psm1 extension that contain valid PowerShell commands. Binary modules are compiled DLLs that contain cmdlets and providers.

GETTING STARTED

This section explores some of the basics to get started with PowerShell. The first step is to actually start the PowerShell console application which, for SharePoint 2010, is the SharePoint 2010 Management Console. The PowerShell cmdlet conventions are explained in this section as well. Finally, command discovery explains how you may figure out what cmdlets are available.

Starting PowerShell

Windows PowerShell is available on any server in the SharePoint farm with the exception of the database server. To launch Windows PowerShell, simply select SharePoint 2010 Management Shell from the Start menu. The command window opens as shown in Figure 7-1.

We utilize a SharePoint-specific shortcut rather than the built-in PowerShell one. The SharePoint 2010 Management Shell shortcut actually executes the following command:

```
C:\Windows\System32\WindowsPowerShell\v1.0\PowerShell.exe  -NoExit  " & '
   C:\Program Files\Common Files\Microsoft Shared\Web Server
   Extensions\14\CONFIG\POWERSHELL\Registration\\sharepoint.ps1 ' "
```

www.it-ebooks.info

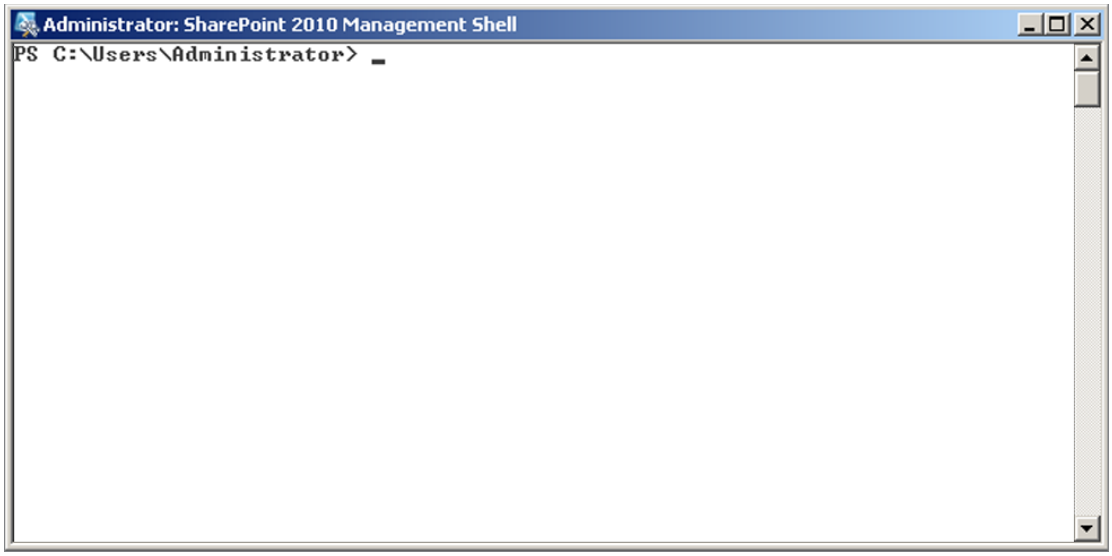


FIGURE 7-1: SharePoint 2010 Management Shell

The command opens the Windows PowerShell console (PowerShell.exe) and executes sharepoint.ps1. The *NoExit* parameter forces the console to remain open when execution of sharepoint.ps1 completes. Looking inside the sharepoint.ps1 script file shows several lines of code:

```
$ver = $host | select version
if ($ver.Version.Major -gt 1)  {$Host.Runspace.ThreadOptions = "ReuseThread"}
Add-PsSnapin Microsoft.SharePoint.PowerShell
Set-location $home
```

Sharepoint.ps1 sets the `ThreadOptions` of the runspace to `"ReuseThread"`. The command is executed only if the host version is greater than 1. Because the `ThreadOptions` property was added in PowerShell 2.0, the check is necessary. Setting the `ThreadOptions` to `"ReuseThread"` tells PowerShell to use the same thread for all command invocations. Next, the script registers the `Microsoft.SharePoint.PowerShell` snap-in via the `Add-SPSnapin` cmdlet. This makes the SharePoint-specific cmdlets available during the session. Finally, sharepoint.ps1 uses the `Set-location` cmdlet to change the current folder to the logged-in user's home directory.

PowerShell Conventions

Before beginning a discussion on the specifics of using SharePoint and PowerShell together, anytime a new cmdlet is introduced, the full syntax is presented in the following format for reference:

```
Set-Location [-LiteralPath] <string> [-PassThru] [-UseTransaction]
[<CommonParameters>]
```

In most cases, these listings are taken directly from the help output of the command discussed. Having a reference to the syntax and available parameters is quite useful and serves as a jumping-off point for any discussion. Space considerations prevent detailing every available parameter or

www.it-ebooks.info

even every single cmdlet in a single chapter; however, you can easily find full documentation on Microsoft TechNet at `http://technet.microsoft.com/en-us/library/ee890108.aspx`. In addition, the `Get-Help` cmdlet discussed in the next section provides a significant amount of information regarding specific cmdlets.

Command Discovery

The `Microsoft.SharePoint.PowerShell` snap-in contains more than 500 commands, so figuring out exactly what you're looking for can be a challenge. The easiest way to discover Windows PowerShell commands is by using the built-in `Get-Command` cmdlet. You may start off by listing all the available SharePoint commands. Because the SharePoint-specific Windows PowerShell commands are contained in the `Microsoft.SharePoint.PowerShell` snap-in, it is simple to produce a list of available commands:

```
Get-Command -module Microsoft.SharePoint.PowerShell
```

This produces a (long) list of all commands available in the module indicated by the module parameter, as shown in Figure 7-2. The list is quite long and useful only for getting a broad overview.

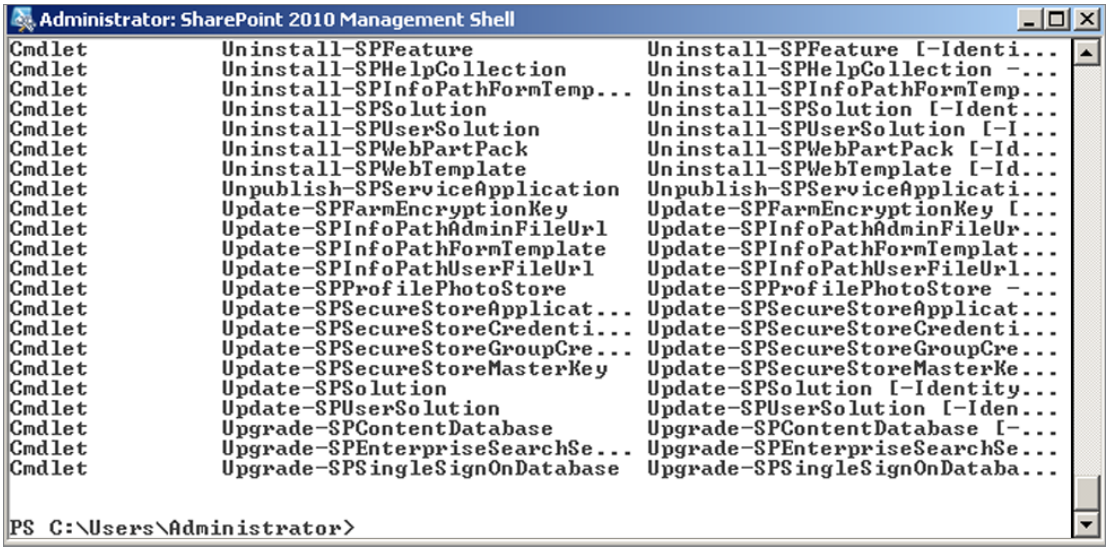


FIGURE 7-2: Available SharePoint PowerShell commands

You can refine the command to limit the list. For example, to list all of the commands that operate on features, use this syntax:

```
Get-Command -module Microsoft.SharePoint.PowerShell -name *feature*
```

The list of commands produced is much more manageable, as shown in Figure 7-3.

www.it-ebooks.info

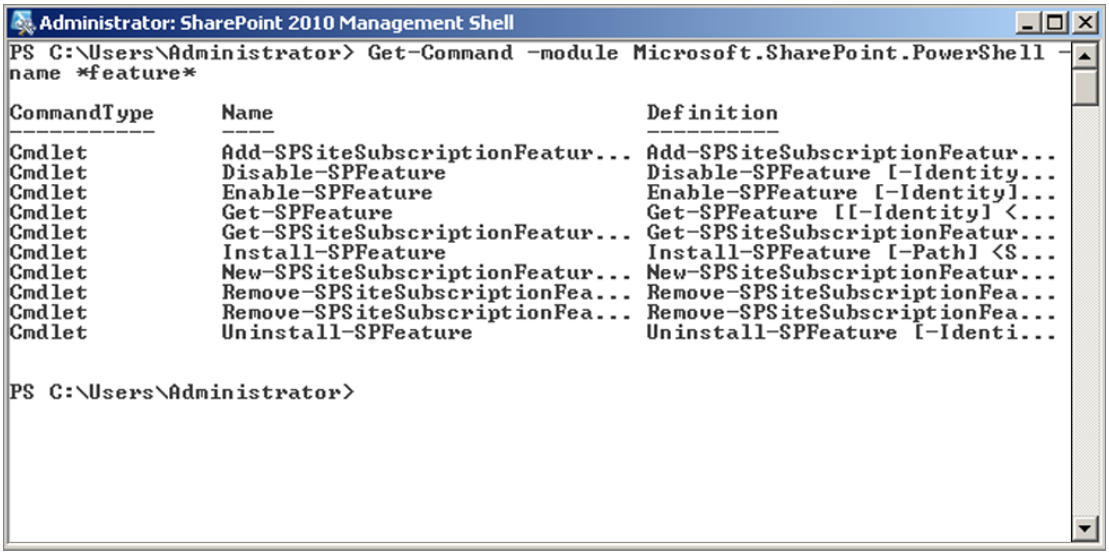


FIGURE 7-3: List of commands that operate on features

You may now select a single command from the list and use the `Get-Help` cmdlet to learn more about it, as shown in Figure 7-4.

```
Get-Help -name Enable-SPFeature*
```

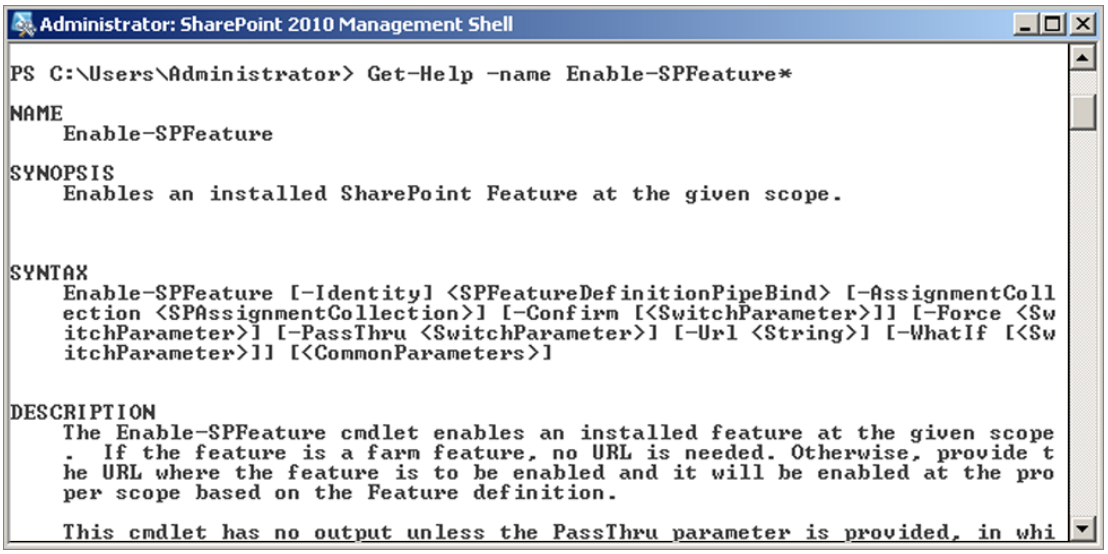


FIGURE 7-4: Getting help for a command

www.it-ebooks.info

The `Get-Help` cmdlet returns information regarding the specified command including a description, syntax details, and a list of related commands. You can also use the optional `-examples` parameter to the `Get-Help` cmdlet to see examples of how to use the command. For all available information regarding the command, add the *`-full`* parameter. You may also issue a cmdlet with *`-?`* to display the main help text.

Function Key Shortcuts

You can use several function key operations in PowerShell that are similar to Command Prompt window operations. These assist in repeating commands or at least help you start with a similar command that was previously executed. Pressing the F3 function key in the console window displays the last executed statement on the prompt line. When this appears, pressing the up and down arrows pages through any other previously executed commands. This is useful for repeating or correcting previous command entries. You can use the left and right arrow keys to move the cursor within the command-line text for corrections. Pressing the Insert key enables you to overwrite the text instead of inserting because by default the operation is insertion.

Pressing the F7 function key pops up a menu of previously executed commands from your PowerShell session. You can use the arrow keys to highlight the appropriate item in the menu. When highlighted, pressing Enter immediately executes the command selected. However, pressing F9 and entering the command number of the selected menu item simply inserts that command on the current command line without automatically executing it.

Command Prompt System Commands

PowerShell uses aliases to facilitate the backward compatibility with command prompt system commands. These are the commands that stem from the original disk operating system (DOS). Example commands are listed here:

- `cd` or `chdir`
- `cls`
- `dir`
- `echo`
- `ls`
- `print`
- `type`

The `ls` command is interesting because it actually stems from UNIX and lists out the items of the current location (similar to `dir`). The `ls` command was previously supported in the Windows Command prompt; however, it does not appear to work in Windows 7.0. It is indeed functioning in PowerShell and is mapped as an alias for `Get-ChildItem`, which is exactly what `dir` is mapped as. To review all system aliases use the `Get-Alias` on a PowerShell command line prompt.

www.it-ebooks.info

SHAREPOINT MANAGEMENT WITH POWERSHELL

One of the first things that typically needs to be done within SharePoint post-installation is the setup of a site structure. When setting up a new farm, you typically start with a web application, one or more site collections, and one or more webs — collectively referred to as the site structure. When the site structure is established, specific lists are created, permissions are applied, and various features and customizations are deployed. This is a process that will be repeated many times over the life of a SharePoint farm and is a perfect place to start scripting with PowerShell.

Before working with web applications and sites in PowerShell, you need to understand the concept of an `SPAssignment` object. Anyone familiar with the SharePoint object model knows the importance of disposing of `SPSite` and `SPWeb` objects. These objects allocate large amounts of memory and, when not properly disposed of, can cause performance and memory issues in an application. Developers typically employ a Using statement to ensure proper object disposal. PowerShell is no different; working with a large list of webs or sites can consume large amounts of memory. Luckily, PowerShell for SharePoint includes the `Start-SPAssigment` and `Stop-SPAssignment` cmdlets.

```
Start-SPAssignment [-AssignmentCollection <SPAssignmentCollection>]
[-Global <SwitchParameter>] [<CommonParameters>]

Stop-SPAssignment [[-SemiGlobal] <SPAssignmentCollection>] [-AssignmentColl
 ection <SPAssignmentCollection>] [-Global <SwitchParameter>] [<CommonParameters>]
```

These cmdlets enable groups of objects to be allocated within a specific store and disposed of when they are no longer needed. Assignments can be made to either a global or a specific named store. The key difference is that using a specific named store allows much finer-grained control over which objects are disposed and when. Using the global store essentially puts all objects into a single bucket and simultaneously disposes of them all.

SharePoint Farm Management

This section describes the various farm management cmdlets that are available to manage and configure farm-level settings. Microsoft has scripts available for download to automate the installation and creation of a new SharePoint 2010 farm (`http://technet.microsoft.com/en-us/library/cc262839.aspx`). Once the farm is created, the cmdlets explained in this section may be used to modify configuration values.

Farm Configuration

Use the `SPFarmConfig` command to configure farm settings. Use `Get-SPFarmConfig` to review the current status of these values, as shown in Figure 7-5.

You may change these values by using the `Set-SPFarmConfig` cmdlet. The `Set-SPFarmConfig` cmdlet has syntax as follows:

```
Set-SPFarmConfig [-AssignmentCollection <SPAssignmentCollection>]
[-ASPScriptOptimizationEnabled <SwitchParameter>]
[-Confirm [<SwitchParameter>]]
[-DataFormWebPartAutoRefreshEnabled <$true | $false>]
[-Force <SwitchParameter>]
```

www.it-ebooks.info

```
[-InstalledProductsRefresh <SwitchParameter>]
[-ServiceConnectionPointBindingInformation <String>]
[-ServiceConnectionPointDelete <SwitchParameter>]
[-WhatIf [<SwitchParameter>]] [-WorkflowBatchSize <Int32>]
[-WorkflowEventDeliveryTimeout <Int32>]
[-WorkflowPostponeThreshold <Int32>] [<CommonParameters>]
```

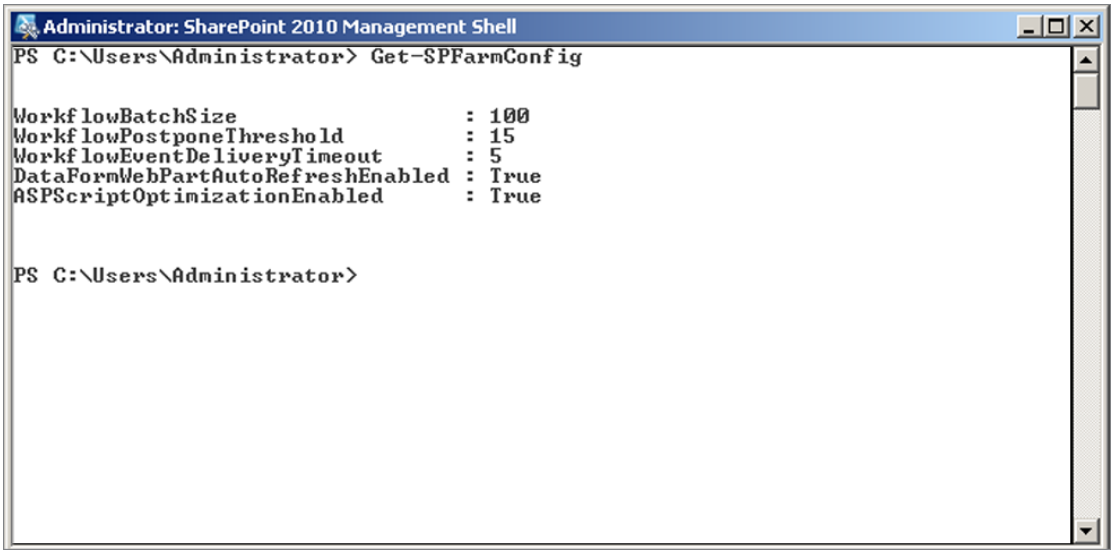


FIGURE 7-5: Displaying the farm configuration

The *ASPScriptOptimizationEnabled* parameter specifies if ASP scripting is allowed to be optimized by IIS. The *DataFormWebPartAutoRefreshEnabled* parameter specifies if the data from web parts should automatically refresh via the asynchronous Ajax settings. The *WorkflowBatchSize* value specifies the amount of events that may be processed within a single workflow instance. The *WorkflowEventDeliveryTimeout* value determines the number of minutes that a workflow can run before a timeout occurs. The *WorkflowPostponeThreshold* value determines how many workflows may be executed at the same time.

Farm Pass Phrase

The farm pass phrase is a separate password used when installing and modifying the SharePoint farm. If the pass phrase needs to be changed or reset, the only method to do this is from the SharePoint PowerShell cmdlet `Set-SPPassPhrase`, which has two syntaxes as follows:

```
Set-SPPassPhrase -ConfirmPassPhrase <SecureString>
-PassPhrase <SecureString>
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-WhatIf [<SwitchParameter>]] [<CommonParameters>]

Set-SPPassPhrase -PassPhrase <SecureString>
```

www.it-ebooks.info

```
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-LocalServerOnly <SwitchParameter>]
[-WhatIf [<SwitchParameter>]] [<CommonParameters>]
```

The pass phrase needs to be a secured string, and there you need to use the `ConvertTo-SecureString` PowerShell function, as shown in Figure 7-6.

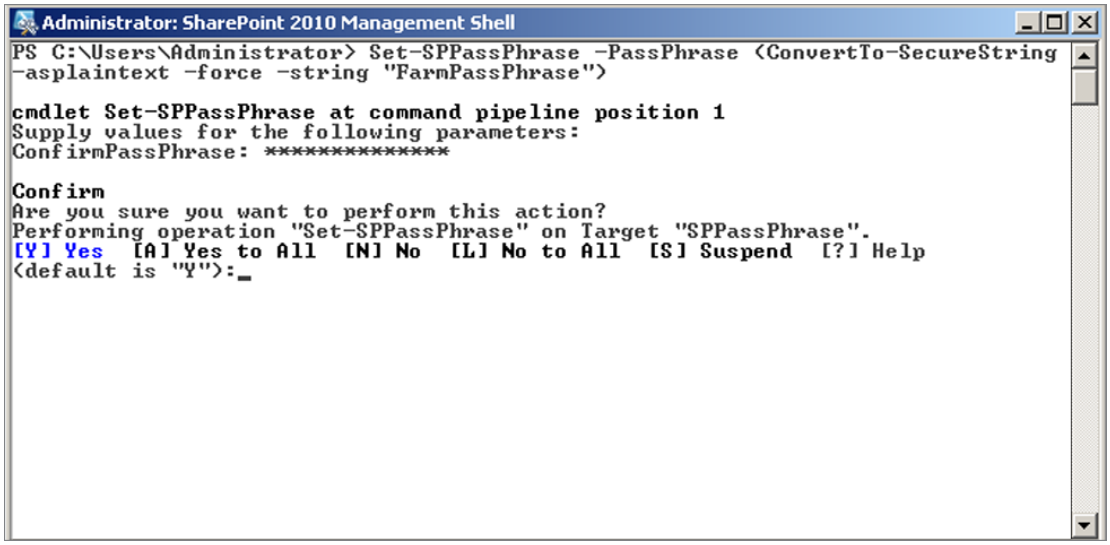


FIGURE 7-6: Setting the farm pass phrase

As you can see, you will be prompted to confirm the pass phrase as well as the overall operation. You may use the *`-ConfirmPassPhrase`* and *`-Confirm:$false`* parameters to execute the cmdlet without further prompting.

Central Administration HTTP Port

While it might be recommended to let the port used for SharePoint Central Administration remain as the randomly assigned value (during installation), it does make life easier if the port number is a standard value that can be consistently used within multiple environments (for example development, staging, and production).

You may easily change the port of the Central Administration Web site by using the `Set-SPCentralAdministration`:

```
Set-SPCentralAdministration -Port <Int32>
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-WhatIf [<SwitchParameter>]]
[<CommonParameters>]
```

Simply use the `-Port` parameter with a new port value and you are all set.

www.it-ebooks.info

Managed Paths

Managed paths are configured at the farm level and provide additional virtual web folders to create site collections. To review the current managed paths use the `Get-SPManagedPaths` cmdlet with the *`-WebApplication`* parameter as shown in Figure 7-7.

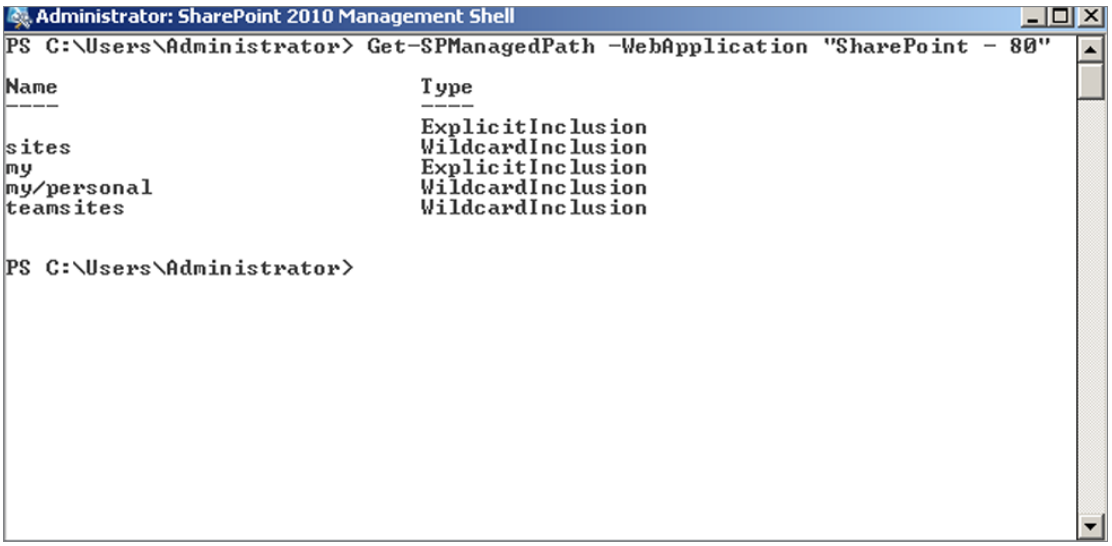


FIGURE 7-7: Displaying the current managed paths

You may use the `New-SPManagedPath` cmdlet to create a new managed path on your farm. The `New-SPManagedPath` cmdlet has two syntax options as follows:

```
New-SPManagedPath [-RelativeURL] <String>
-WebApplication <SPWebApplicationPipeBind>
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-Explicit <SwitchParameter>]
[-WhatIf [<SwitchParameter>]]
[<CommonParameters>]

New-SPManagedPath [-RelativeURL] <String>
-HostHeader <SwitchParameter>
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-Explicit <SwitchParameter>]
[-WhatIf [<SwitchParameter>]]
[<CommonParameters>]
```

Web Application Management

When the SharePoint farm is up and running, the logical starting point is the web application. The first step in validating any new farm, aside from visiting Central Admin, is getting a basic site up

www.it-ebooks.info

and running. Here you can start with the web application, create a site collection, and finally a site. The first cmdlet you need to be familiar with is `New-SPWebApplication`:

```
New-SPWebApplication -ApplicationPool <String> -Name <String>
[-AdditionalClaimProvider <SPClaimProviderPipeBind[]>]
[-AllowAnonymousAccess <SwitchParameter>]
[-ApplicationPoolAccount <SPProcessAccountPipeBind>]
[-AssignmentCollection <SPAssignmentCollection>]
[-AuthenticationMethod <String>]
[-AuthenticationProvider <SPAuthenticationProviderPipeBind[]>]
[-Confirm [<Switch Parameter>]]
[-DatabaseCredentials <PSCredential>]
[-DatabaseName <String>]
[-DatabaseServer <String>]
[-HostHeader <String>]
[-Path <String>]
[-Port <UInt32>]
[-SecureSocketsLayer <SwitchParameter>]
[-ServiceApplicationProxyGroup <SPServiceApplicationProxyGroupPipeBind>]
[-SignInRedirectProvider <SPTrustedIdentityTokenIssuerPipeBind>]
[-SignInRedirectURL <String>]
[-URL <String>]
[-WhatIf [<SwitchParameter>]]
[<CommonParameters>]
```

The cmdlet has quite a few parameters. One of the parameters is the previously mentioned *`AssignmentCollection`*. In this case, create a new web application and a new application pool to go along with it. To create a new application pool, just pass in a name that does not already exist with the *`ApplicationPool`* parameter. The *`ApplicationPoolAccount`* parameter is required when creating a new application pool. The web application will be set up for NTLM, which is the default, so there is no need for the `Authentication` parameter.

Web Application Settings

You can configure many settings on a web application. These may all be modified using a single cmdlet, `Set-SPWebApplication`. This cmdlet is so versatile; it has three distinct parameter sets for accomplishing various configuration goals:

```
Set-SPWebApplication [-Identity] <SPWebApplicationPipeBind>
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-DefaultQuotaTemplate <String>]
[-DefaultTimeZone <Int32>]
[-Force <SwitchParameter>]
[-ServiceApplicationProxyGroup <SPServiceApplicationProxyGroupPipeBind>]
[-WhatIf [<SwitchParameter>]]
[<CommonParameters>]


Set-SPWebApplication [-Identity] <SPWebApplicationPipeBind>
-Zone <Default | Intranet | Internet | Custom | Extranet>
[-AdditionalClaimProvider <SPClaimProviderPipeBind[]>]
[-AssignmentCollection <SPAssignmentCollection>]
```

www.it-ebooks.info

```
[-AuthenticationMethod <String>]
[-AuthenticationProvider <SPAuthenticationProviderPipeBind[]>]
[-Confirm [<SwitchParameter>]]
[-Force <SwitchParameter>]
[-SignInRedirectProvider <SPTrustedIdentityTokenIssuerPipeBind>]
[-SignInRedirectURL <String>]
[-WhatIf [<SwitchParameter>]]
[<CommonParameters>]

Set-SPWebApplication [-Identity] <SPWebApplicationPipeBind>
-SMTPServer <String>
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-Force <SwitchParameter>]
[-OutgoingEmailAddress <String>]
[-ReplyToEmailAddress <String>]
[-WhatIf [<SwitchParameter>]]
[<CommonParameters>]
```

Use the first parameter set to configure the default quota template and default time zone. You can use the second parameter set to configure the web application zone and authentication settings. Use the last parameter set to configure the email settings of the web application.

Site Management

Site management involves the administration of site collections and subsites (or webs in the SharePoint world). Similar to the SharePoint object model, `SPSite` refers to site collections, and `SPWeb` refers to subsites within a site collection. Using these noun commands with various verb prefixes allows you to configure and modify sites and webs. This section explains various actions that you can perform using these cmdlets.

Site Collections

You may display a list of all site collections on your SharePoint farm by using the `Get-SPSite` cmdlet. You may use `Get-SPSite` without any parameters to display all site collections, as shown in Figure 7-8.

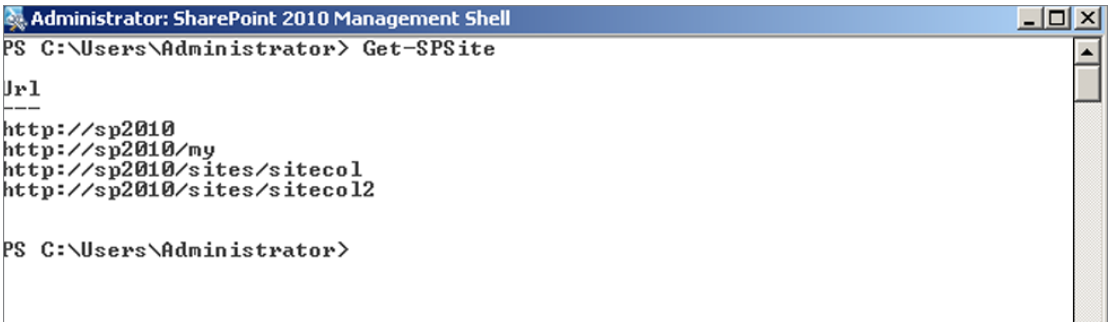


FIGURE 7-8: Displaying all site collections on the farm

www.it-ebooks.info

The `Get-SPSite` cmdlet has various parameter sets that you can use to further refine the site collections returned. Using the *`-WebApplication`* parameter and providing the web application displays all site collections within the specified web application. You may perform a similar action by using the *`-ContentDatabase`* parameter and providing the name of a specific content database. This displays all site collections stored within the specified content database.

You may use the `New-SPSite` cmdlet to create a new site collection. Initially you may use this to create the root site collection; however, subsequent site collections must be created within a managed path. The `New-SPSite` cmdlet has the following syntax:

```
New-SPSite [-URL] <String> -OwnerAlias <SPUserPipeBind>
[-AdministrationSiteType <None | TenantAdministration>]
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-ContentDatabase <SPContentDatabasePipeBind>]
[-Description <String>]
[-HostHeaderWebApplication <SPWebApplicationPipeBind>]
[-Language <UInt32>]
[-Name <String>]
[-OwnerEmail <String>]
[-QuotaTemplate <SPQuotaTemplatePipeBind>]
[-SecondaryEmail <String>]
[-SecondaryOwnerAlias <SPUserPipeBind>]
[-SiteSubscription <SPSiteSubscriptionPipeBind>]
[-Template <SPWebTemplatePipeBind>]
[-URL <String>]
[-WhatIf [<SwitchParameter>]]
[<CommonParameters>]
```

Although this provides great flexibility in generating a new site collection, you may easily create a new site collection using defaults by providing only the *`URL`*, *`Name`*, *`Owner`*, and *`Template`*:

```
New-SPSite –URL http://sp2010/sites/newsite -name NewSite
-owneralias SP\NewSiteAdmin -template STS#0
```

You can find a list of available templates by issuing a `Get-SPWebTemplates` command.

The settings for a site collection can be administered by using the `Set-SPSite` cmdlet. This cmdlet has the following two syntaxes:

```
Set-SPSite [-Identity] <SPSitePipeBind>
[-AdministrationSiteType <None | TenantAdministration>]
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-LockState <String>]
[-MaxSize <Int64>]
[-OwnerAlias <SPUserPipeBind>]
[-QuotaTemplate <SPQuotaTemplatePipeBind>]
[-SecondaryOwnerAlias <SPUserPipeBind>]
[-Template <SPWebTemplatePipeBind>]
[-URL <String>]
[-UserAccountDirectoryPath <String>]
[-WarningSize <Int64>]
[-WhatIf [<SwitchParameter>]]
```

www.it-ebooks.info

```
[<CommonParameters>]

Set-SPSite [-Identity] <SPSitePipeBind>
[-AdministrationSiteType <None | TenantAdministration>]
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-Force <SwitchParameter>]
[-SiteSubscription <SPSiteSubscriptionPipeBind>]
[-WhatIf [<SwitchParameter>]]
[<CommonParameters>]
```

You may use `Set-SPSite` to change settings on a specific site collection (identified by the *`-Identity`* parameter) including the URL, the owner, lock state, and max size.

Subsites (Webs)

You may retrieve subsites available using the `Get-SPWeb` cmdlet. This cmdlet can return all subsites by using the *`-Identity`* parameter or all webs (including the site collection web) by using the *`-Site`* parameter. The syntax for `Get-SPWeb` is as follows:

```
Get-SPWeb [[-Identity] <SPWebPipeBind>]
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-Filter <ScriptBlock>]
[-Limit <String>]
[-Regex <SwitchParameter>]
[-Site <SPSitePipeBind>]
[-WhatIf [<SwitchParameter>]]
[<CommonParameters>]
```

Similar to creating a new site collection, you may create a new subsite by using the `New-SPWeb` cmdlet. You need to provide the site template as you did with the `New-SPSite` cmdlet. The `New-SPWeb` syntax is as follows:

```
New-SPWeb [-URL] <String>
[-AddToQuickLaunch <SwitchParameter>]
[-AddToTopNav <SwitchParameter>]
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-Description <String>]
[-Language <UInt32>]
[-Name <String>]
[-Template <SPWebTemplatePipeBind>]
[-UniquePermissions <SwitchParameter>]
[-UseParentTopNav <SwitchParameter>]
[-WhatIf [<SwitchParameter>]]
[<CommonParameters>]
```

Some interesting parameters are available when creating a subsite. Ensure that the new web is added to the Quick Launch (*`-AddToQuickLaunch`*) and the top navigation menu *`(-AddToTopNav`*). The *`-UseParentTopNav`* makes sure that the site collection top navigation is replicated within the new subsite. Using the *`-UniquePermissions`* switch parameter automatically breaks inheritance from the site collection and provides the ability to administer separate user/group permissions on the subsite.

www.it-ebooks.info

Using the `Set-SPWeb` cmdlet allows you to configure web settings on subsites similar to `Set-SPSite`. The `Set-SPWeb` syntax is as follows:

```
Set-SPWeb [-Identity] <SPWebPipeBind>
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-Description <String>]
[-RelativeURL <String>]
[-Template <SPWebTemplatePipeBind>]
[-Theme <String>]
[-WhatIf [<SwitchParameter>]]
[<CommonParameters>]
```

You may use `Set-SPWeb` to modify the URL, description, template, and theme of the specified subsite.

Solutions and Features

Most custom-coded SharePoint functionality is packaged up and deployed to the farm as a solution that may in turn produce several features that may be activated within a site or site collection. Previously, installing and deploying solutions and features was performed using `stsadm`. Now, these activities can be facilitated using PowerShell commands.

Solution Management

The first step to deploy a solution is adding to the farm. You can easily accomplish this by using the `Add-SPSolution` cmdlet, which in its simplest form needs only the path of the solution file (.wsp).

```
Add-SPSolution -LiteralPath <String>
```

Because the `-LiteralPath` is required and is the first parameter, you can simply omit the `-LiteralPath` and provide the path to the solution file directly, as shown in Figure 7-9.

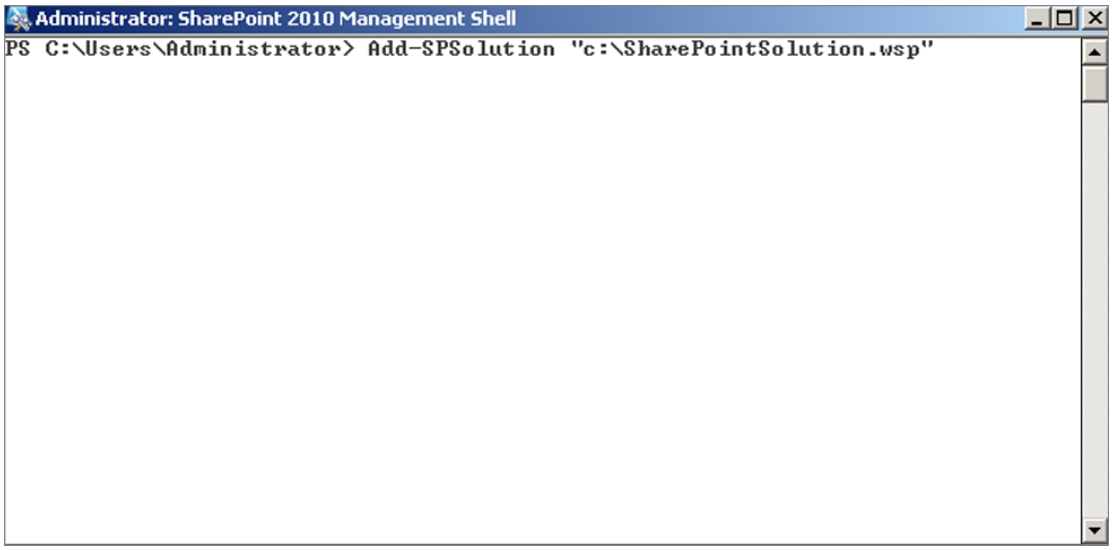


FIGURE 7-9: Adding a solution to the SharePoint farm

www.it-ebooks.info

To make sure the solution is installed, you may use `Get-SPSolution` to list out all solutions within the farm, as shown in Figure 7-10.

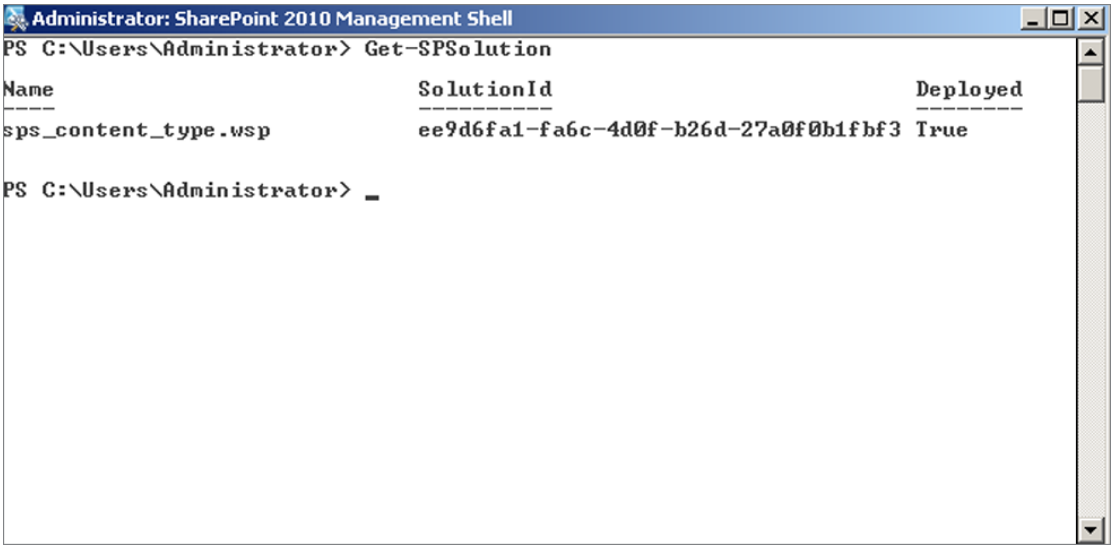


FIGURE 7-10: Displaying solutions on the farm

After adding a solution to the farm, you need to deploy the solution to the intended web application(s). Use the `Install-SPSolution` cmdlet.

```
Install-SPSolution [-Identity] <SPSolutionPipeBind>
[-AllWebApplications <SwitchParameter>]
[-AssignmentCollection <SPAssignmentCollection>]
[-CASPolicies <SwitchParameter>]
[-Confirm [<SwitchParameter>]]
[-Force <SwitchParameter>]
[-GACDeployment <SwitchParameter>]
[-Language <UInt32>]
[-Local <SwitchParameter>]
[-Time <String>]
[-WebApplication <SPWebApplicationPipeBind>]
[-WhatIf [<SwitchParameter>]]
[<CommonParameters>]
```

You may deploy the solution to all available web applications by using the *`-AllWebApplications`* switch parameter, or you may specify a specific web application using the *`-WebApplication`* parameter. If the solution needs to deploy assemblies into the GAC, you must use the *`-GACDeployment`* parameter. When custom permission sets are provided, the *`-CASPolicies`* parameter must be used.

www.it-ebooks.info

Feature Management

Features have various scopes depending on how they are configured. They may be installed at the overall farm, web application, site collection, or site (web) level. To retrieve a list of features at various levels, you may use the `Get-SPFeature` cmdlet.

```
Get-SPFeature [[-Identity] <SPFeatureDefinitionPipeBind>]
[-Limit <String>]

Get-SPFeature [[-Identity] <SPFeatureDefinitionPipeBind>]
[-Farm <SwitchParameter>] [-Limit <String>]

Get-SPFeature [[-Identity] <SPFeatureDefinitionPipeBind>]
[-Limit <String>] [-Sandboxed <SwitchParameter>]
[-Site <SPSitePipeBind>] [<CommonParameters>]

Get-SPFeature [[-Identity] <SPFeatureDefinitionPipeBind>]
[-Limit <String>] [-Web <SPWebPipeBind>]

Get-SPFeature [[-Identity] <SPFeatureDefinitionPipeBind>]
[-Limit <String>] [-WebApplication <SPWebApplicationPipeBind>]
```

There is a separate parameter set for each level. Use the first set to retrieve a specific feature using the *`-Identity`* parameter. For a site collection use the *`-Site`* parameter and provide the URL of the site collection. For the site collection, there is a *-Sandboxed* switch parameter. This enables you to retrieve sandboxed features from the site collection.

Use the `-Web` along with the URL of the specific site to retrieve features activated at the site level. For web application level features, use the *`-WebApplication`* parameter and provide either the root site URL hosted by the web application or the name of the web application, as shown in Figure 7-11.

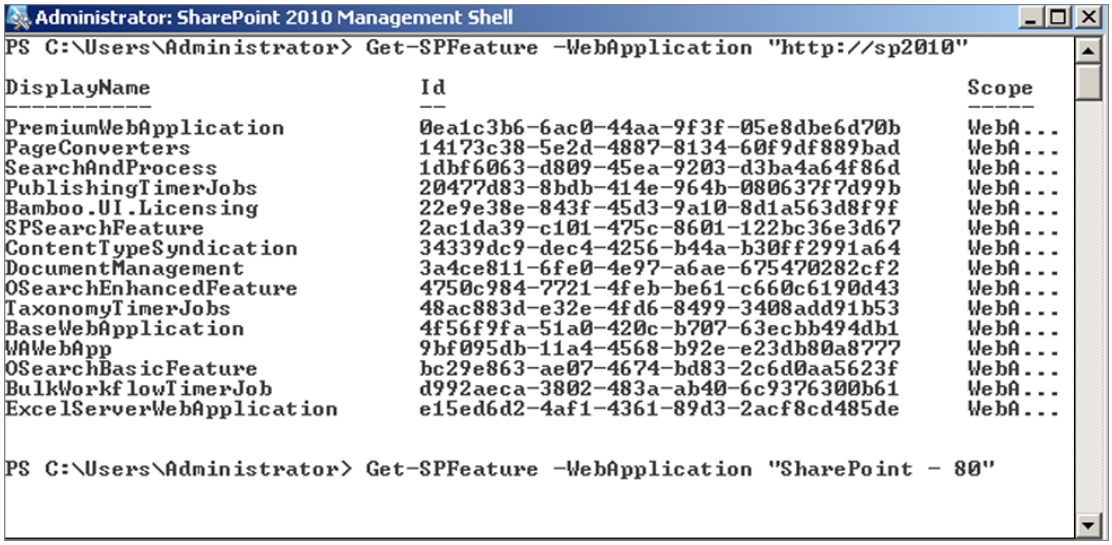


FIGURE 7-11: Displaying features for a web application

www.it-ebooks.info

To install a feature, use the `Install-SPFeature` cmdlet and provide the folder name of the feature to be installed using the *`-Path`* parameter. The feature files must have been placed within the 14\TEMPLATES\FEATURES folder.

```
Install-SPFeature -Path "Custom SharePoint Site Feature"
```

This installs the feature inside SharePoint at the scoped level. The feature is installed but not activated. To activate a feature, use the `Enable-SPFeature` cmdlet, and provide the URL of the location where it should be activated. Conversely, use `Disable-SPFeature` to deactivate a feature.

Enterprise Content Management

Content management and deployment is commonly used when SharePoint hosts both internally facing sites (intranet) and externally facing sites (Internet). The content is created internally first and then, when approved, is deployed out to the public-facing Web site. This involves setting up paths and jobs between two different Central Administration sites.

Content Deployment Paths

When you create a new content deployment path, you are essentially mapping the intranet site to the Internet site. This determines the source of the content and the destination when it is pushed out. To create a new deployment path, use the `New-SPContentDeploymentPath` cmdlet.

```
New-SPContentDeploymentPath
-DestinationCentralAdministrationURL <Uri>
-DestinationSPSite <String>
-DestinationSPWebApplication <Uri> -Name <String>
-PathAccount <PSCredential>
-SourceSPSite <SPSitePipeBind>
-SourceSPWebApplication <SPWebApplicationPipeBind>
[-Authentication <Basic | Windows>]
[-CompressionEnabled <SwitchParameter>]
[-Confirm [<SwitchParameter>]]
[-DeploySecurityInformation <None | WssOnly | All>]
[-DeployUserNamesEnabled <SwitchParameter>]
[-Description <String>] [-EventReceiversEnabled <SwitchParameter>]
[-KeepTemporaryFilesOptions <Never | Always | Failure>]
[-PathEnabled <SwitchParameter>]
```

The *`-PathAccount`* parameter needs a SharePoint account that has full access to the destination site and web application. The account should also have administrator rights on the destination Central Administration site.

Content Deployment Jobs

After a deployment path is configured, a deployment job needs to be created to facilitate the promotion of content from one site to another (as determined by the content deployment path). You can use the `New-SPContentDeploymentJob` cmdlet to create a new content deployment job.

```
New-SPContentDeploymentJob -Name <String>
-SPContentDeploymentPath <SPContentDeploymentPathPipeBind>
```

www.it-ebooks.info

```
[-Confirm [<SwitchParameter>]]
[-Description <String>] [-EmailAddresses <String[]>]
[-EmailNotifications <None | SendEmailOnFailure |
   SendEmailOnSuccess | SendEmailOnAlways>]
[-HostingSupportEnabled <SwitchParameter>]
[-IncrementalEnabled <SwitchParameter>]
[-Schedule <String>] [-ScheduleEnabled <SwitchParameter>]
[-Scope <SPWebPipeBind[]>]
[-SqlSnapshotSetting <None | CreateNew>]
[-WhatIf [<SwitchParameter>]] [<CommonParameters>]
```

The tricky parameter here is *-Schedule*, which takes a string entry. The string must be a valid timer job schedule in any of the following forms:

- Every 5 minutes between 0 and 59
- Hourly between 0 and 59
- Daily at 15:00:00
- Weekly between Fri 22:00:00 and Sun 06:00:00
- Monthly at 15 15:00:00
- Yearly at Jan 1 15:00:00

SharePoint Database Management

This section describes how you can use PowerShell to perform SharePoint database management. One key aspect of SharePoint is the storage of content and configuration information. SharePoint wouldn't be SharePoint without the underlying databases.

To start, use the `Get-SPDatabase` cmdlet to list out all SharePoint databases on a specified SQL Server instance:

```
Get-SPDatabase -ServerInstance "SharePointSQLServer"
```

To obtain a specific database to perform operations, you need to assign a variable to the specific instance of the wanted SharePoint database by using the identity GUID of the database entry along with the `Get-SPDatabase` cmdlet:

```
$dbVar = Get-SPDatabase -Identity 58134294-6b05-22cd-8354-8cd09e43bbc5
```

Content Databases

The content databases store the SharePoint content contained in one or more site collections. Each web application may have multiple content databases. To create a new content database for a specific web application, use the `New-SPContentDatabase` cmdlet:

```
New-SPContentDatabase -Name NewContentDatabaseName
-WebApplication "WebApplicationName"
```

www.it-ebooks.info

To view the new content database listing and all the other content databases for a specific web application, simply use the `Get-SPContentDatabase` cmdlet:

```
Get-SPContentDatabase -WebApplication "WebApplicationName"
```

If you need to remove a content database from a web application, you can use the `Dismount-SPContentDatabase` cmdlet. This does not delete the database but simply "detaches" it from the web application. This is a case where you may want to use a database variable as previously discussed in the beginning of this section:

```
$dbVar = Get-SPContentDatabase -Identity 0c5f3219-eb87-629c-876f-ecc5b0a99db5
Dismount-SPContentDatabase $dbVar
```

Notice here you use the `Get-SPContentDatabase` versus the generic `Get-SPDatabase`. This is good practice because if the database identified by the identity parameter is not an actual content database, the `dismount` command will fail.

You can use the `Mount-SPContentDatabase` to perform the opposite operation, which attaches a content database to a specific web application:

```
Mount-SPContentDatabase -Name "NewContentDatabase"
-WebApplication "WebApplicationName"
```

To completely remove and delete a content database from the SQL Server database server, after dismounting the database from the web application, use the `Remove-SPContentDatabase` cmdlet:

```
$dbVar = Get-SPContentDatabase -Identity 0c5f3219-eb87-629c-876f-ecc5b0a99db5
Remove-SPContentDatabase $dbVar
```

Configuration Databases

The configuration databases store all the settings for the SharePoint farm, web application, or service application. However, when using the SharePoint PowerShell cmdlets for configuration databases, they refer to the SharePoint farm configuration database. To create a new configuration database, use the `New-SPConfigurationDatabase` cmdlet:

```
New-SPConfigurationDatabase -DatabaseServer "SharePointSQLServer"
-DatabaseName NewConfigurationDatabase
```

Deleting the current configuration database instance is too easy. All you need to do is execute the `Remove-SPConfigurationDatabase` cmdlet without any parameters. Because the cmdlet removes only the current configuration database and there can be only one SharePoint farm configuration database associated to the instance, there are no identity parameters.

Backup and Restore

Probably the most common database operations that you need to perform are backups and restores. There are various SharePoint 2010 PowerShell commands to back up and restore configuration databases, site collections, and the entire farm itself:

- `Backup-SPConfigurationDatabase`
- `Backup-SPFarm`

www.it-ebooks.info

- `Backup-SPSite`
- `Restore-SPFarm`
- `Restore-SPSite`

There is no restore cmdlet for the configuration database. To restore a configuration database, you need to use the `Restore-SPFarm` cmdlet with the *`ConfigurationOnly`* switch parameter:

```
Restore-SPFarm -Directory "H:\Backups" -ConfigurationOnly -RestoreMethod Overwrite
```

SEARCH ADMINISTRATION WITH POWERSHELL

Scripting out Search-related operations in PowerShell saves time when trying to replicate configurations across environments — especially when creating new managed properties. Two sets of PowerShell cmdlets involve the Search Service in SharePoint: SharePoint Foundation Search and Enterprise Search. The Enterprise Search functionality is available only with the enterprise license of SharePoint 2010.

SharePoint Foundation Search

If you run SharePoint 2010 Foundation or SharePoint 2010 Standard Edition, your farm is set up to use the SharePoint Foundation Search. This is facilitated by the Search Service application.

You may view the Search Service application by issuing a `Get-SPSearchService` command in the PowerShell window, as shown in Figure 7-12.

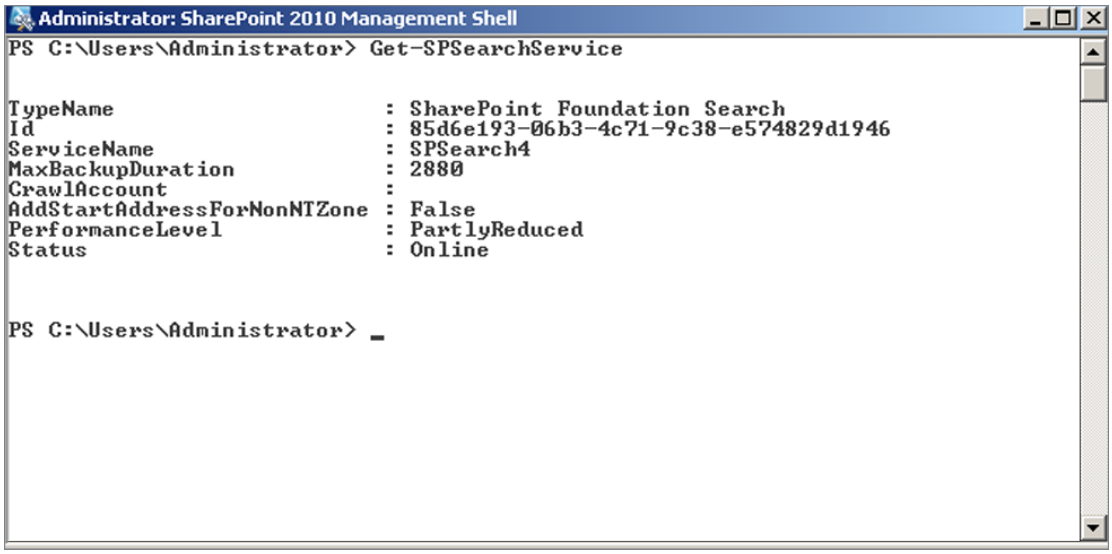


FIGURE 7-12: Displaying the Search Service

www.it-ebooks.info

You need the identity of the Search Service to configure the various settings for SharePoint Foundation Search. There is only one cmdlet that you can use to configure these settings: `Set-SPSearchService`.

```
Set-SPSearchService [[-Identity] <SPSearchServicePipeBind>]
[-CrawlAccount <String>] [-CrawlPassword <SecureString>]
[-AddStartAddressForNonNTZone <String>]
[-MaxBackupDuration <String>]
[-PerformanceLevel <String>]
[-Verbose] [-Debug]
[-ErrorAction <ActionPreference>]
[-WarningAction <ActionPreference>]
[-ErrorVariable <String>]
[-WarningVariable <String>]
[-OutVariable <String>]
[-OutBuffer <Int32>]
```

You may use `Set-SPSearchService` to set the crawl account, configure the maximum backup duration, or tweak the performance level setting.

Enterprise Search

The Enterprise Search Service runs within the Enterprise Edition of SharePoint 2010. This service provides more capabilities in crawling content and providing search results accordingly.

Similar to the SharePoint Foundation Search, you need the identity of the Enterprise Search Service to configure enterprise search settings. To retrieve the Enterprise Search Service identity, issue the `Get-SPEnterpriseSearchService` command. Use the identity provided in the listing to use in the `Set-SPEnterpriseSearchService`.

```
Set-SPEnterpriseSearchService
[[-Identity] <SearchServicePipeBind>]
[-AcknowledgementTimeout <String>]
[-Confirm [<SwitchParameter>]]
[-ConnectionTimeout <String>]
[-ContactEmail <String>]
[-IgnoreSSLWarnings <String>]
[-InternetIdentity <String>]
[-PerformanceLevel <String>]
[-ProxyType <String>]
[-ServiceAccount <String>]
[-ServicePassword <SecureString>]
```

You may use `Set-SPEnterpriseSearchService` to set the crawl account (service account), configure various timeout durations, or tweak the performance level setting of the Enterprise Search Service.

One of the better ways to use the Enterprise Search cmdlets is to script out the creation of crawled and managed properties. To perform these tasks, you first need the actual Enterprise Search Service Application reference. Use the `Get-SPEnterpriseSearchServiceApplication` cmdlet to obtain this information, as shown in Figure 7-13.

www.it-ebooks.info

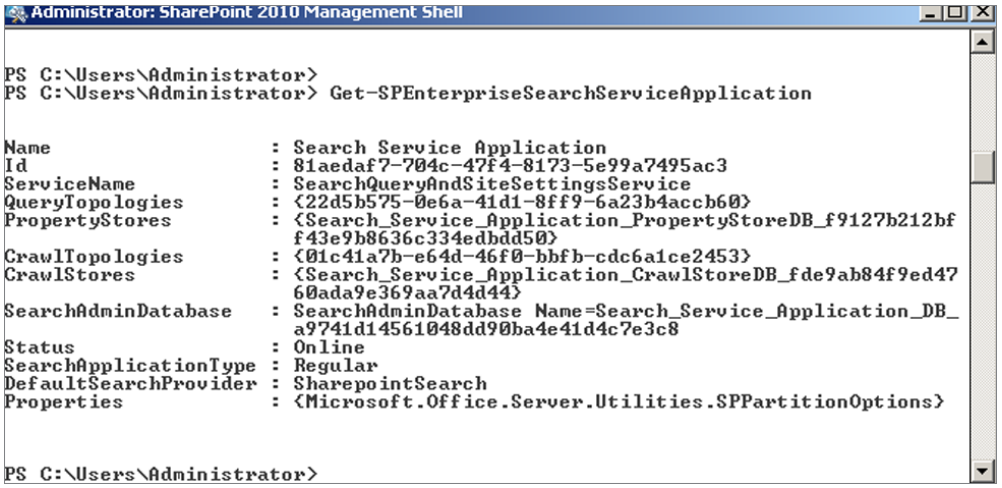


FIGURE 7-13: Displaying the Enterprise Search Service Application

To create new search fields for SharePoint, you need to create crawled properties and managed properties and then provide a metadata mapping between the two. Crawled properties are automatically created when you run a full crawl against a new content source. However, with PowerShell scripting, you no longer need to wait until a full crawl is completed to create a new crawled property. You may use the `New-SPEnterpriseSearchMetadataCrawledProperty` cmdlet to create a new crawled property.

```
New-SPEnterpriseSearchMetadataCrawledProperty
-Category <CategoryPipeBind>
-IsNameEnum <$true | $false>
-Name <String>
-PropSet <Guid>
-SearchApplication <SearchServiceApplicationPipeBind>
-VariantType <Int32>
[-IsMappedToContents <Nullable>]
```

You may use the `Get-SPEnterpriseSearchMetadataCategory` to find a category or use `New-SPEnterpriseSearchMetadataCategory` to create a new custom category.

The *`-IsNameEnum`* switch parameter is required and determines if the crawled property contains integer values but is being converted to a string value.

The *`-PropSet`* parameter requires the property set associated with a category and its crawled properties. Use the `Get-SPEnterpriseSearchCrawledProperty` cmdlet to find the property set with crawled properties that are similar to the one you are creating. The most common property set is 00130329-0000-0130-c000-000000131346.

The *`-VariantType`* parameter determines the type of data stored within the crawled property. The *`VariantType`* values can be any of the following integers:

- 11: Yes/No (boolean)
- 12: - Binary Data

www.it-ebooks.info

- 20: Integer
- 31: Text (string)
- 64: Date and Time

The next step is to create a managed property that can be mapped to the crawled property. To create a managed property, use the `New-SPEnterpriseSearchMetadataManagedProperty` cmdlet.

```
New-SPEnterpriseSearchMetadataManagedProperty
-Name <String>
-SearchApplication <SearchServiceApplicationPipeBind>
-Type <Int32>
[-DefaultForQueryIndependentRank <Nullable>]
[-Description <String>]
[-EnabledForQueryIndependentRank <Nullable>]
[-EnabledForScoping <Nullable>]
[-FullTextQueriable <Nullable>]
[-IncludeInMd5 <Nullable>]
[-NameNormalized <Nullable>]
[-NoWordBreaker <Nullable>]
[-RemoveDuplicates <Nullable>]
[-RespectPriority <Nullable>]
[-Retrievable <Nullable>]
[-UserFlags <Nullable>]
```

Although many optional parameters exist, the required parameters for a managed property aren't as complicated as creating a crawled property. You essentially need only a name for the management property, the search application identity, and the type. The *`-Type`* parameter determines the type of value for the property similar to the *`-VariantType`*. The *`-Type`* may be any of the following integers:

- 1: Text
- 2: Integer
- 3: Decimal
- 4: Date/Time
- 5: Yes/No
- 6: Binary

Now you need to map the crawled property to the managed property by using the `New-SPEnterpriseSearchMetadataMapping` cmdlet. Provide this cmdlet with the search application, the crawled property, and the managed property, and the mapping is set.

It is easier to understand how to use all these property cmdlets together by viewing them in action. Following is an example script that came from a larger needed script to easily re-create search properties in the QA and Production environments.

```
$searchapp = Get-SPEnterpriseSearchServiceApplication "Search Service Application"
$category = Get-SPEnterpriseSearchMetadataCategory
```

www.it-ebooks.info

```
-Identity SharePoint -SearchApplication $searchapp


$crawledproperty = New-SPEnterpriseSearchMetadataCrawledProperty
-SearchApplication $searchapp -Category $category -VariantType 31
-PropSet "00130329-0000-0130-c000-000000131346"
-Name ows_ReportType -IsNameEnum $false

$managedproperty = New-SPEnterpriseSearchMetadataManagedProperty
-SearchApplication $searchapp -Name ReportType -Type 1
-EnabledForScoping 1 -Retrievable 1

New-SPEnterpriseSearchMetadataMapping -SearchApplication $searchapp
-ManagedProperty $managedproperty -CrawledProperty $crawledproperty

$crawledproperty = New-SPEnterpriseSearchMetadataCrawledProperty
-SearchApplication $searchapp -Category $category -VariantType 31
-PropSet "00130329-0000-0130-c000-000000131346"
-Name ows_ReportDescription -IsNameEnum $false

$managedproperty = New-SPEnterpriseSearchMetadataManagedProperty
-SearchApplication $searchapp -Name ReportDescription -Type 1
-EnabledForScoping 1 -Retrievable 1

New-SPEnterpriseSearchMetadataMapping -SearchApplication $searchapp
-ManagedProperty $managedproperty -CrawledProperty $crawledproperty

$crawledproperty = New-SPEnterpriseSearchMetadataCrawledProperty
-SearchApplication $searchapp -Category $category -VariantType 31
-PropSet "00130329-0000-0130-c000-000000131346"
-Name ows_ArchiveDate -IsNameEnum $false

$managedproperty = New-SPEnterpriseSearchMetadataManagedProperty
-SearchApplication $searchapp -Name ArchiveDate -Type 1
-EnabledForScoping 1 -Retrievable 1

New-SPEnterpriseSearchMetadataMapping -SearchApplication $searchapp
-ManagedProperty $managedproperty -CrawledProperty $crawledproperty
```

When creating a script, you need to assign only the category and search application variable once. The rest of the script is the generation of each crawled property and managed property and assigning the result to a variable that then is used in the `New-SPEnterpriseSearchMetadataMapping` cmdlet to perform the mapping between the two.

SERVICE APPLICATIONS

Each service application available in SharePoint 2010 runs as its own process; therefore, each service application has its own set of PowerShell cmdlets. Some are similar in behavior or function, whereas others are unique to the service application they belong. This section touches upon each service application and associated cmdlets.

www.it-ebooks.info

Access Services

Access Services provides the `Set-SPAccessServiceApplication` cmdlet to configure various settings. You may use the name of the Access Services service application as the identity (instead of the GUID). To verify the name use `Get-SPAccessServiceApplication`.

```
Set-SPAccessServiceApplication
-Identity <SPAccessServiceApplicationPipeBind>
[-ApplicationLogSizeMax <Int32>]
[-CacheTimeout <Int32>] [-ColumnsMax <Int32>]
[-Confirm [<SwitchParameter>]]
[-NonRemotableQueriesAllowed <SwitchParameter>]
[-OrderByMax <Int32>] [-OuterJoinsAllowed <SwitchParameter>]
[-OutputCalculatedColumnsMax <Int32>]
[-PrivateBytesMax <Int32>] [-RecordsInTableMax <Int32>]
[-RequestDurationMax <Int32>]
[-RowsMax <Int32>]
[-SessionMemoryMax <Int32>]
[-SessionsPerAnonymousUserMax <Int32>]
[-SessionsPerUserMax <Int32>]
[-SourcesMax <Int32>]
[-TemplateSizeMax <Int32>]
```

Excel Services

Excel Services is similar to Access Services because there is a `Set-SPExcelServiceApplication` cmdlet available to configure the service application settings. You may also use the service application name as the identity. Use the `Get-SPExcelServiceApplication` cmdlet to verify the name.

```
Set-SPExcelServiceApplication
[[-Identity] <SPExcelServiceApplicationPipeBind>]
[-CachingOfUnusedFilesEnable <SwitchParameter>]
[-Confirm [<SwitchParameter>]]
[-CrossDomainAccessAllowed <SwitchParameter>]
[-EncryptedUserConnectionRequired <None | Connection>]
[-ExternalDataConnectionLifetime <Int32>]
[-FileAccessMethod <UseImpersonation | UseFileAccessAccount>]
[-LoadBalancingScheme <WorkbookURL | RoundRobin | Local>]
[-MemoryCacheThreshold <Int32>]
[-PrivateBytesMax <Int32>]
[-SessionsPerUserMax <Int32>]
[-SiteCollectionAnonymousSessionsMax <Int32>]
[-TerminateProcessOnAccessViolation <SwitchParameter>]
[-ThrottleAccessViolationsPerSiteCollection <Int32>]
[-UnattendedAccountApplicationId <String>]
[-UnusedObjectAgeMax <Int32>]
[-WhatIf [<SwitchParameter>]]
[-WorkbookCache <String>]
[-WorkbookCacheSizeMax <Int32>]
```

In addition to the main `SPExcelServiceApplication` set of commands, there are many others to manage trusted locations, data connections, data providers, and so on. These are conveniently shown in Table 7-1.

www.it-ebooks.info

TABLE 7-1: Excel Services Commands

NOUN	VERBS AVAILABLE
SPExcelBlockedFileType	Get, New, Remove
SPExcelDataConnectionLibrary	Get, New, Remove, Set
SPExcelDataProvider	Get, New, Remove, Set
SPExcelFileLocation	Get, New, Remove, Set
SPExcelUserDefinedFunction	Get, New, Remove, Set

InfoPath Forms Services

The InfoPath Forms Services service application (or Forms Services) also has a main cmdlet that you can use to modify the main settings of the service application: `Set-SPInfoPathFormsService`.

```
Set-SPInfoPathFormsService
-Identity <SPFormsServicePipeBind>
[-ActiveSessionTimeout <Nullable>]
[-AllowEmbeddedSqlForDataConnections <String>]
[-AllowUdcAuthenticationForDataConnections <String>]
[-AllowUserFormBrowserEnabling <String>]
[-AllowUserFormBrowserRendering <String>]
[-AllowUserFormCrossDomainDataConnections <String>]
[-AllowViewState <String>]
[-DefaultDataConnectionTimeout <Nullable>]
[-MaxDataConnectionResponseSize <Nullable>]
[-MaxDataConnectionRoundTrip <Nullable>]
[-MaxDataConnectionTimeout <Nullable>]
[-MaxFormLoadTime <Nullable>]
[-MaxPostbacksPerSession <Nullable>]
[-MaxSizeOfUserFormState <Nullable>]
[-MaxUserActionsPerPostback <Nullable>]
[-MemoryCacheSize <Nullable>]
[-RequireSslForDataConnections <String>]
[-ViewStateThreshold <Nullable>]
```

InfoPath Forms Services also provides cmdlets to verify and install administrator-approved form templates. These are InfoPath forms that require full trust, contain code-behind, and need to be installed directly to Forms Services.

You may verify an InfoPath form using the `Test-SPInfoPathFormTemplate` cmdlet and providing the `-Path` parameter with the fully qualified path of the template. When verification is successful, you may install the form template into Forms Services by using the `Install-SPInfoPathFormTemplate` cmdlet. This cmdlet also takes the `-Path` parameter. Conversely, you may use `Uninstall-SPInfoPathFormTemplate` to uninstall a form from Forms Services.

Installing a form template simply makes it available within Forms Services but does not deploy it automatically for use as a site content type. To use the installed form template within a site collection, you must activate the template using `Enable-SPInfoPathFormTemplate` cmdlet.

www.it-ebooks.info

The simplest command line for this cmdlet contains only the *-Identity* and the *-Site* parameters. The *-Identity* is simply the name of the form template installed into Forms Services. The *-Site* is the URL of the site collection where you want to enable the form. Conversely, you may use `Disable-SPInfoPathFormTemplate` with the same parameters to deactivate a form from a site collection.

Metadata Service

The Metadata Service application has one main cmdlet to configure most of the settings for the Metadata Service Application: `Set-SPMetadataServiceApplication`

```
Set-SPMetadataServiceApplication
-Identity <SPMetadataServiceCmdletPipeBind>
[-AdministratorAccount <String>]
[-ApplicationPool <SPIisWebServiceApplicationPoolPipeBind>]
[-AssignmentCollection <SPAssignmentCollection>]
[-CacheTimeCheckInterval <Int32>]
[-Confirm [<SwitchParameter>]]
[-DatabaseCredentials <PSCredential>]
[-DatabaseName <String>]
[-DatabaseServer <String>]
[-DoNotUnpublishAllPackages <SwitchParameter>]
[-FailoverDatabaseServer <String>]
[-FullAccessAccount <String>]
[-HubUri <String>]
[-MaxChannelCache <Int32>]
[-Name <String>]
[-RestrictedAccount <String>]
[-SyndicationErrorReportEnabled <SwitchParameter>]
[-WhatIf [<SwitchParameter>]] [<CommonParameters>]
```

To find the identity of the Metadata Service Application, you need to use the generic `Get-SPServiceApplication` cmdlet and locate the Managed Metadata Services entry. Even though there is a `Get-SPMetadataServiceApplication` cmdlet, it is used to retrieve a specific service application and requires an *-Identity* parameter.

PerformancePoint Services

PerformancePoint Services (PPS) has several cmdlets available to configure the PPS Service Application. The main cmdlet is `Set-SPPerformancePointServiceApplication`.

```
Set-SPPerformancePointServiceApplication
[-Identity] <SPPerformancePointMonitoringServiceApplicationPipeBind>
[-AnalyticQueryCellMax <Int32>]
[-AnalyticQueryLoggingEnabled <$true | $false>]
[-ApplicationCacheEnabled <$true | $false>]
[-ApplicationCacheMinimumHitCount <Int32>]
[-ApplicationPool <SPIisWebServiceApplicationPoolPipeBind>]
[-ApplicationProxyCacheEnabled <$true | $false>]
[-CommentsDisabled <$true | $false>]
[-CommentsScorecardMax <Int32>]
[-Confirm [<SwitchParameter>]]
[-DataSourceQueryTimeoutSeconds <Int32>]
```

www.it-ebooks.info

```
[-DecompositionTreeMaximum <Int32>]
[-ElementCacheSeconds <Int32>]
[-FilterRememberUserSelectionsDays <Int32>]
[-FilterTreeMembersMax <Int32>]
[-IndicatorImageCacheSeconds <Int32>]
[-MSMQEnabled <$true | $false>]
[-MSMQName <String>]
[-SelectMeasureMaximum <Int32>]
[-SessionHistoryHours <Int32>]
[-SettingsDatabase <String>]
[-ShowDetailsInitialRows <Int32>]
[-ShowDetailsMaxRows <Int32>]
[-ShowDetailsMaxRowsDisabled <$true | $false>]
[-TrustedContentLocationsRestricted <$true | $false>]
[-TrustedDataSourceLocationsRestricted <$true | $false>]
[-WhatIf [<SwitchParameter>]] [<CommonParameters>]
```

You may use `Get-SPPerformancePointServiceApplication` or `Get-SPServiceApplication` to obtain the identity of the PPS Service Application.

To create a trusted content location or a trusted data source location, you may use the `New-SPPerformancePointServiceApplicationTrustedLocation` cmdlet.

The *`-TrustedLocationType`* can be specified as either `Content` or `DataSource`. The *`-Type`* value specifies the type of location for the `Content` or the `DataSource` but only certain values are valid for each location type.

```
New-SPPerformancePointServiceApplicationTrustedLocation
-ServiceApplication <SPPerformancePointMonitoringServiceApplicationPipeBind>
-TrustedLocationType <DataSource | Content>
-Type <SiteCollection | Site | List | DocumentLibrary>
-URL <String> [-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-Description <String>]
[-WhatIf [<SwitchParameter>]] [<CommonParameters>]
```

The *`-Type`* values valid for a content location follow:

- `SiteCollection`
- `Site`
- `List`

The *`-Type`* values valid for a data source follow:

- `SiteCollection`
- `Site`
- `DocumentLibrary`

In other words, `List` is valid as a type only if the location type is `Content`, and `DocumentLibrary` is valid as a type only if the location type is `Datasource`.

www.it-ebooks.info

Secure Store Service

The Secure Store Service (SSS) is a service application used in SharePoint 2010 to provide single-sign-on (SSO) capabilities for various applications named target applications. The overall process of using SSS involves securely storing service account credentials that may be used when accessing the configured target applications.

There is a `Set-SPSecureStoreServiceApplication` that may be used to configure the SSS service application. However, you must use the `Get-SPServiceApplication` to obtain the identity of the SSS service application.

```
Set-SPSecureStoreServiceApplication
-Identity <SPServiceApplicationPipeBind>
[-ApplicationPool <SPIisWebServiceApplicationPool>]
[-AssignmentCollection <SPAssignmentCollection>]
[-AuditingEnabled <SwitchParameter>]
[-AuditlogMaxSize <Nullable>]
[-Confirm [<SwitchParameter>]]
[-DatabaseCredentials <PSCredential>]
[-DatabaseName <String>]
[-DatabasePassword <SecureString>]
[-DatabaseServer <String>]
[-DatabaseUsername <String>]
[-FailoverDatabaseServer <String>]
[-Sharing <SwitchParameter>]
[-WhatIf [<SwitchParameter>]] [<CommonParameters>]
```

To create a new SSS entry requires several cmdlets and processes. The new entry requires application fields (used to provide the credentials) and a target application. Therefore the script to generate a new SSS entry involves several command lines and variable creations.

```
$serviceAccount = New-SPSecureStoreApplicationField –Name "SP\ServiceAccount"
–Type WindowsUserName –Masked:$false
$password = New-SPSecureStoreApplicationField –Name "password"
–Type WindowsPassword –Masked:$true
$applicationFields = $serviceAccount, $password
$targetApplication = New-SPSecureStoreTargetApplication –Name "SSSApplication"
–FriendlyName "SSS Application" –ApplicationType Group
New-SPSecureStoreApplication –ServiceContext http://sp2010
–TargetApplication $targetApplication –Fields $applicationFields
```

When creating the application fields, various types may be created. They are as follows:

- `Generic`
- `Key`
- `Password`
- `Pin`
- `Username`
- `WindowsUsername`
- `WindowsPassword`

www.it-ebooks.info

State Service

SharePoint 2010 provides a State Service application to manage the overall session state behavior within the provided SharePoint environment. There is a `Set-SPStateServiceApplication` cmdlet, but that is used only to rename a particular State Service application instance.

```
Set-SPStateServiceApplication
[-Identity] <SPStateServiceApplicationPipeBind>
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-Name <String>]
[-WhatIf [<SwitchParameter>]] [<CommonParameters>]
```

You may use `Get-SPStateServiceApplication` to display all State Service applications running on the farm and therefore retrieve the appropriate *`-Identity`* value.

There are also various database operations that may be performed for the State Service application using the `SPStateServiceDatabase` variations of cmdlets:

- `New-SPStateServiceDatabase`
- `Remove-SPStateServiceDatabase`
- `Get-SPStateServiceDatabase`
- `Set-SPStateServiceDatabase`
- `Mount-SPStateServiceDatabase`
- `Dismount-SPStateServiceDatabase`
- `Initialize-SPStateServiceDatabase`
- `Suspend-SPStateServiceDatabase`
- `Resume-SPStateServiceDatabase`

In addition to managing the State Service Application, you may also manage the actual Session State Service using variations of the `SPSessionStateService` cmdlet:

- `Get-SPSessionStateService`
- `Set-SPSessionStateService`
- `Enable-SPSessionStateService`
- `Disable-SPSessionStateService`

User Profile Service

The User Profile Service is now known as the Profile Service Application within SharePoint 2010. The most common usage of the Profile Service is to synchronize (mainly import) users into SharePoint from Active Directory Services as well as maintain `MySites`. There is a

www.it-ebooks.info

`Set-SPProfileServiceApplication` cmdlet used to configure the Profile Service application settings.

```
Set-SPProfileServiceApplication [-Identity] <SPServiceApplicationPipeBind>
[-ApplicationPool <SPIisWebServiceApplicationPoolPipeBind>]
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-MySiteHostLocation <SPSitePipeBind>]
[-MySiteManagedPath <SPPrefixPipeBind>]
[-Name <String>]
[-ProfileDBCredentials <PSCredential>]
[-ProfileSyncDBCredentials <PSCredential>]
[-SocialDBCredentials <PSCredential>]
[-SyncInstanceMachine <SPServerPipeBind>]
[-WhatIf [<SwitchParameter>]] [<CommonParameters>]
```

You need to use the `Get-SPServiceApplication` to obtain the proper identity of the Profile Service application.

Any ratings, comments, or tags within SharePoint 2010 are managed through the Profile Service and are identified as "social items." Therefore, you may manage (remove) these items by using the `Remove-SPSocialItemByDate` cmdlet.

```
Remove-SPSocialItemByDate
-EndDate <DateTime>
-ProfileServiceApplicationProxy <SPServiceApplicationProxyPipeBind>
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-RemoveComments <$true | $false>]
[-RemoveRatings <$true | $false>]
[-RemoveTags <$true | $false>]
[-SiteSubscription <SPSiteSubscriptionPipeBind>]
[-WhatIf [<SwitchParameter>]] [<CommonParameters>]
```

Visio Services

The Visio Service Application does not have one main cmdlet to configure the service application settings. Instead there are a few different cmdlets used for different purposes.

The `Set-SPVisioServiceApplication` cmdlet, unlike other service applications, is used only to set the application pool for the Visio Service Application.

```
Set-SPVisioServiceApplication
[-Identity] <SPVisioServiceApplicationPipeBind>
[-ServiceApplicationPool] <SPIisWebServiceApplicationPoolPipeBind>
[-AssignmentCollection <SPAssignmentCollection>]
[-Confirm [<SwitchParameter>]]
[-WhatIf [<SwitchParameter>]] [<CommonParameters>]
```

To obtain the identity, use `Get-SPVisioServiceApplication` to retrieve the name and ID. Either one may be used with the *`-Identity`* parameter.

www.it-ebooks.info

Use the `Set-SPVisioExternalData` cmdlet to set up the unattended service account for Visio Services. This requires the proper Secure Store Service entry. The `Set-SPVisioExternalData` cmdlet has the following syntax:

```
Set-SPVisioExternalData
-UnattendedServiceAccountApplicationID <String>
-VisioServiceApplication <SPVisioServiceApplicationPipeBind>
[-AssignmentCollection <SPAssignmentCollection>] [<CommonParameters>]
```

The *-VisioServiceApplication* is essentially the identity of the Visio Service Application. This may be the name or the ID.

The `New-SPVisioSafeDataProvider` may be used to create a new safe data provider that Excel workbooks can access via a data connection.

```
New-SPVisioSafeDataProvider -DataProviderId <String>
-DataProviderType <Int32>
-VisioServiceApplication <SPVisioServiceApplicationPipeBind>
[-AssignmentCollection <SPAssignmentCollection>]
[-Description <String>]
[<CommonParameters>]
```

Again, the *-VisioServiceApplication* is the identity of the Visio Service Application and may be the name or the ID. The *–DataProviderID* is a unique identifier for the data provider. The *–DataProviderType* can be any of the following integer values (based on the type of provider):

- 1: OLE DB
- 2: SQL Server
- 3: ODBC
- 4: ODBC with DSN
- 5: SharePoint List
- 6: Custom

You may configure the performance settings for the Visio Service Application by using the `Set-SPVisioPerformance` cmdlet. These settings determine the maximum thresholds when processing Visio graphics and diagrams.

```
Set-SPVisioPerformance -MaxDiagramCacheAge <Int32>
-MaxDiagramSize <Int64> -MaxRecalcDuration <Int32>
-MinDiagramCacheAge <Int32>
-VisioServiceApplication <SPVisioServiceApplicationPipeBind>
[-AssignmentCollection <SPAssignmentCollection>] [<CommonParameters>]
```

Word Services

Word Services has one main cmdlet to configure all the possible settings for the Word Service Application: `Set-SPWordConversionServiceApplication`:

```
Set-SPWordConversionServiceApplication
-Identity <WordServiceApplicationPipeBind>
```

www.it-ebooks.info

```
[-ActiveProcesses <Nullable>]
[-AddSupportedFormats <String[]>]
[-ApplicationPool <SPIisWebServiceApplicationPoolPipeBind>]
[-AssignmentCollection <SPAssignmentCollection>]
[-ClearSupportedFormats <SwitchParameter>]
[-Confirm [<SwitchParameter>]]
[-ConversionsPerInstance <Nullable>]
[-ConversionTimeout <Nullable>]
[-DatabaseCredential <PSCredential>]
[-DatabaseName <String>]
[-DatabaseServer <String>]
[-DisableBinaryFileScan <SwitchParameter>]
[-DisableEmbeddedFonts <SwitchParameter>]
[-KeepAliveTimeout <Nullable>]
[-MaximumConversionAttempts <Nullable>]
[-MaximumConversionTime <Nullable>]
[-MaximumMemoryUsage <Nullable>]
[-RecycleProcessThreshold <Nullable>]
[-RemoveSupportedFormats <String[]>]
[-TimerJobFrequency <Nullable>]
[-What If [<SwitchParameter>]] [<CommonParameters>]
```

To find the identity of the Word Service Application, use the generic `Get-SPServiceApplication` cmdlet, and locate the Word Automation Services entry.

SUMMARY

PowerShell provides a powerful method of scripting out routine commands using SharePoint cmdlets enabling for the repeatability and consistency across environments. Anything that can be configured visually via Central Administration may be easily executed using a PowerShell cmdlet. Actually, some settings can be modified only using PowerShell.

This chapter concludes Part I, "Using SharePoint Technologies and Exploiting Functionality," of the book. Part II, "SharePoint Solutions," dives into specific SharePoint solutions that provide common customizations requested by organizations that leverage SharePoint within their enterprise.

www.it-ebooks.info

PART II
SharePoint Solutions

- **CHAPTER 8:** UI Design and Branding
- **CHAPTER 9:** Building a Site Provisioning System
- **CHAPTER 10:** Building Scalable Applications in SharePoint
- **CHAPTER 11:** Building a Custom Global Navigation Solution

www.it-ebooks.info

www.it-ebooks.info

8

UI Design and Branding

User interface design and branding is important for SharePoint solutions because it is a representation of the organization's brand and professionalism. Branding in SharePoint 2010 has become easier than with previous versions of SharePoint; however, the type of site you design for depends on the approach you take for branding. There are two types of branding for SharePoint sites: public facing anonymous access sites and authenticated sites (typically intranet sites). This chapter provides a high-level look at branding for SharePoint 2010 including themes, style sheets, master pages, page layouts, and application pages. This chapter also covers the different approaches to branding based on the type of SharePoint site you are branding for.

THEMES

Themes provide a quick way to apply lightweight branding such as colors and fonts to sites, and you can use them with standard site templates or custom master pages. The theme engine in SharePoint 2010 has drastically changed from previous versions of SharePoint. Themes in SharePoint 2007 used CSS and images, whereas the new theme engine in SharePoint 2010 uses Office theme files (`.thmx` format). Two major theme features added to SharePoint 2010 are the capability to inherit a theme from a parent site and the capability to push a theme down to subsites, similar to how master pages work.

A set of predefined themes are available in SharePoint 2010, which you can find at the following locations:

- Stored in a SharePoint Library at `http://siteurl/_catalogs/theme/Forms/AllItems.aspx`
- In your computer's File System at C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\ GLOBAL\Lists\themes

www.it-ebooks.info

Themes modify the following SharePoint elements:

- Ribbon user interface elements
- Site actions buttons
- Drop-down menus
- Shortcut menus (Site Settings links)
- Dialog boxes
- Web Part Chrome
- Highlighting for bulk editing operations
- SPGridView

Creating Themes

A set of predefined themes are available in SharePoint 2010, which can be customized through the browser in SharePoint Server. In SharePoint Foundation, the ability to edit themes in the browser is not available with the exception of changing the assigned theme. You can also create themes using Theme Builder or PowerPoint 2010. Theme Builder is a free tool available for download at `http://connect.microsoft.com/themebuilder`.

Using Themes

Before deciding to use themes for your SharePoint site, the first step is to decide whether themes are an appropriate option for your scenario. Themes are a good option if you want to allow site owners to change colors and fonts. If you want to make changes to other design elements, such as font size and spacing, cascading style sheets is the way to go. Master pages are the other option if you want to completely change the page structure and design.

When you decide to use themes, the next step is to determine who will be responsible for creating the `*.thmx` files. Because you can create the themes with PowerPoint, you do not need a graphic designer. You can create themes from scratch using PowerPoint, or you can edit an existing theme by downloading a copy of the theme from the Theme Gallery, as shown in Figure 8-1.

1. **Creating a new theme:** Open PowerPoint and use a blank presentation; then on the Ribbon click the Design tab to get to the theme settings.
2. **Editing an existing SharePoint theme:** From the top level site, navigate to Site Settings; under the Galleries section, select Themes; select the theme to download, and from the Ribbon, click Download a Copy. After the theme downloads, double-click the `.thmx` file to open it in PowerPoint (Figure 8-2). PowerPoint opens with a blank presentation with your current theme colors and fonts selected. On the Ribbon, click the Design tab to go to the current selected theme settings.

www.it-ebooks.info

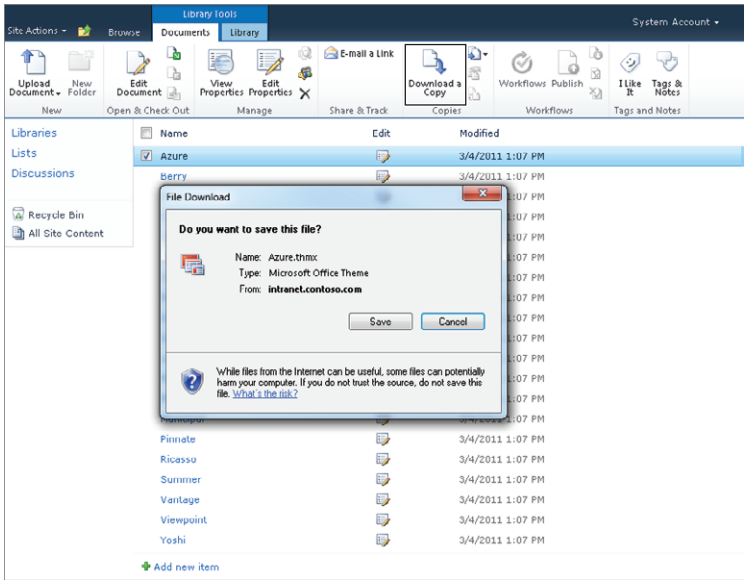


FIGURE 8-1: Download themes from the Theme Gallery

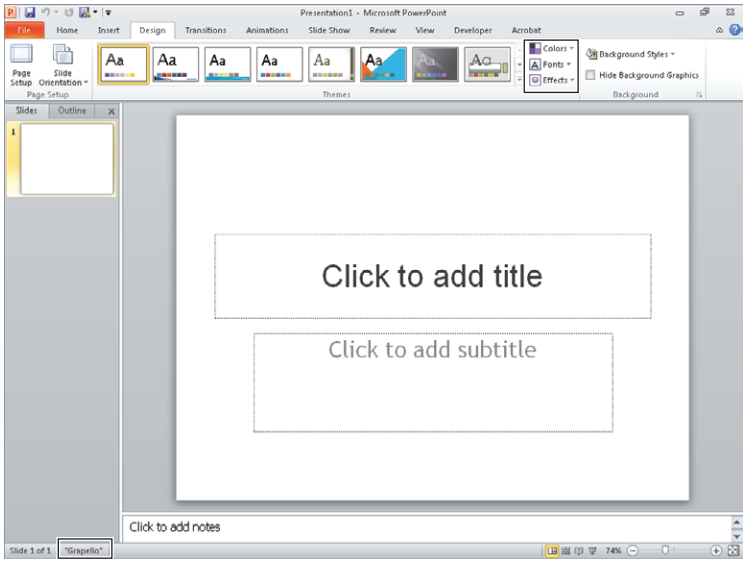


FIGURE 8-2: Open theme in PowerPoint 2010

To modify the colors or fonts, click the Colors or Fonts button, and select Create New Theme Colors or Create New Theme Fonts, as shown in Figure 8-3. This is where it is not intuitive; when you edit existing theme colors and fonts, you still select the Create New options.

www.it-ebooks.info

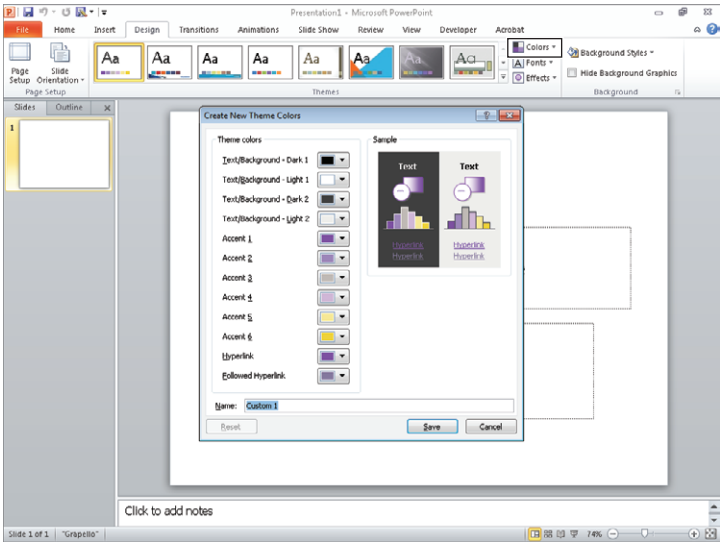


FIGURE 8-3: Update current theme colors

When you finish selecting the colors, change the name, click the Save button, and then repeat the process for the fonts, as shown in Figure 8-4.

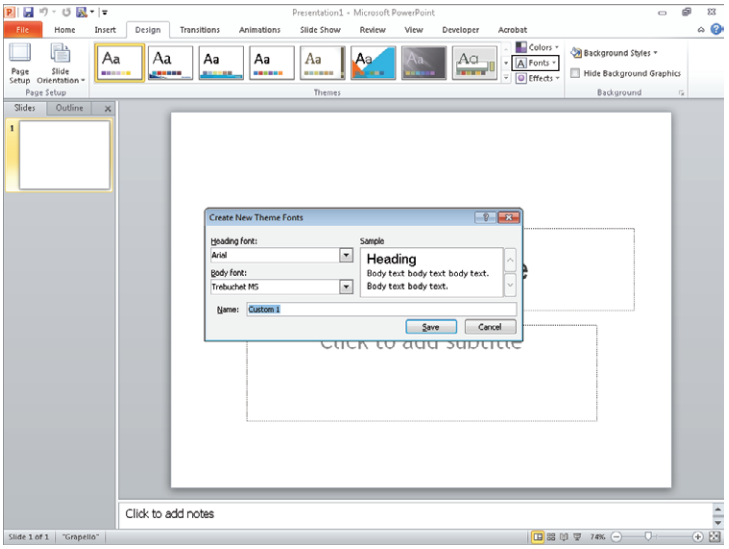


FIGURE 8-4: Update current theme fonts

www.it-ebooks.info

After you make all changes, simply use File ➪ Save As when you are ready to save your theme. Select the Office Theme (`*.thmx`) file type, name the file, and save it to your local hard drive. After your theme is saved, navigate back to the theme gallery and upload your theme (see Figure 8-5). Deploying a theme through the Theme Gallery is a simple approach to use when working with a single site collection.

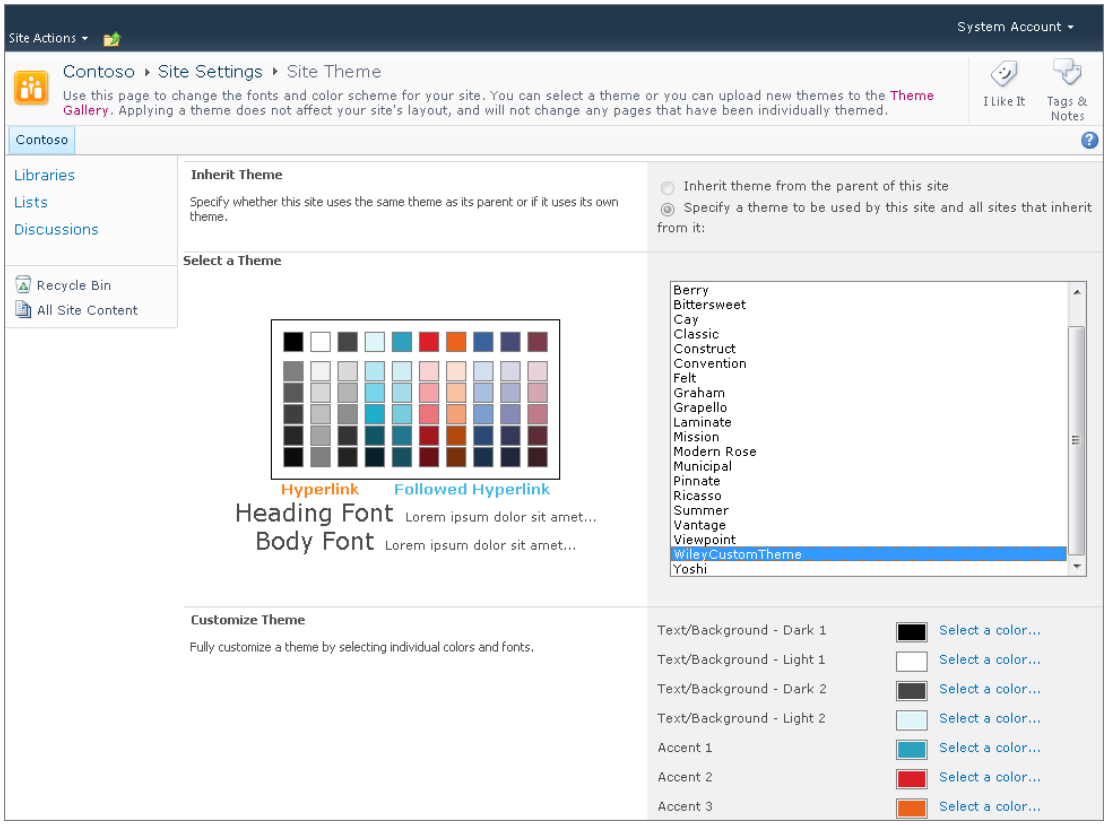


FIGURE 8-5: Change theme to the new uploaded theme

Configuring Themes from the Site User Interface

When the SharePoint Server Publishing Infrastructure features are not enabled on a site, the Site Theme options through the browser only enable the user to select a different theme. It will look like the Nonpublishing Theme Site settings shown in Figure 8-6. However, when publishing is enabled, the user has the ability to modify all uploaded themes through the browser interface, as displayed in Figure 8-7.

www.it-ebooks.info

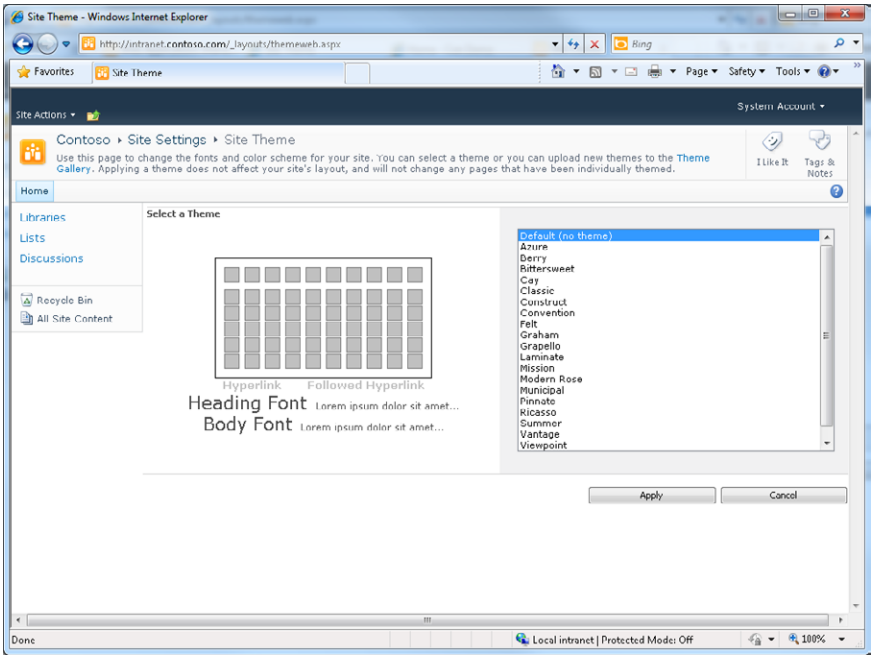


FIGURE 8-6: Nonpublishing Site Theme Site settings

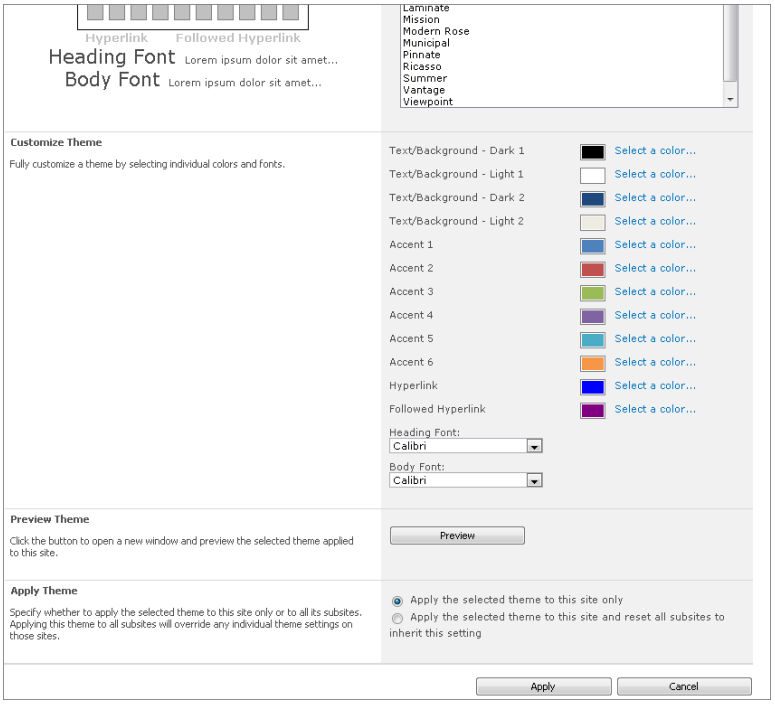


FIGURE 8-7: Publishing Site Theme Site settings

www.it-ebooks.info

Deploying Themes Through a Feature

The best practice for deploying branding is through a Feature, which is the recommended approach when dealing with multiple site collections. All custom files such as themes, master pages, CSS files, and images can be deployed through a SharePoint solution using Visual Studio 2010 and a module element.

The following code snippet is an example of a `Feature.xml` used for deploying a theme through a Module feature.




```
<Feature  Id="28E39FC9-BFE9-4FAE-9E71-C5BFEB818D2E"
          Title="My Custom Theme"
          Description="My Custom Theme"
          Version="1.0.0.0"
          Scope="Site"
          Hidden="FALSE"
          DefaultResourceFile="core"
          xmlns="http://schemas.microsoft.com/sharepoint/">
  <ElementManifests>
    <ElementManifest Location="Elements.xml"/>
  </ElementManifests>
</Feature>
```

ThemeElements.txt

The Elements.xml includes the following:

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Module Name="MyCustomTheme" Url="_catalogs/theme"
Path="MyCustomTheme" RootWebOnly="true">
    <File Url="MyCustom.thmx" Type="GhostableInLibrary" Path="MyCustom.thmx">
    </File>
  </Module>
</Elements>
```

BRANDING WITH CUSTOM CSS

Branding with custom CSS has improved with SharePoint 2010. Table elements are no longer used; instead all items use the `<DIV>` element. The default behavior is the corev4.css, which is applied unless a custom style sheet is referenced.

Themes and CSS

One of the new features with SharePoint 2010 is how the theming engine works on CSS files. When modifying themes, the actual theme files are not modified, but instead the derived CSS classes are replaced. The theming engine works by reading comments in your CSS file, known as annotated

www.it-ebooks.info

styling. When you use annotated styling in your style sheet, themes can replace colors, replace fonts, and recolor images. The following is an example of an annotated style.

```
/* [ReplaceColor(themeColor:"Light2-Darkest")] */ background-color:#C0C0C0;
```

The background color defaults to `#C0C0C0` when the default theme is used; however, when a theme is applied, the background color is replaced with the Light2-Darkest value. The twelve theme colors are represented by the following tokens:

- Dark1, Dark2
- Light1, Light2
- Accent1, Accent2, Accent3, Accent4, Accent5, Accent6
- Hyperlink
- Followed hyperlink

Building and Including Custom CSS

Building custom CSS files for SharePoint 2010 should follow the same best practices for SharePoint 2007. *Never* edit the out-of-the-box CSS files. Instead create your own CSS file, and save it to a location that makes sense to you. You can save the CSS file in two places: the 14 hive or the Style Library. The recommendation is to save it to the 14 hive.

One of the challenges in SharePoint 2007 was getting your custom CSS to load after the `core.css`. Microsoft addressed this issue and has provided a new property with the `<SharePoint:CSSRegistration>` called `After`. The following code snippet shows how this `after` property is applied.

```
<!-- loads custom CSS files after corev4.css -->
<SharePoint:CssRegistration name="customstyle1.css" After="corev4.css"
runat="server"/>
```

If you want to use themes with your custom CSS, include your annotated styling comments.

Using a Centered Fixed Width Design

One of the new challenges faced using SharePoint 2010 branding (especially with public-facing Web sites) is a fixed width design because of the Ribbon. The new Ribbon was intended to stay static and always be visible for contributors at the top of the page; however, this causes an issue when using a fixed width layout. When using a fixed width, the Ribbon needs to scroll with the body of the page; otherwise, a vertical scroll can end up in the middle of the page after it is centered.

A number of helpful resources are available on using a fixed width design with SharePoint 2010, including the following:

- **Fixed width master pages in SharePoint 2010:** Elumination Blog at `http://tinyurl.com/eluminationblog`
- **Fixed width, centered SharePoint 2010 site:** Tom Wilson, Styled Point at `http://tinyurl.com/styledpoint`

www.it-ebooks.info

- **Centered layout with CSS and v4.master on SharePoint 2010:** SharePoint Blues at `http://tinyurl.com/sharepointblues`

Styling Individual Navigation Items

Styling individual navigation items in SharePoint 2010 has changed from previous versions of SharePoint.

Simple Rendering in SharePoint Navigation

One of the new changes in SharePoint 2010 is the addition of the `UseSimpleRendering` property to the `<SharePoint:AspMenu>` control. When `UseSimpleRendering` is set to `true` in the master page on the control, the SharePoint navigation menu renders as a cleaner unordered HTML list format. This provides a much cleaner and more standard way to customize the global navigation than the SharePoint 2007 old table structures. The following are the class styles used for the top navigation.




```
.s4-tn ul.dynamic{
/* [ReplaceColor(themeColor:"Light2")] */ background-color:white;
/* [ReplaceColor(themeColor:"Dark2-Lighter")] */ border:1px solid #D9D9D9;
}
.s4-tn li.dynamic > .menu-item{
display:block;
padding:3px 10px;
white-space:nowrap;
font-weight:normal;
}
a:link{
/* [ReplaceColor(themeColor:"Hyperlink")] */ color:#0072BC;
text-decoration:none;
}
.s4-tn li.dynamic > a:hover{
font-weight:normal;
/* [ReplaceColor(themeColor:"Light2-Lighter")] */ background-color:#D9D9D9;
}
```

NavigationStyling.txt

You can disable simple rendering for SharePoint 2010 `<SharePoint:AspMenu>` controls by setting the `UseSimpleRendering` property to `False`. This sets the control to use the older table-based navigation as used in SharePoint 2007, which includes having unique IDs for each navigation item.

Hiding Navigation Arrows on Dynamic Flyouts

If you want to hide the arrows that appear on the flyout navigation items, use the following CSS code.

```
.menu-horizontal a.dynamic-children span.additional-background,
.menu-horizontal span.dynamic-children span.additional-background{
    padding-right:0px;
    background-image:none;
}
```

www.it-ebooks.info

Changing SharePoint Search Icon with CSS

The Search icon is another component commonly changed with public-facing Web sites. Changing the SharePoint Search icon in SharePoint 2010 is quite simple using custom CSS. First, you need to create your new Search icon, upload it to your wanted location, and then update the following CSS classes.




```
td.ms-sbgo{
    background-color:transparent;
    padding-left:3px;
    padding-right:3px;
    padding-bottom:0px;
    padding-top:0px;
    border:solid 0px #949494;
}
td.ms-sbgo a{
    padding: 0px 0px 0px 0px !important;
    margin: 0px 0px 0px 0px !important;
}

td.ms-sbgo a{
   background-image: url(/_layouts/images/CustomFolderName/CustomSearchImage.gif);
   background-repeat: no-repeat;
   background-position: top left;
}
td.ms-sbgo a img{
    visibility: hidden;
    height: 17px;
    width: 17px;
}
```

SearchIconStyle.txt

MASTER PAGES

Use master pages in SharePoint 2010 to change the design of sites as they were used in SharePoint 2007; however, many changes were made in SharePoint 2010. One of the big changes is the addition of the Office Ribbon and the addition of modal windows, otherwise known as dialog boxes.

Another change is the move from table-based layouts to dynamic positioning, which is the ability to use DIVs and CSS to position elements instead of using absolute positioning. There are some design considerations, however, depending on what type of site you develop. Public-facing Web sites might have issues with cross-browser compatibility with dynamic positioning.

Following are the master pages included in SharePoint 2010:

- **v4.master:** Default team site master page. Provides a Ribbon bar and all other new visual UI changes.
- **default.master:** Old v3 master page available for upgrades that have not transitioned to the new UI. No Ribbon bar. Designed to support Visual Upgrades from SharePoint 2007 to SharePoint 2010.

www.it-ebooks.info

- **Minimal.master:** Slimmed-down master page used by search pages.
- **Simple.master:** Master page used for specific pages. Pages using this master page are `login.aspx`, `signout.aspx`, `error.aspx`, `reqacc.aspx`, `confirmation.aspx`, `webdeleted.aspx`, and `accessdenied.aspx`. Cannot be customized but can be replaced using the `SPWebApplicaiton.SPCustomPage` object.

You can work with master pages either through SharePoint Designer 2010 or Visual Studio 2010. A gap might exist between designers and developers. Designers don't want to use Visual Studio, and most developers don't want to use SharePoint Designer. Designers prefer SharePoint Designer because of the WYSIWYG method with three view options: Code, Design, or Split View. It is fine to use SharePoint Designer in your development environment, but use Visual Studio to create solution packages to deploy your custom branding to your sites.

Browser Support

Since the official release of Internet Explorer 8, many people have problems with their custom branding, especially for public-facing Web sites.

The way IE8 decides its rendering engine is based on certain criteria in your code or master page. IE8 attempts to render a site as follows:

- If it sees a valid `DocType` declared, it attempts to render the site in IE8 Standards Mode.
- If it doesn't see a `DocType`, it attempts to render the site in Quirks mode (otherwise known as pre IE7 rendering mode).

To correct this problem, you can use two different meta tag options:

- `<meta http-equiv="X-UA-Compatible" content="IE=7" />`
- `<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" />`

The first meta tag (`IE=7`) forces a page to render in IE7 mode. The second meta tag (`IE=EmulateIE7`) forces the page to render as IE7 would have historically rendered it. The difference between the two is that the `EmulateIE7` meta tag forces the browser to look for a `DocType` before rendering in IE7 mode, and if it doesn't find one, it renders in Quirks mode.

- If adding the meta tag to your master page, ensure it is added first in the `<head>` before the CSS. Check out META Tags and Locking in Future at `http://tinyurl.com/spmetatags`.

Deploying Master Pages via Visual Studio 2010

As mentioned earlier in this chapter with themes, branding artifacts can be deployed through a SharePoint solution using Visual Studio 2010 and a module element. Deploying branding as a feature through a solution is the recommended best practice to deploy branding for any type of SharePoint site you design for. This approach also makes it easier to deploy branding across multiple site collections.

www.it-ebooks.info

The following code snippet is an example of an `elements.xml` file that contains the module element to deploy a custom master page in the feature.

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Module Name="MasterPageModule" Url="_catalogs/masterpage">
    <File Path="MasterPageModule\CustomMasterPage.master"
Url="CustomMasterPage.master" Type="GhostableInLibrary">
      <Property Name="UIVersion" Value="4" />
      <Property Name="ContentTypeId" Value="0x010105" />
    </File>
  </Module>
</Elements>
```

The `Path` attribute is the path to the physical file located in the Visual Studio solution. Visual Studio will update this path automatically when you add, remove, or modify files within the Solution Explorer. The `Url` attribute uses the site-relative path to the root of the document library, and `_catalogs/masterpage` is the standard path for deploying master pages. The recommendation is to add the `Url` attribute to the `<Module>` tag and then update the `Url` for each `<File>` tag to only include the filename. This example only deploys one master page file; however, when you add multiple files and file types, this method will make it easier to maintain. You will add more files in the next section of this chapter.

The `Type` attribute, set to GhostableInLibrary, tells SharePoint not to copy the physical file into the content database and instead add a reference to the file in the content database. The `UIVersion` property is the SharePoint UI version, and the `ContentTypeId` value for master pages is `0x010105`. Another property that can be used instead of `ContentTypeId` is `ContentType`. The code following code snippet includes the `ContentType` property.

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Module Name="MasterPageModule" Url="_catalogs/masterpage">
    <File Path="MasterPageModule\CustomMasterPage.master"
Url="CustomMasterPage.master" Type="GhostableInLibrary">
      <Property Name="UIVersion" Value="4" />
      <Property Name="ContentType"
       Value="$Resources:cmscore,contenttype_masterpage_name;
" />
    </File>
  </Module>
</Elements>
```

The following two examples use the `FeatureActivated` and `FeatureDeactivating` methods. The code in the `FeatureActivated` method sets the `MasterUrl` and `CustomMasterUrl` properties. The `MasterUrl` property changes the System master page, and the `CustomMasterUrl` changes the site master page. The `Uri` property gets the incoming URL associated with the zone of the request as a URL. The `FeatureActivated` method sets the `MasterUrl` and `CustomMasterUrl` to the customized master page, and the `FeatureDeactivating` method sets the `MasterUrl` and `CustomMasterUrl` back to the v4.master.

www.it-ebooks.info



LISTING 8-1: Setting MasterUrl and CustomMasterUrl in FeatureActivated

```
public override void FeatureActivated
(SPFeatureReceiverProperties properties)
        {
            SPSite site = properties.Feature.Parent as SPSite;

            if (site == null)
                return;

            SPWeb rootWeb = site.RootWeb;

            //Create full master URL
            Uri masterUri = new Uri(rootWeb.Url +
"/_catalogs/masterpage/custommasterpage.master");

            //Master page used by all forms and pages that are not publishing pages
            rootWeb.MasterUrl = masterUri.AbsolutePath;
            //Master page used by all publishing pages on the site
            rootWeb.CustomMasterUrl = masterUri.AbsolutePath;
            rootWeb.Update();
        }
```

MasterURLFeatureActivatedMethod.txt



LISTING 8-2: Resetting MasterUrl and CustomMasterUrl to Default in FeatureDeactivating

```
public override void FeatureDeactivating
(SPFeatureReceiverProperties properties)
        {
            SPSite site = properties.Feature.Parent as SPSite;

            if (site == null)
                return;

            SPWeb rootWeb = site.RootWeb;

            //Create full master URL
            Uri masterUri = new Uri(rootWeb.Url +
 "/_catalogs/masterpage/v4.master");

            //Master page used by all forms and pages that are not publishing pages
            rootWeb.MasterUrl = masterUri.AbsolutePath;
            //Master page used by all publishing pages on the site
            rootWeb.CustomMasterUrl = masterUri.AbsolutePath;
            rootWeb.Update();
        }
```

MasterURLFeatureActivatedMethod.txt

www.it-ebooks.info

Customizing the Ribbon

When customizing your branding, you need to plan for the usage of space of the Ribbon when changing the look and feel of your site. The Ribbon can be moved in the master page and is controlled by a series of `<div>` tags.

Hiding the Ribbon

The Ribbon is used mostly for content creation and site administration; therefore, you must hide the Ribbon for anonymous users for public-facing Web sites. To hide the Ribbon, you must place the `SPSecurityTrimmedControl` control inside the `s4-ribbonrow div` tag and set the `PermissionsString` property. The common permission level used is `ManagePermissions`. You can specify different permission levels for the `SPSecurityTrimmedControl`, which enable you to configure exactly who can see the Ribbon.

The following are available permission levels:

- List permissions
  - `ManageLists`
  - `CancelCheckout`
  - `AddListItems`
  - `EditListItems`
  - `DeleteListItems`
  - `ViewListItems`
  - `ApproveItems`
  - `OpenItems`
  - `ViewVersions`
  - `DeleteVersions`
  - `CreateAlerts`
  - `ViewFormPages`
- Site permissions
  - `ManagePermissions`
  - `ViewUsageData`
  - `ManageSubwebs`
  - `ManageWeb`
  - `AddAndCustomizePages`
  - `ApplyThemeAndBorder`
  - `ApplyStyleSheets`

www.it-ebooks.info

- CreateGroups
- BrowseDirectories
- CreateSSCSite
- ViewPages
- EnumeratePermissions
- BrowseUserInfo
- ManageAlerts
- UseRemoteAPIs
- UseClientIntegration
- Open
- EditMyUserInfo

- Personal permissions
  - ManagePersonalViews
  - AddDelPrivateWebParts
  - UpdatePersonalWebParts

You may notice a horizontal bar after the Ribbon has been removed from the page using `SPSecurityTrimmedControl`. This horizontal bar is a placeholder for the Ribbon content and can be removed by adding the following CSS to your custom style sheet:

```
body #s4-ribbonrow {
    min-height: 0 !important;
    height: auto !important;
}
```

The Catch

The `login` control is now inside the Ribbon for SharePoint 2010, which means if you hide the Ribbon, the `login` control is not available. This can be resolved by moving the `login` control outside of the Ribbon.

WORKING WITH PAGE LAYOUTS

Using page layouts provide greater control over the entire site by providing structure for individual content pages. Both public-facing and internal Web sites can benefit if you use page layouts. Page layouts contain content controls, and each content control in a page layout references a specific content placeholder in the master page. To use page layouts with SharePoint, two features must be activated: Site Collection feature SharePoint Server Publishing Infrastructure feature and the Site feature SharePoint Server Publishing. The publishing infrastructure is only a part of SharePoint Server licensing and is not a part of SharePoint Foundation.

www.it-ebooks.info

SharePoint 2010 includes many OOTB page layouts; however, if you need more layouts or different layouts, you need to create your own custom page layouts. Just like master pages and style sheets, it is recommended to never modify the OOTB page layouts.

Creating a custom page layout isn't much different from creating it in SharePoint 2007. However, the deployment from Visual Studio 2010 has changed. When creating custom page layouts, it is typical to first start in SharePoint Designer; when finished, deploy the custom pages as a feature through a Visual Studio 2010 solution.

Creating Page Layouts Using SharePoint Designer 2010

Page layouts in SharePoint Designer 2010 are accessed via the Page Layouts section of the left navigation. You can create a new page layout in SharePoint Designer by selecting the New Page Layout button in the Ribbon's New group. When creating a new page layout, you have the option of basing the page layout on an existing page layout or creating based on your own custom content types. The recommendation is to create your own custom content types, especially when using the Content Query Web Part for rolling up the content.

After a new page layout is created, you can access the field controls for the content type used. Field controls are accessible only through SharePoint Designer when you edit a page layout. To access field controls for the page layout, you need to open the Toolbox task pane by clicking the Task Panes command in the Ribbon's Workspace group; then click Toolbox.

The following are two sections for fields:

- **Page fields:** Fields (columns) directly inherited from the parent content type
- **Content fields:** Additional fields added to the new page layout

Page Layout UIVersioned Content

Page Layouts were previously available in SharePoint 2007; however, a new server-side control was added to SharePoint 2010 for legacy and upgrade purposes. This new server-side control, called `UIVersionedContent`, identifies two different versions:

- **Version 3:** Corresponds to SharePoint Server 2007
- **Version 4:** Corresponds to SharePoint Server 2010

Each `UIVersionedContent` control instance includes a `ContentTemplate` control that includes markup used based on the version detected.

Adding Web Parts and Web Part Zones to Page Layouts

You can add web parts and web part zones to page layouts; however, you must consider the end user control you delegate as a result of adding web parts and zones to your page layouts. Another thing to consider is that web parts cannot be versioned.

A common example to use an embedded web part in a page layout is the user of the Content Query Web Part. In SharePoint 2007, the OOTB Content Query Web Part (CQWP) did not support dynamic filtering. This meant if you added a CQWP to your page layout, the result of the query

www.it-ebooks.info

would be the same on all pages using that page layout. The CQWP in SharePoint 2010 has improved with the capability to now support dynamic filtering.

Deploy Custom Page Layouts Through Visual Studio 2010

Deploy page layouts the same as master pages using a SharePoint solution and module element. In the following example, a new module was added to the existing `MasterPageModule elements.xml` file to include the custom page layout.



LISTING 8-3: Elements.xml File with New Module Elements Added

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Module Name="MasterPageModule" Url="_catalogs/masterpage">
    <File Path="MasterPageModule\CustomMasterPage.master"
Url="CustomMasterPage.master" Type="GhostableInLibrary">
      <Property Name="UIVersion" Value="4" />
      <Property Name="ContentTypeId" Value="0x010105" />
    </File>
  </Module>
  <Module Name="PageLayoutsModule" Url="_catalogs/masterpage">
    <File Path="MasterPageModule\CustomPageLayout.aspx" Url="CustomPageLayout.aspx"
Type="GhostableInLibrary">
      <Property Name="PublishingAssociatedContentType" Value=";#Welcome
Page;#0x010100C568DB52D9D0A14D9B2FDCC96666E9F2007948130EC3DB064584E219954237AF3
90064DEA0F50FC8C147B0B6EA0636C4A7D4;#" />
    </File>
  </Module>
</Elements>
```

BrandingModuleElementsXML.txt

The `PublishingAssociatedContentType` property is set to use OOTB Welcome Page content type; however, you can change this to any of the other OOTB content page layout content types or set your own custom content type. Images, style sheets, and other branding artifacts can also be added to the `elements.xml` for deployment. Notice there are now two different module elements in the above code snippet. The first module is the one that was automatically created by Visual Studio when you added your module to the solution. The second module element was added manually to separate the master pages from the page layouts. More modules can be added manually to group other files such as images and custom style sheets. The following code snippet is an example of a module for style sheets and a module for images.

```
<Module Name="StyleSheetsModule" Url="Style Library/Custom">
    <File Path=" MasterPageModule\CustomStyle1.css"
Url="metlifesubstyle.css" Type="GhostableInLibrary"
IgnoreIfAlreadyExists="true" />
    <File Path=" MasterPageModule\CustomStyle2.css"
Url="metlifepubstyle.css" Type="GhostableInLibrary"
IgnoreIfAlreadyExists="true" />
</Module>
```

www.it-ebooks.info

```
  <Module Name="ImagesModule" Url="SiteAssets/images/main">
    <File Path="MasterPageModule\images\MyLogo.jpg"
Url="MyLogo.jpg" />
    <File Path="MasterPageModule\images\MyTopHeader.gif"
Url="MyTopHeader.gif" />
  </Module>
```

When deploying files through a Module, the files do not get unprovisioned from the content database when the feature is deactivated, causing the files to be orphaned. The way around this is to add code to delete the files from the content database on `FeatureDeactivating`. The following code snippet is an example of how to delete the custom master page referenced earlier in this chapter.

```
SPFile file = Web.GetFile("_catalogs/masterpage/CustomMasterPage.master");
if (file.Exists)
{
    file.Delete("_catalogs/masterpage/CustomMasterPage.master");
}
```

Branding Layout Pages

In SharePoint 2007, custom master pages could not be applied to application pages which are the pages in the `layouts` directory. The only way to modify the branding was to create a custom theme or customize the `Application.master` file, which was not supported nor recommended. This has changed for SharePoint 2010; you now have the ability to apply custom master pages to your application pages. By default, the application page references the custom master page of the site. The master page used for application pages can be changed by setting the MasterPageFile attribute of the application page element. This attribute must be set if dynamic master pages are not enabled on your SharePoint server.

Following is an example of setting the MasterPageFile attribute:

```
MasterPageFile="~/_layouts/applicationv4.master
```

SharePoint 2010 includes "Safeguarded application pages," which are a set of pages that will automatically revert to use an OOTB master page if your custom master page is broken. Safeguarded Application Pages include:

- `AccessDenied.aspx`
- `MngSiteAdmin.aspx`
- `People.aspx`
- `RecycleBin.aspx`
- `ReGhost.aspx`
- `ReqAcc.aspx`
- `Settings.aspx`
- `UserDisp.aspx`
- `ViewLsts.aspx`

www.it-ebooks.info

CUSTOMIZING APPLICATION PAGES

Application pages are ASP.NET pages designed for use in a SharePoint site and are shared across all sites on the server. An application page is the best type of page to create if you want to contain custom code. With SharePoint 2010, you can now create your own custom error pages and override the OOTB application pages.

The following pages can be overridden:

- **AccessDenied:** Specifies `AccessDenied.aspx`
- **Confirmation:** Specifies `Confirmation.aspx`
- **Error:** Specifies `Error.aspx`
- **Login:** Specifies `Login.aspx`
- **RequestAccess:** `Specifies ReqAcc.aspx`
- **Signout:** Specifies `SignOut.aspx`
- **WebDeleted:** Specifies `WebDeleted.aspx`

Creating Custom Application Pages

Visual Studio 2010 includes an Application Page project item template (see Figure 8-8) that enables you to design application pages using the source view of the VS designer. Unfortunately, the design view of the designer is disabled for application pages; however, you can drag controls from the toolbox to the source view.

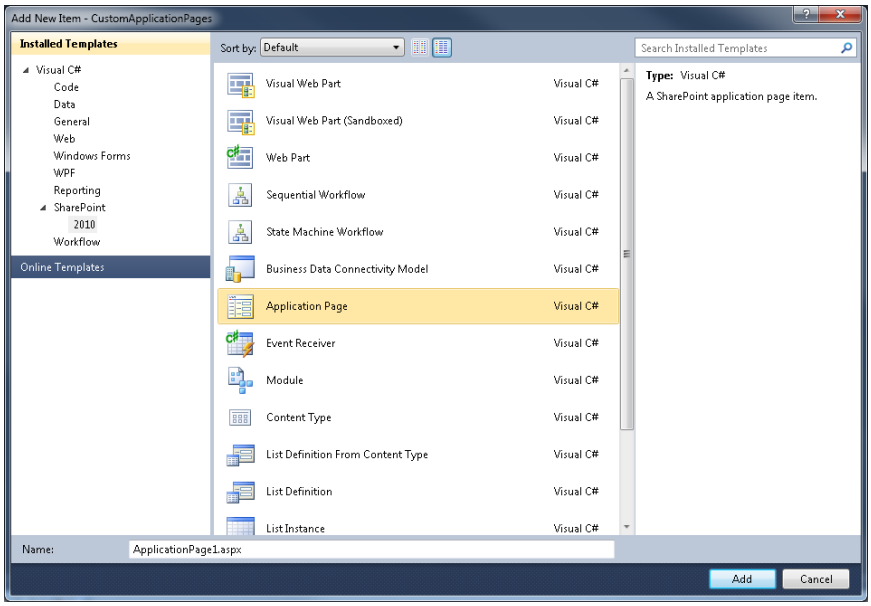


FIGURE 8-8: Visual Studio Application Page template

www.it-ebooks.info

When you create an application page, Visual Studio adds the following folders to the project:

- **Layouts:** Maps to the `_layouts` directory of the SharePoint file system.
- **Layouts subfolder:** Contains the files that make up the application page. By default, the folder has the same name as your project, which can be renamed. When deployed, VS deploys the folder to the `_layouts` virtual directory of the SharePoint site.

Visual Studio also adds the following files to your project:

- **ASP.NET page file (.aspx):** Markup that defines the page
- **Application Page code file:** Code-behind for the application page
- **Application Page Designer code file:** Code generated by the Visual Studio 2010 Designer

Figure 8-9 shows an example of a custom application page.

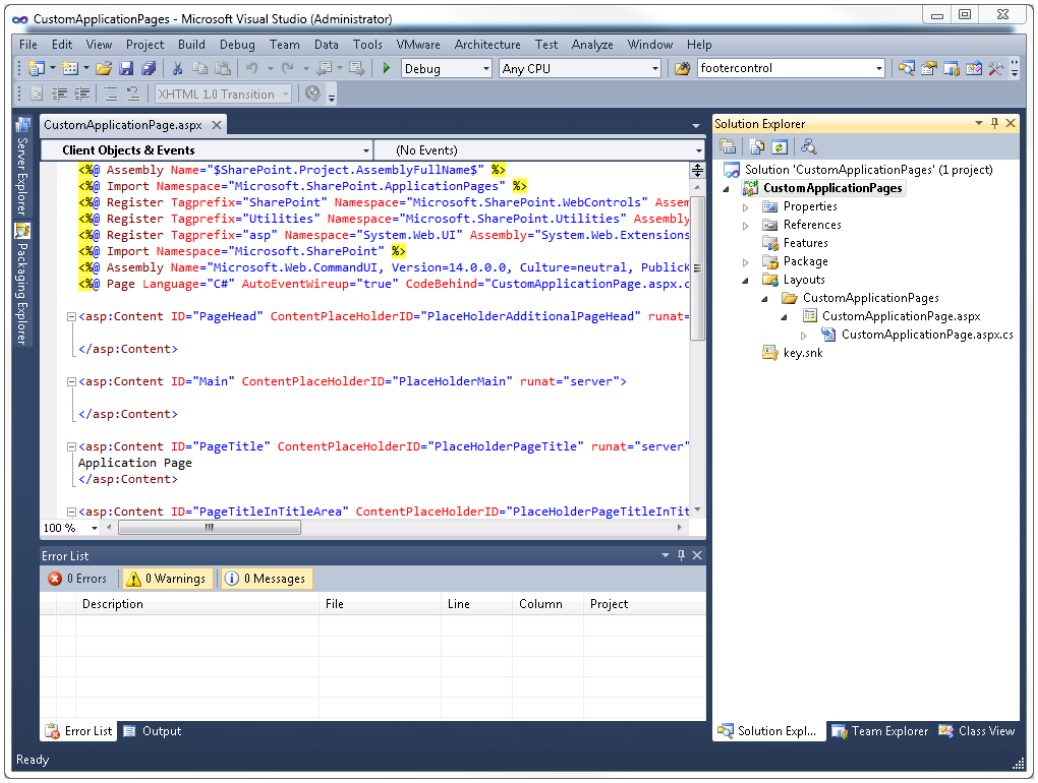


FIGURE 8-9: Visual Studio Application Page project

www.it-ebooks.info

Registering Custom Application Pages from Receivers

Using the SharePoint 2010 API, you can programmatically set the URL of a custom page that replaces one of the default SharePoint application pages using a SharePoint Web Application feature. The following pages can be replaced in code:

- `SPWebApplication.SPCustomPage.AccessDenied`
- `SPWebApplication.SPCustomPage.Confirmation`
- `SPWebApplication.SPCustomPage.Error`
- `SPWebApplication.SPCustomPage.Login`
- `SPWebApplication.SPCustomPage.RequestAccess`
- `SPWebApplication.SPCustomPage.Signout`
- `SPWebApplication.SPCustomPage.WebDeleted`

The following code listings are examples of the FeatureActivated and FeatureDeactivating methods for replacing the RequestAccess application page. The FeatureActivated method sets the RequestAccess page to a custom access page called `MyCustomPage.aspx` when the feature is activated. The FeatureDeactivating method sets the RequestAccess page back to the normal default page.



LISTING 8-4: FeatureActivated Event Receiver Code Sample to Set the RequestAccess Page to Your Custom Page

```
public override void FeatureActivated
(SPFeatureReceiverProperties properties)
        {
            if (properties == null) throw new ArgumentNullException("properties");

            SPWebApplication webApp = properties.Feature.Parent as
SPWebApplication;

            if (webApp != null)
            {

              webApp.UpdateMappedPage(SPWebApplication.SPCustomPage.RequestAccess,
"/_layouts/CustomDirectory/MyCustomPage.aspx");

                webApp.Update();
            }
        }
```

RequestAccessFeatureActivated.txt

www.it-ebooks.info



LISTING 8-5: FeatureDeactivating Event Receiver Code Sample to Set the RequestAccess Page Back to the Default

```
public override void FeatureDeactivating
(SPFeatureReceiverProperties properties)
        {
            if (properties == null) throw new ArgumentNullException("properties");

            SPWebApplication webApp = properties.Feature.Parent
as SPWebApplication;

            if (webApp != null)
            {

                webApp.UpdateMappedPage
(SPWebApplication.SPCustomPage.RequestAccess, null);

                webApp.Update();
            }
        }
```

RequestAccessFeatureDeactivating.txt

Registering Custom Application Pages from Windows PowerShell

In SharePoint 2007, customizing application pages was a headache without editing the OOTB SharePoint root files. That has all changed in SharePoint 2010. Overriding application pages with your custom application pages is easy in SharePoint 2010 with the use of Windows PowerShell. The `Get-SPCustomLayoutsPage` returns a mapping to a custom layout page. The following example returns the mapping of the `AccessDenied` layout page in the web application.

```
Get-SPCustomLayoutsPage -Identity "_layouts/accessdenied.aspx"
-WebApplication "http://webappurl"
```

The `Set-SPCustomLayoutsPage` PowerShell command maps a new path for a custom layout page. The following example replaces the `AccessDenied` (`accessdenied.aspx`) page in the `_layouts` folder with a new custom page.

```
Set-SPCustomLayoutsPage -Identity "Login" -RelativePath
"/_layouts/custompages/accessdenied.aspx"
-WebApplication "{replace with web app url}"
```

SUMMARY

This chapter provided a high-level look at branding for SharePoint 2010 including themes, style sheets, master pages, page layouts, and application pages. Because many approaches to branding exist, you should follow best practices when branding your SharePoint 2010 site. Never touch the OOTB files, and always create your own custom files.

www.it-ebooks.info

9

Building a Site Provisioning System

This chapter outlines the design and construction of a site provisioning system in SharePoint. Common among enterprise implementations of SharePoint, a site provisioning system enforces your site taxonomy, keeps records of all sites and site requests, and ensures a consistent process is followed when new sites are created. Furthermore, an automated provisioning system saves time for end users by eliminating the need for administrators to take action for users to get a new SharePoint site.

While some SharePoint farms make use of self-service site creation to enable end users to provision their own sites, this out-of-the-box solution is suboptimal. The functionality is not very configurable, has no approval process associated with it, and cannot be easily extended to provide the amount of functionality that the custom solution presented in this chapter can provide.

The goal for this chapter is to demonstrate how to build a system that automates the provisioning of new sites in your SharePoint environment. The system enables users to create new site requests that may go through an approval process before the site is actually created. Such a system accomplishes the following high-level objectives:

- Reduces administration
- Enforces your site taxonomy
- Enforces your site provisioning process
- Keeps track of all sites

In addition to meeting these objectives, a number of functional and nonfunctional requirements are practically universal in this type of solution. These include the following:

- Must support multiple types of sites.
- Manager approval will be required to create certain types of sites.

www.it-ebooks.info

- Email alerts to users.
- Managers must approve a site simply by clicking an Approve link within an email alert.
- No long wait times or page loads.
- Highly configurable.
- Easily managed by support staff.



The entire Site Provisioning System solution developed throughout this chapter is available for download from the book's companion website at `www.wrox.com`.

SITES AND SITE COLLECTIONS

Although it's not uncommon to find SharePoint farms with a single site collection of content (with many subsites, security groups, content types, and more contained within), this is not best practice. For a number of reasons, you should use many site collections for your intranet, ideally arranged around functional groups, projects, and ownership of data or processes contained within a site.



Throughout this chapter, when discussing SharePoint "sites," the text specifically refers to SharePoint site collections and not to subsites.

To illustrate, imagine an intranet with single site collections and many subsites, one for each department. In such a scenario, the site collection administrator has full control over the entire intranet, and it's not easy to grant someone full control of a subsite alone. Even if you do secure your subsites, the owners of those areas cannot perform site collection administrator (SCA) level administration over those sites (auditing, search, caching, or any other settings under Site Collection Administration in Settings.) Furthermore, as subsites are locked down, SharePoint security groups must be created to secure visitors, owners, and members of each subsite. With a site collection containing 8 subsites, that means 27 security groups to administer from a single location. By using many site collections, you can keep your data and settings cleanly separated and under the complete control of their respective owners.

Furthermore, a single SharePoint site collection must exist within a single database, and cannot be split across multiple databases. If all sites are placed within a single site collection, there is no way to split that data across multiple databases, significantly limiting scalability and manageability. Ten 100GB databases are much easier to back up, restore, and move across the network when compared with a single 1TB database. In addition to database size, you might assign different maintenance schedules to databases depending on the data contained within, be it slow-changing publishing data or fast-changing collaboration data. By breaking out sites into multiple collections, you can have complete control over this.

www.it-ebooks.info

DETERMINING YOUR SITE TAXONOMY

A site taxonomy for a particular SharePoint farm describes the types of sites within the farm, along with various properties of those sites, such as the site template used to create them, URLs where they may exist, databases they are stored in, and restrictions on users of those sites. A site taxonomy enables you to categorize your sites, keep your data in appropriate types of sites, and enforce appropriate usage for various site types. Without a well-defined and actively enforced site taxonomy, your SharePoint farm grows to contain a hodgepodge of uncategorized sites, creating an administrative and usability nightmare.

Site taxonomies are documented by describing the purpose and attributes of the different kinds of sites you intend to create in SharePoint. At a high level, a site taxonomy defines the following types of sites:

- **Project sites:** Temporary sites used for collaboration on a project.
- **Team collaboration sites:** Permanent sites assigned to individual business units within the organization to collaborate on day-to-day tasks.
- **Intranet sites:** Permanent sites assigned to individual business units and departments such as HR or Finance to enable them to disseminate information organization-wide.
- **Apps or Service sites:** Sites that host applications or services. This would include sites with BI dashboards, SharePoint workflows, and other application or service-like functionality (that is, not collaboration). The site that will host your site provisioning process will be a Service site.
- **"External" sites:** Sites that are shared with third parties outside the organization. These sites could take the form of public-facing publishing websites; alternatively they could take the form of portals used to collaborate with outside groups such as vendors, suppliers, or clients.

These site types serve different purposes and audiences, and may be subject to different rules about how they should be used. Site quotas, security settings, who can request the site, who will administer the site, the site template used to create the site, and more must be considered for each type of site.

Example Taxonomy Matrix

It's helpful to create a matrix to describe and communicate the purpose and rules around each type of site in your taxonomy. Table 9-1 shows an example matrix.

TABLE 9-1: Example Taxonomy Matrix

SITE TYPE	PROJECT	TEAM	INTRANET	SERVICE
Purpose	Project Collaboration	Team Collaboration	Information Publishing	Application or Service
Audience	Project Members	Team Members	Entire Organization	Any Audience

continues

www.it-ebooks.info

TABLE 9-1 *(continued)*

SITE TYPE	PROJECT	TEAM	INTRANET	SERVICE
Template	Team Site	Team Site	Publishing Collaboration Site	Blank Site
Quota	1GB	5GB	1GB	2GB
Site Admins	Designated Project Member	IT Admins	IT Admins	Service Owner
Who Can Request	Anyone	Managers	Managers	Anyone
Approver	Manager	None	IT Admins	Manager
Site Location	proj.domain.com	team.domain.com	domain.com/depts	apps.domain.com

Of course, this matrix is just an example, and planning for a full implementation of SharePoint for collaboration, publishing, and external access is outside the scope of this work. Regardless, take time to think about how your organization will interact with SharePoint and what breakdown of site types will be most effective in facilitating those interactions, all while maintaining manageability and usability for your end users.

Additional Factors

Many other factors can come into play when determining your full taxonomy, such as which web applications, managed paths, and content databases should be utilized with each type of site. You must determine the appropriate mix of site types and rules that work for your organization. There is no single correct way to structure your taxonomy.

For example, you could place all project sites in `http://projects.mycompany.com/sites`, or you could place all project sites in their own managed path within the main web application, such as `http://mycompany.com/projects`. Both scenarios would enable you to define custom search scopes limited to project sites, but only the first option (putting project sites in their own web application) would enable for the use of a web application policy to grant or deny access to all project sites without affecting other types of sites. On the other hand, hosting project sites in their own web application can consume more resources on the server, especially if that web application is run within its own application pool.

Furthermore, because each type of site has different usage scenarios and quotas, it's helpful, though not necessary, to group them in web applications according to type, as content databases are configured per web application. This way, no content databases would contain more than one type of site, and you can set up your content databases according to the usage expected. So for project sites with a 1GB quota, you may preallocate databases to support twenty sites per database; whereas for team sites with a 5GB quota, you would be better off allowing for four or five sites per database.

www.it-ebooks.info

It is possible to create different types of sites in a single web application, with sites stored in different content databases according to site type. To keep the example in this chapter simple, the code will make use of SharePoint's underlying ability to automatically select the content database used to store a new site collection based on the web application where that site is created, using one of the preconfigured content databases for that web application.

The rest of this chapter demonstrates a method to automate the provisioning of sites according to the taxonomy described here. The final provisioning system enforces who may request, approve, and administer sites, as well as manage the process of requesting a site using a SharePoint workflow. Part of this solution includes a "site type" list used to store metadata about each of the site types in your taxonomy. The site creation workflow uses this metadata to enforce your taxonomy and configure new sites.

THE SITE REQUEST PROCESS

After you work out the details of your site taxonomy, you are ready to design the process to enforce that taxonomy. According to the complexity of your taxonomy, you may need additional steps in your process. For example, to implement two levels of approval, a more complex process is required. In the following section, a site request process that you implement later in this chapter is described.

An Example Site Request Process

Before you can work out an appropriate site provisioning solution, you need to define the process. This process is similar to a lot of request-based processes in SharePoint, in which approval is required and parties must be notified when action is taken.

A Typical Request and Approval Process

For the example solution defined in this chapter, a SharePoint site is used to host a list that accepts requests for new sites from end users who visit your application. If a user requests a site type that requires approval, the approver will be notified that approval is required. Figure 9-1 shows a flowchart describing the typical site request process.

When approved, the system creates the site and notifies the requestor that its site has been created. Additionally, each time a site is created, the system creates an entry for it in a separate list that will be used to store metadata about the site in question. This way, the system can keep your site requests separate from your list of created sites, enabling administrators to delete old (completed or rejected) requests while retaining information about the sites that were created. The list used to store created site data can then be used by other components if necessary, such as navigational web parts or a site contact information web part.

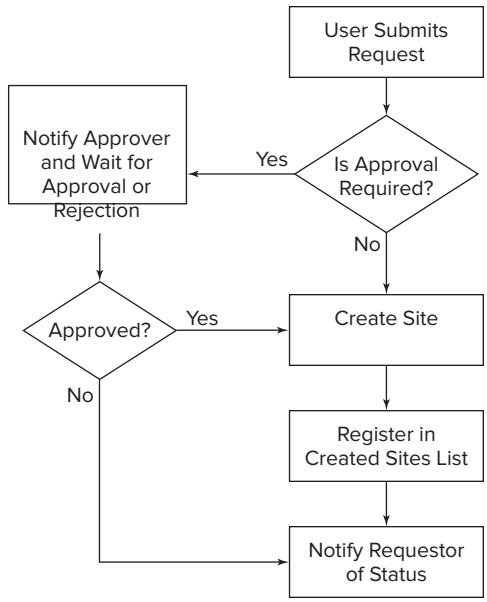


FIGURE 9-1: Site request process

www.it-ebooks.info

In this case, the approver is determined by traversing the corporate hierarchy upward from the submitter's position. The first user found whose job title contains a certain matching string (such as `dir` for director-level resources) is selected as the approver. The `matching` string to be searched for is configurable for each type of site in the taxonomy. This solution requires the correct configuration of corporate hierarchy within Active Directory (AD), and that user imports from AD have been set up for your farm within the User Profile Service. Furthermore, users' titles must be set appropriately for matches to be found. If no `matching` string is provided, the system selects the submitter's direct manager. If the system cannot find a match, it defaults to a preset user configurable per site type.

Additional Complexity

Depending on the complexity of your organization, additional roles and approvals (second or even third level) may be required. You may prefer that support staff have the ability to reassign approvers without creating sites. This functionality can certainly be built by extending this solution, but for the sake of a clear example, you can implement a system in which a group of Site Creation Administrators have complete control over requests to fix issues with the request process. (The approver is unavailable, but a higher-level manager wants the site created, for example.)

As the solution is utilized, Site Creation Administrators also must perform periodic maintenance. This involves the following:

- Checking for sites that failed during the creation process
- Deleting old or rejected site requests
- Reviewing created sites to ensure they fit the established site taxonomy
- Following up on "forgotten" requests that were never approved

To make these tasks easier for administrators, you can create List Views that display the data administrators need to perform the administrative tasks previously listed. Because SharePoint offers excellent capabilities in customizing List Views via the browser, you can configure these views by hand through the web interface on the production site.

Site Request Process State Machine

To simplify and clarify the request process, it's helpful to conceive of the process as a finite state machine with transitions between states. These states and transitions will translate directly to pieces of your implementation. You can implement this process by tracking the following states and transitions for the site request, as shown in Figure 9-2.

Following are five distinct states:

1. **Waiting for approval:** Any new requests (where the approver is not the requestor) will begin in this state.
2. **Rejected:** When the approver rejects a request, it is placed in this state.
3. **Waiting on site creation:** In this case, once approved, the request enters this state and the site will be created.

www.it-ebooks.info

4. **Site created:** The request process is complete and the site has been successfully created.
5. **Site creation failed:** The request was approved but an error prevented the site from being created.

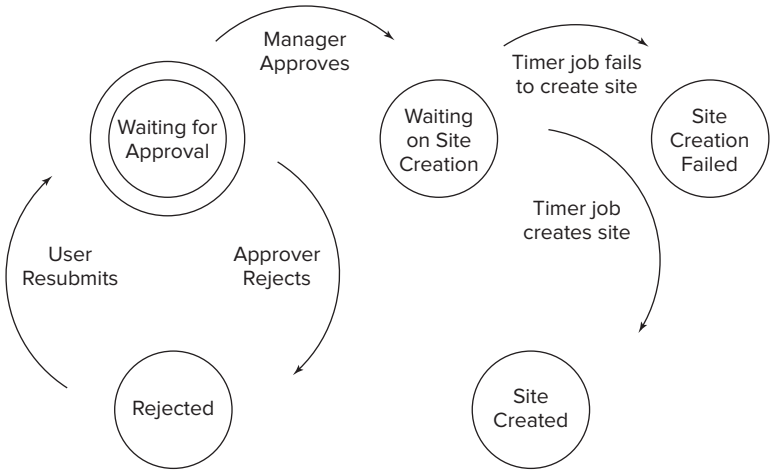


FIGURE 9-2: Site request state machine diagram

Looking at the preceding list, it becomes clear how we might need to modify these states to support additional functionality. For example, if a second level of approval is needed, we would need a "Waiting for second level approval" state, indicating that the first level of approval has been passed, but another approval must occur before the site can be created. These states are critical to our implementation because they will be used to determine how to display the request form and when to take certain actions, such as sending an email or creating a new site.

Implementing the Process

You could make use of Workflow Foundation to implement this process, but because the state transitions and workflow logic are mostly implemented within the Site Request form, and to keep the example as simple as possible, this chapter does not make use of Workflow Foundation, but takes a simpler, yet just as effective (and configurable) approach.

You can create a custom ASPX page used to add and edit site requests. That ASPX page implements the logic needed to request, approve, reject, and resubmit. The function used to create the sites implements the logic required to transition to the Site Created or Site Creation Failed states.

In addition, to implement the required email notifications, you may implement a workflow in SharePoint Designer (directly on the production site) that sends emails to the appropriate parties. By using SharePoint Designer, site administrators can later alter the email messages sent by the solution. If your farm does not allow SharePoint Designer workflows, you must add email notification to the form code itself. In that case, build a configuration list to store the email message text and settings so they may be altered later by solution administrators.

www.it-ebooks.info

Security Challenges

In addition to the approval workflow and roles, some additional challenges determine how the process works from a technical perspective. A common trap that SharePoint developers fall into is that they fail to use different accounts for different services when configuring SharePoint on their development servers. This leads to scenarios in which code that runs without a problem on the development server fails when put into a test environment or, worse, production (when the testing and staging servers are misconfigured as well). Site provisioning systems are particularly prone to this problem.

A Common Issue

All site provisioning systems must create new site collections, which involves writing data to the central farm configuration database. The farm configuration database contains sites, sitemaps, and sitecount tables that list every site collection in the farm. These tables are necessary because they allow SharePoint to find a particular site collection without having to connect to and query every database that might be contained within a web application. Luckily, SharePoint configures the configuration database with a SQL Server role called `WSS_Content_Application_Pools` that is granted the necessary rights to create site collections, and little else. All content application pool accounts (aka web content accounts) are mapped to this role.

Unfortunately, in a properly configured SharePoint farm, the web content account cannot create site collections without being added to the Farm Administrators group in central administration, and therefore any code that attempts to create a site collection from within the web application process will fail. The only exception to this is with self-service site creation — the code for which has been decompiled by this author, only to find an obfuscated function, the contents of which could not be viewed. Analysis of calls to the database made by this functionality indicate that this function bypasses code used by the traditional site creation functionality, which checks for farm administrator group membership before creating a new site.

Site Creation Options

This leaves you with a number of basic options for implementing the functionality used to create new sites:

1. Call the self-service site creation method from your code.
2. Alter the Farm Administrators group to enable site creation from the timer job process.
3. Impersonate the farm account from the web or timer job process.
4. Create a custom service application to handle site creation.
5. Create a new service in `/_layouts` that is called through central administration.
6. Add the web content account to the Farm Administrators group and elevate privileges when creating new site collection.

Option #1 presents two difficulties. First is that when using self-service site creation, you can only create a site within the current web application. Second, you would need to enable self-service site creation function from central administration, which would make the OOTB self-service site creation page visible to end users. This page could be blocked, of course, but there are better approaches.

www.it-ebooks.info

Option #2 may be out of the question depending on how your production farm is locked down. By default, the timer job account does not have the necessary rights. Furthermore, moving site creation functionality to a timer job introduces unnecessary complexity to the solution. Option #3 requires that you store farm account credentials somewhere the web or timer job can read them, which introduces a potential security hole.

Option #4 is a lot of effort to solve a small piece of the problem. If you have extremely rigid security policies, you must go with option #4 and take the extra step to create a custom service application to handle the site creation step of the process. Such a task is outside the scope of this chapter because creating a service application is an advanced task that most people who create site provisioning systems in SharePoint find unnecessary.

Option #5 requires that the web application has access to central administration via HTTP. This may be difficult if loopback connections are disabled on your farm servers.

Most environments do not require this level of lockdown, so you should go with option #6 and alter the Farm Administrators group of your farm to allow the web content account to create site collections. To update your farm's administrators group, browse to central administration and perform the following steps:

1. Under Security, click Manage Farm Administrators Group. The Manager Farm Administrators Group page opens.
2. Click New to add a user to the group. A pop-up appears.
3. Enter the login name of your web content account, and click OK. The user is added to the group.

If you choose to provision sites from the timer service or additional web content accounts, you must add those account names as well.

Alternatively, you could create a managed service account, add it to the farm administrators group and `WSS_Content_Application_Pools` SQL role in the config DB, and impersonate that account when creating sites. This is more secure in that it doesn't allow the web content account to create new sites, but would also require storing the credentials for the managed account in a place where the web content account could access it. For simplicity, the solution in this chapter will forego this extra layer of security.

Our Example Solution

Given the initial requirements, site taxonomy, and process, you have the information you need to devise the solution architecture. The following items provide for a working solution:

- Public site collection to host the solution.
- Site Types list to describe the types of available sites. Configurations for various types of sites are stored in this list. It will be maintained by administrators, enabling them to control the types of sites available for creation.
- New Site Requests list to track site requests. This list will contain one item for each site requested. A custom form will be used to control how users interact with this data.

www.it-ebooks.info

- Created Sites list to track created sites. Once new sites are created, they will be listed here. This list could be used for a number of additional purposes, such as tracking the current "Owners" of particular sites and driving navigational controls.
- Some way to flag the site as the location of this solution. More on this later in this chapter.
- Security configuration to prevent users from overriding rules or altering configuration.
- Workflow to manage the site request process. For simplicity, the workflow will be a combination of form logic and SharePoint designer workflows that notify users at various stages of the process.

It would be nice to have a single feature you can activate to enable site requests on the host site. You want to make the feature hidden to avoid accidental activations of your feature, so you must activate the feature using `stsadm or PowerShell` under the farm account. You don't want your site administrators activating this by mistake!

The solution architecture takes shape:

- Site Scoped feature
  - Three list definitions and instances, one with a custom Add/Edit form
    - **Site Types list:** Describes the types of sites available
    - **Site Requests list:** Stores requests for new sites
    - **Created Sites list:** Stores details of sites that have been created
  - Custom Site Request ASPX form used to add and edit items in the "New Site Requests" list.
  - Feature receiver to set security settings and perform other miscellaneous tasks upon activation.

Alternatives in Implementation

The solution architecture incorporates some options that are a matter of preference or style. Feel free to deviate from the specific implementation in this chapter if you are more comfortable with designing your own alternative.

Custom Forms

For example, I chose to use a custom ASPX form because I'm comfortable with the technology and like the fully customizable nature of the option. You may choose to implement a custom form using InfoPath forms, for example. In that case, although the specifics of the implementation described in this chapter will not always apply, you should be able to follow the general form logic explained in the Buildout section of this chapter.

Lists

To use another example, this architecture makes use of two different lists to track sites, the "New Site Requests" list and the "Created Sites" list. These lists could be combined into a single list

www.it-ebooks.info

for a more simple architecture, but in my experience, it is easier to keep these entities separate. Administration of list data is easier when site requests and created sites are stored in different places, and there may be many more site requests (including historical, rejected requests) than created sites, such that keeping them in the same list will make queries for created site data less efficient. Finally, although the data they store is very similar, the entities represented by these lists are distinct, such that keeping them in separate lists produces less confusion among users interacting with the solution. As with the preceding example, this is a matter of style and you should feel free to alter the implementation as you see fit.

Workflows

Finally, the workflow logic needed for this solution involves two pieces: managing state transitions and notifying users. These capabilities could be implemented using Workflow Foundation or simply by having the custom form perform the duties of both state transitions and sending email alerts to various parties. I chose to use custom form logic to manage state transitions and SharePoint Designer workflows to notify users because I believe it provides a good mix of simplicity and configurability. State transitions are unlikely to change as much as the messages sent to users, for example, and having messages defined within a SharePoint Designer workflow will allow administrators to make alterations easily without changing code. Alternatives include creating your own email configuration list and using custom code to read from that configuration. Obviously that is more work than simply creating a SharePoint Designer workflow, but as previously mentioned, feel free to go with an implementation that you feel most comfortable with.

BUILDOUT

Now that you have worked out the details of your site taxonomy and provisioning process, and used that information to determine your solution architecture, it's time to actually build out the solution. In this section, you build a complete provisioning solution from scratch. You start by creating a new site collection to host your solution and then create and build out a new project that references that site collection.

When using the built-in SharePoint deployment functionality within Visual Studio, your site collection level feature is reactivated whenever you deploy, causing your solution lists to be re-created each time you redeploy and test your solution. This is great when you are developing your lists, but after you have sample data and configuration in your solution, this behavior may be undesirable.

Consider creating a PowerShell script to upgrade the existing solution. Place that script in your output directory. You may then reference this file in the SharePoint predeployment command (in your project settings) to automatically deploy using your script instead of the default functionality. Just be sure to create and use a new deployment configuration that runs only your batch file and resets the application pool upon deployment.

Alternatively, you can simply view the properties of the list instance folders within your project (defined in the following sections) and set the `DeploymentConflictResolution` property value to `None` to stop Visual Studio from attempting to re-create those lists on each deployment.

www.it-ebooks.info

Creating the Project Framework

Before you create your project in Visual Studio, it's helpful to create a site collection to host the process on your development server. This enables you to easily set up the Visual Studio project to deploy to that site. If you don't have a web application set up on your development farm, create one at `http://localhost`. Otherwise, you may use any existing content web application to host the solution. In this case, you create a site collection called Site Requests because the primary reason for users to visit the site will be to submit or approve site requests through your solution.

To begin, create your site collection:

1. Open central Administration.
2. Select Application Management from the left nav. The Application Management page displays.
3. Select Create site collections. The Create site collection page displays.
4. Ensure the web application is set to the correct application where you would like to host your site request process.
5. Enter **Site Requests** for the title, **/sites/SiteRequests** for the URL, and **Blank Site** for the site type.
6. Enter your username under the Primary Site Collection Administrator field.
7. Select No Quota.
8. Click OK to create the site collection.
9. Copy the site URL that appears after the site is created.

Next, create your project in Visual Studio (see Figure 9-3):

1. Open Visual Studio 2010, and click New Project. The New Project window opens.
2. Within Visual C# project types, select SharePoint, and click Empty SharePoint Project.
3. Type a name for your project (**CompanyName.SiteProvisioning**) and make sure the project location is appropriate. Click OK.
4. On the next screen that appears, under What Local Site Do You Want to Use For Debugging? paste or type the URL of the site collection you created to host your solution.
5. Under What Is the Trust Level for This SharePoint Solution? select Deploy as a Farm Solution.
6. Click Finish. Your project is created.

Creating the Necessary Lists

Now that you have a project and site set up, you can begin the work of creating your list structures to store sites that have been requested and created. To begin, create a site collection level feature that functions to create the necessary lists.

To create the site collection level feature, follow these steps:

1. In the Project Explorer view, right-click the `Features` folder, and select Add Feature. A new feature file titled `Feature.feature` displays.

www.it-ebooks.info

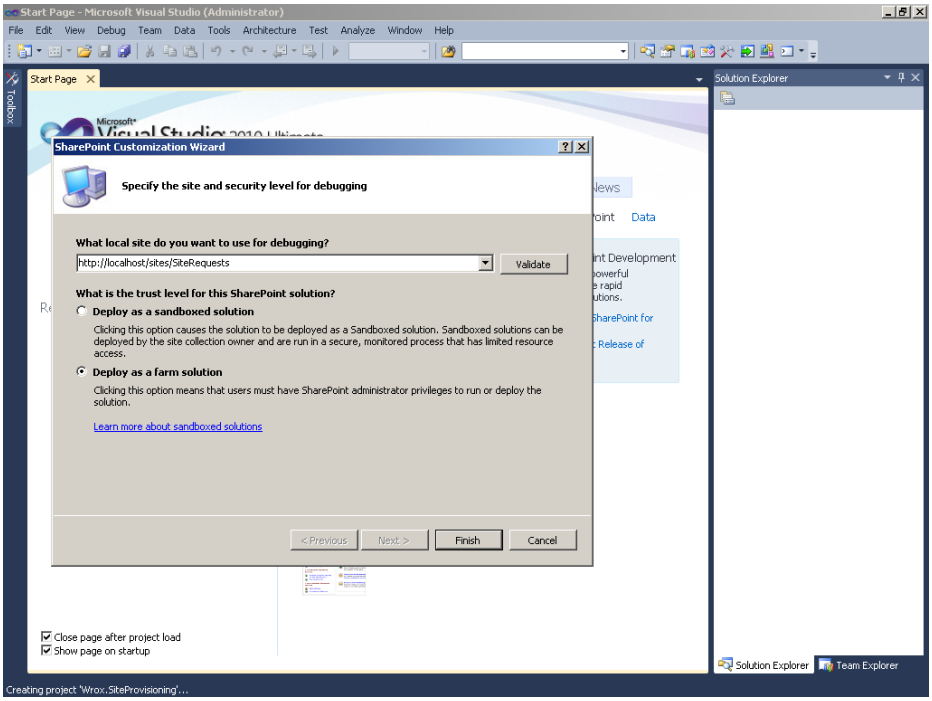


FIGURE 9-3: Creating the Project

2. Update the title to **Site Provisioning Process** and enter an appropriate description.
3. Set the scope to Site and save the file.
4. In the Project Explorer view, select the feature folder you just created. Click the title to rename it.
5. Rename the folder **CompanyName.SiteProvisioning.Site**. This naming convention indicates a site level feature called Site Provisioning.

Site Type Configuration List

Next, create a list definition for your Site Types list using the following steps:

1. In the Project Explorer, right-click the project node (not the solution node) and select Add, and then New Item.
2. Under Installed Templates to the left, within Visual C# project types, select SharePoint, then select 2010.
3. From the available types, select List Definition, and enter a name of **Site Types**.
4. Click Add. The Choose List Definition Settings dialogue appears.
5. Enter **Site Types** for the display name of the list definition.

www.it-ebooks.info

6. Under What Is the Type of the List Definition, select Custom List.
7. Ensure Add a List Instance for This Definition is checked.
8. Click Finish. The list definition is created.

At this point, Visual Studio has created the necessary files needed to provision a list within a site collection. This includes a list instance that was automatically created when you created your list definition. Both the instance and definition have been added to the feature you created earlier. (Visual Studio automatically adds items to features where appropriate.)

Before continuing, you must rename the list instance and set the details of the instance:

1. In the Project Explorer, select the `ListInstance1` folder beneath the `Site Types` list definition you just created.
2. Click the title to rename to **Site Type Configuration**.
3. Double-click the Site Type Configuration node to bring up the list instance `Elements.xml` file and edit as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <ListInstance Title="Site Type Configuration"
        OnQuickLaunch="FALSE"
        TemplateType="10000"
        Url="Lists/SiteTypes"
        Description="Used to configure site types available for creation.">
  </ListInstance>
</Elements>
```

Next, update the details of the list definition to include the fields necessary to support your site request process. Table 9-2 describes each field you must define.

TABLE 9-2: Site Type Configuration Fields

FIELD NAME	TYPE	DESCRIPTION
Title	Text	Title of site type.
Description	Note	Describes the appropriate use for this site type.
Manager Approval Required	Boolean	Indicates if approval is required.
Manager Title Match	Text	Used to configure the level of manager required for approval; the system will search for managers up the corporate hierarchy until one is found whose title contains the text in this field; for example, Manager or Dir for director-level approval.

www.it-ebooks.info

FIELD NAME	TYPE	DESCRIPTION
Fallback Approver	Person	If an approver cannot be found, this person will be set as the approver.
URL Prefix	Text	The URL location where sites of this type will be created.
Site Template	Text	The SharePoint site template used to create this type of site.
Site Features	Multiline Text	List of feature GUIDs to enable upon site creation.
Web Features	Multiline Text	List of feature GUIDs to enable upon site creation.
Default Site Administrators	Person	For security configuration.
Default Owners	Person	For security configuration.
Default Members	Person	For security configuration.
Default Viewers	Person	For security configuration.
Site Requestors	Person	People who may request this type of site.

To update the definition to include these columns, perform the following steps:

1. In Project Explorer, under the `Site Types` folder, double-click `Schema.xml` to open the list definition schema file.
2. Delete the following `ContentTypes` node entirely:

```
<ContentTypes>
  <ContentTypeRef ID="0x01">
    <Folder TargetName="Item" />
  </ContentTypeRef>
  <ContentTypeRef ID="0x0120" />
</ContentTypes>
```

3. Update the following XML in the `Fields` node:

```
<Fields>
  <Field Type="Note" Name="Description" DisplayName="Description"/>
  <Field Type="Boolean" Name="ApprovalRequired"
         DisplayName="Manager Approval Required"/>
  <Field Type="Text" Name="ManagerTitle"
         DisplayName="Manager Title Match"/>
  <Field Type="User" Name="FallbackApprover"
         DisplayName="Fallback Approver"/>
  <Field Type="Text" Name="URLPrefix" DisplayName="Site URL Prefix"/>
  <Field Type="Text" Name="SiteTemplate" DisplayName="Site Template"/>
  <Field Type="Note" Name="SiteFeatures" DisplayName="Site Features"/>
  <Field Type="Note" Name="WebFeatures" DisplayName="Web Features"/>
  <Field Type="UserMulti" Name="DefaultAdministrators"
         DisplayName="Default Site Administrators"/>
```

www.it-ebooks.info

```
  <Field Type="UserMulti" Name="DefaultOwners"
         DisplayName="Default Owners"/>
  <Field Type="UserMulti" Name="DefaultMembers"
         DisplayName="Default Members"/>
  <Field Type="UserMulti" Name="DefaultViewers"
         DisplayName="Default Viewers"/>
  <Field Type="UserMulti" Name="SiteRequestors"
         DisplayName="Site Requestors"/>
</Fields>
```

New Site Requests List

Next, create a list definition for your Site Requests list:

1. In the Project Explorer, right-click the `Project` node (not the `Solution` node) and select Add and then New Item.
2. Under Installed Templates to the left, within Visual C# project types, select SharePoint, 2010.
3. From the available types, select List Definition, and enter a name of **Site Requests**.
4. Click Add, and the Choose List Definition Settings dialogue appears.
5. Enter **Site Requests** for the display name of the list definition.
6. Under What Is the Type of the List Definition, select Custom List.
7. Ensure Add a List Instance for This Definition is checked.
8. Click Finish. The list definition is created.

As with the site type list, you must rename the list instance and set the details of the instance:

1. In the Project Explorer, select the `ListInstance1` folder beneath the `Site Requests` list definition you just created.
2. Click the title to rename to **New Site Requests**.
3. Double-click the `New Site Requests` node to bring up the list instance `Elements.xml` file, and edit as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <ListInstance Title="New Site Requests"
     OnQuickLaunch="TRUE"
     TemplateType="10001"
     Url="Lists/Requests"
     Description="Stores site requests submitted to this site for provisioning.">
  </ListInstance>
</Elements>
```

Next, update the details of the list definition to include the fields needed for this list to store new site requests. Table 9-3 describes the fields you need to add to this list definition.

www.it-ebooks.info

TABLE 9-3: New Site Request Fields

FIELD NAME	TYPE	DESCRIPTION
URL	Text	The suggested URL for the new site
Type	Text	The type of site
Description	Multiline text	A description of how the site will be used
Approver	Person	The user who must approve this request
Approver Comments	Multiline text	Comments entered by the approver
Status	Choice	Status of the request
Site Owner	Person	Intended site owner
Provisioning Status	Text	Useful when troubleshooting, these fields are updated with provisioning details during site creation.
Provisioning Log	Multiline Text	

To update the definition to include these columns, perform the following steps:

1. In Project Explorer, under the `Site Requests` folder, double-click `Schema.xml` to open the list definition schema file.
2. Delete the `ContentTypes` node entirely.
3. Update the following XML in the `Fields` node:

```
<Fields>
  <Field Type="Text" Name="SiteURL" DisplayName="Site URL"/>
  <Field Type="Text" Name="SiteType" DisplayName="Site Type"/>
  <Field Type="Note" Name="Description" DisplayName="Description"/>
  <Field Type="User" Name="Approver" DisplayName="Approver"/>
  <Field Type="Note" Name="ApproverComments"
         DisplayName="Approver Comments"/>
  <Field Type="Choice" Name="Status" DisplayName="Status">
    <CHOICES>
      <CHOICE>Waiting for Approval</CHOICE>
      <CHOICE>Rejected</CHOICE>
      <CHOICE>Waiting on Site Creation</CHOICE>
      <CHOICE>Site Created</CHOICE>
      <CHOICE>Site Creation Failed</CHOICE>
    </CHOICES>
    <Default>Waiting for Approval</Default>
  </Field>
  <Field Type="User" Name="SiteOwner" DisplayName="Site Owner"/>
  <Field Type="Text" Name="ProvisioningStatus"
         DisplayName="Provisioning Status"/>
  <Field Type="Note" Name="ProvisioningLog"
         DisplayName="Provisioning log"/>
</Fields>
```

www.it-ebooks.info

Created Sites List

Finally, create a list definition for your Created Sites list:

1. In the Project Explorer, right-click the `Project` node (not the `Solution` node) and select Add, then select New Item.
2. Under Installed Templates to the left, within Visual C# project types, select SharePoint, 2010.
3. Select List Definition from the available types and enter a name of **Created Sites**.
4. Click Add; the Choose List Definition Settings dialogue appears.
5. Enter Created Sites for the display name of the list definition.
6. Under What Is the Type of the List Definition, select Custom List.
7. Ensure Add a List Instance for This Definition is checked.
8. Click Finish. The list definition is created.

As with the site type list, you must rename the list instance and set the details of the instance:

1. In the Project Explorer, select the `ListInstance1` folder beneath the `Created Sites` list definition you just created.
2. Click the title to rename to **Sites**.
3. Double-click the `Sites` node to bring up the list instance`Elements.xml` file and edit as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <ListInstance Title="Sites"
    OnQuickLaunch="TRUE"
    TemplateType="10002"
    Url="Lists/Sites"
    Description="Records sites created via the site provisioning process.">
  </ListInstance>
</Elements>
```

Next, add the fields described in Table 9-4 to the Sites list definition. This list definition will be used for your Created Sites list and will be used to store records of sites that have already been created.

TABLE 9-4: Created Sites Fields

COLUMN NAME	TYPE	DESCRIPTION
Title	Text	Title of the created site
URL	URL	Site URL
Site Type	Text	Type of site created
Description	Note	Site request description

www.it-ebooks.info

COLUMN NAME	TYPE	DESCRIPTION
Site Owner	Person	Intended site owner
Requested By	Person	Person who requested the site
Approver	Person	Site approver
Approver Comments	Multiline Text	From site creation process

To update the definition to include these columns, perform the following steps:

1. In Project Explorer, under the `Created Sites` folder, double-click `Schema.xml` to open the list definition schema file.
2. Entirely delete the `ContentTypes` node.
3. Update the following XML in the `Fields` node:

```
<Fields>
  <Field Type="Text" Name="SiteURL" DisplayName="Site URL"/>
  <Field Type="Text" Name="SiteType" DisplayName="Site Type"/>
  <Field Type="Note" Name="Description" DisplayName="Description"/>
  <Field Type="User" Name="SiteOwner" DisplayName="Site Owner"/>
  <Field Type="User" Name="RequestedBy" DisplayName="Requested By"/>
  <Field Type="User" Name="Approver" DisplayName="Approver"/>
  <Field Type="Note" Name="ApproverComments"
         DisplayName="Approver Comments"/>
</Fields>
```

Save all your files, and then build, package, and deploy to test as follows:

1. Save All.
2. In Project Explorer, right-click the `Project` node, and select Build to build the project.
3. Right-click the `Project` node and select Package to create the WSP.
4. Right-click the `Project` node and select Deploy. The project is deployed to the server and your feature is enabled on the site collection you built earlier.
5. Visit the site collection you built earlier.
6. Click New Site Requests and view the form that displays.
7. Do the same for the other two lists, confirming that the list fields were properly created.

Creating a Custom Feature Receiver

Before you can create your custom form, you must ensure security settings are properly configured on your site. To accomplish this, create a feature receiver for your site level feature. This enables you to execute custom code whenever the feature is enabled. In this case, the feature receiver creates a new security group called Site Creation Administrators used to secure the solution. This group will

www.it-ebooks.info

be granted full control over the request site. You will also enable all users to visit the site so that they may submit a request. Finally, use the feature receiver to set a web application property indicating the location of your Site Requests site collection. This will be used to prevent activation of this feature on more than one site in a given web application, as well as allow for other functionality that might need to find and access data located within the site collection. Because the feature sets a web application property, it also needs to remove that property during the feature deactivating event.



Web application properties are made up of a property bag containing key-value pairs, accessible via `SPWebApplication.Properties` *and persisted to the central configuration database by calling* `SPWebApplication.Properties.Update()`. *This feature stores the ID of its parent site collection in the parent web application's property bag.*

Although visitors may not have the rights needed to add new list items themselves (in this case, site requests), you can make use of a custom form that elevates privileges to bypass the list security settings, allowing visitors to submit new site requests. This setup controls visitors' access to the underlying list data while preserving full control for administrators.

Creating the Receiver Class

To begin, create a feature receiver for your site level feature:

1. In Solution Explorer, under the `Features` folder, right-click the Site feature created earlier (`CompanyName.SiteProvisioning.Site`) and select Add Event Receiver. An event receiver is added to your feature.
2. Double-click the newly created event receiver file (`CompanyName.SiteProvisioning.EventReceiver.cs`). The file opens for editing.
3. Uncomment the `FeatureActivated()` method and insert the code in Listing 9-1.



Available for download on Wrox.com

LISTING 9-1: Feature Activation

```
//get the current site
SPSite site = properties.Feature.Parent as SPSite;

//check the webapp property bag value
SPWebApplication webApp = site.WebApplication;
string key = "Wrox.SiteProvisioning.SiteID";
if (webApp.Properties.ContainsKey(key))
{
    if (!webApp.Properties[key].Equals(site.ID))
    {
        throw new ApplicationException(
                    "Provisioning Site Already Exists"
                    );
    }
}
else
```

www.it-ebooks.info

```
{
    webApp.Properties.Add(key, site.ID);
    webApp.Update();
}

//check for the admin group's existence
SPGroup admins = null;
try
{
    admins = site.RootWeb.SiteGroups["Site Creation Administrators"];
}
catch { ; }//eat the error if the group can't be found

if (admins == null)
{
    //Create the administrators group
    site.RootWeb.SiteGroups.Add(
                "Site Creation Administrators",
                site.RootWeb.SiteAdministrators[0],
                null,
                "This group has full control over site creation."
                );

    admins = site.RootWeb.SiteGroups["Site Creation Administrators"];

    //grant them full control over the site
    SPRoleAssignment assignment = new SPRoleAssignment(admins);

    assignment.RoleDefinitionBindings.Add(
                    site.RootWeb.RoleDefinitions["Full Control"]
                    );

    site.RootWeb.RoleAssignments.Add(assignment);
    site.RootWeb.Update();
}

//Get an SPUser object representing all users
SPUser allusers =
       site.RootWeb.EnsureUser("NT AUTHORITY\\authenticated users");

//Add to the associated visitors group
if (site.RootWeb.AssociatedVisitorGroup != null)
{
    try
    {
        site.RootWeb.AssociatedVisitorGroup.AddUser(allusers);
        site.RootWeb.AssociatedVisitorGroup.Update();
    }
    catch { ; }//ignore the error if allusers has already been added

    //grant visitors access to the requests list
    SPRoleAssignment assignment =
                new SPRoleAssignment(site.RootWeb.AssociatedVisitorGroup);

    assignment.RoleDefinitionBindings.Add(
```

continues

www.it-ebooks.info

LISTING 9-1 *(continued)*

```
                site.RootWeb.RoleDefinitions["Contribute"]
                );

    SPList requestsList = site.RootWeb.Lists["New Site Requests"];
    requestsList.DisableGridEditing = true;
    requestsList.BreakRoleInheritance(true);
    requestsList.RoleAssignments.Add(assignment);
    requestsList.Update();

}


//finally, we'll set the custom add/edit form for your list
SPContentType requests = site.RootWeb.Lists["New Site Requests"].ContentTypes[0];
requests.EditFormUrl = "_layouts/Wrox.SiteProvisioning/SiteRequests.aspx";
requests.NewFormUrl = requests.EditFormUrl;
requests.Update();
```

Wrox.SiteProvisioning\Features\Wrox.SiteProvisioning.Site\Wrox.SiteProvisioning.EventReceiver.cs

This code performs the following:

- Checks that this feature isn't already enabled on a site within the current web application.
- Sets a web application property indicating that this site will be used to host the request process.
- Creates a Site Creation Administrators group if none exists.
- Grants full control to Site Creation Administrators.
- Ensures that all users will have visitor access to the site.
- Ensures that all users have contributor access to the New Site Requests list. This enables them to use the out-of-the-box SharePoint functionality to add items to the new site requests list. Although this is not necessary, as your custom form will elevate privileges when creating and updating site requests, it is helpful when testing.
- Disables datasheet editing on the New Site Requests list. This ensures that only your custom form is used to interact with the data.
- Updates the New and Edit form URLs for your site requests list to point to your custom page. (You will create that later.) You need to alter the URL set in this code snippet to match your project.

Configuring Sample List Items

If you've chosen to leave deployment conflict resolution enabled for the Site Type Configuration list (more on this later in this chapter), you may modify and add the following code snippet to the bottom of the `FeatureActivated()` event to make debugging easier.

```
//create sample site type configuration for testing.
SPWeb web = site.RootWeb;
```

www.it-ebooks.info

```
SPListItem item = web.Lists["Site Type Configuration"].Items.Add();
item["Title"] = "Test";
item["URLPrefix"] = "http://localhost/sites/";
item["ApprovalRequired"] = false;

SPUser user = site.RootWeb.EnsureUser("RDADEV\\ALICE");
item["FallbackApprover"] =
             new SPFieldUserValue(site.RootWeb, user.ID, user.LoginName);

user = site.RootWeb.EnsureUser("RDADEV\\TEST");
item["SiteRequestors"] =
             new SPFieldUserValue(site.RootWeb, user.ID, user.LoginName);

item["SiteTemplate"] = "STS#0";
item.Update();
```

This ensures that a sample site type configuration item is created when your solution deploys, enabling you to avoid re-creating a sample site type over and over. You must replace `http://localhost`, `RDADEV\\ALICE`, and `RDADEV\\TEST` to match your environment. If you choose to disable deployment conflict resolution for the Site Type Configuration list, add a condition to this code to ensure it is not executed more than once.

Implementing Feature Deactivation

When using a feature receiver, it always good practice to reverse your code changes when the feature is deactivated. In some cases, you want to ensure that all elements of the feature have been removed, such as lists and libraries. In other cases, it's more helpful to design the feature such that it can be deactivated and reactivated with no loss of underlying data, making the feature "safe" to deactivate and reactivate at any time. This is the course you will take with this feature; in this instance, the feature receiver simply has to remove the flag on the web application that indicates that the current site is the site dedicated to hosting the site provisioning process.

Uncomment the `FeatureDeactivating()` method and update with this code:

```
//get the current site
SPSite site = properties.Feature.Parent as SPSite;
SPWebApplication webApp = site.WebApplication;

//remove the property identifying this
//site as the site provisioning site for the web application
webApp.Properties.Remove("Wrox.SiteProvisioning.SiteID");
webApp.Update();
```

This simply ensures that the web application property is removed when the feature is deactivated.

Building and Deploying the Solution

Build, package, and deploy your solution to check for errors. You should find that your site contains a new group called Site Creation Administrators, and that all users have the rights to contribute to the new site requests list. Also, attempts to create new site request list items will fail until you deploy the custom form referenced in your feature receiver. You create that form in the next section.

www.it-ebooks.info

Your lists will be re-created and your feature reactivated each time you deploy. You may want to disable automatic activation to avoid losing configuration stored within your site between deployments.

To disable re-creation of lists on each deployment, perform the following steps:

1. Expand the list definition node for the list in question in Project Explorer.
2. Select the `list instance` node (under the `list definition` node) for the list.
3. Right-click and view properties. The properties window should display the details of this list instance.
4. Set the `Deployment Conflict Resolution` property to a value of **None**. Visual Studio will no longer attempt to re-create the list upon deployment.

Note that if the list definition or instance changes in some way, you will need to delete the list instance in question before redeploying.

Creating a Custom List Item Form

Now it's time to build out your site request form. This form can function as both a new and edit form and contain most of your workflow logic. Before you build the form, you should examine the roles and states that this form must implement. Table 9-5 shows the various roles and states, as well as how the form will be displayed for each combination. If you qualify as an approver, you see the Approver view, even when first requesting the site. The same applies if you are an Administrator.

TABLE 9-5: Form State According to Status and Role

STATUS/ROLE	SITE CREATION ADMINISTRATOR	APPROVER	REQUESTOR
New Entry	All fields editable	N/A (There is no approver set yet.)	Approver Field disabled. Submit and Cancel buttons only.
Awaiting Approval	All fields editable	Type field disabled. Approve and Reject buttons only.	Type and Approver fields disabled. Resubmit and Cancel buttons only.
Rejected	All fields editable	Type field disabled. Approve and Reject buttons only.	Type and Approver fields disabled. Resubmit and Cancel buttons only.
Waiting on Site Creation	All fields editable	All fields disabled.	All fields disabled.
Site Created	All fields editable	All fields disabled.	All fields disabled.
Site Creation Failed	All fields editable	All fields disabled.	All fields disabled.

www.it-ebooks.info

This can help you work out the logic for your form. You can see by looking at the preceding matrix that the site creation administrator always has the same view of the form. Furthermore, for the other roles, the last three statuses result in the same view with all fields disabled. Going a little deeper you can see that the approver sees the same view as the requestor, except for the ability to edit the Approver fields (Approver and Approval Comments), which further simplifies the logic you use.

To start with your form, create a custom application page that hosts your form functionality:

1. In the Project Explorer, right-click the `Project` node (not the `Solution` node) and select Add; then select New Item.
2. Under Installed Templates to the left, within Visual C# project types, select SharePoint, 2010.
3. Select Application Page from the available types, and enter a name of **SiteRequests.aspx**. The page is created.

At this point, an ASPX page is created, and you can begin to customize it. First though, the code presented next makes use of SharePoint User Profiles as well as Active Directory libraries. To ensure that you have the correct references in your project, perform the following steps:

1. In Solution Explorer, right-click the `References` node.
2. Select Add Reference.
3. Select the .NET tab and wait for references to load.
4. Click the Component Name header to sort by component name.
5. Find the reference for `Microsoft.SharePoint.Server` and click OK. The reference is added to your project.
6. Repeat these steps to add the following additional references to your project:
   - `Microsoft.Office.Server.UserProfiles`
   - `Microsoft.Office.Server.Security`
   - `System.DirectoryServices.AccountManagement`

To ensure that your form code properly compiles, you must add the necessary `using` statement to the top of your code file:

1. In Solution Explorer, expand the `SiteRequests.aspx` node, and double-click the `SiteRequests.aspx.cs` code-behind file.
2. Add the following `using` statements to the top of your code file:

```
using System.Collections;
using System.DirectoryServices.AccountManagement;
using System.Collections.Generic;
using Microsoft.Office.Server.UserProfiles;
```

www.it-ebooks.info

Your form can keep a local variable referencing the list item being created or edited, as well as a reference to an `SPSite` object created under elevated privileges. To continue, declare these variables in your class file:

1. In Solution Explorer, expand the `SiteRequests.aspx` node, and double-click the `SiteRequests.aspx.cs` code-behind file.
2. Add member variables to the page class that can store the `SPListItem` for the item added or edited and the `SPSite` utilized during page execution:

```
protected SPListItem item;
protected SPSite elevatedSite;
```

Creating Form Helper Functions

Because you will be checking to see if a user is eligible to request a certain type of site, you'll also need to determine if the user belongs to a given Active Directory group. Add this function to your page:

```
private bool IsUserInGroup(string userName, string groupName)
{
    PrincipalContext pc = new PrincipalContext(ContextType.Domain,
                                               Environment.UserDomainName);
    PrincipalSearchResult<Principal> searchResults;
    UserPrincipal user;

    user = UserPrincipal.FindByIdentity(pc, userName);
    searchResults = user.GetGroups(pc);

    foreach (Principal grp in searchResults)
    {
        string name = grp.Context.Name + "\\" + grp.Name;
        if (groupName.ToUpper() == name.ToUpper())
        {
            return true;
        }
    }

    return false;
}
```

Next, add a helper function called `IsRequestorEligible()` to determine if the current user is allowed to request a given type of site. This function can accept the list item storing the site type in question and make use of the `IsUserInGroup()` function you just created. Add the function in Listing 9-2 to your form.



LISTING 9-2: Determining Requestor Eligibility

```
private bool IsRequestorEligible(SPListItem siteType)
{

    string currentLogin = SPContext.Current.Web.CurrentUser.LoginName;
```

www.it-ebooks.info

```
        //if the current user operates as system, this will be the login name
        if (currentLogin.ToUpperInvariant() == "SHAREPOINT\\SYSTEM")
        {
            return true; //we'll let system request anything
        }

        string requestors = null;
        try
        {
            requestors = siteType["SiteRequestors"].ToString();
        }
        catch(System.NullReferenceException) { ;} //item might be null

        if (string.IsNullOrEmpty(requestors))
        {
            return true;
        }

        SPFieldUserValueCollection Multiusers =
            new SPFieldUserValueCollection(SPContext.Current.Web, requestors);
        foreach (SPFieldUserValue user in Multiusers)
        {

            if (user.User.IsDomainGroup)
            {
                if (IsUserInGroup(currentLogin, user.User.LoginName))
                {
                    return true;
                }
            }
            else
            {
                if (user.User.LoginName == currentLogin)
                {
                    return true;
                }
            }
        }

        return false;
    }
```

Wrox.SiteProvisioning\Layouts\Wrox.SiteProvisioning\SiteRequests.aspx.cs

Next, add a function that determines the appropriate approver for a given request according to the site type requested and the user performing the request. This function also accepts the list item for the site type in question as a parameter. Add the function in Listing 9-3 to your form.

www.it-ebooks.info



LISTING 9-3: Finding the Approver

```
private SPUser GetApprover(SPListItem siteType)
{
    SPWeb web = SPContext.Current.Web;

    if ((Boolean)(siteType["ApprovalRequired"] ?? false))
    {
        SPServiceContext ctx = SPServiceContext.Current;
        UserProfileManager pm = new UserProfileManager(ctx);
         UserProfile currentUser = null;
        try
        {
            currentUser = pm.GetUserProfile(web.CurrentUser.LoginName);
        }
        catch
        {
            //no user profile could be found, return the fallback approver
            string fallback = siteType["FallbackApprover"].ToString();
            return new SPFieldUserValue(web, fallback).User;
        }

        Dictionary<Guid, bool> found = new Dictionary<Guid, bool>();
        UserProfile manager = null;
        try
        {
            manager = currentUser.GetManager();
        }
        catch { ;}//there may not be a manager...

        if (!string.IsNullOrEmpty((string)siteType["ManagerTitle"]))
        {
            while (manager != null && !found.ContainsKey(manager.ID))
            {
                found.Add(manager.ID, true);

                string jobTitle = "";
                try{
                    jobTitle = manager["Title"].Value.ToString();
                }catch{;}//property may not exist or may be null

                if (jobTitle.Contains((string)siteType["ManagerTitle"]))
                {
                    string mgr = (string)manager["AccountName"].Value;
                    return web.EnsureUser(mgr);
                }

                try
                {
                    manager = manager.GetManager();
                }
                catch {
                    manager = null;
                }//there may not be a manager...
            }
```

www.it-ebooks.info

```
            }
            else
            {
                if (manager != null)
                {
                    string mgr = (string)manager["AccountName"].Value;
                    return elevatedSite.RootWeb.EnsureUser(mgr);
                }
            }

            return new SPFieldUserValue(
                        web,
                        siteType["FallbackApprover"].ToString()
                        ).User;

        }
        else
        {
            return null;
        }

    }
```

Wrox.SiteProvisioning\Layouts\Wrox.SiteProvisioning\SiteRequests.aspx.cs

Examining this function, consider a few things:

- The code makes use of user profiles to determine the manager of a given user. This means that you must have your User Profile Service configured, and a profile import complete, for this code to function. Any time you alter the data in Active Directory (such as changing a manager or job title), you need to re-import Active Directory data into the User Profile Service for this code to properly function.
- The code keeps track of "found" users in a hash table and stops executing if the same user is found twice. This is to avoid an infinite loop while searching for managers because many organizations require that all users have an assigned manager, often with the CEO defined as his own manager. If a loop is encountered, the code defaults to the fallback approver.
- The code defaults to the fallback approver if a manager cannot be found.
- If manager title match is not specified in the Site Type Configuration list, the user's direct manager is chosen.

Creating Form Controls

Now that your form's helper functions are defined, you may continue by adding the necessary controls to your form.



1. In Solution Explorer, double-click the new ASPX page you just created.
2. Supply text for the `PageTitle` and `PageTitleInTitleArea` content controls with the name of your list, in this case `New Site Requests`.

www.it-ebooks.info

3. Add the following ASPX markup to the `Main` content control.

```
<table>
<tr>
    <td class="ms-formlabel" valign="top" width="190" nowrap>
        <h3 class="ms-standardheader">
            <nobr>Site Type</nobr><span class="ms-formvalidation">*</span>
        </h3>
    </td>
    <td class="ms-formbody" valign="top">
        <asp:DropDownList runat="server" ID="ddlSiteType"
             AutoPostBack="true" />
        <asp:RequiredFieldValidator runat="server"
             ControlToValidate="ddlSiteType"
             Text="You must select a site type." />
    </td>
</tr>
```

This markup opens a table tag and creates a row with two columns, starting a form with a two-column layout. The first column will be used to display a field name (in this case "Site Type"), while the second column will contain the control used to edit the field data. Note the use of `ms-formlabel`, `ms-formbody`, `ms-standardheader` and `ms-formvalidation` CSS classes. These are provided by SharePoint and are used here to provide a consistent look and feel for forms. Furthermore, the site type drop-down must be set to post back automatically because setting the site type will change the view of the form. Also note the use of a RequiredFieldValidator control to prevent the user from submitting the form without providing a site type.

4. Next, add the site URL field to your form:

```
<tr>
    <td class="ms-formlabel" valign="top" width="190" nowrap>
        <h3 class="ms-standardheader">
            <nobr>Site URL</nobr><span class="ms-formvalidation">*</span>
        </h3>
    </td>
    <td class="ms-formbody" valign="top">
        <asp:Literal runat="server" ID="litUrlPrefix"/><asp:TextBox
             ID="txtURL" Enabled="false" runat="server" />
        <asp:RequiredFieldValidator  runat="server"
             ControlToValidate="txtURL" Text="URL is Required." />
    </td>
</tr>
```

Note that the `litUrlPrefix` literal control immediately precedes the `txtURL` text control used to enter the site URL. The literal control will be used to display the prefix (hostname and path) configured for the site type chosen by the user.

5. Now add the site title and site description fields to your form:

```
<tr>
    <td class="ms-formlabel" valign="top" width="190" nowrap>
        <h3 class="ms-standardheader">
```

www.it-ebooks.info

```
            <nobr>Site Title</nobr>
            <span class="ms-formvalidation">*</span>
        </h3>
    </td>
    <td class="ms-formbody" valign="top">
        <asp:TextBox ID="txtTitle" runat="server" />
        <asp:RequiredFieldValidator runat="server"
             ControlToValidate="txtTitle"
               Text="Site Title is Required." />
    </td>
</tr>
<tr>
    <td class="ms-formlabel" valign="top" width="190" nowrap>
        <h3 class="ms-standardheader">
            <nobr>Site Description</nobr>
            <span class="ms-formvalidation">*</span>
        </h3>
    </td>
    <td class="ms-formbody" valign="top">
        <asp:TextBox ID="txtDescription" TextMode="MultiLine" Rows="4"
             runat="server" />
        <asp:RequiredFieldValidator runat="server"
             ControlToValidate="txtDescription"
             Text="Description is Required." />
    </td>
</tr>
```

These fields are pretty straightforward and follow the conventions used by the other fields in this form.

6. Next, add the Site Owner and Site Approver fields to your form:

```
<tr>
    <td class="ms-formlabel" valign="top" width="190" nowrap>
        <h3 class="ms-standardheader">
            <nobr>Site Owner</nobr>
            <span class="ms-formvalidation">*</span>
        </h3>
    </td>
    <td class="ms-formbody" valign="top">
        <SharePoint:PeopleEditor runat="server" ID="pplOwner"
             AutoPostBack="true" AllowEmpty="false"
             MultiSelect="false" />
    </td>
</tr>
<tr runat="server" id="trApprover">
    <td class="ms-formlabel" valign="top" width="190" nowrap>
        <h3 class="ms-standardheader">
            <nobr>Approver</nobr>
        </h3>
    </td>
    <td class="ms-formbody" valign="top">
        <SharePoint:PeopleEditor runat=server ID="pplApprover"
             AutoPostBack="true" MultiSelect="false" />
    </td>
</tr>
```

www.it-ebooks.info

The Site Owner and Site Approver field makes use of the SharePoint `PeopleEditor` control, which the form is using to allow the end user to select the site owner. Since the SharePoint `PeopleEditor` control does not support the ASP.NET `RequiredFieldValidator` control, your form will make use of another method to validate these fields, shown in the next section.

7. Next, add the Approver Comments and Status fields to your form and close out the table:

```
<tr runat="server" id="trApproverComments">
    <td class="ms-formlabel" valign="top" width="190" nowrap>
        <h3 class="ms-standardheader">
            <nobr>Approver Comments</nobr>
        </h3>
    </td>
    <td class="ms-formbody" valign="top">
        <asp:TextBox ID="txtApproverComments" TextMode="MultiLine"
             Rows="4" runat="server" />
    </td>
</tr>
<tr runat="server" id="trStatus">
    <td class="ms-formlabel" valign="top" width="190" nowrap>
        <h3 class="ms-standardheader">
            <nobr>Status</nobr>
        </h3>
    </td>
    <td class="ms-formbody" valign="top">
        <asp:Literal ID="litStatus" runat="server" />
    </td>
</tr>
<tr runat="server" id="trProvisioningStatus">
    <td class="ms-formlabel" valign="top" width="190" nowrap>
        <h3 class="ms-standardheader">
            <nobr>Provisioning Status</nobr>
        </h3>
    </td>
    <td class="ms-formbody" valign="top">
        <asp:Literal ID="litProvisioningStatus" runat="server" />
    </td>
</tr>
</table>
<br />
```

Note that the status fields are display-only, shown via literal controls. Provisioning Status is used to display the status of a site once approved for creation, whether it is in the process of being created, is successfully created, or if an error occurred during creation.

8. To complete your form markup, add the following elements:

```
<asp:Button Text="Save" ID="btnSave" runat="server" />
   
<asp:Button Text="Approve" ID="btnApprove" runat="server" />
   
<asp:Button Text="Reject" ID="btnReject" runat="server" />
   
```

www.it-ebooks.info

```
<asp:Button Text="Cancel" ID="btnCancel" runat="server"
CausesValidation="false" />
<asp:Literal runat="server" ID="litJSPlaceHolder" />
```

The `litJSPlaceHolder` literal control is provided to enable the form code to inject JavaScript into the form HTML. You can use this JavaScript placeholder to close the form when in Dialog mode and to alert the user when necessary. You have added `RequiredFieldValidators` to all required fields, except for the `Site Owner` field. This enables you to avoid writing an extensive validation function because no actions can be taken on the form without data supplied to all the validated fields. The Cancel button has a property of `CausesValidation` set to `false` to enable canceling of incomplete forms, and table rows that may be hidden depending on form state have been provided with IDs and `runat=` server attributes, enabling you to show and hide these rows from code. The `txtURL TextBox` control, used to store the URL, is disabled by default until a site type is chosen from the drop-down list.

The `Site Owner` field makes use of the SharePoint People Editor control, which the form uses to enable the end user to select the site owner. Because the SharePoint People Editor control does not support the ASP.NET `RequiredFieldValidator` control, your form can make use of another method to validate this field, as shown in the next section.

Wrox.SiteProvisioning\Layouts\Wrox.SiteProvisioning\SiteRequests.aspx

Building Out the Custom Form Code

Now that you have your controls and layout defined, you can get back to the form code. Before you build the main form code, however, you'll need to add some helper functions to your form. These helper functions will perform tasks such as validating the site owner, closing the form, and loading site types from your "Site Types" configuration list.

Validating the Site Owner

Although the form markup makes use of `RequiredFieldValidator` controls for most required fields, the SharePoint `PeopleEditor` control does not support the `RequiredFieldValidator` control. You need a function to validate the Site Owner field. By setting the `AllowEmpty` property of the `PeopleEditor` control to `false` in the ASPX markup, you can simply call `PeopleEditor.Validate()`, then check the `PeopleEditor.IsValid` field to detect whether a Site Owner was supplied. This also ensures that an error message is displayed to the user if the field is empty. The example code that follows demonstrates how to check the number of entities selected instead of checking the `IsValid` property. This example is helpful if you require more than one site owner to be selected or wish to make the number of site owners required dependent on the site type selected. Use the following steps to create the function:

1. In Solution Explorer, expand the `SiteRequests.aspx` node, and double-click the `SiteRequests.aspx.cs` code-behind file.
2. Add the following function to your code file:

```
private bool CustomValidate()
{
```

www.it-ebooks.info

```
        //since allowempty is set to false, calling "Validate" will cause an
        //error message to be displayed to the user by default
        pplOwner.Validate();
        if (pplOwner.ResolvedEntities.Count == 0)
        {
            return false;
        }
        else
        {
            return true;
        }
    }
```

Closing the Form

Because the form may be opened in a dialogue or as a separate page, you need to check the `Form` mode to determine whether to write some JavaScript to the page that will close the dialogue or simply redirect back to the default view for the list.

It's useful to have a helper function to close the form. Add the following function to your code file:

```
    private void CloseForm()
    {
        if (IsDialogMode)
        {
           litJSPlaceHolder.Text =
"<script>window.frameElement.commitPopup();</script>";
        }
        else
        {
            Response.Redirect(item.ParentList.DefaultViewUrl);
        }
    }
```

Loading the Site Type Drop-down

Your form controls contain a Site Type drop-down list, used to select which type of site is being requested. Because the form offers the user a choice of site types, you need to load the Site Type drop-down list with values from the Site Type Configuration list.

Add the following function to your code file:

```
    private void LoadSiteTypes()
    {
        SPList list = SPContext.Current.Web.Lists["Site Type Configuration"];

        ddlSiteType.Items.Add(new ListItem("Select a Site Type...", "0"));

        foreach (SPListItem item in list.Items)
        {
            ddlSiteType.Items.Add(
               new ListItem(item.Title, item.ID.ToString()));
        }
    }
```

www.it-ebooks.info

Loading the Form

With all your helper functions implemented, you are now ready to implement the main logic of your form. Add the following code to the `Page_Load()` function in your form:

```
SPSecurity.RunWithElevatedPrivileges(delegate()
{
using (elevatedSite = new SPSite(SPContext.Current.Site.ID))
{
    elevatedSite.RootWeb.AllowUnsafeUpdates = true;
    if (!string.IsNullOrEmpty(Request.QueryString["ID"]))
    {
        //parse the item ID from the query string
        int ID = int.Parse(Request.QueryString["ID"]);
        //get the requests list
        SPList List = elevatedSite.RootWeb.Lists["New Site Requests"];
        //get the item by ID
        item = List.GetItemById(ID);
    }

    //assign event handlers to controls
    btnSave.Click += new EventHandler(btnSave_Click);
    btnCancel.Click += new EventHandler(btnCancel_Click);
    btnApprove.Click += new EventHandler(btnApprove_Click);
    btnReject.Click += new EventHandler(btnReject_Click);

    ddlSiteType.SelectedIndexChanged += new
                EventHandler(ddlSiteType_SelectedIndexChanged);

    if (!IsPostBack)
    {
        //Load site type dropdown list
        LoadSiteTypes();
        if (item != null)
        {
            //load site request values into form controls
            LoadFields();
        }
        //lock down the form
        SecureForm();
    }

}
});
```

This code checks the query string variable ID and loads the `SPListItem` for that particular ID from your New Site Requests list. If no ID is supplied, "item" remains null. This functionality is required due to the way that out-of-the-box list form works in SharePoint, where the list form is loaded with a query string parameter containing the ID of the list item to be viewed or edited.

After checking for a list item ID, the code then assigns event handlers to handle button-click events, followed by an event handler to handle the `SelectedIndexChanged` event of the site type drop-down list. You'll see why this is necessary later.

www.it-ebooks.info

Finally, if the page is not posted back (in other words, it is loaded for the first time, not reloaded as a result of a button press or other event), the code performs the following:

- Loads the Site Type drop-down list with values by calling `LoadSiteTypes()`.
- If an item has been loaded, form fields are loaded with the appropriate data by calling `LoadFields()`, defined in the following section.
- Secures form fields by calling `SecureForm()`, defined later.

These tasks need only be performed on page load because they set various controls' state. These controls' state will be serialized into the page ViewState on subsequent postbacks, eliminating the need to set that state on every page request.

Loading Form Data

Next, implement `LoadFields()` by adding the following function to your form:

```
private void LoadFields()
{
    txtTitle.Text = item.Title;
    txtURL.Text = (item["SiteURL"] ?? "").ToString();
    ddlSiteType.SelectedValue = (item["SiteType"] ?? "0").ToString();
    txtDescription.Text = (item["Description"] ?? "").ToString();
    litStatus.Text = (item["Status"] ?? "").ToString();
    txtApproverComments.Text =
                        (item["ApproverComments"] ?? "").ToString();
    litProvisioningStatus.Text =
                        (item["ProvisioningStatus"] ?? "").ToString();

    if (item["Approver"] != null)
    {
        string login = new SPFieldUserValue(item.ParentList.ParentWeb,
                            item["Approver"].ToString()).User.LoginName;
        pplApprover.CommaSeparatedAccounts = login;
        pplApprover.Validate();
    }

    if (item["SiteOwner"] != null)
    {
        string login = new SPFieldUserValue(item.ParentList.ParentWeb,
                            item["SiteOwner"].ToString()).User.LoginName;
        pplOwner.CommaSeparatedAccounts = login;
        pplOwner.Validate();
    }
}
```

The preceding code retrieves field values from the `item` member variable of the page class, which is loaded during `Page_Load()`, and populates the form controls with those field values. If you add fields to your requests list, you need to alter this function to enable loading of those fields' values into the form's controls.

www.it-ebooks.info

Securing the Form

Now that you have the code required to load list items into the form, you need to lock down certain fields according to role and status by implementing the `SecureForm()` function. Add the function in Listing 9-4 to your code.



LISTING 9-4: Securing Form Controls

```
private void SecureForm()
{
    SPWeb web = SPContext.Current.Web;

    SPGroup admins = web.SiteGroups["Site Creation Administrators"];
    bool isAdmin = admins.ContainsCurrentUser;

    string currentLogin = SPContext.Current.Web.CurrentUser.LoginName;

    //if the current user operates as system, this will be the login name
    if (currentLogin.ToUpperInvariant() == "SHAREPOINT\\SYSTEM")
    {
        isAdmin = true; //we'll let system act as an administrator
    }

    if (isAdmin)
    {
        return;
    }


    trProvisioningStatus.Visible = false;
    ddlSiteType.Enabled = false;
    btnApprove.Visible = false;
    btnReject.Visible = false;

    if (item == null)
    {
        //new item, no approver yet
        trApprover.Visible = false;
        trApproverComments.Visible = false;

        //submit and cancel only
        ddlSiteType.Enabled = true;
    }
    else
    {

        bool isApprover = false;
        if (item["Approver"] != null)
        {
            SPUser approver = new SPFieldUserValue(web,
                                                   (string)item["Approver"]
                                                   ).User;
            if (approver.LoginName == web.CurrentUser.LoginName)
```

continues

www.it-ebooks.info

LISTING 9-4 *(continued)*

```
            {
                isApprover = true;
            }
        }

        if (item["Status"].ToString() == "Waiting for Approval"
            || item["Status"].ToString() == "Rejected")
        {
            if (isApprover)
            {
                //approve and reject only
                btnSave.Visible = false;
                btnApprove.Visible = true;
                btnReject.Visible = true;
            }
            else
            {
                //resubmit and cancel only, approver fields disabled
                pplApprover.Enabled = false;
                txtApproverComments.Enabled = false;
                btnSave.Text = "Resubmit Request";
            }
        }
        else
        {
            btnSave.Visible = false;
            btnApprove.Visible = false;
            btnReject.Visible = false;
            txtTitle.Enabled = false;
            txtURL.Enabled = false;
            txtDescription.Enabled = false;
            txtApproverComments.Enabled = false;
            ddlSiteType.Enabled = false;
            pplApprover.Enabled = false;
            pplOwner.Enabled = false;
        }
    }
}
```

Wrox.SiteProvisioning\Layouts\Wrox.SiteProvisioning\SiteRequests.aspx.cs

This code simply sets various fields' `Visible` and `Enabled` properties as needed. If the user is an administrator (checked by determining if the user is either the system account, or belongs to the Site Creation Administrator group created in your feature receiver), the function returns before locking down any fields because administrators always have full access to the data. Otherwise, the code checks to see if the current user is the approver, securing fields as needed depending on the status of the request.

www.it-ebooks.info

Saving Form Data

At this point you have all the code you need to load a list item and display it. To enable saving, you simply need to implement the event handlers for the various buttons and Site Type drop-down list, along with one helper function, `SaveFields()`. Add the `SaveFields()` function in Listing 9-5 to your code.



LISTING 9-5: Saving the Form Data

```
private void SaveFields()
{
    item["Title"] = txtTitle.Text;
    item["SiteURL"] = txtURL.Text;
    item["Description"] = txtDescription.Text;
    item["SiteType"] = int.Parse(ddlSiteType.SelectedValue);
    item["ApproverComments"] = txtApproverComments.Text;

    SPWeb web = SPContext.Current.Web;

    if (pplApprover.ResolvedEntities.Count > 0)
    {
        string ApproverLogin =
               ((PickerEntity)pplApprover.ResolvedEntities[0]).Key;
        int approverID =
               elevatedSite.RootWeb.EnsureUser(ApproverLogin).ID;
        item["Approver"] =
               new SPFieldUserValue(web, approverID, ApproverLogin);
    }

    if (pplOwner.ResolvedEntities.Count > 0)
    {
        string OwnerLogin =
               ((PickerEntity)pplOwner.ResolvedEntities[0]).Key;
        int ownerID =
               elevatedSite.RootWeb.EnsureUser(OwnerLogin).ID;
        item["SiteOwner"] =
               new SPFieldUserValue(web, ownerID, OwnerLogin);
    }

    //manually set the author and editor fields...
    SPFieldUserValue author;
    author = new SPFieldUserValue(web,
                                  web.CurrentUser.ID,
                                  web.CurrentUser.LoginName);

    if (item["Author"] == null)
    {
        item["Author"] = author;
        item["Created"] = DateTime.Now;
    }

    item["Editor"] = author;
    item["Modified"] = DateTime.Now;
}
```

Wrox.SiteProvisioning\Layouts\Wrox.SiteProvisioning\SiteRequests.aspx.cs

www.it-ebooks.info

This code is simply the corollary to `LoadFields()`. Instead of taking list item values and loading them into form fields, this function takes the form field values and loads them into the current list item. As mentioned with `LoadFields()`, any additional fields you choose to create in your site requests list need to be saved back to your list item within this function.

This code manually sets the `Author`, `Created`, `Editor`, and `Modified` fields to appropriate values. This is necessary because the code elevates privileges during the save operation, which would normally cause the `Editor` and `Author` fields to be set to `System Account`. Because you want to track the person who actually created the request, your code must set those fields manually and then call `SPListItem.SystemUpdate()` to update the item. `SystemUpdate()` differs from `Update()` in that `Update()` automatically sets the `Editor` and `Author` fields to the current user, whereas `SystemUpdate()` cannot overwrite those fields.

Implementing the Save Button

Next, add the Save button event handler to your code using the code in Listing 9-6.



LISTING 9-6: Save Button Logic

```
void btnSave_Click(object sender, EventArgs e)
{
    if (!CustomValidate())
    {
        return;
    }

    SPSecurity.RunWithElevatedPrivileges(delegate()
    {
    using (elevatedSite = new SPSite(SPContext.Current.Site.ID))
    {
    elevatedSite.RootWeb.AllowUnsafeUpdates = true;
    SPList siteRequestList =
                    elevatedSite.RootWeb.Lists["New Site Requests"];

    if (item == null)
    {
        //get a new list item
        item = siteRequestList.GetItems(
                                new SPQuery() { Query = "0" }).Add();
    }
    else
    {
        item = siteRequestList.GetItemById(item.ID);
    }


    //look up configuration details
    SPList configList =
             SPContext.Current.Web.Lists["Site Type Configuration"];
    SPListItem siteType =
             configList.GetItemById(int.Parse(ddlSiteType.SelectedValue));
```

www.it-ebooks.info

```
    //if item requires approval, status is set to awaiting approval
    if ((bool)(siteType["ApprovalRequired"] ?? false))
    {
        item["Status"] = "Waiting for Approval";

        SPUser Approver = GetApprover(siteType);
        item["Approver"] = new SPFieldUserValue(Approver.ParentWeb,
                                                Approver.ID,
                                                Approver.LoginName);
    }
    else
    {
        //otherwise status is waiting on site creation
        item["Status"] = "Waiting on Site Creation";
    }


    SaveFields();
    item.SystemUpdate();
    CloseForm();
    }
    });
}
```

Wrox.SiteProvisioning\Layouts\Wrox.SiteProvisioning\SiteRequests.aspx.cs

This is the most complex of the event handlers. After elevating privileges (necessary to support various security configurations because the requestor need not have contribute access to the New Site Requests list), this code creates a new `SPListItem` if needed, loads the Site Type Configuration list item for the Site Type selected by the user, and uses that configuration list item to determine the approver (if needed). Finally, the code sets the appropriate status for the item, saves the item, and closes the form.

Implementing the Site Type Drop-down

Next, implement the Site Type drop-down list event handler by adding the following function to your code:

```
void ddlSiteType_SelectedIndexChanged(object sender, EventArgs e)
{
    //if no site type is selected, we disable the URL field and return
    if (ddlSiteType.SelectedValue == "0")
    {
        litJSPlaceHolder.Text = "";
        litUrlPrefix.Text = "";
        txtURL.Text = "";
        txtURL.Enabled = false;
        return;
    }

    SPList list = SPContext.Current.Web.Lists["Site Type Configuration"];
    SPListItem siteType =
                  list.GetItemById(int.Parse(ddlSiteType.SelectedValue));
```

www.it-ebooks.info

```
        //check that this user is eligible
        if (!IsRequestorEligible(siteType))
        {
            //warn user
            litJSPlaceHolder.Text =
            string.Format(
                        @"<script>
                            alert('You may not request sites of type '{0}'.);
                            </script>",
                           ddlSiteType.SelectedItem.Text);
            ddlSiteType.SelectedValue = "0";
            litUrlPrefix.Text = "";
            txtURL.Text = "";
            txtURL.Enabled = false;
        }

        litUrlPrefix.Text = (siteType["URLPrefix"] ?? "").ToString();
        txtURL.Enabled = true;
    }
```

Note the use of the helper function `IsRequestorEligible()` that you defined earlier, as well as the use of `litJSPlaceHolder` to write JavaScript to the page that pops up an alert to the user. This function also places the URL Prefix for the selected site type into a literal control on the page, enabling the user to visualize the full URL to the site he requests. If the user is not eligible to request a certain type of site, the site URL textbox is cleared and disabled. This way, your form validation does not let the user submit the form without a valid site type selected. The URL textbox is re-enabled when a valid site type is chosen.

Implementing the Approve and Reject Buttons

Lastly, the event handlers for the Approve, Reject, and Cancel buttons must be implemented. Add the following code to your form:

```
    void btnApprove_Click(object sender, EventArgs e)
    {
        if (!CustomValidate())
        {
            return;
        }

        SPSecurity.RunWithElevatedPrivileges(delegate()
        {
            using (elevatedSite = new SPSite(SPContext.Current.Site.ID))
            {
                elevatedSite.RootWeb.AllowUnsafeUpdates = true;
                SPList list = elevatedSite.RootWeb.Lists["New Site Requests"];
                item = list.GetItemById(item.ID);
                item["Status"] = "Waiting on Site Creation";
                SaveFields();
                item.SystemUpdate();
                ProvisionAsync();
                CloseForm();
            }
```

www.it-ebooks.info

```
        });

    }

    void btnReject_Click(object sender, EventArgs e)
    {
        if (!CustomValidate())
        {
            return;
        }

        SPSecurity.RunWithElevatedPrivileges(delegate()
        {
            using (elevatedSite = new SPSite(SPContext.Current.Site.ID))
            {
                elevatedSite.RootWeb.AllowUnsafeUpdates = true;
                SPList list = elevatedSite.RootWeb.Lists["New Site Requests"];
                item = list.GetItemById(item.ID);
                item["Status"] = "Rejected";
                SaveFields();
                item.SystemUpdate();
                CloseForm();
            }
        });

    }

    void btnCancel_Click(object sender, EventArgs e)
    {
        CloseForm();
    }
```

Both approve and reject event handlers are largely identical with the exception that the status is set on the request item. The `cancel` event simply closes the form.

At this point, the form is ready for an initial test. Build and deploy the solution, and then navigate to your New Site Requests list using your web browser. Clicking Add New Item should bring up your custom form, as shown in Figure 9-4.



FIGURE 9-4: Custom form

www.it-ebooks.info

Enter values for the form, and click Save to test the form. The dialogue closes and the page reloads. Your newly created item appears in the list. Select the item and edit it to confirm that the fields are loaded back into the controls. Depending on your site type (and configuration), you see different views depending on status and current user.

To fully test the form, configure users with job titles and managers in Active Directory, followed by a full profile import in the User Profile Service. You can then log in as those different users to see how the form behavior differs among users and item status. The form does not create any sites, but you should monitor how the status changes to confirm that your logic is correct. Refer to the state machine diagram at the beginning of this chapter as a reference for how the status of the item should change. You may want to add additional fields to the default List View to assist with testing.

Creating the Core Site Provisioning Functionality

Now that you have a working request form, you can build out the necessary functionality to actually provision the site based on the request list item. Start by creating a utility class that will handle some basic site provisioning duties, such as configuring security, registering the site in the "Created Sites" list, and enabling features:

1. In Solution Explorer, right-click the `Project` node and select Add, and then Class.
2. Enter `Utility.cs` in the Filename textbox.
3. Click Add. The class file is created.
4. Add the following using statements to the top of the file:

```
using Microsoft.SharePoint;
using Microsoft.SharePoint.Administration;
```

This enables this utility class to access the SharePoint object model used to create the site.

Registering the Site in the "Created Sites" list

Add the following helper functions to the class module.




```
private static void RegisterSiteInSitesList(SPListItem siteRequest,
                                            string siteURL,
                                            SPListItem configItem)
{
    siteRequest.ParentList.ParentWeb.AllowUnsafeUpdates = true;
    SPList sitesList = siteRequest.ParentList.ParentWeb.Lists["Sites"];
    SPListItem siteItem = sitesList.GetItems(
                          new SPQuery(){Query = "0" }).Add();
    siteItem["SiteURL"] = siteURL;
    siteItem["Title"] = siteRequest.Title;
    siteItem["SiteType"] = configItem.Title;
    siteItem["Description"] = siteRequest["Description"];
    siteItem["SiteOwner"] = siteRequest["SiteOwner"];
    siteItem["RequestedBy"] = siteRequest["Author"];
    siteItem["Approver"] = siteRequest["Approver"];
```

www.it-ebooks.info

```
    siteItem["ApproverComments"] = siteRequest["ApproverComments"];
    siteItem.Update();
```

Wrox.SiteProvisioning\Layouts\Wrox.SiteProvisioning\SiteRequests.aspx.cs

This function will create a new item in the "Created Sites" lists, copying data from the site request list item used to request the site. Conveniently, the site request list item contains all the data needed to create the "Created Sites" list item.

Enabling Site Features

Continue with the following function:

```
private static void EnableFeatures(string[] featuresToEnable,
                                   SPFeatureCollection features,
                                   StringBuilder log)
{
    foreach (string featureId in featuresToEnable)
    {
        try
        {
            features.Add(new Guid(featureId.Trim()));
        }
        catch(Exception ex)
        {
            //log it
            log.AppendLine("Feature activation failed for " +
                           featureId.ToString() + "\n\n" +
                           ex.Message + "\n\nStack Trace:\n" +
                           ex.StackTrace);
        }
    }
}
```

This helper function is used to enable features at the site or web level. When called, the `SPFeatureCollection` for a given site or web will be supplied, allowing this function to work at any scope level, as `SPFarm`, `SPWebApplication`, `SPSite` and `SPWeb` objects all have a `.Features` property that could be passed to this function. We will use it to turn on both the web and site level features configured for the site type being created.

Provisioning Site Security

Continue with the following function:

```
private static void ProvisionSiteSecurity(SPWeb newWeb,
                                          string ownerLogin,
                                          string Owners,
                                          string Members,
                                          string Visitors,
                                          string Admins,
                                          SPWeb requestWeb)
{
```

www.it-ebooks.info

```
        newWeb.AssociatedOwnerGroup = CreateGroup(
                                                newWeb.SiteAdministrators[0],
                                                newWeb,
                                                "Full Control",
                                                newWeb.Title + " Owners");

        newWeb.AssociatedMemberGroup = CreateGroup(
                                                newWeb.SiteAdministrators[0],
                                                newWeb,
                                                "Contribute",
                                                newWeb.Title + " Members");

        newWeb.AssociatedVisitorGroup = CreateGroup(
                                                newWeb.SiteAdministrators[0],
                                                newWeb,
                                                "Read",
                                                newWeb.Title + " Visitors");

        newWeb.AssociatedOwnerGroup.AddUser(newWeb.EnsureUser(ownerLogin));

        AssignGroupMembers(newWeb,
                           newWeb.AssociatedOwnerGroup,
                           Owners,
                           requestWeb);
        AssignGroupMembers(newWeb,
                           newWeb.AssociatedMemberGroup,
                           Members,
                           requestWeb);

        AssignGroupMembers(newWeb,
                           newWeb.AssociatedVisitorGroup,
                           Visitors,
                           requestWeb);

        SPFieldUserValueCollection admins =
                           new SPFieldUserValueCollection(requestWeb, Admins);

        foreach (SPFieldUserValue admin in admins)
        {
            newWeb.SiteAdministrators.Add(admin.User.LoginName,
                                          admin.User.Email,
                                          admin.User.Name,
                                          "Default Administrator");
        }
        newWeb.Update();

    }
```

When creating new site collections via the SharePoint API, default groups are not created automatically. Furthermore, you may want to assign a number of administrators as well. To accomplish both goals, the site creation functionality will provision security groups and site administrators using the code above. Alter this function to provide for additional default security groups or administrative settings. The code is structured to be able to create and assign groups as easily as possible.

www.it-ebooks.info

Note the use of the `CreateGroup()` and `AssignGroupMembers()` helper functions, defined in the following code sections. Continue with the following function:

```
private static void AssignGroupMembers(SPWeb newWeb,
                                       SPGroup group,
                                       string members,
                                       SPWeb requestWeb)
{
    SPFieldUserValueCollection users =
                    new SPFieldUserValueCollection(requestWeb, members);

    foreach (SPFieldUserValue user in users)
    {
        group.AddUser(newWeb.EnsureUser(user.LookupValue));

    }

    group.Update();

}
```

Called when provisioning site security, this assigns members to that group. Continue with the following function:

```
private static SPGroup CreateGroup(SPUser Owner,
                                   SPWeb newWeb,
                                   string sPermissionLevel,
                                   string sGroupName)
{

    newWeb.SiteGroups.Add(sGroupName, Owner, null, "");
    SPGroup newMembers = newWeb.SiteGroups[sGroupName];


    SPRoleAssignment assignment = new SPRoleAssignment(newMembers);
    assignment.RoleDefinitionBindings.Add(
                                newWeb.RoleDefinitions[sPermissionLevel]
                                );

    newWeb.RoleAssignments.Add(assignment);
    return newMembers;
}
```

Called by the preceding security provisioning code, this function creates a group with the provided name, owner, and permissions, on a particular web.

Logging Site Creation

Finally, add the following Logging helper functions to the file:

```
private static void WriteLog(SPListItem item, string Status, string Log)
{
    using (SPSite site = new SPSite(item.ParentList.ParentWeb.Site.ID))
    {
```

www.it-ebooks.info

```
            site.RootWeb.AllowUnsafeUpdates = true;
            SPList requestList = site.RootWeb.Lists["New Site Requests"];
            SPListItem newItem = requestList.GetItemById(item.ID);
            newItem["ProvisioningStatus"] = Status;
            newItem["ProvisioningLog"] = Log;
            newItem.SystemUpdate();
        }

    }

    private static void LogEvent(SPListItem item,
                                 string Status,
                                 StringBuilder Log)
    {
        Log.AppendLine(Status);
        WriteLog(item, Status, Log.ToString());

    }
```

These functions will make logging site creation details easy from our main provisioning code. Data is logged to fields within the original site request list item. Writing status and log data to the site request item while the site is being created allows you to monitor the site creation process by viewing the associated request item.

Provisioning the Site

Now that you have defined some helper functions to be used by the site provisioning process to assign security, enable features, and write log data, you are ready to implement the `ProvisionSite()` function. Use the code in Listing 9-7 to do so.



Available for download on Wrox.com

LISTING 9-7: Provisioning the Site

```
public static void ProvisionSite(SPListItem siteRequest)
{
    SPSecurity.RunWithElevatedPrivileges(delegate()
    {

    bool success = false;
    StringBuilder provisioningLog = new StringBuilder();
    string siteURL = null;

    LogEvent(siteRequest, "Initializing", provisioningLog);

    //get parent web and configuration item
    SPWeb web = siteRequest.ParentList.ParentWeb;
    SPList configList = web.Lists["Site Type Configuration"];
    int siteTypeId = (int)siteRequest["SiteType"];
    SPListItem configItem = configList.GetItemById(siteTypeId);

    try
    {
        LogEvent(siteRequest,
```

www.it-ebooks.info

```
            "Finding Web Application",
            provisioningLog);

//build URL
siteURL =
      configItem["URLPrefix"].ToString() + siteRequest["SiteURL"];

//Load webapplication and site owner details
string hostURI =
      new Uri(siteURL).GetLeftPart(UriPartial.Authority);
SPWebApplication webApp =
      SPWebApplication.Lookup(new Uri(hostURI));

LogEvent(siteRequest,
            "Loading Owner and Features",
            provisioningLog);

//load site owner
string siteOwner = siteRequest["SiteOwner"].ToString();
SPUser owner = new SPFieldUserValue(web, siteOwner).User;

//load features to enable
string featureString;
featureString = ((string)(configItem["WebFeatures"] ?? ""));
string[] webFeatures = featureString.Split( new char[] { '\n' },
                        StringSplitOptions.RemoveEmptyEntries);

featureString = ((string)(configItem["SiteFeatures"] ?? ""));
string[] siteFeatures = featureString.Split(new char[] { '\n' },
                         StringSplitOptions.RemoveEmptyEntries);

LogEvent(siteRequest, "Creating Site", provisioningLog);

using (SPSite site = webApp.Sites.Add(
                         siteURL,
                         configItem.Title,
                         siteRequest["Description"] as String,
                         1033,
                         configItem["SiteTemplate"] as String,
                         owner.LoginName,
                         owner.Name,
                         owner.Email))
{

    LogEvent(siteRequest,
              "Provisioning Security",
              provisioningLog);

    //configure groups
    ProvisionSiteSecurity(
           site.RootWeb,
           owner.LoginName,
           (configItem["DefaultOwners"] ?? "").ToString(),
           (configItem["DefaultMembers"] ?? "").ToString(),
```

continues

www.it-ebooks.info

LISTING 9-7 *(continued)*

```
                (configItem["DefaultViewers"] ?? "").ToString(),
                (configItem["DefaultAdministrators"] ?? "").ToString(),
                web);

            LogEvent(siteRequest, "Activating Features", provisioningLog);

            //enable features
            EnableFeatures(webFeatures,
                           site.RootWeb.Features,
                           provisioningLog);

            EnableFeatures(siteFeatures,
                           site.Features,
                           provisioningLog);


            //update log
            LogEvent(siteRequest,
                     "Creating Item in Created Sites List",
                     provisioningLog);

            RegisterSiteInSitesList(siteRequest, siteURL, configItem);

            //update item status
            success = true;        }
    }
    catch (System.Exception ex)
    {
        success = false;
        provisioningLog.AppendLine(
            "Error provisioning site collection: " +
            ex.Message + "\n\nStack Trace:\n" + ex.StackTrace);

        LogEvent(siteRequest, "Site Creation Failed", provisioningLog);
    }
    finally
    {
        using (SPSite site = new SPSite(web.Site.ID))
        {
            LogEvent(siteRequest,
                     "Updating Request Status",
                     provisioningLog);

            int ID = siteRequest.ID;
            SPList list = site.RootWeb.Lists["New Site Requests"];
            SPListItem item = list.GetItemById(ID);
            item["Status"] = success ?
                             "Site Created" :
                             "Site Creation Failed";
            item.Update();
            LogEvent(siteRequest,
                     (string)item["Status"],
                     provisioningLog);
```

www.it-ebooks.info

```
            }


        }
        });
    }
```

Wrox.SiteProvisioning\Layouts\Wrox.SiteProvisioning\SiteRequests.aspx.cs

Finally, you've reached the heart of the solution. The preceding code provisions a new site collection given a site request list item object. Although the function appears to be long, much of the length is due to logging and status updates. Step by step, this function performs the following:

- Retrieves the configuration list item for the given site type
- Builds the new site's full URL
- Locates the web application where that site will be created
- Determines the site owner and required features to activate
- Creates the site collection
- Provisions security groups



The step to provision security groups is necessary because site collections created via the object model do not have default security groups created automatically.

- Enables required site and web scoped features
- Creates a new list item in the Created Sites list, used to keep tracks of successfully provisioned sites
- Updates the status of the site request list item

Calling the Provisioning Code

Depending on the security configuration of your farm, this code may be run from a number of places. You could invoke this functionality from within the custom form itself (upon approval or save if approval is not required), from an event handler, from a timer job, from a web service, from a workflow, or even from a custom service application. In this case, you call this code directly from your form code, but in asynchronous fashion, allowing the form to be closed while site creation is occurring. To wire in this functionality, complete the following steps:

1. In Solution Explorer, expand the `SiteRequests.aspx`, and double-click `SiteRequests.aspx.cs` to open the class file for your custom page.
2. Add the following function to the class:

```
private void ProvisionAsync()
{
```

www.it-ebooks.info

```
        //provision
        if (item["Status"].ToString() == "Waiting on Site Creation")
        {
            System.Threading.ThreadPool.QueueUserWorkItem(
                o => Utility.ProvisionSite((SPListItem)o),
                item
                );
        }
    }
```

This function makes use of the thread pool to queue a work item that can provision the site. By asynchronously calling the `ProvisionSite()` method, you can avoid making the user wait while the potentially long-running operation executes. To enable this functionality within your form, perform the following steps:

1. Update both the `btnSave_Click` and `btnApprove_Click` event handlers in your form code by adding a call to `ProvisionAsync()` immediately before the `CloseForm()` function is called:

```
item.Update();
ProvisionAsync(); //<-- Insert function call here
CloseForm();
```

   At this point the solution code is complete.

2. Save, build, and deploy your code to test the site creation functionality.

Deploying the Solution

Once you have integrated the provisioning code, the code for your solution is ready to be deployed to a server for installation, configuration, and actual use. You may deploy the solution as many times as you wish, which is important as you may wish to alter the code for the solution in iterations. Unlike deploying to your development environment, where Visual Studio largely takes care of the deployment details for you, there are a few steps to take when deploying to a nondevelopment environment, such as a test, staging, or production environment.

Create a Deployment Package

The first step in deploying your solution is to create a package that yourself or an administrator might use to deploy the solution. In this case, the package will include the WSP (SharePoint Solution Package) created by Visual Studio, as well as some PowerShell scripts to configure the solution. Depending on the configuration of the environments where the solution is to be deployed, these scripts may need to be altered for each environment.

To create your deployment package, complete the following steps:

1. Set the solution configuration to "Release" and build the solution.
2. Copy the `.wsp` file from the `/bin/Release` folder of your solution into a destination deployment package directory.

www.it-ebooks.info

3. Create a PowerShell script to install the solution to the farm and deploy to the correct web application. If the web application that will be hosting the solution does not already exist, you may choose to create the web application within the script as well. Otherwise, your solution will have to be deployed after its host web application has been created. This script may resemble the following:

```
Start-SPAssignment -Global # Take care of the disposable objects to prevent
memory leak.
Add-SPSolution "SiteProvisioning.wsp"
New-SPWebApplication -Name "Intranet" -Port 80 -HostHeader intranet.mycompany.com
  -URL "http://intranet.mycompany.com" -ApplicationPool "AppPool"
  -ApplicationPoolAccount (Get-SPManagedAccount "DOMAIN\serviceaccount")
Install-SPSolution  -Identity "SiteProvisioning" -GACDeployment
  -WebApplication "Intranet"
```

4. Create a PowerShell script to create the site collection that will host the solution, and enable the `SiteProvisioning.Site` feature. This script may resemble the following:

```
Start-SPAssignment -Global # This cmdlet takes care of the disposable
objects
New-SPSite http://intranet.mycompany.com/NewSite Requests
  -OwnerAlias "DOMAIN\SiteOwner" -Language 1033
  -Name "New Site Requests" -Template "STS#0"
Enable-SPFeature -Identity "SiteProvisioning.Site"
  -URL http://intranet.mycompany.com/NewSite Requests
```

5. Finally, include a `readme.txt` file with your deployment package. This file should detail each of the installation scripts, what order to run them in, what effect installing the solution will have on the farm, and other pertinent details. Don't forget to mention how the scripts may need to be altered for different environments, including how to redeploy the solution without creating a new web application or site collection.

You may also want to include instructions for resolving errors during deployment, including how to delete stuck deployment jobs. Such instructions are outside the scope of this chapter.

Execute the Deployment Package

Once your package is ready, you can pass it off to an administrator for deployment, or deploy it yourself if you have the luxury of direct access to servers in your environment. Simply follow the instructions you created for the deployment package, executing scripts as needed. It's highly encouraged for you to test your deployment scripts in a development environment first to ensure that you haven't missed any details.

Once you have completed deployment, your code is installed on the server and you can begin to configure the solution. Once you have a functioning web application and site collection, with your site provisioning feature enabled, you are ready to begin configuring the solution for real-world use.

In the next section, you will see ways of configuring the site collection that hosts the site requests process for usability.

www.it-ebooks.info

Configuring the Solution

Although you have all the code you need to request, approve, and create new site collections, you still need to configure the solution for usability. After successfully testing your code, you should configure a number of items in the site collection that hosts your solution, such as List Views, web parts on pages, and workflows to enable email alerts to requestors, approvers, and site administrators. Although you should configure these items in your development environment, you do not need to fully build out these last items until the solution reaches production because List Views, page configurations, and (sometimes) workflows are easy to build in place but difficult to move between environments. You are encouraged to configure these items in code called from your feature receiver, to maintain consistency and avoid having to re-create these items in each environment that you deploy to, but such an exercise is outside the scope of this chapter. Whether you automate these items of create them by hand, the following information applies.

Configuring List Views

To configure List Views for your solution, first consider how the data will be accessed by users. At the very least, the default view of the New Site Requests list should display the site requestor, site owner, and request status, sorted by descending request date. You may decide to limit this view to active requests, where the status is either Waiting for Approval or Waiting on Site Creation. Additional views should be created for approvers and administrators that show the data they need to accomplish their tasks within the site provisioning process.

Configure List Views as described here:

List: New Site Requests

View Name: Failed Requests

Use: For administrators to reapprove failed site creations

Fields: Title, URL, Provisioning Status, Provisioning Log

Filter: Only display items with a status of Site Creation Failed

List: New Site Requests

View Name: Awaiting My Approval

Use: For approvers to view sites awaiting their approval

Fields: Title, URL, Requestor, Site Owner, Approver

Filter: Only display items where the Approver is [ME]

List: New Site Requests

View Name: New Requests

Use: For use on the home page of the site

Fields: Title, URL, Requestor, Site Owner, Approver

www.it-ebooks.info

Sort: Created Date (descending)

Limit: 10 Items Max

List: Created Sites

View Name: Newly Created Sites

Use: For any users who want to view recently created sites

Fields: Site URL, Site Title, Site Owner

Sort: Created Date (descending)

These views can make it easy for users to see what is happening on the site in terms of new site requests and created sites. Next, configure the home page of your site with views of these lists and instructions for users.

Configuring Web Parts on Pages

To ensure that this solution is user-friendly, you should provide a useful interface for users. By configuring List View and content editor web parts on a page within your site, you can create an effective interface with views of relevant data. Ideally this page would be the home page of the site collection that hosts the provisioning solution. If the home page is already in use for another purpose, you can create a web part page to act as the home for the site provisioning process. Be sure to provide clear links to this page somewhere in your navigation.

The solution home page should contain at least three web parts. First, add a content editor web part to display instructions for end users, along with descriptions of the various types of sites available for request. You may want to add links to pertinent List Views, as well as a direct link to your site request form labeled Request a Site.

Next, add a web part for the New Requests view of your New Site Requests list. If you choose to grant all users contributor rights to the New Site Requests list, end users can simply click Add New Item to request a new site. If you choose to disable contributor access to the New Site Requests list for users, you must provide a link to your custom edit form in the content editor web part to enable them to request sites. As mentioned earlier, your custom form elevates privileges as needed to allow any users to submit new site requests (provided they are eligible to request the type of site selected by the user).

Next, display titles, owners, and URLs for your most recently created sites by adding a web part to display your Newly Created Sites List View to the page. Users and Administrators can use this view to access newly created sites.

Finally, if you choose, you can add the Awaiting My Approval view to the home page. This makes it easy for approvers to view a list of sites pending their approval. This view is not entirely necessary because approvers still receive email notifications when their approval is required. Figure 9-5 shows an example view of the completed home page with web parts fully configured.

www.it-ebooks.info

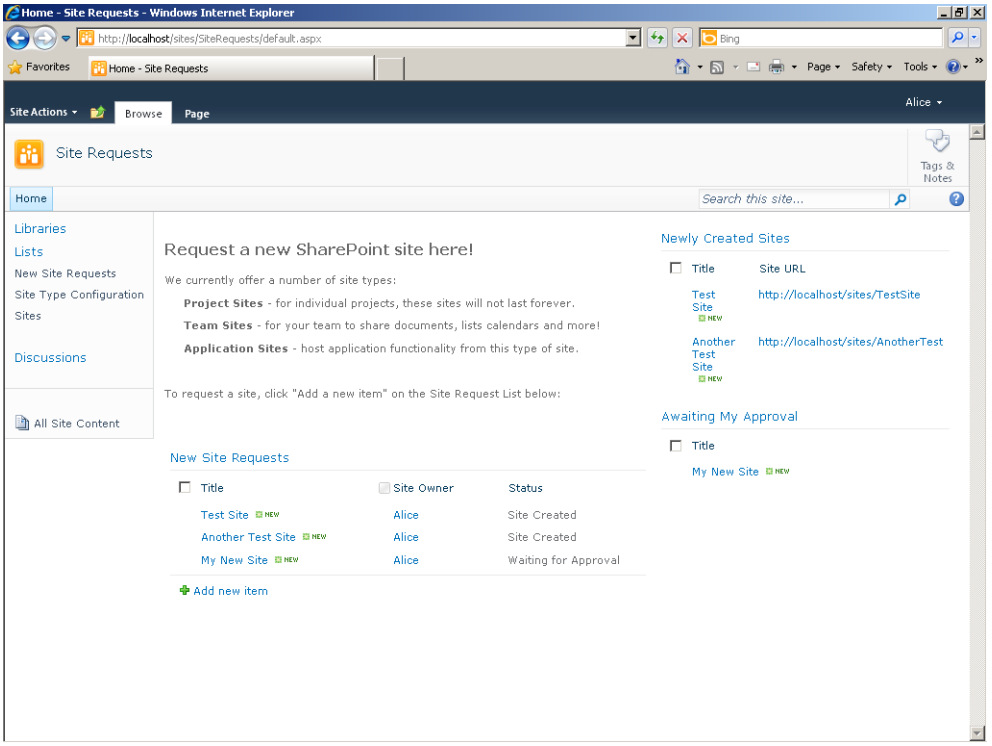


FIGURE 9-5: Home page web parts

Configuring a Notification Workflow

To enable highly configurable email messages for the site request process, you may choose to make use of SharePoint Designer (SPD) workflows.

This type of workflow can be edited quickly and easily by administrators, should they choose to alter the emails sent to end users.

This approach does have a drawback: these workflows must be reconfigured within each environment the solution is deployed to. If you are deploying to multiple environments and need to ensure that the email sending logic implemented in one environment is identical to the email sending logic in another environment, a different approach is necessary. Consider sending emails from the form code itself, using `SPUtility.SendEmail()`. You may retrieve email message text and settings from a custom email settings list; be sure to include placeholders that can be replaced by dynamic data pertaining to the site request, such as site name, approver, a link to the form to view and edit the request, and more. By populating this list upon feature activation, you can avoid having to reconfigure email messages in each environment you deploy your solution to.

In order to keep the code samples as clean as possible, and to demonstrate how SPD workflows enable solution administrators to fully control email messages sent from the system, this section will focus on configuring these messages via SPD workflows.

www.it-ebooks.info

In the case of SPD workflows, your workflows simply send emails according to item status. When a request is awaiting approval, for example, the workflow sends an email to the approver containing a message and a link to the request item. You also want the workflow to email administrators when site creation fails or encounters errors, as well as requestors when an item is rejected. Finally, both requestor and site owner should receive an email upon site creation. For a complete description of how to create workflows using SharePoint Designer, refer to Chapter 6, "Workflow." Figure 9-6 shows how to create a new workflow from a view of an existing list.

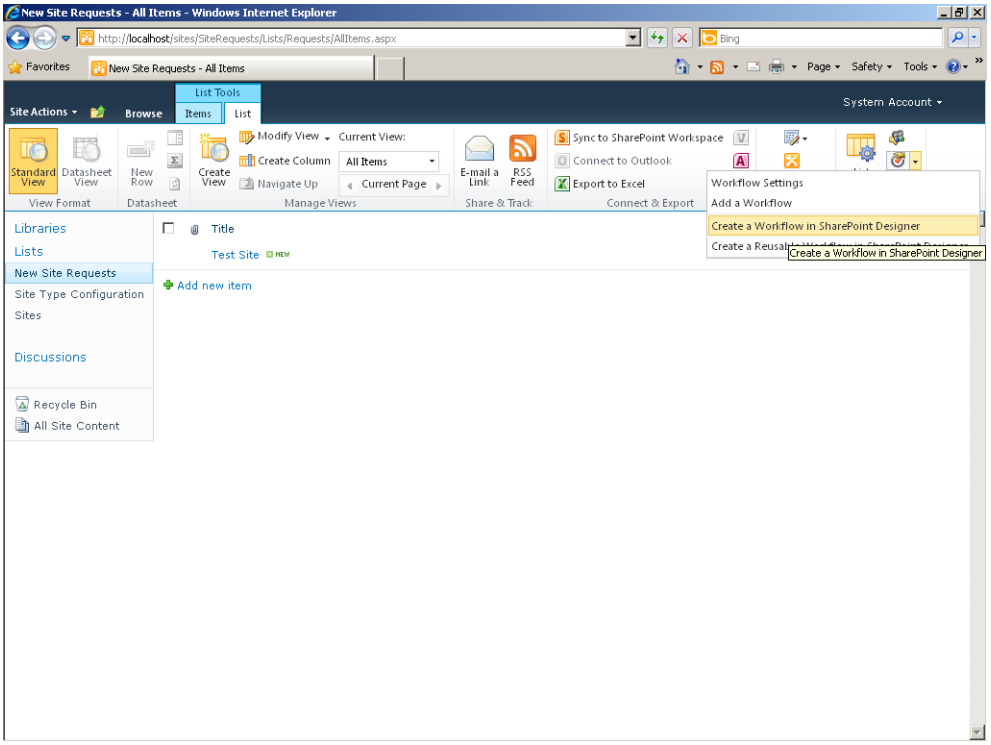


FIGURE 9-6: Creating a workflow

You need to build two workflows. The first executes whenever a New Site Request list item is created or changed, to notify approvers, submitters, and administrators of various events as needed. The other executes whenever a new Created Sites list item is created, notifying site requestors and owners that their sites have been created. Be sure to provide a useful name and description for your workflows as shown in Figure 9-7.

To build the first workflow, create a workflow on the New Sites Request list titled **Site Request Process**. Next, configure the following four conditional blocks. Each can be created by inserting an `If Current Item Field Equals Value` condition followed by a Send an Email action:

- `If Current Item:Status` equals Waiting for Approval
  - Send an email
  - To: `Current Item:Approver`

www.it-ebooks.info

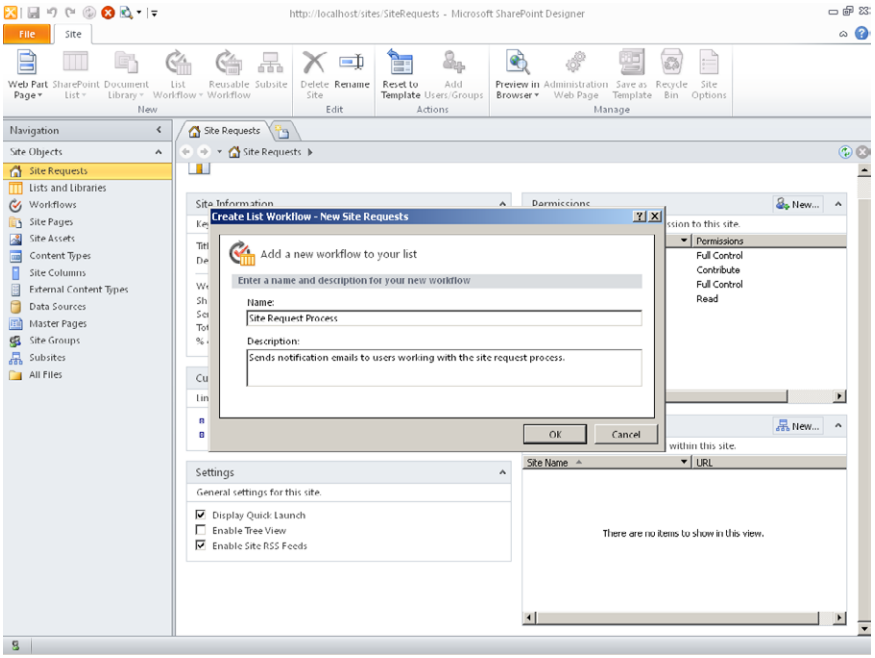


FIGURE 9-7: Naming your workflow

- Subject: Approval Required
- Body:

  Your approval is required for a new site request. Please `<a href="[%Current Item:Encoded Absolute URL%]">approve or reject the item by clicking here.<a/>`

When selecting the approver of the current item as the message recipient, be sure to select Email Address in the Return Field As drop-down list of the lookup dialog. Also, note that the workflow is checking the status of the item and sending the approver an email if the status is Waiting for Approval. Note that the `[%Current Item:Encoded Absolute URL%]` value cannot simply be typed in.

To add the item URL to the body of the message, click the Add or Change Lookup button, and select Current Item for the data source. Then select Encoded Absolute URL in the Field from Source drop-down. The remaining three blocks follow the same pattern:

- If `Current Item:Status` equals Rejected
  - Send an email
    - To: `Current Item:Created By`
    - Subject: Site Request Rejected

www.it-ebooks.info

- Body:

  Your site request for `[%Current Item:Title%]` was rejected.

  Approver Comments: `[%Current Item: Approver Comments%]`

  Please `<a href="[%Current Item:Encoded Absolute URL%]">click here if you would like to resubmit your request.<a/>`

- If `Current Item:Status` equals
  - Send an email
    - To: Site Request Administrators
    - Subject: Site Creation Failed
    - Body:

      Provisioning has failed for the following site:

      `<a href="[%Current Item:Encoded Absolute URL%]">[%Current Item:Title%]</a>`

      Please examine the provisioning log for the site request previously shown to determine why site creation failed. When the problem is resolved, edit the request item, and click Approve to restart the site creation process. Be sure to check for an invalid or already-used Site URL.

- If `Current Item:Status` equals `Site Creation Failed and Current Item:Provisioning` log contains failed
  - Send an email
    - To: Site Request Administrators
    - Subject: Site Created with Errors
    - Body:

      The following site was created with errors:

      `<a href="[%Current Item:Encoded Absolute URL%]">[%Current Item:Title%]</a>`

      Please examine the provisioning log for the site request to determine the problem and attempt to resolve.

Figure 9-8 shows what this workflow might look like when configured in SharePoint Designer 2010.

www.it-ebooks.info

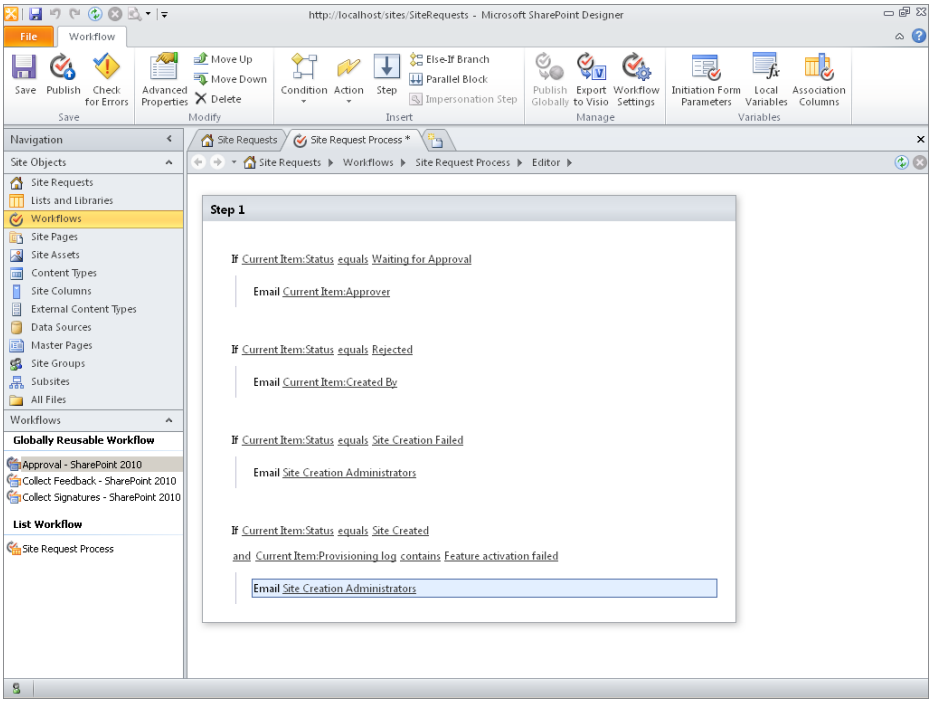


FIGURE 9-8: Example workflow

Of course, the content of the emails you configure should probably contain much more descriptive text, direct links to relevant List Views, and more. The specification shown is meant to keep the example simple. Save and publish the workflow; then test to ensure emails are sent properly and with the correct links.

Finally, you should configure a workflow on the Created Sites list that will email site requestors and owners when new sites are successfully provisioned. This workflow should execute only when new items are created in the Created Sites list.

To build the workflow, create a workflow on the New Sites Request list titled **Site Created Notification**. Simply insert a single Send an Email action in the workflow:

- Send an email
  - To: `Current Item:Approver; Current Item:Requested By`
  - Subject: Site Created
  - Body:

    The following site has been successfully provisioned:

    ```
    <a href="[%Current Item:Site URL%]">[%Current Item:Title%]</a>
    ```

    If you are the owner of this site, you may use `<a href="[%Current Item:Site URL%]/_layouts/user.aspx">configure security accounts now</a>`.

www.it-ebooks.info

You could provide links directly to site settings, a view of recently created sites, or anywhere else you might want to direct new site owners upon site creation.

NEXT STEPS

Once you have created a basic site provisioning solution, a number of steps can be taken to enhance functionality, ease-of-use, look and feel, and more. Consider extending the solution to meet the specific needs of your organization. Although the specifics of how to extend the solution are left as an exercise for the reader, this section will outline a number of areas where you should consider improving the solution.

User Experience

One of the first enhancements you should consider making is to the user experience provided by the solution. This chapter did not focus on providing clear instructions for your end users or on customizing the UI to match your organization's branding. Form and List Views were kept simple to make the examples easy to follow. Take some time to work on these items so you can ensure that your users will find working with the system intuitive and not be confused by the solution. Testing with real end users is recommended so that you can gather feedback and incorporate it into the solution.

The following items should be considered when trying to improve user experience:

- Reduce the number of clicks needed to access frequently used items. You might decide to place the new site request form on the home page of the site that hosts the solution, for example.
- Provide better messaging and feedback to users: make sure that instructions are readily available, all status and error messages are clear, and users are told what actions they should take to resolve issues.
- Provide a live view of site provisioning status. Since the statuses and log entries created during site provisioning are constantly written back to the new site request item, you could create a pop-up window that refreshes a view of that data for the item being created. Users would see the log update as the pop-up refreshes every few seconds. You could even go as far as providing a progress bar within this display.
- Ensure you have an appropriate set of site types that is well communicated to your user base. Be sure to describe these sites from the users' perspective so they are not requesting the wrong types of sites.

Site Request and Configuration

Note that in the end, the system provisions your sites according to the combination of the specific request details and the preset defaults for a given site type. You may add any number of fields to either the request itself or the site type configuration to further extend the configurability of site types and site requests.

www.it-ebooks.info

The following items are examples of this type of modification:

- Adding a site theme or custom CSS URL to the "Site Types" or "New Site Requests" lists.
- Adding a "Parent Site" drop-down to the "New Site Requests" list that references the "Created Sites" list. This would enable a hierarchy of site collections that could be displayed to users via tree-view or traditional navigation controls, resulting in effective cross-site collection navigation.
- Adding a site quota option to the "Site Types" or "New Site Requests" lists. This could be a setting per site type, changeable in individual requests by approvers, or simply something the requestor can specify. It's up to you.
- Adding an expiration date to the "Site Types" or "New Site Requests" lists. This will have to be enforced by admins or via workflow.
- Adding a primary support contact for each newly created site or, alternatively, a checkbox that indicates whether to contact the site owner vs. help desk for support. You could even automatically assign individual support agents to new sites in round-robin fashion.
- Create a custom form for the "Created Sites" list that will allow administration of certain site settings when editing items. Since the "Created Sites" list items each correspond to their own site collection somewhere on the farm, all the data you need to make API calls is available. Alternatively, you could implement event handlers that respond to changes to data in this list (such as a site quota setting), reflecting those changes within the settings of the corresponding site.

As you can see, this list could go on and on. The possibilities are up to you and your organization to work through.

Additional Uses for Site Data

Additional items configured on the request site, such as a site disposition workflow that enforces a site expiration date, can be easily added to the solution without altering code. The following list describes a number of ways to extend the use of this solution beyond the site request and configuration process:

- Navigation or site lookup controls: These controls would read from the "Created Sites" list to display lists of sites according to type, owner, or other criteria. Feel free to add your own criteria to the "Created Sites" list if you would like to customize your navigation. For example, you might mark sites as "Public" vs. "Private" to control how they appear on your navigation controls. Be sure to cache the list data being used by controls such as this, so you are not querying the database for large amounts of information every page load.
- Site archiving workflow: This workflow would ask owners of unused sites whether those sites are still needed, disposing or archiving sites as needed. This could be driven by the site type configuration or a prespecified site expiration determined by the site requestor at the time it was provisioned.

www.it-ebooks.info

- Site Owners or Site Support control: These controls would display contact and support information for the current site. Ideally they would be displayed on the home pages of created sites and would read from the "Created Sites" list to retrieve information about the current site's owners and support requirements.
- Site quota management workflow: This workflow would detect when sites are at or near their quota limits and initiate a process to raise the quota for the site once approved. Ideally administrators or site owners could initiate this workflow from within their site.

SUMMARY

This chapter covered a lot of ground, including demonstrations of list definitions and instances, a custom feature receiver, a custom `.aspx` list item form, asynchronous execution of site creation, manager lookups, AD group lookups, and more. In addition to the technical implementation details, it is my hope that the discussion of site taxonomy and site provisioning processes has been enlightening for the reader. As a developer, it's easy to focus on the technical details while forgetting to plan for the longer term, whether it's support, maintenance, or the ability to constrain a system so that it doesn't grow out of control. This chapter has taken a broader view for that reason.

Although there is a lot of code included, it is by no means complete. Code was constructed to be as simple as possible while still providing highly relevant examples that solve the core problems you encounter when building a site provisioning process. As discussed previously, additional logging, error handling, and form field validation is highly encouraged. The code presented, for example, does not validate that the requested URL is available, or even valid. (It could contain invalid characters such as & or /.) Nor does the code prevent someone else from requesting the same URL while a request is being processed. Implementing this level of robustness is left as an exercise for you.

Regardless, the system described in this chapter provides a level of configurability and control over site creation that far surpasses what is available with SharePoint out-of-the-box. With this system, your farm can adhere to a more stable, well-enforced taxonomy than it would otherwise, and new sites can be made available to end users without the need for an administrator to take action.

Furthermore, this framework can be extended to provide functionality relating to site management, whether it's a globally available navigation component, site support web part, or site quota exception process. When you see all of the capabilities enabled by a robust site provisioning system, the value becomes clear.

www.it-ebooks.info

www.it-ebooks.info

10

Building Scalable Applications in SharePoint

This chapter covers performance and scalability issues across all tiers of a SharePoint application, from high-level farm architecture and configuration to the client-side implications of design decisions. Although they are somewhat disparate topics, this chapter discusses farm configuration details, application coding techniques, client-side optimization techniques, and much in between, all in service of better performance and scalability.

Performance and scalability are two closely linked topics. At a basic level, a better performing farm or application is more scalable, as better performance allows the server to accomplish more tasks in the same amount of time, enabling it to host a larger number of users or serve larger amounts of data. For this reason, this chapter also includes information about how to make your code execute as quickly and as efficiently as possible. In addition to SharePoint-specific scalability and performance optimizations, you will learn how to tackle some general C# and custom database scalability and performance issues, since these issues can certainly come into play when developing high-performance applications within SharePoint.

Before delving into SharePoint-related performance and scalability issues, a quick rundown of general performance problems is in order. By understanding what makes a computer system slow or inefficient in general, it's easier to understand why the SharePoint-specific recommendations presented in this chapter make sense.



An example solution that demonstrates many of the optimization techniques described throughout this chapter is available for download from the book's companion website at `www.wrox.com`.

www.it-ebooks.info

GENERAL PERFORMANCE ISSUES

The CPU of a modern PC is no different in basic function from that of a decades-old PC: The CPU loads data from memory, alters data in its registers, and stores it back to memory. At a basic level, that's all the CPU does. Information to be displayed on-screen is written to a specific location in memory (known as a buffer), which is read by the video hardware and displayed on-screen. Key presses on your keyboard are written to a buffer where the keystroke data can be read and processed by the CPU. The same is true of network and disk I/O: Data is moved back and forth via a buffer in memory. Although this description is simplified, it is helpful when trying to understand why even fast processors are subject to the performance limitations of the systems it interacts with, such as main memory, disks, and the network.

Memory, Disk, and Network Access

In addition to manipulating the memory locations stored in main memory, the CPU also brings and keeps the most recently used data in local, smaller, faster caches. Local means that these memory caches reside on the same silicon wafer as the CPU. These caches are known as L1 (Level 1) and L2 (Level 2) caches, where L1 is the smallest (usually about 128KB), and closest to the CPU. The L2 cache is further away and larger (usually 2 to 8 MB on modern processors).

Because L1 is closest to the CPU, it is the fastest cache to access. Data from this cache is available to the CPU within 2 clock cycles. Data from L2 is available within about 20 clock cycles, and data from main memory typically takes somewhere between 400 and 800 cycles. The further away the memory from the CPU, the larger and slower the memory becomes. When you add the hard disk to this equation, you can see that it's even further from the CPU, much larger than main memory, and orders of magnitude slower. How much slower is the CPU compared to main memory? An example timescale is helpful here:

Suppose that for this example timescale, it takes the CPU 1 second to retrieve data from the L1 cache. Getting a piece of data from L2 cache would then take approximately 10 seconds, whereas main memory access takes approximately 3 minutes. Because hard disks are mechanical devices, getting data from a location on a physical hard disk requires that the hard disk read head be rotated into position, and the drive spun to the correct location, before data can be retrieved. At the example timescale, this can take 115 days, or approximately 4 months! Solid state disks (SSD) are much faster than traditional hard disks because they have no moving parts that must be in position before data can be read. With an SSD, at the example timescale, data can be retrieved in approximately 55 hours, a little more than 2 days. This explains why SSDs provide such a dramatic performance increase over traditional drives.

Of course, this example is the most dramatic possible. For sequential data access, often the disk head is already where it needs to be when the next write takes place. Also, modern RAID disk configurations can speed up disk access significantly by simultaneously accessing data from multiple drive spindles. This significantly increases throughput but does not help much with initial latency of data access (the cost to retrieve the next piece of nonsequential data). For these reasons, a set of SSDs in a RAID configuration can provide extremely high performance and redundancy, although at a high cost.

It's clear to see from this analysis that physical disk I/O is a performance killer. The more you can avoid reading from and writing to physical disks, the better your applications will perform. By keeping your most actively accessed data in memory, you can avoid a lot of unnecessary disk access and

www.it-ebooks.info

improve performance. This is known as application level caching, where software keeps data in variables in memory, instead of rereading the same data over and over from a file on disk.

Similar to the disk access latency issue, network calls have significant overhead compared to accessing main memory. The overhead of network hardware and protocols means that at the preceding example timescale, a network call on a local 1Gbps network can result in latency of days or weeks, much faster than a physical disk, but slower than a local SSD. Of course, the remote system being called needs to access its own data out of memory or, worse, from disk. This means that database calls can be faster or slower depending on the data requested, whether that data resides in memory, and the level of additional overhead in processing your query, such as parsing and optimization. You can use a number of database optimization techniques to improve the performance of your SharePoint farm, and they are discussed later in this chapter.

Algorithmic Efficiency

The efficiency of the algorithms used within an application has a major effect on overall performance and scalability. Knowledge of algorithms and data structures is the key to improving the efficiency of your code. The trick to most efficient algorithms is how they structure data in memory. In the following discussion, algorithmic performance is described in terms of loop iterations. Although no real unit of time is associated with a given loop iteration, the abstraction is useful because each loop iteration can be assumed to take a constant amount of time. In other words, think of algorithms in terms of how many times they have to perform a particular task over and over again without worrying about how long each task takes.

Suppose you have to find out if a particular value exists in a list containing 1 million strings. The simplest way to accomplish this is to loop through the list, comparing each string to the value you are looking for until you find a string that matches. Suppose you need to perform this task 1,000 times in a row. In this case, each lookup would require an average of 500,000 loop iterations before the title is found, resulting in 500 million iterations to complete the operation.

Alternatively, you could build a hash table from the list of items you are searching. A hash table is a data structure that enables fast insertion and retrieval of data by a key value (in this case, the title is your key). Hash tables are clever. To understand how they work, you need to understand what a hash function is. A hash function is a function that converts a value to a number within a given range. The hash function must always convert a given value to the same number, given the same range. A hash table makes use of this function to find the location of a given value within an array of items.

Sometimes, passing two different values to the same hash function can return the same result. This is known as a hash collision and is dealt with a number of ways, such as by just picking the next open slot in the array to store or retrieve the value. When searching for an item into a hash table, the code must check the values in the hash table's internal array, starting from the numerical address returned by the hash function, then checking the next item in the array, and so on until the correct item is found. Other collision resolution schemes exist, such as chain bucket hashing and quadratic hashing, which may be of interest to you.

When you create a hash table, you may provide the expected number of items that the table can hold. This enables the hash table to allocate an array of sufficient size to ensure that there are not

www.it-ebooks.info

too many hash collisions. In this way, a hash table enables you to insert and retrieve items by a key value in a short period of time, eliminating the need to loop through every item in the list.

For the previous example, in which a list of strings is searched, using a hash table would first require inserting 1 million strings into the hash table's internal array. Assuming insertions take an average of 20 loop iterations, it can take 20 million loop iterations to create the hash table. To perform the rest of the algorithm, the computer must simply perform 1,000 looped operations that perform a key-based lookup on the hash table. Assuming each key-based lookup requires an average of 35 loop iterations to run the hash function and compare values, the result is a total of 20,035,000 loop iterations, a far cry from the 500,000,000 you ended with in the previous example. This is why you need to understand algorithms and data structures, and to choose the correct ones when coding your functionality.

LOAD BALANCING AND FARM CONFIGURATION

The first step to ensure you have a SharePoint farm that can scale to support a large number of users is to plan your overall farm architecture for scalability. This section discusses the high-level farm configuration options that you encounter when designing a farm for high performance, scalability, robustness, and uptime. Your architecture must make use of redundancy and the appropriate amount of processing capacity to deliver these qualities: Redundancy where possible to provide for load-balanced and failover-capable services, and capacity where needed to scale those services to support the expected load. When talking about scaling capacity, two general options exist: scaling up and scaling out. *Scaling up* means provisioning more powerful servers, whereas *scaling out* means provisioning more servers into your farm overall. You should scale up *and* out to provide a suitable farm architecture. This section discusses the trade-offs between scaling up and out, and when each strategy might be appropriate.

Server Configuration Options

Within a SharePoint farm there are typically three types of servers:

- **Web frontend:** Respond to web requests from end users
- **Application:** Backend processing, search indexing, and query serving
- **Database:** Respond to database queries from other servers

Each of the servers in your farm performs one of these roles. The type of processing that a given server performs relates directly to its role within the farm and the services configured to run on it.

Database servers, for example, spend much of their time filtering, sorting, and retrieving data within the database, whereas web frontend servers spend much of their time building and transforming strings to be output as HTML and sent over the network to a client machine's web browser. Understanding the type of work these servers perform enables you to better spec out your farm and makes the most of your hardware resources.

For example, because the database server is responsible for serving data to multiple web frontends, fast disks, excellent network connectivity, plenty of memory, and multiple processor cores are needed for the best performance. Web frontends, on the other hand, can do a lot of caching and string processing, so multiple processor cores and plenty of memory is important, but disk speed is not as critical as with the database servers.

www.it-ebooks.info

Application servers in SharePoint typically perform work not directly related to serving web pages to client machines, such as search indexing, document conversions, certain backend timer jobs, and more. The number of application servers needed for your farm is directly related to the use and configuration of a number of backend services. If you farm does not make use of many backend services such as search or document conversions, you may not even need to provision a separate application server role. On the other hand, if you build a heavily used collaboration platform with frequent search crawls and lots of document conversions, you may want to provision multiple application servers. SharePoint makes use of an internal load-balancer service, which distributes backend job processing between the available application servers.

Figure 10-1 shows a typical farm architecture diagram, showing two web frontends, an application server, and clustered database servers. This configuration is the minimum required to provide full redundancy for your farm. If one web frontend server fails, the other is there to handle incoming requests. If one of the database servers fails, the other database server takes over. If the application server fails, the application server's roles can temporarily failover to one or both of the web frontend servers.

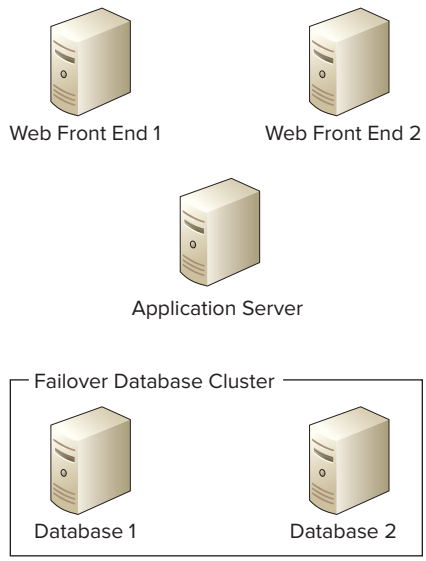


FIGURE 10-1: Farm architecture diagram

You need to understand how load is balanced between servers of different types in a scenario such as this. The web frontend servers can be configured in two ways, either for load-balanced operation or failover capability. In a failover configuration, only one web frontend would respond to requests at a time. If the currently active web frontend fails, requests are routed to the other web frontend. In a load-balanced configuration, both web frontend servers handle different incoming requests at the same time. Unlike the failover configuration, your capacity and performance is affected if one of your web frontend servers fails.

From a licensing perspective, especially with publicly facing Internet sites, it is generally more cost-effective to scale up your web frontend servers before you scale them out. In other words, fewer (at least two), more powerful web frontends should be utilized instead of more, less powerful web frontend servers. This reduces your server administration costs as well as total server licensing costs because per-server licenses can cost tens of thousands of dollars. Of course, if your farm requires additional web frontends in the future, they can always be added to the existing farm, but in general, it's best to start with fewer servers and add frontend and application servers as needed.



When discussing ways to increase computing power, scaling up *means increasing the power of individual servers, while* scaling out *means adding additional servers to your infrastructure.*

www.it-ebooks.info

All too often SharePoint farms are overbuilt, resulting in unnecessary licensing and operating costs. Two powerful web frontends can easily host a few thousand concurrent users in a typical collaboration environment, and potentially many more users in a properly optimized publishing environment.

Load-Balancing Options

Load-balancing your web frontends is possible because the HTTP protocol is stateless, meaning that information is generally not persisted on the server between postbacks. Consequently, a sequence of requests can be handled by multiple, different web frontend servers available within your SharePoint farm. Each web frontend shares the same state and data as the other web frontends, with the exception that different data may be cached in memory depending on what distinct pieces of data each frontend has previously retrieved. This picture gets a bit fuzzier when session state is turned on, which requires either sticky sessions or centralized storage of session data. Regardless, any data that can be altered through the web frontend servers must reside within the databases, each of which acts as the single point of storage for the data contained within. This means that for a single database, there is no possibility of load balancing. Only failover clusters are possible.

Also, though application servers are internally load-balanced, SharePoint manages that automatically and there is little for you to configure. Consequently, this section specifically discusses load-balancing web frontend servers.

A number of options are available for load-balancing your web frontend servers:

- Hardware load balancing
- Windows network load balancing (aka WNLB)
- Round robin DNS load balancing

Hardware load balancing makes use of a dedicated piece of hardware (or dedicated server with appropriate software) to accept incoming requests from clients, routing each to a single web frontend server. This requires no additional work from your web frontend servers because they simply serve incoming requests from clients with no knowledge of how those requests were routed. Hardware load balancers attempt to balance loads using a number of schemes, such as round robin routing, random routing, load-based routing, and sticky sessions. Round robin load balancing simply routes each subsequent request to the next server in the list, allocating requests between all servers equally, without regard to the client making the request. Random routing simply routes each request to a randomly selected server, regardless of where the last request was routed to. Load-based routing attempts to route each request to the least loaded server at the time of the request. Sticky session load balancing attempts to always send the same clients' requests back to the same server, enabling the use of state-based sessions within your applications. Sticky sessions are not necessary with SharePoint out-of-the-box because SharePoint comes configured out-of-the-box with sessions disabled for performance reasons. When session state is enabled within SharePoint, a backend database is used to store session state, making sticky sessions unnecessary even when making use of session state.

According to the type of load balancer used, different options are available in terms of the routing methods available and the heuristics used to route requests. Some balancers may directly monitor your servers using operating-system-provided interfaces such as Windows management instrumentation (WMI), whereas others may simply monitor the lengths of incoming requests and existing connections

www.it-ebooks.info

to determine existing server loads. You need to understand the options available when selecting or configuring your load balancer so that you can make the most of your budget, ensure farm performance, and reduce administrative costs.

In addition to balancing requests between servers, a hardware load balancer is responsible for managing server failover. Hardware load balancers generally check each web frontend server in the farm by sending a test HTTP request to a preconfigured page on a regular basis to ensure that each server is still serving requests. If a server doesn't respond to the load balancer's test request, the load balancer stops routing requests to that particular server until the server comes back online. Hardware load balancers provide the best and most common option for distributing load and managing failover across multiple web frontend servers because they are easy to configure, provide a single point of entry to your farm, and can handle high loads without invoking the overhead of WNLB, which is discussed next.

Windows Network Load Balancing (WNLB or just NLB) is a function provided by Windows servers that enables multiple servers to load-balance between themselves. It does not require a dedicated piece of hardware to sit in front of the farm to route requests but does generate overhead as your web frontends chatter with each other to determine which servers should handle which requests. With WNLB, each load-balanced server has a frontend and backend IP address. The servers are configured to share the same frontend IP address, so every server in the rotation receives each request made to the farm. When a request comes in, the servers talk to each other over their backend IP addresses to determine which server will respond to that request. Typically, servers are configured with multiple NICs for performance reasons, so the backend traffic does not interfere with the frontend.

Although not the best solution from a performance perspective, WNLB provides load-balancing capabilities for much lower cost than a dedicated hardware load balancer. WNLB operates at the network level, and although it monitors servers for general availability (are they still running at all), WNLB has no way to know if IIS or SharePoint has failed at an application level, meaning that true automatic failover is not possible without some additional management software or custom scripts.

Finally, DNS load balancing (aka round robin) is the most basic but least powerful way to direct different users to different servers on your farm. By configuring DNS entries for your farm to contain multiple IP addresses (one for each web frontend in your farm), client computers connecting to your farm can randomly choose one of the IP addresses to use, routing themselves to a single server.

This does not guarantee an even load between servers; certain versions of Windows XP do not truly randomly choose the IP address, resulting in a skewed load where the first IP address listed receives most of the traffic. Some DNS server implementations have mitigated this by returning lists of IP addresses in different orders each time they are requested.

Another issue with DNS load balancing is that when a server fails, there is no automatic failover mechanism. The IP address for a failed server must be moved to one of the remaining functional servers, or entirely taken out of the DNS entry, which can take up to the Time-to-Live (TTL) of the DNS entry to propagate to client machines. For this reason, when using DNS load balancing, the TTLs of your DNS entries should be kept to short timeframes (on the order of minutes). This results in a higher load on your DNS servers because the DNS records are more frequently requested by downstream DNS servers that can't cache the DNS records for long due to the short TTL.

DNS load balancing may also be used to balance between entire server farms, each of which makes use of their own hardware load balancer. You can see this for yourself by looking up the DNS

www.it-ebooks.info

entries for http://google.com. Run `nslookup google.com` from the command prompt to view a number of IP addresses that may be used to access Google, each of which routes to a different farm.

Closely related to DNS load balancing, geographic DNS services rely on DNS servers that serve location-specific IP addresses to client machines. By using a geographically aware DNS infrastructure, you can route end users to local farms by providing them with the geographically closest IP address when responding to DNS requests for your domain. This is not a foolproof scheme because determining the geographic location of an IP address depends on imperfect methods and cannot be guaranteed.

Because all web frontends serving the same content must connect to the same databases, there is a limit to the number of web frontends that can be load-balanced before the database becomes the bottleneck (more on this later). This depends on the usage scenario, but performance typically does not improve beyond 8 to 12 web frontend servers. This results in a perverse rule about load balancing: The more I/O an application requires to and from a single database, the fewer web frontends are needed because the database becomes a choke point that additional web frontends cannot alleviate.

Database Allocation

How can you ensure that your database doesn't become a choke point for your application? First, you can ensure that you have high-performance hardware on the servers running your databases. The more disk spindles (or better, SSDs) in your storage array, the more redundancy the database server can exploit at the hardware level to perform disk I/O. The more CPU cores, the more simultaneous requests can be served (as long as disk I/O isn't holding things back). The more memory available, the more data can be cached, reducing the need for disk I/O. Second, you can make use of the fact that SharePoint relies on multiple databases, which can then be served up by different database servers. By spreading the database load between a number of failover database clusters, you can achieve higher performance and scalability.

In practice, this means splitting your data across multiple site collections, which may each reside in their own databases. An understanding of the usage patterns between different types of sites and how that impacts database usage is helpful here. Publishing sites that are authored elsewhere and make of use content deployment jobs to receive content, for example, may make heavy use of caching and only occasionally update data in the database, meaning actual disk I/O can be kept low. In this case, adding more memory for caching can provide a larger performance boost compared with adding faster disks. Collaboration sites, on the other hand, may see frequent updates of large documents in the megabyte range, meaning that disk I/O could be heavy, and database maintenance more frequent. Faster disks can be helpful in this type of situation.

If you have a mix of publishing and collaboration sites on your farm, you generally would not put all the collaboration sites on one DB cluster and the publishing sites on another because the result will be an unbalanced load between the two database clusters. On the other hand, this scenario might make sense if the DB cluster used to host the publishing sites is less powerful, or if a lot of content editing and workflow happens on those sites that would increase disk I/O.

Figure 10-2 represents a server farm that has been built out to support heavy capacity and full redundancy for all tiers of the system. By placing different databases on different physical systems, you can scale out the backend database across many clusters. In this case, the farm is serving all system and backend databases, such as config, search, user profiles, state service (used by InfoPath

www.it-ebooks.info

among others), and other from a single cluster, while two content databases are each served from their own failover SQL cluster. In this configuration, search crawls, search queries, user profile crawls, and similar backend processing does not affect the performance of the content databases. Furthermore, access to one content database cannot impact access to the other because they are on different systems. This setup helps to alleviate the potential for database resources becoming a bottleneck for your farm. You have also made use of four web frontend servers and two application servers in this configuration. By keeping web frontend server #4 and application server #2 on standby, the farm cannot lose any processing capacity when an application or a web frontend server fails because the standby servers are available to pick up the load.

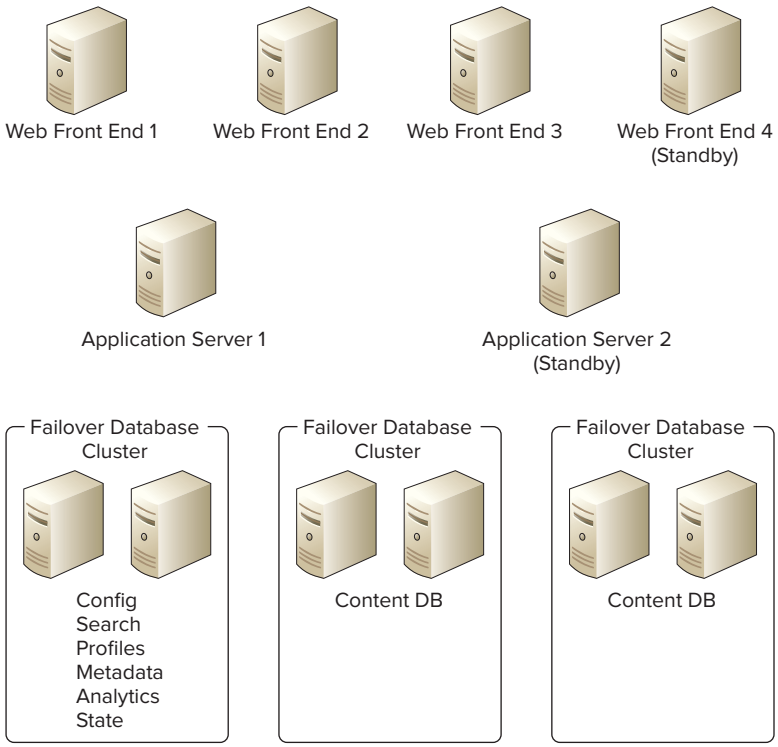


FIGURE 10-2: Extended farm architecture diagram

Data Sharding

Finally, if you have a custom database used by your own custom application, you can make use of data "sharding" to split your data across many databases. With data sharding, entities in your application are stored in a particular database according to some fixed identifiers on those entities. For example, if your system stores customer profiles, you may choose to store those profiles on different databases according to the first letter of the customer's last name. Two different database servers, each storing the data for customers A–L and M–Z, respectively, can provide twice the throughput capacity when compared to a single database. In this case, anytime a customer or customer service

www.it-ebooks.info

agent needs to access a customer's profile, the system would issue a query to the appropriate server depending on the customer's last name. The system would also need to handle the special case when a customer's last name changes because the record may need to be deleted from one database and inserted into another. For this reason, it is best to choose an identifier that cannot be changed, such as the customer's account number, to use as the basis for sharding your data.

Of course, many limitations come into play when sharding your data across databases, such as the inability to report on or query all data in a single database call, or the need to synchronize supporting entities (such as drop-down list options) between databases. You must carefully consider your use cases when designing a data sharding scheme, and ensure you can support them with the architecture you choose.

What can you do if you need to support reporting on all data at once, but need to split your data across databases for performance reasons in other use cases? Consider making use of a secondary reporting database kept on a separate system to support reporting across all data (such as aggregating all custom profile data). An extract, translate, and load (ETL) process can then be used to copy (and denormalize if necessary for reporting speed) your sharded data into the reporting database on a regular basis. Although this secondary database does not have the throughput of the sharded databases, as long as it will not be used in the same way or by the same number of users, this approach can provide reporting capabilities that don't impact the performance of the system of record (where the single version of the truth is stored).

SHAREPOINT CACHING OPTIONS

As shown previously in this chapter, disk I/O is expensive, and too much of it can kill performance. One of the best ways to avoid unnecessary disk I/O is to cache as much data as possible in memory, where it can be retrieved without disk I/O when needed. Fortunately, SharePoint provides a number of built-in caching mechanisms that can be enabled to improve performance. The only way to ensure a successful caching strategy is implemented is to create a baseline, apply your caching techniques and then come back and measure the impact. There is no one-size-fits-all solution here. Every implementation calls for its own caching requirements. In this section you learn about the various options, when they apply, and how to configure them.

Page Output Caching

Page Output Caching is functionality provided by SharePoint that enables publishing sites to cache publishing pages' rendered HTML output. Publishing sites in SharePoint are sites that have the publishing feature turned on and make use of publishing pages to deliver content. Enabling page output caching reduces not only database I/O, but also page processing by the web frontend server because the page HTML does not need to be regenerated each time a user requests a page.

Page output caching makes use of profiles that are configured at the site collection level, but may be applied to pages at the site, or page layout level, enabling you to control cache settings for entire sites or specific sections of sites as needed. When configured at the page layout level, for example, all pages that make use of a particular page layout can utilize the output caching options specified in

www.it-ebooks.info

the caching profile assigned to that page layout. This is useful for sites that contain some static pages alongside pages with dynamic content. If a page with dynamic content displays information specific to the current user, for example, output caching should be turned off for that page because cached content for one user could be served to another user by mistake. Also, if your site contains pages with frequently updated content, you may want to configure a cache profile for those pages with a short duration or **Check for Changes** enabled so that users don't see stale data server from the cache. At each level, you can specify a cache profile to use for both authenticated and anonymous access, which is useful because those scenarios require vastly different caching behaviors. You may also enable the **Perform ACL Check** setting to ensure that users see only content they are authorized to see. This is not necessary when all site content is available to anonymous users, as there are no resources that need to be secured.

You may want to disable page output caching when making use of many fine-grained permissions (ACLs). Although when you create a Cache profile at the site collection level you have the option to check **Perform ACL Check**, depending on the number of ACLs this may impose more overhead on the farm than the time required to query the backend to retrieve the content. Output cache profiles provide you with fine-grained control over how the output cache operates. Before looking at specific cache profile settings, it's helpful to understand how the output cache determines whether a page can be retrieved from the cache or needs to be rendered from scratch. Each request to a page that uses the output cache can result in a cache key being generated. The cache key is based on the request details and the cache profile used. For example, if a request is made to `default.aspx`, and the profile is set to cache by user rights and browser version, the cache key might look like `IE9;Reader;default.aspx` for a user with IE9 and Reader rights.

SharePoint then checks the cache for page output based on the key generated and returns that output to the client. If the key is not found in the cache, SharePoint renders the page and stores the output in the cache using that key. Cache profiles contain a few Vary By settings that provide you with full control over which values are included in the cache key.

You can access cache profiles by visiting the site settings page on a publishing site and selecting the Site Collection Cache Profiles link under Site Collection Administration. The following list briefly describes each configuration option available in a cache profile:

- **Title:** Simply a title used to identify the profile.
- **Display Name:** Similar to title, just used when displaying profile in a list.
- **Display Description:** Used to provide a description of the profile.
- **Perform ACL Check:** When set to "yes," the system checks security settings to determine if the current user has rights to view a page before returning page output from the cache. Because security checks typically happen during page execution, this step is necessary when serving pages out of the cache to authenticated users. Disable this check on anonymous sites to improve performance.
- **Enabled:** Whether the cache will be enabled at all.
- **Duration:** Amount of time in seconds that items can live in the cache before becoming invalidated.

www.it-ebooks.info

- **Check for Changes:** When set to "yes," upon each page request the system checks for any changes on the site since the page was cached. This is because SharePoint has no way to know what data on the site may be included in the page output, so any site changes invalidate items in the cache when this is enabled. If this setting is disabled, items remain in the cache until the duration time limit is met.
- **Vary by Custom Parameter:** This setting enables you to set any parameters you want to be included in the key used to cache the page. You must provide a custom handler for any custom parameters that you specify that can convert the parameter name into the parameter value. The `Browser` parameter is built in and supported by default in SharePoint. If `Browser` is included here, pages will be cached according to the client browser used to request them.
- **Vary by HTTP Header:** This setting enables you to include HTTP header values in the key used to cache the page. The most common value here is the `Accept-language` header, when you have page output that varies according to your end users' language settings.
- **Vary by Query String Parameters:** Similar to the other Vary By options, this setting enables you to include query string variables in the key used to cache the page.
- **Vary by User Rights:** Not to be confused with the Perform ACL Check option, this option specifies whether the system should cache separate versions of the same page according to user rights. In other words, this includes the current users' rights in the cache key used to store and retrieve cached versions of page output.
- **Cacheability:** Specifies the headers sent in responses served from the BLOB cache. These headers indicate whether and how page output is cached by downstream caches or HTTP proxies.
- **Safe for Authenticated Users:** This setting has no effect on the output cache itself but simply marks a given profile as safe for use in authenticated scenarios. This results in the profile being available in the drop-down list for cache profiles that can be applied to authenticated pages.
- **Allow Writers to View Cached Content:** Enabling this setting tells the output cache to server cached pages to users with page edit rights. The danger here is that checked-out or unpublished content could be saved to the cache. In general, this setting should stay disabled to avoid caching content that isn't checked in or fully published.

Each copy of page output that is cached consumes memory according to the following formula: `Memory Required in KB = 2 × [Size of page output] + 32`. This means that if a page's output is 50KB in size, it consumes up to 132K in memory for a single copy. Assuming pages are cached according to browser and user rights, and assuming there are three different browsers and three different sets of user rights in use among your user base, you could have up to nine versions of the same page cached in memory. So in this scenario, a single page could consume 9 × 132KB or over 1MB of memory when fully cached. Keep this in mind when determining how your output cache keys "vary by" certain parameters. The more `Vary By` parameters used, the more potential memory will be required for the output cache.

Cache profiles may be enabled and applied to all pages in a site collection by visiting the site settings page and clicking Site Collection Output Cache under Site Collection Administration. To configure profiles for a single site, visit the site settings page for that site, and under Site Administration, click Site Output Cache. To specify profiles to be used for a particular page layout, visit the Master Pages and Page Layouts gallery for your site (accessible through site settings under Galleries), and locate

www.it-ebooks.info

the page layout in question. Each page layout includes properties that specify the cache profile to be used for both anonymous and authenticated access. Alter those properties as needed, and complete your changes by checking in, publishing, and approving the page layout.

One way to get around the fact that you cannot serve personalized content to users within cached pages is to dynamically request the personalized content via an AJAX request or IFRAME include. This enables SharePoint to serve up that base page content from the cache, while dynamically generating personalized content from a web service or backend ASPX page, which should have much lower processing overhead to retrieve when compared to rendering a full SharePoint publishing page. The dynamic AJAX call or IFRAME include may even execute on a different system altogether, providing yet another way to scale your application further as dynamic content processing is kept entirely separate from serving of static HTML content.

SharePoint Object Cache

The SharePoint object cache is a mechanism built into the SharePoint Object Model, the server-side API used to interact programmatically with SharePoint. Whenever the object cache is enabled on a site, initial requests for certain objects (such as site navigation data and cross list query results used by the Content Query Web Part) from that site are cached in memory, along with security information about those objects. This reduces database access by retrieving site data directly from memory on subsequent page loads instead of querying the database server on each page load.

Because the security settings of the objects are stored in the cache, the object cache automatically trims results for the current user according to that user's security settings. To provide cache objects for end users with potentially different security settings, the object model retrieves more data from the database than the current user requests when first populating the cache. This is where the cross list query multiplier, a cache configuration option, comes into play. When configuring the cache, the default query multiplier is 3. This means that SharePoint attempts to retrieve up to 3 times the amount of data initially requested so that users with different security settings can retrieve a sufficient number of different items from the same cache, reducing additional database load.

The query multiplier may be set to a value from 1 to 10. If you have a site where all users have the same permissions (or are anonymous), you should set this value to 1 to avoid retrieving more data than you will ever need from cross site queries because all users will see the same data. On the other hand, if you have a site with heavily fragmented security settings among site contents and among your user base, set the multiplier higher.

When querying the database for data, the object cache must use accounts that have high privileges to retrieve and cache data that the current user may not have access to, but the next user might. In addition to the rights users have to view particular entities, some users may see draft versions of those same entities, so the cache must hold both at the same time. To accomplish this, the query mechanism requests data from the database that is available to two different users: the Portal Super User and the Portal Super Reader. These user accounts should be configured so that they can view draft and fully published items, respectively. Each time the object cache retrieves data from the database, it issues two queries, one for each super account. This results in both draft and published items to be returned and stored in the cache, resulting in fewer cache misses for all users.

By default, the system uses the web application pool account for the Portal Super User and `NT Authority\Local Service` for the Portal Super Reader. Unfortunately, this is not always ideal. For

www.it-ebooks.info

example, the web application pool account functions as System, which may have items checked out to it. Because you never want to return checked-out items from the cache (only drafts and published items should be returned), this can be a problem. Furthermore, the local service account used as the Portal Super Reader cannot work at all on sites that use forms-based authentication.

To configure your portal super accounts, you first need to create two new accounts (either forms-based accounts or active directory accounts if NT Authentication is available for your site) that satisfy the following requirements:

- Must not be used to login to the site
- Are two different accounts
- Are not used as application pool accounts

After you have created the accounts to be used, you must configure SharePoint to grant those users with the rights needed to read all items, including draft items for the super user account. To set security for these accounts, visit central administration and under Application Management, select Manage Web Applications. Click the web application you want to configure, and click User Policy on the Ribbon. Click Add Users, select All Zones from the drop-down provided, and click Next. Enter the name of the portal super reader account, and check the Full Read check box. Do not check Account Operates as System. Click Finish. Repeat this procedure for the portal super user account, except that you check Full Control instead of Full Read.

After you click Finish, the Policy for Web Application dialog appears. Record the text shown in the User Name column for each of the portal super accounts that you just configured. Those values will be used to set web application properties indicating that those accounts should be used as the portal super accounts. To set the necessary web application properties, execute the following PowerShell commands:

```
$WebApp = Get-SPWebApplication "http://localhost/"
$WebApp.Properties["portalsuperuseraccount"] = "domain\superuser"
$WebApp.Properties["portalsuperreaderaccount"] = "domain\superreader"
$WebApp.Update();
```

Be sure to replace the web application URL value with the URL of the web application you are configuring, and replace the account names with the values you recorded from the Policy for Web Application dialog. For forms-based authentication users, the usernames should look something like i:0#.w|domain\superreader instead of just domain\superreader.

The object cache is most effective on sites that make heavy use of the Content Query Web Part (CQWP). The CQWP uses cross list queries to retrieve data, so the object cache will be used for these queries. SharePoint's navigation controls also make use of the object cache when retrieving data because the navigation nodes need to be retrieved quickly by the page, but also security trimmed for the end user. Additionally, some publishing, search, and metadata retrieval functionality makes use of this cache as well. Because this functionality always makes direct use of the object cache, the object cache cannot be entirely disabled. Although you may set the cache size to a small number to effectively disable it, this is generally not needed or recommended.

When writing code for SharePoint, you may choose to make use of the object cache. By querying list data through the `PortalSiteMapProvider` object, you can automatically take advantage of the

www.it-ebooks.info

object cache. The `PortalSiteMapProvider` abstracts the details of the cache from your code, such that you can simply query for list items through this object without worrying about whether the items are coming from the database or out of the cache. It's all taken care of for you by the object cache API. Given how the object cache retrieves data using portal super accounts, you cannot use this method to retrieved checked-out items. Furthermore, because the cache is designed to run within a web process, you must always have a valid `HTTPContext` (that is, `HTTPContext.Current` must not be null) when utilizing the object cache. This means you must take care not to attempt to use the object cache from within timer jobs, console applications, or other applications domains outside of the web application process.

Before you can use the object cache API, be sure that your project has a reference to the SharePoint.Publishing.dll, as well as the following `using` statements at the top of your code file:

```
using Microsoft.SharePoint;
using Microsoft.SharePoint.Navigation;
using Microsoft.SharePoint.Publishing;
using Microsoft.SharePoint.Publishing.Navigation;
```

The following snippet demonstrates how to use the `PortalSiteMapProvider` to query list data:

```
SPWeb web = SPContext.Current.Web;

//retrieve the provider
PortalSiteMapProvider Provider =
     PortalSiteMapProvider.CurrentNavSiteMapProviderNoEncode;

//Retrieve the site map node for the web in question
PortalWebSiteMapNode Node =
     Provider.FindSiteMapNode(web.ServerRelativeUrl) as PortalWebSiteMapNode;

//build your query
SPQuery query = new SPQuery();
Query.Query = @" <Where>
   <Eq>
     <FieldRef Name='Title' />
      <Value Type='Text'>SomeValue</Value>
   </Eq>
</Where>";

//Query the object cache api for the list items in question
SiteMapNodeCollection Items = portalProvider.GetCachedListItemsByQuery(
                        Node, "Your List Name", query, SPContext.Current.Web);

//process the response in some way
foreach (PortalListItemSiteMapNode item in Items){
     Response.Write(item["Title"].toString());
}
```

The object cache is configured at the site collection level. To configure the cache, visit the site settings page, and under Site Collection Administration, select Site Collection Object Cache. From the cache setting page, you have a number of available options. You may specify the size of the cache in megabytes (100MB by default); although, generally this does not need to be altered because most

www.it-ebooks.info

site collections do not use more than 500KB or so of cache memory. If this limit is reached, the object cache discards the least recently used items to ensure the total size of the cache is under this limit. You may select to flush the object cache, including forcing all servers on the farm to flush their caches. Use this option when stale data becomes noticeable or when altering the Super User and Super Reader accounts used by the cache.

Next, you can select the method by which items in the cache become invalid and must be reloaded from the database. The default is to set a timeout duration in seconds (60 seconds default) after which items in the cache become invalid. The other option is to check the database server for changes on every request, which incurs database latency overhead on each page load, but never results in users seeing stale data. Weigh the need for performance against the need for up-to-date information when choosing your cache invalidation method. Finally, you may select the cross list query multiplier, as described earlier in this section.

In addition to the site collection level object cache configuration, SharePoint 2010 introduces a new object cache configuration option available at the web application level: the object cache *maxSize* override. The maxSize override setting limits the output cache size for the entire web application. This setting is 100MB by default, which curiously matches the default limit for a given site collection. This setting overrides any site collection level setting, but at a global level for the web application, meaning that the site collection limit may never be reached if there are other site collections sharing this cache in the same web application. Consequently, on systems with many site collections, consider raising this limit, ideally to support at least 500KB per site collection, as previously mentioned.

To alter the maxSize setting, locate the `ObjectCache` node in the `web.config` for your web application, and set the maxSize attribute to the maximum number of megabytes to use. This value may be between 1 and 2048.

The BLOB Cache

The last high-level caching functionality within SharePoint is the BLOB cache. This cache stores copies of BLOBs on the file systems of web frontend servers in a SharePoint farm. BLOB stands for Binary Large OBject, which is the data type used to store file content in SharePoint content databases. The purpose of this cache is to avoid querying the database for full file content each time a user requests a file from a document library. Note that this functionality does not apply to site folders, which are different from document libraries. Most effective with resource files that don't change often and are requested frequently, the BLOB cache is highly recommended for any high-traffic SharePoint environment where users regularly download resources stored in document libraries. Because the cache is stored on the filesystem of the WFE servers, disk access may still be involved in retrieving the files from the disk, but at least the database does not need to perform disk access each time the file is requested, which frees the database up for other tasks. Also, because the operating system caches file system I/O, your most commonly accessed cache files will most likely be cached in memory by the OS, assuming your WFE server has enough memory available.

If you host an environment in which many people are collaborating on many different files, you may not see much of a performance improvement with this option. In this type of environment, different users download different sets of files with potentially little overlap between them, making caching less effective.

www.it-ebooks.info

You need to consider a trade-off when enabling the BLOB cache. Specifically, items retrieved from the cache take a performance penalty the first time they are retrieved because the system must query the database for full metadata and permission information about the items in question so that they may be served securely from the cache in subsequent requests. If the file is not often requested, or is often updated, this first-retrieval penalty applies more often. Also, if your site contains many resources that will be accessed through the BLOB cache, each resource requires 800 bytes of memory within the BLOB Cache index (used to quickly find and securely trim requested items), as well as space on disk. Consider moving large numbers of infrequently accessed files to a web application where the BLOB cache is not enabled, or a site folder that is not a document library to avoid using unnecessary disk and memory resources when those items are accessed.

SharePoint 2010 introduces the ability to effectively stream media from the BLOB cache in a way that greatly reduces the time to first byte for large files such as audio and video resources. Previously, SharePoint would have to load a given file into the BLOB cache in its entirety before the server could transmit the file data to the client. This could result in delays of many seconds before a media file would even begin to download if it didn't already exist in the cache. Because these files typically must buffer on the client before they can be played, the wait was noticeable. SharePoint 2010 streams file data from the BLOB cache to the client as soon as bytes are received from the database.

Furthermore, the BLOB cache supports HTTP range requests. This means that clients can request a particular range of bytes for a given file, allowing media players to seek to specific parts of a video or audio file without having to download the entire file. With these features, SharePoint 2010 provides a much better user experience when consuming large media files through the BLOB cache.

The BLOB cache is subject to some additional considerations that you should consider. First, the BLOB cache is configured with a maximum size (default of 10GB). When the BLOB cache grows larger than the maximum size, the least popular entries in the cache are removed. This process is known as cache compaction and utilizes server resources when performed. For this reason, you should tune your cache to avoid frequent compaction by ensuring that the cache is large enough to store the most popular content. BLOB cache compactions may be monitored through the Total Number of Cache Compactions WMI performance counter, available within the SharePoint Disk-Based Cache counters.

Second, although files within SharePoint do not suffer from a URL character limit, the Windows file system imposes a 260-character limit on file paths. BLOB cache files are written to disk within a folder structure that mimics the URLs of those files, and because the BLOB cache can append additional strings to the paths of those files in the cache, files whose path on disk would exceed the file system's 260-character limit cannot be served from the BLOB cache. In practical terms, this means that files with server-relative URLs longer than approximately 160 characters will not generally be BLOB-cached. To avoid this limit, place your most popular and large files in locations with short paths.

Third, files and even entire folders that exist in the cache can become invalidated during certain events, causing them to be removed from the cache and reloaded from the database, paying a first-hit penalty for each invalidated file. Actions such as changing permissions on a document library invalidate all cached items from that library because permission data for each item must be rebuilt. This applies to entire sites as well. Changes in any security settings for a given site invalidate all items in the cache from that site. Because frequently invalidated items have a negative performance impact on your SharePoint farm, try to ensure that changes to your farm that may invalidate cached items are

www.it-ebooks.info

scheduled for nonpeak usage times. As with compactions, you can monitor the number of objects discarded from the cache using the Total Object Discards from Blob Cache WMI performance monitor available within SharePoint Disk-Based Cache counters.

Finally, although the files in the BLOB cache are stored on disk, the index used to access those files is stored in memory and regularly written to disk. When the IIS application pool hosting your site is shut down, the BLOB cache index is flushed to disk one last time. The index is then loaded back into memory when the application restarts for quick access. If the IIS application pool crashes while the index is being written, or if the shutdown time limit for the application pool does not enable enough time for the index to be written to disk during shutdown, the index will be corrupted, and the entire BLOB cache will be flushed upon application restart. The same is true if the application pool startup time limit does not allow enough time for the index to be populated in memory. If there are many items in the BLOB cache, flushing the cache can incur a significant performance penalty because each file needs to be reloaded into the BLOB cache, incurring a first-hit penalty. To avoid this, increase the startup and shutdown time limits for the application pool to 300 seconds for large caches.

Also keep in mind that altering any web application policy settings for a web application invalidates all items in the cache and results in a cache flush because the security metadata for each item needs to be regenerated. Consider keeping track of the Total Number of Cache Flushes WMI performance monitor within the SharePoint Disk-Based Cache counters to ensure that cache flushes are not occurring too frequently.

The BLOB cache has a number of available configuration options. It is configured by altering the `web.config` file, which means that any settings applied take effect on the entire web application. The BLOB cache can be configured to cache files whose path matches a particular regular expression. By default this expression is loaded with a number of file extensions to match, so you can easily configure which types of files are cached. You can also specify the folder path where the BLOB cache will be stored on the local machine, as well as the maximum size of the cache in gigabytes. The default is 10GB. If virus scanning or active file monitoring is enabled on your servers, be sure to exclude the BLOB cache folders from this processing to avoid locking files and impacting performance.

To configure the BLOB cache, open the `web.config` file for the web application and locate the BLOBCache node; then set the `enabled` attribute to `true`. A number of attributes are listed here that may be configured on this XML node to tune the cache with specific settings. Not all these attributes will exist by default, but you may add them to the BLOBCache XML node as needed.

- **Location:** A local filesystem path where the cache will be stored. Ideally this location has a short path and is not located on the system drive to avoid disk contention with the OS. Utilizing small SSDs on your frontend servers for this purpose should work well.
- **Path:** A regular expression used to match against file URLs to determine whether they should be included in the BLOB cache. The default path matches against a number of common asset file extensions, such as jpg and png.
- **maxSize:** The maximum size of the cache in GB.
- **maxSize:** Controls the content expiration header sent to client browsers. (See the following paragraph for more on this.)
- **changeCheckInterval:** The amount of time in seconds that an item can exist in the cache before the system checks for changes to that item. By default this is 5 seconds. Increasing this

www.it-ebooks.info

value can improve performance at the expense of seeing slightly more stale data. For publishing environments, 30 seconds or more may be ideal. For collaboration environments, this setting should not be changed.

- **writeIndexInteval:** The interval, in seconds, on which the BLOB cache index is written to disk. By default this is 60 seconds. If your BLOB cache is large, consider increasing this interval to up to 1 hour to reduce the load associated with index writes.
- **deleteOldCacheFolders and deleteCheckedInFiles:** These settings control the maximum number of seconds that folders and files, respectively, exist in the cache after being invalidated. Instead of deleting files as soon as they are invalid, the cache marks them for deletion. Both values default to 600 seconds, or 10 minutes. These settings should not need to be altered.
- **largeFileSize:** The size in bytes that defines a "large file" in the cache. Any files larger than this value will be handled differently by the BLOB cache. As previously mentioned when discussing large media files and HTTP range requests, the BLOB cache is designed to ensure that end users may begin to receive large files' contents before the entire file has been received by your web frontend server from the database. The default largeFileSize is 1MB.
- **readChunkSize:** When large files are read from the database through the cache, they are read in chunks of this size, in bytes. A smaller value results in shorter times to first byte, but more round trips to the SQL server, shortening latency but lowering overall throughput. A larger value results in longer latency but better throughput. This value generally does not need to be tuned.
- **throttleLimit:** When retrieving large files from the database, the BLOB cache spins up multiple threads, each requesting parts of the file, to increase throughput. This value indicates the maximum number of threads that the BLOB cache creates during this process. A higher number negatively impacts SQL server load. This value defaults to 10 and should generally remain untouched. Consider reducing this value (or raising the largeFileSize) if you serve a large number of large files and your SQL servers become heavily loaded.

One of the nicest features of the BLOB cache is the max-age setting. The max-age setting is used to control the content expiration header sent to client browsers when files from the cache are downloaded. This prevents the client browsers from reloading the cached items each time a page is loaded which references them, providing an additional decrease in client, server, and network processing when enabled. Without this setting, most client browsers still check to see if the item has been modified each time a user requests the file. The max-age setting is defined in seconds, and defaults to 86,400 seconds, which is 24 hours. Consider reducing this value if you have content that frequently changes. If you do choose to use a long max-age value, don't cache CSS files that reference cached images because those CSS files, when cached by the client, can't be altered to redirect users to a new image within seconds. With cached CSS and images, it could take up to the max-age setting for users to see content or style changes. By not caching your CSS, you have the opportunity to immediately redirect imagery for all users.

Occasionally, the BLOB cache may get corrupted, perhaps if a file is removed from the cache by mistake. Alternatively, the BLOB cache can become entirely out of date during system changes, such as when a content database is restored. When this happens, you may want to consider flushing the cache completely so that SharePoint can rebuild it. You have two options here. You can disable the BLOB cache, remove the BLOB cache folder on disk, and then re-enable the BLOB cache. Alternatively, you

www.it-ebooks.info

can use the object model (via console app, PowerShell, or other .NET environment) to flush the cache. To flush the cache using C#, simply execute the following commands:

```
using(SPSite s = new
 SPSite("http://mywebapplication.com")){
  Microsoft.SharePoint.Publishing.PublishingCache.FlushBlobCache(s.WebApplication);
}
```

To flush the cache using PowerShell, execute the following commands:

```
$WebApp = Get-SPWebApplication "http://mywebapplication.com"
[Microsoft.SharePoint.Publishing.PublishingCache]::FlushBlobCache($WebApp)
```

You must execute these commands on each web frontend server in your farm because they flush only the cache on the current machine.

CACHING IN CODE

When it comes time to create custom controls, web parts, pages, or other functionality that displays dynamic content to the user, one of the best ways to ensure good performance is to cache data that is frequently accessed by your controls. As you saw earlier in this chapter, reducing disk access by keeping data stored in memory is one of the most effective techniques to increase the speed of your code. For caching data, many caching mechanisms are available to choose from, each with its own advantages and disadvantages. This section discusses how these caching methods differ and when to make use of each.

Application Object

ASP.NET provides a number of objects during page execution, such as the `Application` and `Session` objects, which are used to store data at different scopes. Access to these objects is provided via the current `HttpContext` object for a particular web request. The `Application` object is a global dictionary, common to all users and page requests within a given web application. Any .NET object may be placed in the `Application` object, along with a unique string identifier (also known as a cache key) used to retrieve the value. Any items placed in the `Application` object are stored in memory on the current web server and are not shared between web servers, meaning that if one web server places an item in the `Application` object, it will not be available to the other web servers on the same farm until they retrieve the item themselves and place it in their `Application` objects. The `Application` object is useful for storing global data that you do not want to expire, but that may be regenerated when the IIS application pool is reset.

The following code demonstrates how to access and modify the `Application` object.

```
//set a value
HttpContext.Current.Application["SomeKey"] = "SomeValue";

//retrieve the value
string myValue = (string)HttpContext.Current.Application["SomeKey"];

//remove the value
HttpContext.Current.Application.Remove("SomeKey");
```

www.it-ebooks.info

Session Object

In traditional ASP.NET, the `Session` object is similar to the `Application` object, in that objects may be stored in memory on the current server along with a string cache key. The difference is that the `Session` object is not shared between users and expires entirely after a user has stopped interacting with the Web site for some period of time (typically approximately 20 minutes). This makes the `Session` object ideal for storing user-specific information that is needed only for the current browsing session, such as shopping cart data or user profile data. ASP.NET provides the ability to store session data in a database or in memory, as well as on a single session server within your farm that centralizes session data for all servers, but keeps it in memory.

When configured within SharePoint, however, session state data is always stored in a database. This database is configured in the `web.config` file of your web application, just like any ASP.NET application, but you can use a PowerShell command to enable it. The session state database may be placed on an entirely separate database server if your farm makes heavy use of the session. By default, sessions are disabled on SharePoint farms. Only enable session state if it will be used by components on your farm because the overhead of creating sessions for each user impacts scalability.

One advantage to storing session state in the database is that the data stored in sessions is available to all web frontends and does not need to be reconstructed when users' requests land on different web frontend servers because all servers can see the same session data, not just their own copies. For this reason, the session object is good for storing user-specific data used during a single browsing session that is costly to construct. Security trimmed, user-specific navigational data, for example, may fit this model.

The following code snippet demonstrates how to work with the `Session` object. As you can see, it works just like the `Application` object discussed previously.

```
//set a value
HttpContext.Current.Session["SomeKey"] = "SomeValue";

//retrieve the value
string myValue = (string)HttpContext.Current.Session["SomeKey"];

//remove the value
HttpContext.Current.Session.Remove("SomeKey");
```

View State

When coding custom ASP.NET pages, one option available for caching data is the page's view state. A page's view state contains a description of the state of the page, including control values and properties. When an ASP.NET page posts back to the server (when a button is clicked, for example), view state is included in the data posted back to enable the server-side code to know the state of the page on the client. In addition to the auto-populated control data, you may add your own data to the view state of the page. This data will be available when the page posts back. This data isn't stored in memory on the server between postbacks, but within the page itself, inside a hidden input variable in HTML.

The view state is great for storing values needed while the user is interacting with a particular page. For example, if you create a wizard that runs within a single ASP.NET page, you might store the current

www.it-ebooks.info

wizard step in view state because it is needed only on the current page during postbacks. This would enable you to avoid turning on session state in addition to writing to disk each time the user advances a step. Also, the data can be posted back to any web frontend because the server itself is not storing state between requests.

The following code snippet demonstrates how to interact with the `ViewState` object within an ASP.NET page or control.

```
//set a value
this.ViewState["SomeKey"] = "SomeValue";

//retrieve the value
string value = (string)this.ViewState["SomeKey"];
```

HTTPContext

Another fundamental ASP.NET object is the `HTTPContext`. This object is created each time a request is handled by IIS, and provides access to all the other objects (such as the application and session objects) within the current page execution context. The `HTTPContext` contains an `Items` property bag, which, like the session and application objects, enables you to store objects using a string key as a reference. The difference here is that the `HTTPContext.Items` property bag is destroyed after each page request completes, making this a short-lived cache option, useful when trying to avoid querying the same data twice on the same page load.

For example, you may have a page containing multiple web parts, each of which must display different pieces of information from the current user's profile. Instead of having each web part independently query the profile database, resulting in the same data being retrieved multiple times, you could have the web parts check the `HTTPContext.Items` property bag for the profile object before attempting to retrieve that data from the database. If the profile object doesn't exist in the property bag, the web parts can retrieve the profile from the database and place it in the `HTTPContext.Items` property bag for the next web part to make use of. This technique is great for when you need to only cache data for a single page request, but consider caching data in a more persistent mechanism (so that it is cached between page loads) if it is anticipated that the same user will reload that page multiple times in a short period of time.

The following code snippet demonstrates how to work with the `HTTPContext.Items` collection. As you can see, it works just like the `Application` and `Session` objects discussed previously.

```
//set a value
HttpContext.Current.Items["SomeKey"] = "SomeValue";

//retrieve the value
string myValue = (string)HttpContext.Current.Items["SomeKey"];

//remove the value
HttpContext.Current.Items.Remove("SomeKey");
```

www.it-ebooks.info

HTTP Cache

The most powerful and versatile caching mechanism within SharePoint is the ASP.NET HTTP cache mechanism. This cache was designed for use by ASP.NET Web sites and includes a number of useful features specifically designed for Web site caching. Like the other caching mechanisms discussed in this section, the HTTP cache requires that each item added to the cache have a string key used to set and retrieve the item. Unlike the session cache, the HTTP cache is not specific to any one user, so the cache key used to cache user-specific items, for example, should include the user's login name or some other unique identifier.

The HTTP cache contains mechanisms to automatically expire items, as well as to free up items in memory when the cache grows too large. This is known as *scavenging*. Each item in the cache is associated with a value indicating how expensive it is computationally to generate that item. The HTTP cache frees up items that are less expensive to generate before deleting other items. Items that have not been accessed recently or have expired are also more likely to be deleted from the cache.

In addition to the scavenging capability, the HTTP cache also supports sliding expirations, such that as long as an item is retrieved from the cache within a given time period, the expiration time will reset to a later time. Furthermore, the cache supports dependencies, such that items in the cache can be configured to become invalid when a file or set of files on disk is altered in some way. Because SharePoint generally stores items in content databases, this feature is of limited use in custom SharePoint applications. Finally, you may register a callback function with each item in the cache that will be called when that item is removed from the cache.

The HTTP cache is stored in memory and is unique to the process that hosts it, in this case an IIS application pool. Access to data is fast, and cached items persist for a variable amount of time depending on the memory available for caching, but the cache will be cleared each time the application pool resets. Also, because the cache lives in memory, each web frontend has its own copy of the cache, causing cached items to be regenerated for each frontend server that requires the items.

To use the HTTP cache in your code, ensure that your code will always run within the context of a web application. (Timer jobs and console apps need not apply and should make use of another caching mechanism.) You may access the cache via the `HttpContext.Current.Cache` or via `Page.Cache`. (Or just `Cache` when your code is running within the context of a `Page` class directly.) To add an item to the cache, you may use one of two methods: `Insert()` and `Add()`. The only difference between the two is that `Insert()` has a number of overloaded implementations that enable you to pass fewer parameters if needed, whereas `Add()` requires all configuration parameters to be supplied. The following line of code can add an item to the cache:

```
HttpContext.Current.Cache.Add(
     "Your cache key",
     "The value to cache",//this can be any object
     null, //you could register a dependency on a filesystem file here
     DateTime.Now.AddMinutes(30), //absolute expiration.
     new TimeSpan(0, 5, 0), //sliding expiration of 5 min
     System.Web.Caching.CacheItemPriority.Default, //many options here
     null //you could register a callback here
);
```

www.it-ebooks.info

To retrieve an item from the cache, use the `Get()` method. As shown in the following code snippet, this method simply accepts the cache key as an argument and returns the object in the cache that corresponds to that key. If the object is not found in the cache (perhaps because it was flushed from the cache or because it was never added in the first place), the `Get()` method returns null. You must check this and regenerate or reload the item from the database if needed.

```
object myCachedItem = HttpContext.Current.Cache.Get("Your cache key");
if(myCachedItem == null){
     //recreate the item and add it back to the cache
}
```

Static Variables

The caching mechanism that provides you, the programmer, with the most amount of control is simply the use of static variables. A static variable exists in one location for the life of a process, and there can never be two instances of the same static variable in a given process. This is useful because you can define generic dictionaries or lists to store anything you want in memory and can manage that memory as needed. You may provide your own mechanism to flush your cache as needed, as well as provide a background mechanism to persist these in a memory variable to a custom database, local file, or SharePoint list.

The most common use of static variables for caching involve creating a generic dictionary with a key of type "string" and a value of whatever type you need to cache. Your code can then determine the appropriate string cache key with the correct level of uniqueness for whatever items you need to store or retrieve, and make use of that key when interacting with the static dictionary object. This is the basis for the custom-caching mechanism presented later in this chapter.

Within a web process multiple requests can execute at once. This means that you must be sure to access your static variable in thread-safe fashion, using a locking mechanism (such as the `lock` keyword) to prevent two different threads from trying to alter the same object at the same time.



See "A Reusable Caching Method" in the following sections for an example of how to cache using static variables while making use of the `lock` *keyword to ensure thread safety.*

Local Files and the SharePoint Content DB

Although the file system is the slowest place where data can be stored on the system, it has the advantage of not being volatile. In other words, it persists between page loads, sessions, process resets, and machine resets. For storing large or computationally expensive items, the file system may be an appropriate place to cache data. For similar reasons, you may also choose to cache in the SharePoint Content DB, ideally for when you don't want individual web frontends each generating the same expensive items. You can make use of a custom list, a file residing in a site folder, or web property (accessible via `SPWeb.AllProperties`) to store data in a SharePoint Content DB. Values stored in site properties may be as large as 4GB in size.

www.it-ebooks.info

The following code snippet demonstrates how to use a site property to store data in the site's content DB.

```
 //retrieve the root web for the current site
SPWeb rootWeb = SPContext.Current.Site.RootWeb;

//set a property value
rootWeb.AllProperties["SomeKey"] = "SomeValue";
//update the web to persist the value to the content DB
rootWeb.Update();

//retrieve the value
string someValue = (string)rootWeb.AllProperties["SomeValue"];
```

Of course, if this data is to be frequently accessed, you may also choose to cache it in memory as well. This enables you to avoid recalculating the expensive item every time the application resets, while still taking advantage of fast access to the data in question when present in memory.

A Reusable Caching Method

Often, code needs to access a specific list of data on a regular basis. Whether it's configuration data, security data, or other frequently accessed data, it's sometimes useful to just load the data all at once into memory and then read individual items out of the cache as needed. This reduces overall server round trips because items aren't individually retrieved, but within entire groups of items at one time. Additionally, this data can often be served somewhat stale, especially if response time and throughput are more important than up-to-date data. To handle cases like this, you can build a reusable caching object used to store any kind of data you want.

To build such a class, create a new class module in an existing SharePoint project within Visual Studio, or in an external shared library if you want. Add the following `using` statement to the top of your code file:

```
using System.Threading;
```

Because the reusable cache class can perform asynchronous loads, the previous `using` statement can make your threading code more compact. Next, replace the empty class declaration with the following:

```
class CachedDictionary<SomeObject, ObjectKey>
{
}
```

This defines your cache class as a generic class that supports any type of cached object and any type of key (as long as that type may be used within a hash table, meaning that it must implement the `GetHash()` function). Next, add the following properties to your class definition:

```
//these delegates define method signatures used to retrieve values and keys
public delegate List<SomeObject> ListRetriever();
public delegate ObjectKey KeyRetriever(SomeObject obj);

//the underlying dictionary that stores the cache
private Dictionary<ObjectKey, SomeObject> _cache;
```

www.it-ebooks.info

```
//time when the cache was loaded
private volatile DateTime _loaded;
//maximum amount of time before data is reloaded into the cache
private int _timeoutMinutes;
//maximum stale time for served files
private int _maxStaleMinutes;

//used to store the methods used to retrieve values and keys
private ListRetriever _getItems;
private KeyRetriever _getKey;

//used to indicate when a reload is running
private volatile bool _reloadRunning = false;
//used for thread synchronization
private object _syncRoot = new object();
```

Although the comments in the previous snippet give you some idea of how this object can work, the following function implementations and their associated comments should make this more clear. Insert the class constructor:

```
//main constructor, used to initialize the cache
public CachedDictionary(
                    ListRetriever GetItems,
                    KeyRetriever GetKey,
                    int TimeoutMinutes,
                    int MaximumStaleMinutes)
{
    _getItems = GetItems;
    _getKey = GetKey;
    _timeoutMinutes = TimeoutMinutes;
    _maxStaleMinutes = MaximumStaleMinutes;
    ReloadCache();
}
```

Only four parameters are required to initialize the cache. The first two parameters are methods that can retrieve the items to be cached and their keys, respectively. The first method must return a generic list of type `SomeObject` (your first generic type), whereas the second method must accept an object of type `SomeObject` and return an object of type `ObjectKey` (our second generic type). `TimeoutMinutes` represents the amount of time in minutes that the items will be cached for before being reloaded, whereas `MaximumStaleMinutes` represents the maximum age of the items in the cache that can be served to end users.

Those two values are different because this class can serve data that has reached or exceeded the timeout limit to end users, to provide those users with some quick data while the cache is reloaded in the background. If the data is older than `MaximumStaleMinutes`, users must wait until fresh data is reloaded before they can access the data. This scheme ensures that on loaded systems, client requests are not held up waiting for cache reloads. Those cache reloads operate in the background unless the data has reached the `MaximumStaleMinutes` setting, in which case the data probably hasn't been requested recently enough to kick off a background cache reload, meaning that the system isn't actually loaded at all (or at least not this functionality).

www.it-ebooks.info

Add the `ReloadCache()` function to the class:

```
private void ReloadCache()
{
    try
    {
        _reloadRunning = true;

        //initialize new storage
        Dictionary<ObjectKey, SomeObject> tempCache =
                                            new Dictionary<ObjectKey, SomeObject>();

        //get the items for the cache
        List<SomeObject> tempList = _getItems();

        //load them into the dictionary, calling _getKey on each item
        foreach (SomeObject obj in tempList)
        {
            tempCache.Add(_getKey(obj), obj);
        }

        //set the cache reference to our newly created dictionary
        //in a threadsafe way
        lock (_syncRoot)
        {
            _cache = tempCache;
            _loaded = DateTime.Now;
        }
    }
    catch (System.Exception ex)
    {
        //log this error however you wish
        throw;
    }finally{
        //we must execute this in a finally block
        //otherwise a separate thread could get into an infinite loop
        //waiting for a reload to complete
        _reloadRunning = false;
    }

}
```

The previous code does not operate on the existing cache object but creates a new dictionary and sets only the objects `_cache` member variable after performing the data load. This keeps the cache available to other threads for as long as possible. The final piece of code is the implementation of the `Items` property. This code checks for timed out or stale content and kicks off asynchronous or synchronous (in the case of stale data) reloads. Add the following code to the class to complete the class implementation:

```
//used to return a reference to a valid cache dictionary
public Dictionary<ObjectKey, SomeObject> Items
{
    get
    {
```

www.it-ebooks.info

```
            //check if data is too stale
            if (_loaded.AddMinutes(_maxStaleMinutes) < DateTime.Now)
            {
                //data is too stale, reload the list and return...

                //check to see if data is being reloaded on another thread
                if (!_reloadRunning)
                {
                    ReloadCache();
                }
                else
                {
                    //if it is already being reloaded, wait for it to complete.
                    while (!_reloadRunning)
                    {
                        System.Threading.Thread.Sleep(10);
                    }
                }

            }else if (_loaded.AddMinutes(_timeoutMinutes) < DateTime.Now)
            {
                //data has timed out
                //async reload, only if one isn't already running.
                if (!_reloadRunning)
                {
                    _reloadRunning = true;
                    ThreadStart threadStart = new ThreadStart(delegate() {
                                                                this.ReloadCache();
                                                            });
                    Thread thread  = new Thread(threadStart);
                    thread.Start();

                }

            }

            //return the cache in a threadsafe way (while it isn't being modified)
            lock (_syncRoot)
            {
                return _cache;
            }
        }

    }
```

To use this cache, you can simply declare static instances of initialized cache objects in your code. So if you have a particular page or component that needs to frequently access cached data from a list, you could declare the following variable to load all items from a list called `TestList` from the root site at `"http://localhost"`:

```
static CachedDictionary<SPListItem, string> CachedItems =
     new CachedDictionary<SPListItem, string>(
          delegate()
            {
                using (SPSite site = new SPSite("http://localhost"))
```

www.it-ebooks.info

```
                {
                  return
                    nsite.RootWeb.Lists[glist].Items.Cast<SPListItem>().ToList();

                }
            }, o => o.Title, 5, 10
        );
```

In the previous listing you can see the method that retrieves the list items clearly defined; it's simple and impractical for larger lists (you could run into throttling problems) but works as an example. The function that accepts an object and returns the object key is defined as a lambda expression for brevity; in this case the key returned is the title of the list item. This code will fail if two items in the list have the same title, so care should be taken when selecting keys in the real world. In this case, the timeout is set to 5 minutes, whereas the maximum stale value is 10. That means that as long as users request this data at least every 10 minutes, they never need to wait for the cache to load. (It will happen in the background.) Meanwhile, if someone makes a request after just 5 minutes of inactivity, the cache reloads in the background to provide fresher data for the next request.

To retrieve data from this cache, you must know the key of the items to be retrieved. That's because this cache was originally designed to store key-value pairs from a configuration list; although many other potential uses exist. To access an item from the cache as previously defined, simply access the `Items` property and pass the key (in this case `"someItemTitle"`) in as an index:

```
int Id = CachedItems.Items["someItemTitle"].ID;
```

As you can see, this caching mechanism completely abstracts the loading and reloading of cached items while eliminating wait times for users in most cases. This class can be easily copied and modified to support a generic list instead of a generic dictionary behind the scenes. Consider using such a class for scenarios that don't involve key-value pairs.

MANAGING LARGE LISTS

A common performance problem that creeps up on SharePoint users involves lists that grow large over time. Often, list views perform well when first created; it isn't until the content in the list grows to a point in which performance slows that the problem becomes evident. Usually this happens after a few thousand items or more are added to a list that has not been optimized for large numbers of items. The problem comes down to how the list data is retrieved from the database and processed by the web frontend for display to the end user. Suppose you limit your list views to display only a certain number of items using an item limit setting. Even with this setting, SharePoint must run a query to retrieve all the items, potentially sort and filter them, and only then can the items be trimmed down to the item limit that was set on the view. For a table with millions of items, this can take an unacceptable amount of time.

Query Throttling

To mitigate the effect that inefficient queries can have on your SharePoint farm, SharePoint 2010 introduces Query Throttling, a mechanism that restricts processing of queried list data to a certain number of items. When a view attempts to query more than a preconfigured number of list items (the

www.it-ebooks.info

list view threshold), the query operation halts. The query returns the information it has processed, along with a warning to the user indicating that the query did not complete and some results may be missing. Any custom code that runs into this limit encounters an expensive query exception that must be handled.

The purpose of the query throttling mechanism is to mitigate the potential for an overloaded system by preventing the server from unnecessarily executing expensive queries. From an administrative perspective, this is quite nice because it ensures that SharePoint applications are "playing nice" with the rest of the system. If queries are throttled, the application owners become aware of the issue and are forced to fix it before causing problems for other applications. From a developer's perspective, however, you need to anticipate problems that may arise as a result and mitigate them before query throttling affects the usability of your application.

Query throttling is configured at the web application level. By default, users are subject to a 5,000-item limit. This limit applies to the rows returned to the web frontend for processing when executing queries and matches the SQL server's default limit where it will escalate from row-level locks to a table-level lock. When a table is locked by the SQL server during a query, overall throughput is degraded because other queries may not operate on that table during that time. For a content database that is queried often, this can result in a significant performance penalty. Administrators, on the other hand, are subject to a 20,000-item-view threshold by default. The purpose of this is to provide administrators with the ability to manage larger lists without allowing end users to impact performance.

Query throttling settings for each web application may be altered via Central Administration. To access these settings, under Application Management, click Manage Web Applications; select the web application you want to modify; on the Ribbon, click the General Settings button; and select Resource Throttling.

From there you can configure a number of settings including list view thresholds for both end users and administrators; specify if developers may override throttling via the object model; and limit the number of lookup (including person and workflow status fields) that may be returned in a single query. Lookup fields take extra time to retrieve because the system must essentially join data across tables for each lookup field in a query. You may also configure query throttling to be disabled during a certain time of day. Doing so will allow the system to perform more data-intensive operations during times of nonpeak usage, allowing for reporting and exporting of large data sets that would otherwise fail.

If the object model override is enabled, developers may alter query throttling settings for queries over a single list or an entire site, as well as disable query throttling entirely for individual lists. To disable throttling on an individual query by overriding throttling settings, simply set the `QueryThrottleMode` property as shown in the following snippet. This property is also available on the `SPSiteDataQuery` object as well.

```
SPQuery qry = new SPQuery();
qry.QueryThrottleMode = SPQueryThrottleOption.Override;
```

To disable throttling entirely on an individual list, set the `EnableThrottling` property to false as shown in the following snippet:

```
SPList list = site.RootWeb.Lists["List Name"];
list.EnableThrottling = false;
List.Update();
```

www.it-ebooks.info

Unless the object model override is enabled, the two preceding code snippets have no effect.

If the object model override is disabled, there is a simple way to get around the query limit by coding around the problem. By making use of the `RowLimit` property on the `SPQuery` and `SPSiteDataQuery` objects, you can limit the number of rows returned from the query and avoid triggering throttling limits.

In addition, you can then page through the entire result set for your query beyond the initial `RowLimit` by making use of the `ListItemCollectionPosition` property. When retrieved from an `SPListItemCollection` that was returned from a query, this property indicates the location of the next set of data in the results set that couldn't be returned due to the `RowLimit`. When set for a query, this property indicates the position of items in the result set that should be retrieved when the query executes. This is useful because you can simply pass the `ListItemCollectionPosition` from the results of one query into the call for the next query, until the item is null. The following code demonstrates how easy it is to page through results using the most common implementation:

```
using(SPSite site = new SPSite("http://yoursite")){

     SPList list = site.RootWeb.Lists["SomeList"];
     SPQuery query = new SPQuery();

     //set the RowLimit, or page size
     Query.RowLimit = 10;
     Query.Query = @"<OrderBy><FieldRef Name=""ID""/></OrderBy>";

     do{
          //query the database
          SPListItemCollection results = list.GetItems(query);

          foreach(SPListItem item in results){
               Response.Write(item.ID.ToString());
          }

          //set the next item to retrieve from the current results
          query.ListItemCollectionPosition = results.ListItemCollectionPosition;

     }while(query.ListItemCollectionPosition != null);
}
```

In addition to the benefits of avoiding expensive query exception, the preceding code listing is a good way to manage the retrieval of large data sets from lists in SharePoint. Even if throttling is disabled, it's better to page through lengthy resources than to retrieve them all at once. If anything, doing so reduces the memory footprint of your code because only so many list items need to be held in memory at one time.

Indexed Fields

As long as the views used to retrieve list items are configured for performance, you can store millions of items in a SharePoint list without problems. The key is to ensure that you retrieve only a small subset of the list data, located using views with filters on indexed fields. An indexed field is a field that can

www.it-ebooks.info

be used to filter and sort list items quickly when compared to a non-indexed field. This is similar to how an index in the back of a book works. Without the index, you would need to scan every page of a book, checking each word, to find the locations of a particular word or phrase. With the index, you can scan a sorted list of terms and quickly find the pages containing the word or phrase in question. So by indexing a field, you ensure that SharePoint can quickly jump to any list items that have particular values for that field.

Each additional field indexed can incur additional storage costs, as well as slower inserts, updates, and deletions of list items. This is because SharePoint must store additional information in an index each time an item is created or modified. For this reason, you should index only the columns that will be used to filter data in your views. Furthermore, views may take advantage of filters only when specific values or a specific range of values are selected. Filtering on an indexed field with a `Contains` operation causes the index not to be utilized for that query. For that reason, SharePoint doesn't even enable you to create indexes for multiline text fields; it would be useful only when searching for the entire multiline value, which is not a likely scenario. To perform substring searches on list fields, instead query the search service. SharePoint search is optimized for full-text queries, enabling for efficient retrieval of list items filtered by substring matches.

Configuring Views

When configuring list views for large lists, you should always try to add a filter to your view that operates against an indexed field. Ideally, this field should have high "selectivity," meaning that there is a good distribution of different values for this field among the total set of items in the list. This way, the filter can reduce the number of list items returned by a significant amount. Keep in mind that only the first filter can make use of an index when retrieving items for a list view. Any filters beyond the first will be implemented by scanning through the remaining values after the first filter has been applied, not by making use of any indexes. For this reason, you should always try to use an indexed field with the most "selectivity" as the first filtered field.

One common way to filter large lists is to filter by Created or Modified date fields, where you set the filter to display items greater than `[Today] - 7`. This can restrict the view to items created or modified in the last week, usually enough to limit views to a reasonable size. To limit views to items the user has personally created, you can filter on the Created By field where the value equals `[ME]`. Otherwise, it's a good idea to make use of choice fields with fixed options to use for filtering. These types of fields are particularly well suited to indexing because they contain a small number of distinct possible values. Because indexes don't help with partial (substring or "contains") lookups, these fields provide uniform values to filter against.

Another way to limit the number of items that need to be processed when executing list queries is to make use of folders. By placing items into organized folders, such that each folder never has more than a few thousand items, you can ensure that your queries never need to process more than a few thousand items at a time. If you need to view items by some criteria other than how they are organized into folders, you may still create folderless views that display the list items without regard to their placement in folders. If you create a folderless view on this type of list, you must then make use of a highly selective indexed field to ensure that your query performs.

www.it-ebooks.info

CLIENT-SIDE OPTIMIZATIONS

Client-side performance is subject to the same problems that impact server-side performance: latency and throughput. Just as the web frontend servers in your farm must efficiently retrieve data from the SQL server and cache as much as possible, client-side web browsers must efficiently retrieve data from the web frontends and cache as much data as possible themselves.

Luckily, because HTML can link to external images, JavaScript, CSS, and other resource files, caching of these resources occurs independent of the pages that link to them. Unfortunately, those same external resources introduce additional latency for the client because they must be downloaded one by one, typically sequentially unless stored on different hosts. Furthermore, even when these resources are locally cached, client machines still check the server for each file (again, one by one, sequentially) for each page request to see if any of those files have expired and need to be redownloaded (unless 'no-check' cache headers are used). Still, there are a number of steps you can take to improve client-side performance, some of which can also take load off your servers as well.

Content Distribution Networks

A content distribution network or content delivery network (CDN) is a service that hosts web content for delivery to client machines. CDNs typically implement edge servers to provide geographically distributed sources for the web content that they store and deliver. With such edge servers configured, a CDN becomes a large, distributed cache used to store and deliver content to end users with redundancy, high throughput, and low latency. Geographically aware DNS services are used to direct client machines to the most local server available for the content that is requested.

Because the CDN is geographically distributed among many edge servers, it can take some time to push new content to all edge servers in the network. This means that you should distribute only relatively static resources over such a network, or be sure that you can quickly link to a different resource should the need arise to alter the content on a page linking to resources on a CDN.

When you move content from your SharePoint farm onto a CDN, you eliminate the need for client machines to request that content from your farm, reducing the load on your servers. There's no need to buy more SharePoint licenses and additional servers just to keep up with end users downloading images and other static content. Because the CDN provides local, fast retrieval of that content, your end users can see better performance loading that content as well.

The best candidates for content that should be offloaded onto a CDN are clearly larger image, video, and audio files because these items are large and typically do not change when published. Furthermore, CDN networks typically support controlled streaming of media files in ways that SharePoint does not. Other resources such as linked JavaScript and CSS files may also be placed on a CDN. Even though these files are small, the client must incur some network latency cost when downloading them, so it may be worth it for you to place them on a CDN.

When using a CDN, you need to link to files in such a way that your content can be quickly updated without having to wait for content changes to propagate across the CDN. For example, imagine that you place your images and CSS in a single folder (the same way SharePoint themes are built) and reference all images via CSS. This way, your page content never links directly to images, and you can distribute the images via the CDN or your local farm as needed. A single change to the referenced

www.it-ebooks.info

CSS link on the page can cause all image files to be loaded from one place or the other. Alternatively, you can simply host the CSS file locally at all times and link to images in the CDN as needed. If you need to alter an image, you can quickly update the CSS to point to a local file until your altered content has been fully distributed across the CDN.

Reducing Server Round Trips

When a client web browser loads a web page, much of the time is spent loading resources such as images, JavaScript, and CSS. Each of these resource files requires a round trip to the server and back. Depending on the cache and expiration headers received when a resource file is first retrieved, the browser may continue to check the server to see if the items have changed on subsequent requests for that file. Often, even if the items have not changed, a server round trip, with its associated latency, is still required. Clearly, reducing these round trips is important for client-side performance, just as reducing disk access is important for server-side performance. As an added bonus, any reduction in server round trips also reduces the load on your SharePoint servers as well.

SharePoint 2010 manages cache and expiration headers to ensure that files that almost never change are always cached by the client. Files located in `_layouts`, for example, can be cached on the client for up to a year because SharePoint transmits a "Cache-Control: max-age= 31536000" header with such files. Files stored within document libraries, on the other hand, are transmitted with a max-age value of 0, meaning that the browser checks for file changes on the server each time those files are requested.

The easiest way to ensure that SharePoint instructs clients to cache resources located in document libraries is to enable the BLOB cache for your web application, as mentioned previously in this chapter. If you don't want to enable the BLOB cache or want to have complete control over how client browsers cache files requested from your farm, you can configure a URL Rewrite Module or HTTP Module to alter the max-age setting on the Cache-Control header for whichever files you deem appropriate. Files that might need to be altered often (or even at unpredictable times) and dynamic content should not be cached by the client. If you need to ensure that client machines download a changed file that had previously been served with a long max-age setting, consider adding a random query string value (such as `?random=19386`) to the referenced URL for that file. Client browsers will consider this a new, different resource and reload the file. The random value will not affect the content downloaded, as the extra query string value is simply ignored by the server.

The simplest way to reduce server round trips is to reduce the number of resource files that your HTML references. Multiple JavaScript or CSS files, for example, can be combined together into a single JavaScript or CSS file, unless they are included on different pages or need to load at different times.

Similar to how JavaScript or CSS resource files can be combined to reduce round trips, image files can also be combined into a single file. This technique is known as *CSS Sprites*. With CSS sprites, a number of images (typically icons) are combined in a single image file, and CSS is used to render the original images by showing only portions of the combined image where needed on the page. This technique can be used on graphics or photos, but you should not mix and match image types when creating combined images. For example, a typical gif image (used for graphics) would consume far more bandwidth when combined into a jpeg (used for photos), and a jpeg image would either lose quality or consume far more bandwidth when combined into a gif.

www.it-ebooks.info

In addition to combining these files, JavaScript and CSS that is used on a single page can be included inline on the page itself because the major advantage to linking to external resources is reuse on other pages across the site. This technique can be counterproductive, however, if that one page is loaded over and over again by the same user. In that case the user ends up redownloading the same JavaScript and CSS for each request, which may be wasteful.

Finally, if your site is not configured for anonymous access, each resource requires two server round trips to be retrieved. This is due to the way that authentication is implemented in the browser and on the server. Instead of sending established credentials with each HTTP request to the server, the browser first tries to anonymously download the content. If the content requires authentication, the server responds with a message indicating that credentials are needed. Only then can the browser send credentials to the server, after the first request has failed. This is true of requests that simply check whether an item has expired as well.

To avoid these extra round trips in authenticated environments, create a document library, enable anonymous access for that library, and place your nonsecure resource files in that location. This way, those files will be available without the extra server round trip. Because resource files such as CSS, JavaScript, and images don't typically need to be secured, this technique can improve performance on almost any authenticated environment in SharePoint.

WRITING EFFICIENT CODE IN SHAREPOINT

Much of the processing that occurs on a web frontend involves retrieving, transforming, and transmitting data. A number of SharePoint and C#-specific coding techniques can be applied to many common problems encountered in SharePoint applications. These techniques can make a significant difference in performance and should be used whenever applicable.

Adding and Retrieving List Data

When using the object model to interact with lists, particularly large lists, you must take care to avoid inadvertently querying large amounts of data. For example, many code snippets online suggest the following way to add a new item to a list:

```
SPList.Items.Add();
```

Unfortunately, although this method works to add an item to the list, it also has the side effect of retrieving every list item from the server as well, something that is completely unnecessary when simply adding an item to the list. The reason for this is that simply by calling `SPList.Items`, the object model retrieves all items in the list at once. This is due to a choice by the object model developers to preload all items in the Items collection to ensure that code such as the following does not have to query the database on each loop iteration:

```
foreach(SPListItem item  in List.Items){
     //do some work
}
```

Although this is nice when dealing with smaller lists, it's almost never a good idea for large lists. Because the only way to create a new list item is to call the `Add()` method on a `SPListItemCollection`

www.it-ebooks.info

object, many programmers assume that you must call `SPList.Items.Add()`. Actually, you may retrieve an empty `SPListItemCollection` without querying the database by calling `SPList.GetItems()` and passing an `SPQuery` object designed to return no rows. So to add an item to a SharePoint list, use the following code instead:

```
SPListItem newItem = SPList.GetItems(new SPQuery(){Query = "0"}).Add();
```

This way, no matter how large your list becomes, you can always add new items without incurring a performance penalty that increases as the list grows.

In addition to adding items, SharePoint programmers sometimes make the mistake of looking up the number of items in the list using the following method:

```
SPList.Items.Count
```

As with the previous example, this code loads all items in the list before returning the count of those items. To avoid this, instead make use of the `.ItemCount` property:

```
int itemCount = SPList.ItemCount;
```

When querying list data, unless your code actually needs to iterate over every single item in the list, you should always use CAML queries to retrieve the list items you need. Just as with list views, if your query filters on an indexed field, it performs much better. Furthermore, by specifying an item limit with your query you can avoid running into query throttling limits.

If the list you try to query is extremely large (millions of rows), or you need to retrieve data from more than one list, your only option to efficiently retrieve data may be to make use of search. SharePoint search can effectively query millions of items and with high performance. The only drawback is that the search results are effectively always out of date as the data is retrieved from a crawled "snapshot" of the data, the search index. Keep in mind that this effect can become pronounced if search crawls are not performed frequently enough.

Search administration and configuration, as well as the full details about how to query search from code are outside the scope of this book, but assuming that your farm is set up with a basic search configuration and has completed at least one crawl, and assuming you have created a search scope called `"MyListItems"` that includes only items from the list you want to search, the following snippet queries that list for items matching the title `"MyTitle"`:

```
//required library references
using Microsoft.SharePoint;
using Microsoft.Office.Server.Search.Query;
using System.Data;

//create an spsite
using (SPSite site = new SPSite("http://localhost"))
{
    //create your query and set necessary properties
    KeywordQuery query = new KeywordQuery(site);
    query.QueryText = @"Title:""MyTitle""";
    query.ResultTypes = ResultType.RelevantResults;
    query.SelectProperties.Add("Id");
    query.SelectProperties.Add("Path");
```

www.it-ebooks.info

```
        query.SelectProperties.Add("Title");
        query.HiddenConstraints = @"scope:""MyListItems""";
        query.Execute();

        //Execute the query
        ResultTableCollection results = query.Execute();
        //retrieve the results table
        ResultTable resultsTable = results[ResultType.RelevantResults];
        //create an empty data table and load it with the search results
        DataTable dataTable = new DataTable();
        dataTable.Load(resultsTable, LoadOption.OverwriteChanges);

        foreach (DataRow row in dataTable.Rows)
        {
            //do something with the result
        }
    }
```

The previous code snippet makes use of the `KeywordQuery` class to query using search and mimics the search queries you would type into a site search box to find a particular item. You may also make use of the `FullTextSqlQuery` class as well if you prefer to use SQL full text query syntax to query items. Although searching list items using search is different from using the object model to query items, it can be extremely powerful if correctly used. If you do plan to make use of search to access list items or documents, understand that there is much to know about search before diving in. It will be worth it for you to spend the time required to understand how SharePoint search can be configured, including scopes, crawled properties, managed properties, and crawl details.

Lastly, when administering SharePoint sites (especially publishing sites), try to ensure that files and list items are not unnecessarily left checked out or unpublished. When files are checked out or unpublished, SharePoint must check the security settings of the current user to determine whether to display the unpublished or checked-out versions of those files. Because this security check requires additional server resources, it impacts performance. You may want to consider implementing a workflow or timer job that checks for unpublished or checked-out items and notifies users that these items should be published or checked in.

When a publishing page is in edit mode, you may access the unpublished items report for that page. The unpublished items report shows you any items that the page may be accessing that are unpublished. To access the unpublished items report, open the Page ribbon section at the top of the page while in edit mode; then click Draft Check to execute the report.

Batch Operations

Think about the last time you bought groceries. Hopefully, you didn't travel back and forth between your home and the grocery store to retrieve each of the items on your shopping list one by one. Instead you went to the store and bought all the items you needed at one time. This is known as a batch operation because you have batched together your items in a single trip.

Suppose that you have to perform a large number of inserts on a SharePoint list. The code that performs this operation executes from a web frontend server, calling the database as needed to insert data. Your code could perform a number of sequential operations, one after the other, to insert data into the list

www.it-ebooks.info

item by item. The problem with this approach is that each call to the database involves back-and-forth communication that is subject to latency, just like driving back and forth to the grocery store.

Assume you insert 1,000 items in a list; each database call introduces a latency of 1ms, and each insertion takes 1ms for the database to process. If you insert these items sequentially, it takes 2 seconds: 1,000ms of latency and 1,000ms of processing. If you batch these items together and make a single database call, it takes 1 second and 1ms: 1ms of latency and 1,000ms of processing. That's half the time! If these were the only operations your servers process, you could eliminate the need for half of your web frontends simply by batching these calls. Of course, this is a simplistic example, but the benefits of batching are real and noticeable in the real world.

SharePoint enables you to batch operations through the `SPWeb.ProcessBatchData()` method. This method accepts XML that describes the batch operations to be performed and returns XML that describes the result of those operations. You can configure the operations to stop executing if one fails, or to keep running through all operations regardless of success of previous operations. Because this method is defined within the SPWeb class, you may only batch operations across a single site at one time. Consequently, if you find that your code would benefit from batching operations on lists that exist in different sites, consider combining the lists into a single site.

Supported Batch Operations

The operations supported by the `ProcessBatchData()` method are those defined in the SharePoint Foundation RPC Protocol, the full details of which are available on MSDN. Supported operations include methods that perform the following operations:

- Create, retrieve, update, and delete:
  - Lists
  - Views
  - List Items
  - List Fields
- Miscellaneous operations:
  - Provision Site
  - Retrieve list views as XML or HTML
  - Create page
  - Export list schema
  - Hit Counter (used to track hits on a page)
  - Reorder List Fields

To perform a batch operation in SharePoint, you must first build the correct XML document to submit to the `ProcessBatchData()` method. That method returns an XML document containing results for each method executed. Your code should parse the resulting XML to determine whether

www.it-ebooks.info

errors occurred. The following code snippet demonstrates how to execute a batch method. It defines a function that inserts a list of strings into a particular list on a site:

```
private void BatchInsert(Guid siteGuid, List<string> stringsToInsert){

using (SPSite site = new SPSite(siteGuid)) {
     SPList list = site.RootWeb.Lists["Your List Name"];

     StringBuilder sb = new StringBuilder();


     //process batch data
     sb.Append(@"<?xml version=""1.0"" encoding=""UTF-8""?>
             <ows:Batch OnError=""Return""> ");


     foreach (string stringToInsert in stringsToInsert)) {
       string sMethod = string.Format(
       @"<Method>
        <SetList Scope=""Request"">{0}</SetList>
        <SetVar Name=""Cmd"">Save</SetVar>
        <SetVar Name=""ID"">New</SetVar>
        <SetVar Name=""urn:schemas-microsoft-com:office:office#Title"">{1}</SetVar>
       </Method>",
       list.ID.ToString(),
       stringToInsert
       );
       sb.Append(sMethod);
     }

     sb.Append(@"</ows:Batch>");

     //execute the batch request
     string resultXML = site.RootWeb.ProcessBatchData(sb.ToString());

     // Load the result XML and see if an error occurred
     XmlDocument doc = new XmlDocument();
     doc.LoadXml(resultXML);
     XmlNode errorNode = doc.SelectSingleNode("//ErrorText");
     string errorText = (errorNode != null) ? errorNode.InnerText ?? "": "";
     if (String.IsNullOrEmpty(errorText) == false){
          throw new ApplicationException("Error saving batch data: " + errorText);
     }
}
}
```

The preceding code snippet issues a number of `Save` commands, with the ID set to `New`. This is standard for creating new items in a SharePoint list. If the code were executing an `Update` command, the ID of the item to be updated would be supplied instead. The code only sets the `Title` field's value, referenced by the field's schema `(urn:schemas-microsoft-com:office:office)` and internal field name `(Title)` separated by `#`. Most list fields (including custom fields you create through code and the UI) share the same schema as the `Title` field shown here.

www.it-ebooks.info

Because the root `Batch` node in the submitted XML contains an `OnError` value of `Return`, the batch operation stops processing and returns when an error is encountered executing any of the methods in the batch. This value can also be set to `Continue` if you would like the batch operation to execute every method in the batch regardless of errors encountered. If you use `Continue`, you must check for multiple potential error messages in the result, not just for a single error message as in the previous snippet.

As with CAML list and site queries, all date and time values must be formatted using the `SPUtility.CreateISO8601DateTimeFromSystemDateTime()` function before being inserted into the XML stream. Whenever you perform numerous object model operations within a loop, consider whether it's possible to use the `SPWeb.ProcessBatchData()` method instead.

Multithreaded Operations

Another programming technique that can significantly improve the performance of your application is to make use of multiple threads for certain types of operations. By using multiple threads, you can exploit multicore server architectures to simultaneously process multiple streams of data.

Going back to the grocery shopping example, the analogy can be extended to include multithreading by imagining that you split your grocery list in half and have a partner shop for half the items while you shop for the other half. This won't help you get back and forth to the store any faster, but you will be done retrieving the items on your shopping list in half the time when at the store. If there are two checkout lines, you can check out twice as fast as well. As you can see from this example, the relative increase in performance from multithreading is dependent on the capacity of the system to simultaneously perform operations. With enough cores and sufficient disk throughput, multithreaded code can offer significant and noticeable increases in performance.

Remember the batch example in which you inserted 1,000 items into a list? Imagine that you split the 1,000 items to be inserted into 4 batches of 250 items each. Running each batch simultaneously could theoretically result in a total time of 251ms (1ms latency + 250ms operations — all simultaneous), assuming at least 4 cores and sufficient disk and network throughput on the SQL server. Theoretically, this is an 8-fold improvement over a sequential insert operation. Realistically, however, due to resource contention (such as SQL table locking, disk and network contention, and so on) you usually do not see such dramatic improvement in real-world applications. By testing your code with different numbers of threads (ideally by making this number configurable), you can determine the optimal number of threads to use when performing various operations.

The following code snippet demonstrates how you might perform multiple batches at the same time, making use of the `BatchInsert()` function defined previously to perform batches:

```
//number of threads is configurable
int numberOfThreads = 4;
//marking this volatile makes it thread safe so we can avoid locking
List<Thread> runningThreads = new List<Thread>();
//alter this code to retrieve a list from somewhere
List<string> stringsToInsert = … //get list of strings from somewhere

//calculate items per thread
int itemsPerThread = stringsToInsert.Count / numberOfThreads;

//set a counter to 0, this will be used to get ranges of strings from our list
```

www.it-ebooks.info

```
int currentIndex = 0;
for(int i = 0; i < numberOfThreads; i++){
     //on the last iteration, ensure the number of items to insert is correct
     if(i == numberOfThreads - 1){
          itemsPerThread = stringsToInsert.Count - currentIndex;
     }

     //get the current range to be executed in this thread
     List<string> range = stringsToInsert.GetRange(currentIndex, itemsPerThread);

     //increment the index for use in the next iteration
     currentIndex = currentIndex + itemsPerThread;

     //start the thread
     ThreadStart ts = new ThreadStart(delegate(){
          BatchInsert(siteGuid, range);
     });
     Thread thread = new Thread(ts);
     runningThreads.Add(thread);
     Thread.Start();
}

//wait for all threads to complete processing
foreach(Thread thread in runningThreads){
     thread.Join();
}
```

The preceding code snippet does not employ any sort of locking or thread synchronization beyond the final `Join()` calls to ensure that all threads have completed execution before the code completes. The `Join()` function pauses the current thread until the thread that the function is being called on completes. Although this example is simple, in the real world, multithreaded code can quickly become convoluted . For example, errors raised in a separate thread cannot be caught in other threads. Each thread must have its own error handling logic, often with some mechanism to report errors back to the main thread. Furthermore, you must take care that multiple threads do not attempt to operate on the same data structures at the same time. You can make use of the `lock()` statement to synchronize access to objects, but be careful not to create deadlocks that stop your code dead in its tracks and can be notoriously hard to debug. Although this chapter contains a few multithreaded examples, a full discussion of multithreading in .NET is outside the scope of this book. It is strongly recommended that you learn the intricacies of multithreaded programming and familiarize yourself with multithreading best practices before attempting to write multithreaded code for use in production systems.

In addition to speeding up certain operations, multithreading can also be used to speed up response times and improve the end users' experience. As seen in the custom-caching example shown previously, asynchronous operations were used to prevent a user request from waiting a long time for long running operations to complete. Executing long-running operations asynchronously from user requests allows those requests to complete faster without unnecessarily tying up resources such as the network connection. This is one of the reasons that Mac software is often perceived as extremely responsive: it is a rule in Mac software design that all UI interactions result in asynchronous execution of code, such that the UI never waits on backend code to render, resulting in a fluid UI that never appears to "stick."

www.it-ebooks.info

String Processing

Because much of the code executing on a web frontend builds HTML string to be sent to the client browsers, you need to ensure that string processing is as efficient as possible. The largest performance impact associated with string processing is because within .NET environments, strings are immutable. This means that when strings are created in memory, they cannot be changed. This is for good reason: A string is stored as an array of characters stored contiguously in memory and cannot grow in size without allocating a larger array and copying the data over. This means that each time a string is created, a specific amount of memory must be allocated for that string. If a string is altered in any way, the .NET runtime must allocate a new string and copy the resulting characters into it.

You need to be careful about unnecessarily creating new strings in memory. You can avoid this in a number of ways. The first is to avoid creating strings within loops. Consider the following code:

```
String stringToFind = "some value";
foreach(string stringToCheck in someStrings){
     if(stringToCheck.ToUpper() == stringToFind.ToUpper()){
          return true;
     }
}
```

In this case, for each loop iteration, the code creates two additional strings: one uppercase version of the string to check and one uppercase version of the string to find. You can certainly avoid half of these string creations by using the following code instead:

```
String stringToFind = "some value";
String stringToFindUpper = stringToFind.ToUpper();
foreach(string stringToCheck in someStrings){
     if(stringToCheck.ToUpper() == stringToFindUpper){
          return true;
     }
}
```

Although this is an improvement, you can go a step further if you make use of the `String.Equals()` function, with its support for case-insensitive comparisons, like so:

```
String stringToFind = "some value";
foreach(string stringToCheck in someStrings){
 if(stringToCheck.Equals(stringToFind, StringComparison.CurrentCultureIgnoreCase)){
      return true;
 }
}
```

By passing an `IgnoreCase` option to the `String.Equals()` function, you can avoid creating new strings. This version of the code runs much faster than the original. By keeping track of where new strings are created in your code, you can try to discover additional ways of doing without the creation of new strings if possible.

Because strings are immutable, and each time your code appends additional text to an existing string, a new string must be created. Consider the following code:

```
String s = "some text";
while(s.length < 100000){
```

www.it-ebooks.info

```
        s = s + "some additional text";
    }
    return s;
```

When running this code, subsequent loop iterations take longer and longer because larger and larger string allocations occur within them. To avoid this allocation problem, your code should make use of a `StringBuilder` whenever you build large strings. The `StringBuilder` object is provided to enable your code to build large strings in a way that avoids reallocating memory each time text is appended. To accomplish this, a `StringBuilder` maintains an internal character array of a fixed size. As strings are appended to the `StringBuilder`, they are copied into the internal character array. Only when the internal character array can no longer hold all the text being appended will the `StringBuilder` allocate a larger array and copy the existing characters over. This is why you have the option to provide the expected string length to the `StringBuilder` constructor: It enables the `StringBuilder` to pre-allocate a character array of that size, to avoid having to reallocate memory during subsequent append operations.

The following code demonstrates how to use a `StringBuilder` to accomplish the same function as previously shown:

```
StringBuilder sb = new StringBuilder(100020);
sb.append("some text");
While(sb.length < 100000){
     sb.Append("some additional text");
}
return sb.ToString();
```

This code runs many orders of magnitude faster than the first example because only a single string allocation occurs at the end of the snippet when `sb.ToString()` is called. Using a StringBuilder is not always faster than simply concatenating strings. When concatenating a small number of strings, `string.Concat()` often performs faster than using a StringBuilder. This is simply because the overhead of creating the StringBuilder can be greater than that of calling `string.Concat()`. Also, keep in mind that when concatenating strings within a single line of code, calling `string.Concat()` is not necessary. In this case, a line of code that concatenates strings using the + operator is automatically optimized to make use of `string.Concat()`. Consequently, the following two lines of code exhibit identical performance characteristics.

```
string test = "One" + "Two" + "Three";
string test = string.Concat("one", "two", "three");
```

Another method commonly used when processing strings is the `string.Format()` method. This method enables you to pass a format string along with a number of objects to be merged into the format string. Although this method can make your code much easier to read and modify, it does not perform particularly well. Actually, it can run many times slower than simply concatenating your values and surrounding strings together. This is because `FormatString()` must first parse the format string and allocate intermediate strings before it can build the output string. If your code absolutely needs to run as fast as possible, you may consider foregoing the syntactic benefits of the `string.Format()` function and using `string.Concat()` or a StringBuilder instead.

www.it-ebooks.info

Exception Handling

Throwing and catching exceptions can be expensive operations for the .NET CLR to execute. When possible, your code should avoid throwing unnecessary exceptions by checking for the conditions that would cause an exception before an exception occurs. In addition to better performance, limiting exceptions will allow debugging tools to more effectively track first-chance exceptions, as those exceptions will be more meaningful when they do not include unnecessary exceptions used to trap expected, preventable errors. Otherwise, a number of best practices should be followed in your error handling strategy, which is outside the scope of this book.

SPMonitoredScope

SharePoint 2010 introduces the `SPMonitoredScope` class, which enables you to monitor sections of code for performance and resource use. When blocks of your code are enclosed in a monitored scope, the `SPMonitoredScope` class logs usage statistics to the SharePoint ULS logs, as well as to the developer dashboard when enabled. You can make use of monitored scopes to ensure your code isn't using excessive resources or causing a bottleneck in some way, to determine which component on a page is slowing execution, or to be alerted when your code is running slowly or inefficiently.

Although this functionality is useful in development, it is also used to monitor performance issues in a production environment as well because monitored scopes can be configured with thresholds that when exceeded result in log entries being written to the ULS logs with a "high" trace level to alert administrators. Otherwise, the entries are written with a "verbose" trace level, which typically is not logged in a production system. In fact, almost all code in SharePoint out-of-the-box is enclosed in monitored scopes to log performance problems.

To make use of the `SPMonitoredScope` functionality to monitor sections of your custom code, simply wrap the blocks of code you want to monitor in the following way:

```
using(new SPMonitoredScope("Scope Name")){
     //your custom code here
}
```

When your wrapped code executes, statistics are written to the ULS logs and the developer dashboard. The developer dashboard provides a view of page execution details by reporting on all monitored scopes executed during page load, as well as thread, process, and machine-specific performance monitor values. The developer dashboard is only available to site owners by default and must be enabled to use. To enable the developer dashboard on all pages of your farm, execute the following `stsadm` command:

```
Stsadm.exe –o setproperty –pn developer-dashboard –pv on
```

To disable the developer dashboard entirely, execute the following `stsadm` command:

```
Stsadm.exe –o setproperty –pn developer-dashboard –pv off
```

Finally, to enable an `OnDemand` developer dashboard, where a site owner can turn the dashboard on or off for a particular page (instead of running the dashboard on every page), execute the following `stsadm` command:

```
Stsadm.exe –o setproperty –pn developer-dashboard –pv OnDemand
```

www.it-ebooks.info

The `OnDemand` setting is recommended because it enables site owners to allow the dashboard on a particular page by clicking an icon on the top right of the page. This enables temporary access to the dashboard on individual pages without running additional configuration commands, while also avoiding running the developer dashboard on every page of the site.

Because the developer dashboard is only available on SharePoint web pages, monitored scopes that run outside the context of a SharePoint web page (such as on the application server or within timer jobs) can write only to the ULS log for your farm.

General Tips for Solving Hard Problems

Whenever you run into a performance problem or optimization challenge, consider a number of angles to find and implement a solution. Start by asking yourself the following questions:

- Is the data stored in a way conducive to the type of processing that needs to occur?

  Consider using a different data structure to store your data while processing. Use timer jobs or an extract, translate, load (ETL) process to preprocess your data on a schedule for quick processing later. If the transformed data is to be stored in a database, consider loading the data into a separate database server so that the subsequent processing or retrieval of that data does not impact the source system.

- Is the in-memory data stored in an efficient data structure for the type of processing that needs to occur?

  Consider using hash tables, stacks, heaps, b-trees, linked lists, or other structures depending on the operations you need to perform on your data.

- Is your code performing I/O sequentially when it could be batching?

  Be sure to batch when possible, and consider multithreading if the conditions are conducive to improved performance with multiple threads.

- Is your code computing or retrieving the same thing more than once?

  Always avoid performing the same expensive operations over and over by caching the results of previous operations.

- Is your code waiting for I/O when it could be performing additional processing?

  Make use of multithreading to perform the additional processing while waiting for I/O to complete.

- Can you split the data or process execution across servers?

  Consider using a service-oriented architecture to distribute processing or data across systems. By sharding, or splitting your data across systems, where entities are stored in different locations according to some fixed identifier, you can distribute storage and retrieval of those entities across many servers.

- Are queries to a custom database taking too much time to execute?

  Make sure you index columns appropriately in your database. High-selectivity Columns used in `where` and `join` SQL clauses are good candidates to be indexed, but keep in mind

www.it-ebooks.info

that additional indexes slow down inserts and updates as the indexes must also be updated when data changes.

Consider splitting your database into multiple files across multiple drives to increase throughput. Even a single table can be split across drives using table partitioning, a feature available in the Enterprise Edition of SQL server.

- Is it OK if the outcome isn't "real time"?

  You may want to execute your process in the background or on a schedule and then save the result for quick access later.

- What is the hardware bottleneck your code is bounded by?

  There is always a limiting factor in code execution. By determining the resource (or lack thereof) that is responsible for most of the execution time, you can specifically target code or architectural improvements to reduce the use of that resource or improve the performance of that resource.

If you answered yes to any of these questions, you have an opportunity to improve the performance of your code as described. Keep a global view of your code in mind that includes how it physically operates within the computing resources available, with an eye out for how those resources could be more effectively used.

SUMMARY

Performance and scalability are wide-ranging topics that are encompassed in the entirety of processing on server systems, the networks they run on, and the clients that connect to them. Although this chapter has touched on many of these topics, it is by no means complete. If there were a single message that you should take from this chapter, it is this:

To optimize performance, you must understand how your code operates at the physical level of the system executing it because that is the level from which limitations in speed and scale arise.

Only by understanding the physical basis of your code and system behaviors can you make the most of your physical resources and reduce the fundamental bottlenecks in a system. Understanding how strings are allocated in memory physically, for example, is indispensible in writing string processing code that performs. Understanding that disk I/O can be expensive, but is also independent from network I/O on a physical basis, provides the ability to determine that multiple threads can be used to speed up the execution of your network and disk I/O code and make use of those resources in parallel.

Going further, you must also understand the implications of your design choices and the intricacies of the systems that interact with your designs. For example, understanding the nature of browser caching and cache directives sent in HTTP headers is necessary to optimize your applications for client-side caching and ensure that the system operates as expected, without your users experiencing stale content.

Keeping the concepts explained in this chapter in mind when architecting and developing systems can ensure that your applications perform well and have the capacity to scale to handle large data sets or large numbers of users.

www.it-ebooks.info

11

Building a Custom Global Navigation Solution

In SharePoint, every site collection has its own unique navigation structure. This can be desirable when each site collection is a distinct entity, but many times, it can be desirable to share navigation across site collections. This chapter uses a fictional company, ACME Corp, to help illustrate reasons why a global navigation solution might be implemented and to provide a set of requirements that the solution must deliver.

ACME Corp is a large, fictitious, multinational company that utilizes SharePoint 2010 for its intranet. Like many large companies, ACME's intranet is split into a number of different site collections. As is typical for many information architectures, the site collections have been created less because the content is truly meant to be separated and more to keep content database sizes manageable. ACME wants to share a standardized navigation menu across multiple site collections. The rest of this chapter is devoted to implementing such a solution.



The code for the solution created in this chapter is available for download at the book's companion Web site, `www.wrox.com`.

DEFINING THE REQUIREMENTS

Any navigation menu solution is expected to support certain basic functionality, such as the capability to display a hierarchal list of ordered links that supports an unlimited depth. In addition to standard menu functionality, ACME management has identified these additional requirements.

- **Requirement #1:** Scalability — ACME has approximately 10,000 employees, and if projections hold true, it expects to add another 5,000 employees within the next three years. As such, any solution implemented must be scalable and handle anticipated growth.

www.it-ebooks.info

- **Requirement #2:** Security trimming — ACME management is extremely security conscious and likes the security trimming that SharePoint does with its out-of-the-box (OOTB) navigation. As such, any solution delivered must also support security trimming.
- **Requirement #3:** Delegation of Administration — Another aspect ACME management likes with the OOTB SharePoint navigation is that site collection administrators or site owners can manage their own navigation menus. This functionality doesn't translate 100% to a global navigation solution, but management would like to allow site owners to enable/disable the global navigation, and it would also like to delegate administration of subsections of the menu to particular users.
- **Requirement #4:** Support for multiple types of menu items — The global navigation will primarily be made up of links to different sites within the corporate intranet, but some additional types of links have been identified as needing to be supported:
  - External URLs
  - Links to Document Libraries/Lists
  - Links to individual documents or list items
- **Requirement #5:** Maintain the look and feel of the OOTB — SharePoint Navigation Menu Management likes the way the OOTB SharePoint navigation menu appears and, if possible, would like to maintain its existing look and feel.
- **Requirement #6:** Support auto-population of menu items based on subsite or list content — ACME also likes the capability the OOTB menu has to automatically generating submenus and pages from a page library and would like the retain this capability.
- **Requirement #7:** Support multiple top-level menus and to specify menus used on a per-site collection basis. Although there is currently only the need for a single shared global navigation menu, it is anticipated that there will be a need for multiple global navigation menus, which can be shared across any number of site collections.

That's quite a list of requirements, and the road to get there is long and a little complicated. The solution will touch upon a large number of different aspects of SharePoint development from list definitions to timer jobs to custom actions. You might curse this fictitious ACME Corp, Microsoft, or Wrox once or twice before it's all over, but if you can fight through to the end, you'll have a solid navigation solution and hopefully will have picked up a thing or two that will be useful when developing future solutions.

REVIEWING THE ARCHITECTURE

The following section provides an overview of the architecture that will be used to implement the previous requirements. First, you need to understand how the OOTB navigation works in SharePoint to understand how you can plug a custom solution into the SharePoint model.

www.it-ebooks.info

Understanding OOTB SharePoint Navigation

Before trying to create the global navigation solution, it is helpful to understand how the OOTB SharePoint Navigation works. Although your first inclination might be to just rip out the SharePoint navigation components and replace them with your own, it is usually more beneficial in the long run to do things the "SharePoint Way." By playing nicely within SharePoint's box, you help ensure that your solution continues to work with (and take advantage of) future versions of SharePoint.

SharePoint leverages many standard ASP.NET components to provide its navigation functionality. The top navigation menu control that you see on a page is a control derived from the ASP.NET menu control that utilizes standard ASP.NET Site Map Providers (and derivatives) to provide the navigation structure.

As Figure 11-1 shows, the menu control retrieves the menu items it needs to render from another standard .NET class called a site map provider, of which SharePoint provides several custom implementations.

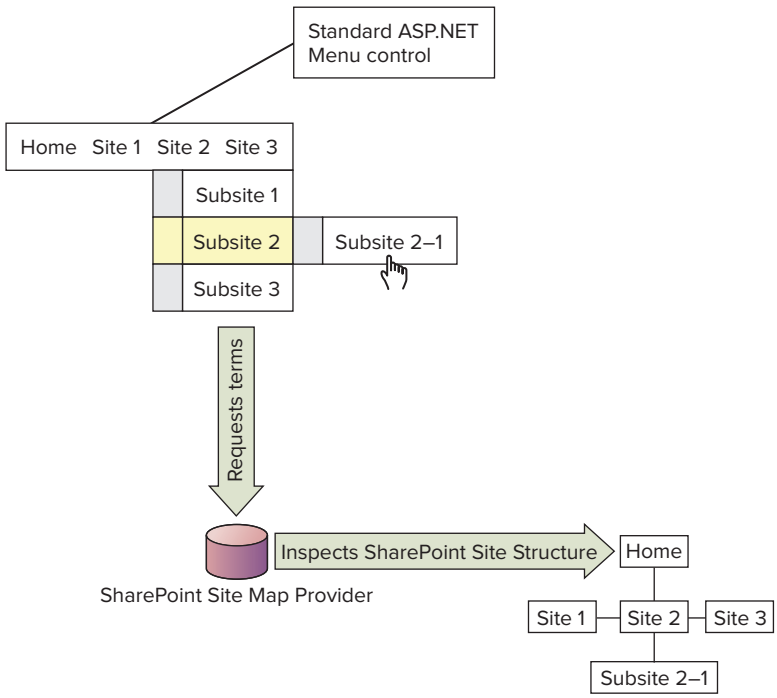


FIGURE 11-1: OOTB SharePoint navigation components

As a developer, you have the ability to replace these controls with your own custom implementations, which could enable you to change the rendering of the menu control or the menu items that will be rendered or both. In this case, ACME is happy with the look and feel of the navigation menu, so the work focuses on replacing the site map provider with your own implementation. Figure 11-2 displays where your piece fits in.

www.it-ebooks.info

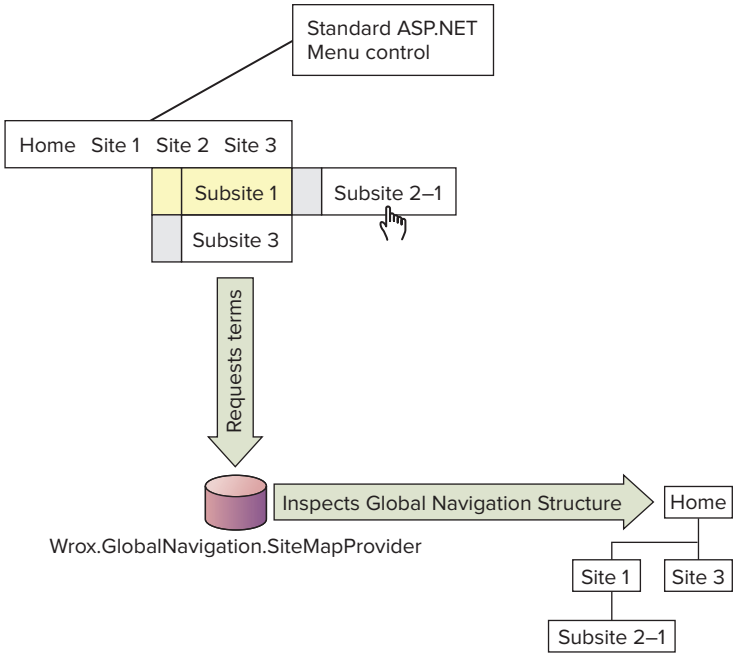


FIGURE 11-2: Where your solution fits in

The difference between Figure 11-1 and Figure 11-2 is subtle but is meant to show that your custom provider can provide an arbitrary site structure back to the menu control that deviates from the actual site structure shown in Figure 11-1.

Pieces of the Solution

As mentioned in the previous section, the heart of the global navigation solution lies with writing a custom site map provider, but several additional pieces of functionality require the solution to provide complete functionality.

Visual/UI Elements

The solution needs to provide several UI elements to enable administrators to manage the global navigation menu structures. It needs some sort of interface that enables designated administrators to maintain the menu structures and also needs some sort of interface that enables site collection administrators to specify which menu the site collection displays.

As far as the ability to maintain the navigation menu structures, following is the minimum functionality that any solution must allow:

- Adding/editing/deleting menus
- Adding/editing/deleting menu items
- Ordering menu items

www.it-ebooks.info

- Nesting menu items
- Delegate management of menus and submenus

This list of requirements sounds like a great fit for a SharePoint list.

The interface that must be provided to site collection administrators is significantly simpler that the maintenance UI and just needs to support these operations:

- Display a list of available menus.
- Allow administrators to pick from the list of menus.

Nonvisual Elements

Three nonvisual elements need to be written as part of the solution. The first is the aforementioned site map provider, which provides SharePoint's menu control with the custom navigation structure.

Second, there needs to be a mechanism to store the custom navigation structure. You could store the navigation structure in a variety of ways, such as in an XML file, some other text file, or an external database, but the most flexible option for ACME's purposes is to store the navigation structure within a custom SharePoint list.

Finally, two of ACME's requirements are around security trimming and auto-population of menu items. Security trimming simply means that only links a user has access to will be rendered within your navigation. An auto-population refers to the capability of the solution to automatically include links to subsites or list items when wanted.

At the simplest level, security trimming could be performed every time the site map provider returns a menu item. However, checking security is a rather expensive operation because it requires opening the SharePoint object the link is pointing to (a site, a list, and so on) and then checking the user's access. Auto-population comes with similar overhead.

Because one of the requirements is to deliver a solution that scales to tens of thousands of users, "expensive" operations are things that need to be minimized if at all possible. One thought to minimize the number of operations would be to cache individual user permissions, which would lessen the number of times an individual's security must be checked per visit. The downside with that approach is that the security information for every user needs to be stored in memory somewhere, and as the number of users increases, the memory footprint of this solution increases as well. Another downside of caching security credentials or site structure is that any changes made to a user's access or the site structure won't be reflected in the navigation provider until the cache expires.

The most scalable approach, and the one is used for this solution, is to create a timer job that iterates through the menu structure periodically and caches security information for every SharePoint object (rather than every end user) in the menu structure. The advantage here is that individual user permissions won't need to be cached, so the memory footprint can be minimal, but this comes at the expense of some small amount of security processing while rendering the menu. As with any caching solution, there is still the potential for cached security to become temporarily out of sync with actual security, but this is likely to be an infrequent occurrence, and SharePoint can still protect the content, so for ACME this is an acceptable trade-off.

www.it-ebooks.info

CREATING THE SOLUTION

The remainder of this chapter walks you through building out the solution. It starts with tackling the administration features, which include building out the list that stores the navigation structure and creating a mechanism for activating the menu on the sites; then it moves on to the navigation provider and other details around the timer job and caching.

Creating the Visual Studio Solution and Projects

The global navigation solution consists of a single SharePoint solution that contains three features. The first feature, Wrox.GlobalNavigation.Administration, is site collection scoped and provides the functionality around creating the global navigation list where the actual navigation structure is stored. It is typically activated on a single site collection within the web application. This feature also contains the timer job that periodically refreshes the cached structure.

The second feature, Wrox.GlobalNavigation, also is a site-collection-scoped feature and contains the functionality around displaying the custom menus on each site collection in the farm. It switches the Navigation Menu data source to point to the custom site map provider. Additionally, this feature provides a mechanism for site collection administrators to specify which navigation menu they want the site collection to use.

The final feature, Wrox.GlobalNavigation.SiteMapProvider, is a web-application-scoped feature that adds the custom SiteMapProvider to the list of available site map providers available to SharePoint. Additionally, there is some common code that is shared across the various bits of functionality, which assist in things such as retrieving your menu structure, and so on.

Now create the site structure. Use the following steps to create the solution:

1. Open up Visual Studio, and select File ➪ New ➪ Project.
2. Under Visual C# select SharePoint ➪ 2010 ➪ Empty SharePoint Project.
3. Specify the name of the project: Wrox.GlobalNavigation.
4. Click OK.
5. The SharePoint Customization Wizard displays, asking you to specify the URL of a local site to use for debugging and for the trust level for the SharePoint solution. Enter in the URL of your local SharePoint environment.
6. Select Deploy as a Farm Solution.
7. Click Finish.

There are a number of reasons the solution must be a farm-level solution rather than a sandboxed solution, but the two biggest ones are that the solution needs to make changes to the web application's `web.config`, and it needs to communicate outside of the site collection the solution runs in. Both of those types of operations are prohibited within sandboxed solutions.

A number of references need to be added to the project, so go ahead and get that out of the way:

1. Right-click References, and select Add Reference.

www.it-ebooks.info

2. Click the .NET tab, and add the following reference:

```
System.Configuration
```

To keep the code organized, some folders need to be created to contain the various bits of written code:

3. Right-click the project, and select Add ➪ New Folder.
4. For Name enter **Administration**.
5. Repeat this step and create these additional folders:

   Name: **Shared**

   Name: **SiteMapProvider**

   Name: **NavigationConsumer**

And finally, you need to deploy some files to various locations within SharePoint's 14 hives, so to do that you need to add some Mapped Folders:

6. Right-click the project, and select Add ➪ SharePoint Layouts Mapped Folder.
7. Right-click the project, and select Add ➪ SharePoint Mapped Folder ➪ Resources.

If you're following along, your solution should look like Figure 11-3.

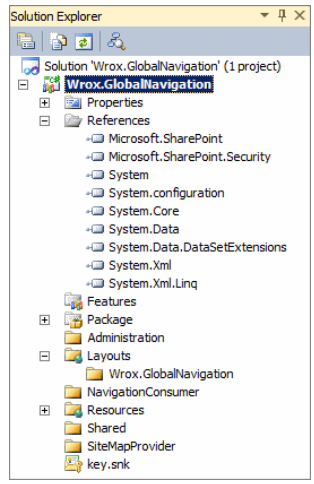


FIGURE 11-3: Solution structure

Hopefully the structure is self-explanatory, but to avoid any confusion, following is the breakdown of what will be going where:

- **Features:** The three features are going to be stored under the `Features` folder.
- **Administration:** The various administration components such as the List Definition appear here.
- **Layouts\Wrox.GlobalNavigation:** Any files that need to be deployed to SharePoint's 14 Hive `LAYOUTS` folder go here. This is where the page that enables the site collection administrators to select which global navigation menu to use goes.
- **NavigationConsumer:** This is where the code that is specific to swapping out the OOTB SharePoint site map provider goes. This is the functionality that will be enabled on each site collection that "consumes" the global navigation menus.
- **Resources:** Best practices dictate that a resource file is used for specifying things such as your list name, and so on. Resources enable you to use different text depending on the end user's culture. Implementation of the resource file is left as an exercise for you.
- **Shared:** Common code is located beneath the Shared folder.
- **SiteMapProvider:** The site map provider implementation code goes beneath here.

www.it-ebooks.info

Building the Navigation Menu UI

As stated earlier, there are actually just two identified pieces of functionality that require a user interface. The first is some sort of interface for maintaining the navigation menus and hierarchies, and the second needs to provide site collection administrators with the ability to choose which navigation menu they'd like to use.

Navigation menus are simply hierarchical structures of links that probably have some additional settings that can be set such as a Display Name or whether the administrator wants the link to be security trimmed. A sample navigation hierarchy might look something like this:

- Primary Global Navigation
  - Home
  - HR
    - Policies
    - Benefits
  - Payroll
    - Expense Report
    - Timesheet
  - IT
    - Technical Support FAQs
- Project Site Global Navigation
  - Home
  - Projects
    - <active project 1>
    - <active project 2>
    - <etc>
    - New Project Request

A variety of mechanisms and UIs could be developed to support the creation and maintenance of such a structure, but the simplest and most "SharePoint-y" implementation available is to use a SharePoint List to store and interface with the menu data. Two types of navigation data can be created within the UI. The first is a top-level Menu (Primary Global Navigation or Project Site Global Navigation from the previous example) and the second are Menu Items (Home, HR, and Policies from the previous example). Because a navigation menu structure is hierarchical, the two content types inherit from the Folder content type in SharePoint. This allows menu items to nest items beneath each other to reflect the menu structure.

The top-level Menu type is meant to support ACME's requirement around the ability to have multiple global navigation menus. The Menu type will allow specifics about the menu to be defined, such as menu name and description. Another benefit to use Folder as the base content type, beyond the

www.it-ebooks.info

ability to support nesting, is that Folders support breaking permission inheritance, which enables the solution to meet the requirement around delegating menu maintenance at a granular level.

Next, determine what type of information needs to be collected about the top-level menus. The information that needs to be captured around top-level menus includes the following (an asterisk [*] indicates a required field):

- Name of the menu *

  This is the name site collection administrators see when selecting which menu to use.

- Description of the menu

  This is an informational description of the menu, which might further help users pick the correct menu to use.

- Owner(s) of the menu

  This is purely informational in nature but is meant to provide site administrators a list of people they can go to to ask questions or request changes.

- Whether it is enabled *

  This is meant to allow menu administrators to work on a menu prior to making it visible to site collection administrators for selection.

Beneath this folder type, the hierarchical menu item structure is captured using a secondary content type called a Navigation Menu Item. This content type captures information about each individual menu item. The information that needs to be collected for menu items follows:

- Display Name *
- URL
- Link Type (choice)
  - SharePoint Site/Site Collection
  - SharePoint List
  - SharePoint List Item
  - External Link
- Tool Tip
- Security Trim
- Auto-Populate (choice)
  - Subsites
  - List Items
- Auto-Populate Depth
- Order
- Visible

www.it-ebooks.info

Most of those should be self-explanatory, but three of the fields, Security Trimming, Link Type, and Auto-Population, deserve a bit more explanation. One of ACME's requirements is around security trimming, which is the ability to restrict which menu items are visible to users based on their access to the target of the link. So for example, if a link on the menu points to a private subsite of the HR department that only HR employees have access to, ACME would like only HR employees to see that particular menu item.

As previously mentioned, security trimming can be computationally intensive, and there are no doubt links that all users have access to (the main portal home page, for example) or should see even if they don't have access (so that they can request access). Furthermore, the provider knows how to trim only SharePoint content, and the menu might include links to external systems. Because of this, security trimming is an optional configuration setting that should be used sparingly.

Auto-Population refers to the ability of the navigation provider to automatically generate child menu items based on the site structure of the target (in the case of a Site) or based on the contents of a list or document library. The Auto-Populate Depth refers to how deeply into the site structure the provider can delve. A setting of one would show only the immediate subsites beneath the target link, whereas a higher depth would build a deeper subsite structure. Like security trimming, auto-population can be computationally intensive, so it is important that it is used where appropriate.

Now that the types of information that need to be collected have been defined, it is time to create the list definition that defines the list and the types of content and information it supports.

Creating the List Definition

Chapter 5, "Custom Field Types, Content Types, and List Definitions," provides an in-depth treatment of content types and list templates, so the treatment here is minimal. The important thing to know about the code is that it defines a list template that contains two content types (Global Navigation Menu and Global Navigation Menu Item) which contain the fields previously discussed.

Following are the steps to create a list definition:

1. Right click the Administration Folder, and select Add ➪ New Item.
2. Select Visual C# ➪ SharePoint ➪ 2010 ➪ List Definition.
3. In the Name field, enter **GlobalNavigationMenuList**. The SharePoint Customization Wizard appears.
4. In the Display Name field, enter **Global Navigation Menu List**.
5. In the Type of List Definition field, enter **Custom List**.
6. Check the box next to Add a List Instance for This List Definition.
7. Click Finish.
8. Right-click the `ListInstance1` file, and select Rename.
9. In the Name field, enter **GlobalNavigationMenuListInstance**.

www.it-ebooks.info

Unfortunately, list definitions and instances require that a large amount of XML be written. There shouldn't be too many additional times in which there will be such a large information dump in the remainder of the chapter. The following steps lead you through the XML writing process.

1. Replace the existing `Elements.xml` (which should be open in VS) for the list definition with the following XML:

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
    <!-- Do not change the value of the Name attribute below. If it does not match
the folder name of the List Definition project item, an error will occur when the
project is run. -->
    <ListTemplate
        Name="GlobalNavigationMenuList"
        Type="10000"
        BaseType="0"
        OnQuickLaunch="TRUE"
        SecurityBits="11"
        Sequence="410"
        DisplayName="Global Navigation Menu List"
        Description="Stores menus used by the Wrox.GlobalNavigation solution."
        Image="/_layouts/images/itgen.png"/>
</Elements>
```

2. Open up the `Schema.xml` file beneath the GlobalNavigationMenuList, and replace the contents with this XML:




```
<?xml version="1.0" encoding="utf-8"?>
<List xmlns:ows="Microsoft SharePoint"
        Title="Global Navigation Menus List"
        EnableContentTypes="TRUE" DisableAttachments="TRUE"
        FolderCreation="FALSE" Direction="$Resources:Direction;"
        Url="Lists/Wrox.GlobalNavigation"
        BaseType="0" xmlns="http://schemas.microsoft.com/sharepoint/">
    <MetaData>
        <ContentTypes>
            <ContentType ID="0x012000c74015edf2a040da8b43a21ea788fde8"
                                    Name="Navigation Menu"
                                    Group="Wrox.GlobalNavigation"
                                    Description="Top Level Navigation Menu"
                                    Inherits="TRUE"
                                    Overwrite="TRUE"
                                    Version="0">
                <FieldRefs>
                    <FieldRef ID="{A47D2CD5-5025-45D8-9E55-2712DF58168D}"
                              Name="Title" DisplayName="Menu Name"/>
                    <FieldRef ID="{503f1caa-358e-4918-9094-4a2cdc4bc034}"
                              Name="Edit" DisplayName="Edit"/>
                    <FieldRef ID="{EBD9B2D9-3018-4BE4-BBE4-A00499DF8FC9}"
                              Name="Wrox_GlobalNavigation_Description"/>
                    <FieldRef ID="{1A5FA981-0201-4AC6-A764-C59ED8628AF0}"
                              Name="Wrox_GlobalNavigation_IsEnabled"/>
                    <FieldRef ID="{B4CC1D3A-05CB-450B-BAAF-12A4D7EC6F79}"
```

www.it-ebooks.info

```
                    Name="Wrox_GlobalNavigation_Owners"/>
            </FieldRefs>
        </ContentType>
        <ContentType ID="0x0120007FD79FB73E484A279578E643720665DD"
                                Name="Navigation Menu Item"
                                Group="Wrox.GlobalNavigation"
                                Description="A link on a menu"
                                Inherits="TRUE"
                                Overwrite="TRUE"
                                Version="0">
            <FieldRefs>
                <FieldRef ID="{A47D2CD5-5025-45D8-9E55-2712DF58168D}"
                          Name="Title" DisplayName="Display Name"/>
                <FieldRef ID="{CCF1A051-7925-45AA-8F63-AF8D83DDD2EB}"
                          Name="Wrox_GlobalNavigation_URL"/>
                <FieldRef ID="{6B464534-8AAC-498F-AAB7-67679700EBCA}"
                          Name="Wrox_GlobalNavigation_LinkType"/>
                <FieldRef ID="{4981B4F2-676B-43E6-994A-288CF956BE49}"
                          Name="Wrox_GlobalNavigation_ToolTip"/>
                <FieldRef ID="{43A43428-0011-482D-8B07-377D6AEAFCEB}"
                          Name="Wrox_GlobalNavigation_AutoPopulate"/>
                <FieldRef ID="{ACB649E3-61E8-4ACB-8948-554EAF16E8A8}"
                          Name="Wrox_GlobalNavigation_AutoPopulateDepth"/>
                <FieldRef ID="{F9EC848A-5A0E-4650-B36E-BC35A7EEE013}"
                          Name="Wrox_GlobalNavigation_SecurityTrim"/>
                <FieldRef ID="{CDDAF7D6-B493-407F-92C3-00B2422ABF7B}"
                          Name="Wrox_GlobalNavigation_Order"/>
                <FieldRef ID="{67D4AB24-417A-4CA2-8636-3285F4398FDA}"
                          Name="Wrox_GlobalNavigation_IsVisible"/>
            </FieldRefs>
        </ContentType>
    </ContentTypes>
    <Fields>
        <Field Name="Wrox_GlobalNavigation_Description"
                          ID="{EBD9B2D9-3018-4BE4-BBE4-A00499DF8FC9}"
                          Group="Wrox.GlobalNavigation" Type="Text"
                          DisplayName="Description"
                          ShowField="True"
                          ShowInDisplayForm="TRUE" ShowInNewForm="TRUE"
                          ShowInEditForm="TRUE" ShowInViewForms="TRUE"/>
        <Field Name="Wrox_GlobalNavigation_IsEnabled"
                          ID="{1A5FA981-0201-4AC6-A764-C59ED8628AF0}"
                          Group="Wrox.GlobalNavigation" Type="Boolean"
                          DisplayName="Enabled"
                          ShowInDisplayForm="TRUE" ShowInNewForm="TRUE"
                          ShowInEditForm="TRUE" ShowInViewForms="TRUE">
            <Default>1</Default>
        </Field>
        <Field Name="Wrox_GlobalNavigation_Owners"
                          ID="{B4CC1D3A-05CB-450B-BAAF-12A4D7EC6F79}"
                          Group="Wrox.GlobalNavigation"  Type="UserMulti"
                          DisplayName="Owners"
                          ShowInDisplayForm="TRUE" ShowInNewForm="TRUE"
                          ShowInEditForm="TRUE"  ShowInViewForms="TRUE"
```

www.it-ebooks.info

```
       Mult="TRUE"/>
<!-- Menu Items -->
<Field Name="Wrox_GlobalNavigation_Url"
       ID="{CCF1A051-7925-45AA-8F63-AF8D83DDD2EB}"
       Group="Wrox.GlobalNavigation"  Type="Text" DisplayName="Url"
       ShowInDisplayForm="TRUE" ShowInNewForm="TRUE"
       ShowInEditForm="TRUE"  ShowInViewForms="TRUE"
       />
<Field Name="Wrox_GlobalNavigation_LinkType"
       ID="{6B464534-8AAC-498F-AAB7-67679700EBCA}"
       Group="Wrox.GlobalNavigation" Type="Choice"
       DisplayName="Link Type"
       ShowInDisplayForm="TRUE"
       ShowInNewForm="TRUE"
       ShowInEditForm="TRUE"
        ShowInViewForms="TRUE">
    <CHOICES>
        <CHOICE>SharePoint Site</CHOICE>
        <CHOICE>SharePoint List</CHOICE>
        <CHOICE>SharePoint List Item</CHOICE>
        <CHOICE>External Link</CHOICE>
    </CHOICES>
    <Default>SharePoint Site</Default>
</Field>
<Field Name="Wrox_GlobalNavigation_ToolTip"
       ID="{4981B4F2-676B-43E6-994A-288CF956BE49}"
       Group="Wrox.GlobalNavigation"  Type="Text"
       DisplayName="Tool Tip"
       ShowInDisplayForm="TRUE" ShowInNewForm="TRUE"
       ShowInEditForm="TRUE"  ShowInViewForms="TRUE"
       />
<Field Name="Wrox_GlobalNavigation_AutoPopulate"
       ID="{43A43428-0011-482D-8B07-377D6AEAFCEB}"
       Group="Wrox.GlobalNavigation" Type="Boolean"
       DisplayName="Autopopulate"
       ShowInDisplayForm="TRUE" ShowInNewForm="TRUE"
       ShowInEditForm="TRUE" ShowInViewForms="TRUE">
    <Default>0</Default>
</Field>
<Field Name="Wrox_GlobalNavigation_AutoPopulateDepth"
       ID="{ACB649E3-61E8-4ACB-8948-554EAF16E8A8}"
       Group="Wrox.GlobalNavigation"  Type="Number"
       Decimals="0" Min="0" DisplayName="AutoPopulate Depth"
       ShowInDisplayForm="TRUE" ShowInNewForm="TRUE"
       ShowInEditForm="TRUE"  ShowInViewForms="TRUE"/>
<Field Name="Wrox_GlobalNavigation_SecurityTrim"
       ID="{F9EC848A-5A0E-4650-B36E-BC35A7EEE013}"
       Group="Wrox.GlobalNavigation"  Type="Boolean"
       DisplayName="Security Trim?" ShowInDisplayForm="TRUE"
       ShowInNewForm="TRUE" ShowInEditForm="TRUE"
       ShowInViewForms="TRUE"/>
<Field Name="Wrox_GlobalNavigation_Order"
       ID="{CDDAF7D6-B493-407F-92C3-00B2422ABF7B}"
       Group="Wrox.GlobalNavigation"  Type="Number" Decimals="0"
```

www.it-ebooks.info

```
            Min="0" DisplayName="Order"
            ShowInDisplayForm="TRUE" ShowInNewForm="TRUE"
            ShowInViewForms="TRUE"
            ShowInEditForm="TRUE" />
        <Field Name="Wrox_GlobalNavigation_IsVisible"
            ID="{67D4AB24-417A-4CA2-8636-3285F4398FDA}"
            Group="Wrox.GlobalNavigation"  Type="Boolean"
            DisplayName="Visible"
            ShowInDisplayForm="TRUE" ShowInNewForm="TRUE"
            ShowInEditForm="TRUE" ShowInViewForms="TRUE">
            <Default>1</Default>
        </Field>
    </Fields>
    <Views>
        <View BaseViewID="0" Type="HTML" MobileView="TRUE" TabularView="FALSE">
            <Toolbar Type="Standard" />
            <XslLink Default="TRUE">main.xsl</XslLink>
            <RowLimit Paged="TRUE">30</RowLimit>
            <ViewFields>
                <FieldRef Name="LinkTitleNoMenu"></FieldRef>
                <FieldRef Name="Wrox_GlobalNavigation_Description"/>
                <FieldRef Name="Wrox_GlobalNavigation_IsEnabled"/>
                <FieldRef Name="Wrox_GlobalNavigation_Owners"/>
                <FieldRef Name="Wrox_GlobalNavigation_Url"/>
                <FieldRef Name="Wrox_GlobalNavigation_LinkType"/>
                <FieldRef Name="Wrox_GlobalNavigation_ToolTip"/>
                <FieldRef Name="Wrox_GlobalNavigation_AutoPopulate"/>
                <FieldRef Name="Wrox_GlobalNavigation_AutoPopulateDepth"/>
                <FieldRef Name="Wrox_GlobalNavigation_SecurityTrim"/>
                <FieldRef Name="Wrox_GlobalNavigation_Order"/>
            </ViewFields>
            <Query>
                <OrderBy>
                    <FieldRef Name="Modified" Ascending="FALSE"></FieldRef>
                </OrderBy>
            </Query>
        </View>
        <View BaseViewID="1" Type="HTML" WebPartZoneID="Main"
              DisplayName="Menus"
              MobileView="TRUE" MobileDefaultView="TRUE"
              SetupPath="pages\viewpage.aspx"
              ImageUrl="/_layouts/images/generic.png" Url="Menus.aspx"
              ContentTypeID="0x012001"
              DefaultViewForContentType="TRUE"
              DefaultView="FALSE"
              >
            <Toolbar Type="Standard" />
            <XslLink Default="TRUE">main.xsl</XslLink>
            <RowLimit Paged="TRUE">30</RowLimit>
            <ViewFields>
                <FieldRef Name="LinkTitle"></FieldRef>
                <FieldRef Name="Edit"></FieldRef>
                <FieldRef Name="Wrox_GlobalNavigation_Description"/>
                <FieldRef Name="Wrox_GlobalNavigation_IsEnabled"/>
                <FieldRef Name="Wrox_GlobalNavigation_Owners"/>
```

www.it-ebooks.info

```
                </ViewFields>
                <Query>
                    <OrderBy>
                        <FieldRef Name="Modified" Ascending="FALSE"></FieldRef>
                    </OrderBy>
                </Query>
            </View>
            <View BaseViewID="2" Type="HTML" WebPartZoneID="Main"
                  DisplayName="Menu Items"
                  DefaultView="TRUE" MobileView="TRUE" MobileDefaultView="TRUE"
                  SetupPath="pages\viewpage.aspx"
                  ImageUrl="/_layouts/images/generic.png" Url="MenuItems.aspx"
                  IncludeRootFolder="FALSE"
                  >
                <Toolbar Type="Standard" />
                <XslLink Default="TRUE">main.xsl</XslLink>
                <RowLimit Paged="TRUE">30</RowLimit>
                <ViewFields>
                    <FieldRef Name="LinkTitle"></FieldRef>
                    <FieldRef Name="Edit"></FieldRef>
                    <FieldRef Name="Wrox_GlobalNavigation_Url"/>
                    <FieldRef Name="Wrox_GlobalNavigation_LinkType"/>
                    <FieldRef Name="Wrox_GlobalNavigation_ToolTip"/>
                    <FieldRef Name="Wrox_GlobalNavigation_AutoPopulate"/>
                    <FieldRef Name="Wrox_GlobalNavigation_AutoPopulateDepth"/>
                    <FieldRef Name="Wrox_GlobalNavigation_SecurityTrim"/>
                </ViewFields>
                <Query>
                    <OrderBy>
                        <FieldRef Name="Modified" Ascending="FALSE"></FieldRef>
                    </OrderBy>
                </Query>
            </View>
        </Views>
        <Forms>
            <Form Type="DisplayForm" Url="DispForm.aspx"
                  SetupPath="pages\form.aspx" WebPartZoneID="Main" />
            <Form Type="EditForm" Url="EditForm.aspx"
                  SetupPath="pages\form.aspx" WebPartZoneID="Main" />
            <Form Type="NewForm" Url="NewForm.aspx"
                  SetupPath="pages\form.aspx" WebPartZoneID="Main" />
        </Forms>
    </MetaData>
</List>
```

Wrox.GlobalNavigation\Wrox.GlobalNavigation\Administration\GlobalNavigationMenuList\Schema.xml

3. Open the `Elements.xml` file beneath the `GlobalNavigationMenuListInstance` and replace the XML with the following:

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <ListInstance Title="Global Navigation Menus"
                OnQuickLaunch="TRUE"
```

www.it-ebooks.info

```
                TemplateType="10000"
                Url="Lists/Wrox.GlobalNavigation"
                Description="Menus used by the Wrox.GlobalNavigation solution.">
    </ListInstance>
  </Elements>
```

That should be the end of the XML you need to write for the solution.

You may have noticed that Visual Studio automatically created a Feature beneath the Features folder when you added the list definition to the project. The next step is to give that feature a more descriptive name and properly set the properties of the feature.

1. Right-click the `Feature1` file beneath the Features folder and select Rename. In the Name field enter **Administration**.
2. Double-click the feature to open it up in Visual Studio, and change these settings:

   Title: Wrox.GlobalNavigation Administration

   Description: Creates a navigation menu list to manage global navigation menus.

   Scope: Site
3. Test the code by pressing F5 to run the solution.

If everything worked as expected, a browser window should display after the solution deploys, and you should see a link named Global Navigation Menus on your quick launch bar. Clicking that link should take you to the Global Navigation Menu list with two content types beneath the New Item menu. Figure 11-4 highlights these menu items.

When you are satisfied that the list was created properly, go ahead and stop debugging. By default, Visual Studio retracts the solution after you stop debugging.

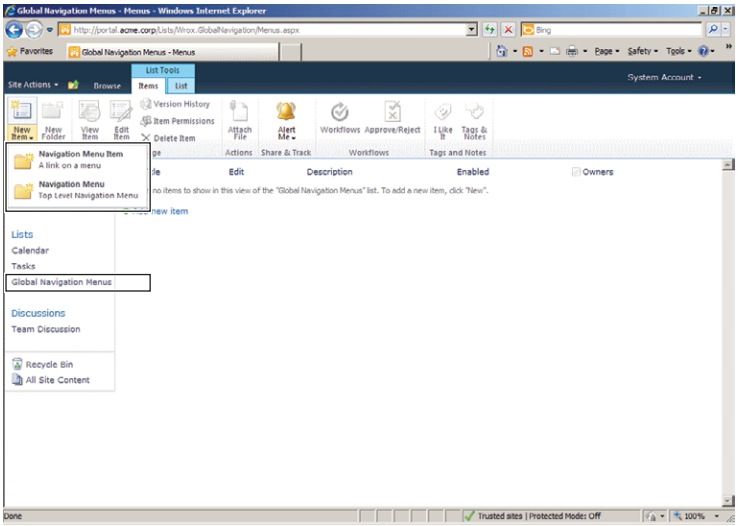


FIGURE 11-4: Menu items

www.it-ebooks.info

How to Turn Off Auto-Retraction of Solution

Auto-retraction of solutions can be desirable, but when you test code both in and out of the debugger, it makes life difficult. If you would like to disable auto-retraction, follow these steps.

1. Right-click the Wrox.GlobalNavigation project, and select Properties.
2. On the left side of the Properties window, click the SharePoint tab.
3. Uncheck the Auto-Retract After Debugging option, and save the project.

Fixing the New Item Menu

The astute observer should notice that the list doesn't enforce any sort of hierarchy on the content types, and you can create both a Menu and a Menu Item at the top level. Ideally, at the top level of the list you should create only items of type Navigation Menu, and in lower levels you would create only Navigation Menu Items.

Fortunately, SharePoint enables you to enforce this type of constraint, but unfortunately it must be done through code. And more complicated still, this code must be executed when the list is first created AND each time a new item is added to the list. This requires two separate pieces of code to handle the enforcement at the different levels. The first restricts the top level or root folder of the list to only new items of type Navigation Menu, and the second restricts any subfolder that gets created so that only Navigation Menu Items can be added.

Before you fix the content types, create a few helper classes to reduce code duplication.

Creating Your Lookup Class

Retrieving SharePoint items such as lists and fields depends on knowing either the name, the unique ID, or the index of the item in question. This can be difficult to remember and is prone to fat-fingering the values, so one strategy that can you can use is a strongly typed lookup class to provide quick access to the keys that will be used.

1. Right-click the Shared folder, and select New ➪ Class. In the Name field, enter **Lookup.cs**.
2. Replace the class definition with the following code:

```
public class Lookup
{
    public class ListNames
    {
        public static string GlobalNavigationList
        {
            get
            {
                return "Global Navigation Menus";
            }
        }
    }
}
```

You continue to add functionality to this class throughout the project, but for now that's all you need.

www.it-ebooks.info

Creating a List Helper Class

The code that needs to run to restrict the content types available at a particular level in the menu structure is the same but must be called both at list creation time and every time an item is added. Because of this, it is useful to create a helper class where this common code can be shared.

1. Right-click the Shared folder, and select New ➪ Class. In the Name field, enter **ListHelper.cs**.
2. Add the following `using` statement:

```
using Microsoft.SharePoint;
```

3. Replace the class definition with the following code:

```
public class ListHelper
{
    public static void RestrictAvailableContentTypes(SPFolder folder,
                          params string[] contentTypeNames)
    {
        if (folder != null)
        {
            List<SPContentType> contentTypes = new List<SPContentType>();
            SPContentTypeCollection availableContentTypes =
                folder.ParentWeb.Lists[folder.ParentListId].ContentTypes;
            foreach (string name in contentTypeNames)
            {
                contentTypes.Add(availableContentTypes[name]);
            }

            folder.UniqueContentTypeOrder = contentTypes;
            folder.Update();
        }
    }
}
```

The preceding code is meant to be a flexible method that takes in a folder object and a list of the names of content types, which should be added to the folder as allowable content types. It then finds the list-level content types that match the supplied names and sets those in the `UniqueContentTypeOrder` property of the `SPFolder` class. Only the content types in this property will be available to be created.

Create a Feature Receiver

Now that you've done a little bit of groundwork, it's time to fix the root-folder issue. You need to run the `RestrictAvailableContentTypes` method against the root folder after the list is created. Because the feature creates a list instance when activated, it makes sense to run the code as part of the `FeatureActivated` logic of the Administration feature.

1. Right-click the Administration Feature, and select Add Event Receiver. This creates and opens a file named `Administration.EventReceiver.cs`.
2. Add the following using statement to the class so that you can leverage the code you created earlier in the Shared folder.

```
using Wrox.GlobalNavigation.Shared;
```

www.it-ebooks.info

3. Within the class `AdministrationEventReceiver`, replace the body of the class with the following code.

```
public override void FeatureActivated(SPFeatureReceiverProperties
                                       properties)
{
    SPWeb web =  ((SPSite)properties.Feature.Parent).RootWeb;

    SPList globalNavigation =
                web.Lists[Lookup.ListNames.GlobalNavigationList];

    ListHelper.RestrictAvailableContentTypes(
        globalNavigation.RootFolder, "Navigation Menu");
}
```

4. Save the file.

The preceding code retrieves the global navigation menu list instance and restricts the available content types to just the Navigation Menu type.

Create an Event Receiver

This takes care of the root folder, but you still need to make all other subfolders of the list allow the use of the Navigation Menu Item type. To fix each individual menu item beneath the root folder, you need to create an Item Event Receiver, which triggers every time an item is created and which sets its available new items to only new menu items.

1. Right-click the Administration folder, and select New Item.
2. Select Visual C# ➪ SharePoint 2010 ➪ Event Receiver.
3. Set Name: NavigationMenuEventReceiver.
4. Click Add and the List Event Item Wizard appears.
5. Set Type of Event Receiver: List Item Events.
6. Set Event Source: Global Navigation Menu List.
7. Handle the following events:

   An item was added
8. Click Finish.
9. The `NavigationMenuEventReceiver.cs` file should be open in Visual Studio. If it isn't, open it.
10. Add the following include statements to the class so that you can leverage the code you created earlier in the Shared folder.

```
using Wrox.GlobalNavigation.Shared;
```

11. Replace the body of the `NavigationMenuEventReceiver` class with the following:

```
public override void ItemAdded(SPItemEventProperties properties)
{
```

www.it-ebooks.info

```
        base.ItemAdded(properties);
        SPListItem li = properties.ListItem;
        ListHelper.RestrictAvailableContentTypes(li.Folder,
                        "Navigation Menu Item");
    }
```

12. Save the file.

Just like with the feature receiver, the event receiver restricts the list of available content types to just Navigation Menu items every time an item is added to the navigation list. Run the project again (F5) and if everything works properly, the Global Navigation Menus list's New button has only a Navigation Menu option. If you create a Navigation Menu and then click into it, you can see only Navigation Menu Item as the only option beneath the New menu.

Preventing the Deployment Conflict Messages

You probably noticed that when you ran the solution this time, a Deployment Conflicts window appeared warning you that a list instance already exists and giving you the option to have Visual Studio automatically resolve the issue. The way Visual Studio automatically resolves this is to delete the list and then to re-create it. This can be a handy feature when you are constantly tweaking your list definition but quickly becomes annoying if you have entered data in the list because it will be deleted. Because you won't be making any further modifications to the list definition, you can disable the automatic resolution for the list.

1. Click the List Instance definition (Wrox.GlobalNavigation\Administration\GlobalNavigationMenuList\GlobalNavigationListInstance).
2. In the Properties window, change the Deployment Conflict Resolution to None.
3. Save the file.

Now when you deploy the solution, your list won't be deleted or re-created. Remember that if you do make any changes to your list definition, you need to either re-enable conflict resolution or manually delete the list from the site before you deploy it again if you want the changes to appear.

Congratulations! You finished the administrative piece of the global navigation solution. Before you move on to the other pieces, you need to create some test data in the menu list so that you have data to consume.

Create Sample Data

Sample data could be created via the list instance definition or by writing code, but in order to keep the solution simple, you will create the sample data manually.

1. Open up the site collection where you deployed the administration solution.
2. Click the Global Navigation Menus link on the Quick Launch.
3. Click the Add New Item link.

www.it-ebooks.info

4. Enter in the following information. (Leave any field not mentioned here blank.)

   Menu Name: Sample Menu

   Description: A sample menu used for testing.

   Enabled: true

5. Click Save.
6. Now click the Sample Menu item you just created to drill down into the menu.
7. Click the Add New Item.
8. Enter in the following information. (Leave any field not mentioned here blank.)

   Display Name: Home

   URL: <the site collection URL you are deploying the solution to>

   Tool Tip: Return home

   Visible: true

9. Click Save.
10. Click Add New Item again.

    Display Name: Pages

    URL: <a site collection URL>/SitePages/Forms/AllPages.aspx

    Link Type: SharePoint List

    Tool Tip: Our pages

    AutoPopulate: true

    AutoPopulate Depth: 1

    Visible: true

11. Click Save.

Building the Processing Layer

You now have a navigation structure that is stored in the SharePoint list, but the SharePoint list tells only half the story as far as the navigation structure is concerned, because the navigation hierarchy may also contain additional items not stored in the list (auto-populated items) and requires more information that is collected from the users (viewer permissions). You need to take the data stored in the list and process it further to handle the security trimming and the auto-population of subsites and list items.

When working with SharePoint list data, it can be helpful to work with strongly typed objects rather than the non-strongly-typed `SPListItem` object. Doing so helps detect coding errors at compile time and allows you to leverage IntelliSense to make writing the code more intuitive. So, before you can start processing the navigation structure, you are going to create a domain model that represents the in-memory menu data structure.

www.it-ebooks.info

Creating the Domain Model

This solution has a relatively simple domain model that consists of three classes (`MenuBase`, `Menu`, and `MenuItem`). `MenuBase` is an abstract base class that stores some of the common data shared across both `Menus` and `MenuItems`. There are also some enums that represent some of the menu item value options.

1. Right-click the Shared folder, and select Add ➪ New Folder. In the Name field, enter **DomainModel**.
2. Right-click the DomainModel folder, and select Add ➪ New Class. In the Name field, enter **AutoPopulateTypes.cs**.
3. Replace the generated class with the following code:

```
public enum AutoPopulateTypes
{
    None,
    Subsites,
    ListItems
}
```

4. Right-click the DomainModel folder, and select Add ➪ New Class. In the Name field, enter **LinkTypes.cs**.
5. Replace the generated class with the following code:

```
public enum LinkTypes
{
    Site,
    List,
    ListItem,
    External
}
```

6. Right-click the DomainModel folder, and select Add ➪ New Class. In the Name field, enter **MenuBase.cs**.
7. Replace the generated class with the following code:

```
public abstract class MenuBase
{
    public MenuBase()
    {
        Children = new List<MenuItem>();
    }

    public Guid Id { get; set; }
    public string Title { get; set; }
    public List<MenuItem> Children { get; set; }
    public MenuBase Parent { get; set; }
    private List<String> _accountsWithAccess = new List<string>();
    public bool HasUniquePermissions { get; set; }
    public bool SecurityTrim { get; set; }
```

www.it-ebooks.info

```
        public List<String> AccountsWithAccess
        {
            get
            {
                if (HasUniquePermissions || Parent == null)
                {
                    return _accountsWithAccess;
                }
                else
                {
                    return Parent.AccountsWithAccess;
                }
            }
        }
    }
```

As mentioned previously, the `MenuBase` class is a common base class shared by both the `Menu` and `MenuItem` classes. The class supports a hierarchical structure (the `Parent` and `Children` properties) as well as the capability to store the security permissions for the target object.

8. Right-click the DomainModel folder, and select Add ➪ New Class. In the Name field, enter **Menu.cs**.

9. Replace the generated class with the following code:

```
public class Menu : MenuBase
{
    public string Url { get; set; }
    public string Description { get; set; }
    public bool IsEnabled { get; set; }
    public List<String> Owners { get; set; }
}
```

The Menu class is basic and represents a top-level menu entry in your Navigation list.

10. Right-click the DomainModel folder, and select Add ➪ New Class. In the Name field, enter **MenuItem.cs**.

11. Replace the generated class with the following code:

```
public class MenuItem : MenuBase
{
    public string LinkUrl { get; set; }
    public LinkTypes LinkType { get; set; }
    public string ToolTip { get; set; }
    public bool AutoPopulate { get; set; }
    public int AutoPopulateDepth { get; set; }
    public int Order { get; set; }
    public bool IsVisible { get; set; }
}
```

12. Save the files.

The `MenuItem` class is similarly simple, and the fields correspond to the fields in your list.

www.it-ebooks.info

Enhancing the Lookup Class

As mentioned previously, the `Lookup` class provides named references to text-based configuration data that you must utilize to look up various items. Previously, you added an entry for your Global Navigation Menus list. Now that you are going to start reading in the menu structure to process it, you need to enhance the `Lookup` class to contain information about the fields contained within the list.

1. Under the Shared folder, open the `Lookup.cs` file.



2. Add the following code to the bottom of the `Lookup` class definition:

```
public class FieldGuids
{
    public class Menu
    {
        public static Guid Description
        {
            get
            {
                return new Guid("EBD9B2D9-3018-4BE4-BBE4-A00499DF8FC9");
            }
        }

        public static Guid IsEnabled
        {
            get
            {
                return new Guid("1A5FA981-0201-4AC6-A764-C59ED8628AF0");
            }
        }

        public static Guid Owners
        {
            get
            {
                return new Guid("B4CC1D3A-05CB-450B-BAAF-12A4D7EC6F79");
            }
        }
    }

    public class MenuItem
    {
        public static Guid LinkUrl
        {
            get
            {
                return new Guid("CCF1A051-7925-45AA-8F63-AF8D83DDD2EB");
            }
        }

        public static Guid LinkType
        {
            get
            {
                return new Guid("6B464534-8AAC-498F-AAB7-67679700EBCA");
```

www.it-ebooks.info

```
            }
        }

        public static Guid ToolTip
        {
            get
            {
                return new Guid("4981B4F2-676B-43E6-994A-288CF956BE49");
            }
        }

        public static Guid AutoPopulate
        {
            get
            {
                return new Guid("43A43428-0011-482D-8B07-377D6AEAFCEB");
            }
        }

        public static Guid AutoPopulateDepth
        {
            get
            {
                return new Guid("ACB649E3-61E8-4ACB-8948-554EAF16E8A8");
            }
        }

        public static Guid SecurityTrim
        {
            get
            {
                return new Guid("F9EC848A-5A0E-4650-B36E-BC35A7EEE013");
            }
        }

        public static Guid Order
        {
            get
            {
                return new Guid("CDDAF7D6-B493-407F-92C3-00B2422ABF7B");
            }
        }

        public static Guid IsVisible
        {
            get
            {
                return new Guid("67D4AB24-417A-4CA2-8636-3285F4398FDA");
            }
        }


    }
}
```

Wrox.GlobalNavigation\Wrox.GlobalNavigation\Shared\Lookup.cs

www.it-ebooks.info

This provides you with quick access to the various underlying GUIDs of the fields in the list. For example, if you need the GUID for the Order field, you can access it simply by typing **Lookup.FieldGuids.MenuItem.Order**. The GUIDS were all defined in the list definition schema.

Enhancing the ListHelper Class

The `ListHelper` class is meant to consist of common functionality related to SharePoint lists. One common need when reading list item data from SharePoint is taking the non-strongly-typed field data and casting it to the expected data type. Now add a few such helper methods to the `ListHelper` class.

1. In Visual Studio, under the Shared folder, open the `ListHelper.cs` file.



2. Add the following code to the end of the class definition:

```
public static SPFieldMultiChoiceValue GetMultiChoice(SPListItem item,
                        Guid fieldId)
{
    return new SPFieldMultiChoiceValue(
            item[Lookup.FieldGuids.MenuItem.AutoPopulate] as string);
}

public static bool GetBooleanValue(SPListItem item, Guid fieldId)
{
    return Convert.ToBoolean(item[fieldId]);
}

public static string GetStringValue(SPListItem item, Guid fieldId)
{
    return item[fieldId] as string;
}

public static int GetIntValue(SPListItem item, Guid fieldId)
{
    return Convert.ToInt32(item[fieldId]);
}

public static decimal GetDecimalValue(SPListItem item, Guid fieldId)
{
    return Convert.ToDecimal(item[fieldId]);
}
```

Wrox.GlobalNavigation\Wrox.GlobalNavigation\Shared\ListHelper.cs

The code is simplistic, and you might question the value of including some of it, but it helps to keep things consistent, and it is easier to remember to just use the `ListHelper` methods than to remember the nitty-gritty details of how to get particular types of data out of a list.

Creating the Repository Class

Now that you have the domain model and have improved some of the helper classes, you need a way to retrieve the underlying list data and return it as the domain objects you just built. The Repository

www.it-ebooks.info

Pattern is a good way to achieve this and also helps abstract away the underlying data source of the navigation menu (in this case a SharePoint list) that allows for greater flexibility in the future if you need to switch to a different data store such as a SQL database.

1. Right-click the Shared folder, and select Add ➪ New Class. In the Name field, enter **Repository.cs**.



2. Replace the generated code with the following code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Wrox.GlobalNavigation.Shared.DomainModel;

namespace Wrox.GlobalNavigation.Common
{
    public class Repository
    {
    }
}
```

3. Add the following method to the class definition to create a `Menu` class from an underlying `SPFolder` object:

```
private static Menu CreateMenu(SPFolder menuFolder, bool loadChildren)
{
    Menu menu = new Menu()
    {
        Id = menuFolder.UniqueId,
        Url = menuFolder.ParentWeb.Url,
        Title = menuFolder.Name,
        Description = ListHelper.GetStringValue(menuFolder.Item,
            Lookup.FieldGuids.Menu.Description),
        Owners = new List<string>(),
        IsEnabled = ListHelper.GetBooleanValue(menuFolder.Item,
            Lookup.FieldGuids.Menu.IsEnabled)
    };

    if (loadChildren)
    {
        foreach (SPFolder child in menuFolder.SubFolders)
        {
            menu.Children.Add(CreateMenuItem(child, menu));
        }

        //Order items based on Order column
        menu.Children =
            menu.Children.OrderBy(mi => mi.Order).ToList<MenuItem>();
    }

    return menu;
}
```

www.it-ebooks.info

The preceding method takes in two parameters: `menuFolder` is the `SPFolder` base object that represents your Menu, and `loadChildren` is a Boolean value that determines whether the `Repository` should load the child objects of the menu.

4. Implement similar methods for creating a `MenuItem` object. Unlike `Menus`, `MenuItems` also can be generated from sources other than an `SPFolder` when auto-population is enabled, so you have a few additional method implementations.

5. Add the following methods to the class definition:

```
private static MenuItem CreateMenuItem(SPFolder menuSource,
    MenuBase parent, bool loadSecurity)
{
    MenuItem menuItem = new MenuItem()
    {
        Id = menuSource.UniqueId,
        Title = menuSource.Name,
        LinkUrl = ListHelper.GetStringValue(menuSource.Item,
            Lookup.FieldGuids.MenuItem.LinkUrl),
        LinkType = GetLinkType(menuSource),
        ToolTip = ListHelper.GetStringValue(menuSource.Item,
            Lookup.FieldGuids.MenuItem.ToolTip),
        AutoPopulate = ListHelper.GetBooleanValue(menuSource.Item,
            Lookup.FieldGuids.MenuItem.AutoPopulate),
        AutoPopulateDepth = ListHelper.GetIntValue(menuSource.Item,
            Lookup.FieldGuids.MenuItem.AutoPopulateDepth),
        SecurityTrim = ListHelper.GetBooleanValue(menuSource.Item,
            Lookup.FieldGuids.MenuItem.SecurityTrim),
        Order = ListHelper.GetIntValue(menuSource.Item,
            Lookup.FieldGuids.MenuItem.Order),
        IsVisible = ListHelper.GetBooleanValue(menuSource.Item,
            Lookup.FieldGuids.MenuItem.IsVisible),
        Parent = parent
    };

    if (loadSecurity)
    {
        LoadAccountsWithAccess(menuItem);
    }

    if (menuItem.AutoPopulate && menuItem.AutoPopulateDepth > 0)
    {
        AutoPopulateItem(menuItem, loadSecurity);
    }

    foreach (SPFolder child in menuSource.SubFolders)
    {
        menuItem.Children.Add(CreateMenuItem(child, menuItem,
            loadSecurity));

        //Order items based on Order column
        menuItem.Children =
            menuItem.Children.OrderBy(mi => mi.Order).ToList<MenuItem>();
    }
```

www.it-ebooks.info

```
    return menuItem;
}

private static MenuItem CreateMenuItem(SPWeb menuSource,
    MenuBase parent, bool loadSecurity)
{
    MenuItem menuItem = new MenuItem()
    {
        Id = menuSource.ID,
        Title = menuSource.Title,
        LinkUrl = menuSource.Url,
        LinkType = LinkTypes.Site,
        ToolTip = menuSource.Description,
        AutoPopulate = (parent as MenuItem).AutoPopulate,
        AutoPopulateDepth = (parent as MenuItem).AutoPopulateDepth - 1,
        SecurityTrim = true,
        Order = 9999,
        IsVisible = true,
        Parent = parent
    };

    if (loadSecurity)
    {
        LoadAccountsWithAccess(menuItem);
    }

    if (menuItem.AutoPopulate && menuItem.AutoPopulateDepth > 0)
    {
        AutoPopulateItem(menuItem, loadSecurity);
    }

    return menuItem;
}

private static MenuItem CreateMenuItem(SPListItem menuSource,
    MenuBase parent, bool loadSecurity)
{
    MenuItem menuItem = new MenuItem()
    {
        Id = menuSource.UniqueId,
        Title = menuSource.DisplayName,
        LinkUrl = menuSource.Web.Url + "/" + menuSource.Url,
        LinkType = LinkTypes.ListItem,
        ToolTip = "",
        AutoPopulate = false,
        AutoPopulateDepth = 0,
        SecurityTrim = true,
        Order = 9999,
        IsVisible = true,
        Parent = parent
    };

    if (loadSecurity)
    {
        LoadAccountsWithAccess(menuItem);
```

www.it-ebooks.info

```
    }

    return menuItem;
}


public static void AutoPopulateItem(MenuItem item)
{
    if (item.AutoPopulate && item.AutoPopulateDepth > 0)
    {
        switch (item.LinkType)
        {
            case LinkTypes.Site:
                using (SPSite site = new SPSite(item.LinkUrl))
                {
                    using (SPWeb web =
                                site.OpenWeb(
                                item.LinkUrl.Replace(site.Url, "")))
                    {
                        foreach (SPWeb subsite in web.Webs)
                        {
                            item.Children.Add(
                               CreateMenuItem(subsite, item));
                        }
                    }
                }
                break;
            case LinkTypes.List:
                using (SPSite site = new SPSite(item.LinkUrl))
                {
                    using (SPWeb web = site.OpenWeb(
                            item.LinkUrl.Replace(site.Url, ""),
                            false))
                    {
                        SPList list =
                           web.GetList(item.LinkUrl.Replace(web.Url, ""));

                        foreach (SPListItem listItem in list.Items)
                        {
                            item.Children.Add(CreateMenuItem(listItem,
                                              item));
                        }
                    }
                }
                break;
        }
    }
}

private static LinkTypes GetLinkType(SPFolder menuItem)
{
    switch (ListHelper.GetStringValue(menuItem.Item,
                  Lookup.FieldGuids.MenuItem.LinkType))
```

www.it-ebooks.info

```
    {
        case "SharePoint Site":
            return LinkTypes.Site;
        case "SharePoint List":
            return LinkTypes.List;
        case "SharePoint List Item":
            return LinkTypes.ListItem;
        default:
            return LinkTypes.External;
    }
}
```

6. That takes care of the individual sources for the menu structure, but you still need some public methods that the other classes can call to retrieve the menus.

7. Add the following methods to the class definition:

```
public static Menu GetNavigationMenu(string siteUrl, Guid menuId)
{
    Menu menu = null;

    try
    {
        using (SPSite site = new SPSite(siteUrl))
        {
            SPWeb web = site.RootWeb;
            SPList navigationList =
              web.Lists[Lookup.ListNames.GlobalNavigationList];
            SPFolder menuFolder = navigationList.Folders[menuId].Folder;
            menu = CreateMenu(menuFolder, true);
        }
    }
    catch { }

    return menu;
}

public static List<Menu> GetNavigationMenus(string siteUrl,
                                            bool loadMenuItems)
{
    List<Menu> menus = new List<Menu>();

    using (SPSite site = new SPSite(siteUrl))
    {
        SPWeb web = site.RootWeb;
        SPList navigationList =
          web.Lists[Lookup.ListNames.GlobalNavigationList];
        menus = (from menu in
                 navigationList.RootFolder.SubFolders.OfType<SPFolder>()
                 select CreateMenu(menu, loadMenuItems)).ToList<Menu>();
    }

    return menus;
}
```

www.it-ebooks.info

These two methods retrieve an individual menu, given the URL and the ID of the menu, or can retrieve all menus.

Another potential need is going to be persisting certain properties at the site collection level. This will primarily be used by the Caching implementation.

8. Add the following methods to handle persisting of properties as well as several methods that will be used by the global navigation consumer to retrieve settings about the active menu.

```
public static void SaveKeyValuePair(SPSite site, string key, string value)
{
    SPWeb rootWeb = site.RootWeb;
    rootWeb.Properties[key] = value;
    rootWeb.Properties.Update();
}

public static string GetKeyValuePairValue(SPSite site, string key)
{
    SPWeb rootWeb = site.RootWeb;
    return rootWeb.Properties[key];
}

public static void SaveSiteCollectionNavigationSelection(SPSite site,
    string navigationSiteUrl, Guid menuId)
{
    site.RootWeb.Properties["__Wrox.GlobalNavigation.MenuUrl__"] =
        navigationSiteUrl;
    site.RootWeb.Properties["__Wrox.GlobalNavigation.MenuId__"] =
        menuId.ToString();
    site.RootWeb.Properties.Update();
}

public static string GetSiteCollectionNavigationSourceUrl(SPSite site)
{
    return GetKeyValuePairValue(site,
            "__Wrox.GlobalNavigation.MenuUrl__");
}

public static Guid GetSiteCollectionNavigationMenuId(SPSite site)
{
    return new Guid(
        GetKeyValuePairValue(site,"__Wrox.GlobalNavigation.MenuId__"));
}

public static Menu GetSiteCollectionNavigationSelection(SPSite site)
{
    Menu menu = null;

    string url = GetKeyValuePairValue(site,
        "__Wrox.GlobalNavigation.MenuUrl__");

    if (!String.IsNullOrEmpty(url))
    {
        Guid menuId = new Guid(
```

www.it-ebooks.info

```
                GetKeyValuePairValue(site,
                    "__Wrox.GlobalNavigation.MenuId__"));
            menu = GetNavigationMenu(url, menuId, false, false);
        }

        return menu;
    }
```

You are almost done with the `Repository` implementation, but there is one final bit of functionality that needs to be written. As mentioned toward the beginning of the chapter, security trimming is a relatively expensive operation, and to reduce the impact, you need to cache user access permissions ahead of time.

9. Add the following method to the `Repository` class:

```
public static void LoadAccountsWithAccess(MenuItem item)
{
    item.AccountsWithAccess.Clear();
    SPSecurableObject so = null;
    SPBasePermissions readPermissions = SPBasePermissions.EmptyMask;

    if (item.LinkType == LinkTypes.External)
    {
        return;
    }

    using (SPSite site = new SPSite(item.LinkUrl))
    {
        using (SPWeb web = site.OpenWeb(
          item.LinkUrl.Replace(site.Url, ""),
          item.LinkType != LinkTypes.Site))
        {
            switch (item.LinkType)
            {
                case LinkTypes.Site:
                    so = web;
                    readPermissions = SPBasePermissions.OpenItems;
                    break;
                case LinkTypes.List:
                    so = web.GetList(item.LinkUrl.Replace(site.Url, ""));
                    readPermissions = SPBasePermissions.OpenItems;
                    break;
                case LinkTypes.ListItem:
                    so = web.GetListItem(
                             item.LinkUrl.Replace(site.Url, ""));
                    readPermissions = SPBasePermissions.OpenItems;
                    break;
            }

            if (so != null)
            {
                if (!so.HasUniqueRoleAssignments)
                {
                    so = so.FirstUniqueAncestorSecurableObject;
```

www.it-ebooks.info

```
                    }

                    foreach (SPRoleAssignment ra in so.RoleAssignments)
                    {
                        foreach (SPRoleDefinition rd in
                                 ra.RoleDefinitionBindings)
                        {
                            if ((rd.BasePermissions & readPermissions) ==
                                 readPermissions)
                            {
                                SPPrincipal securityPrincipal = ra.Member;
                                if (securityPrincipal is SPUser)
                                {
                                    if (
                                        !item.AccountsWithAccess.Contains(
                                           ra.Member.LoginName))
                                    {
                                        item.AccountsWithAccess.Add(
                                           ra.Member.LoginName);
                                    }
                                }
                                else if (securityPrincipal is SPGroup)
                                {
                                    SPGroup group = (SPGroup)securityPrincipal;
                                    foreach (SPUser user in group.Users)
                                    {
                                        if (
                                            !item.AccountsWithAccess.Contains(
                                                    user.LoginName))
                                        {
                                            item.AccountsWithAccess.Add(
                                              user.LoginName);
                                        }
                                    }
                                }

                                break;
                            }
                        }
                    }
                }
                else
                {
                    item.HasUniquePermissions = false;
                }
            }
        }
    }
```

Wrox.GlobalNavigation\Wrox.GlobalNavigation\Shared\Repository.cs

The preceding method gets the underlying `SPSecurableObject` for the SharePoint object and then reads in all the security logins (both users and Active Directory groups) that have read access.

www.it-ebooks.info

Now build the solution just to ensure that everything compiles. There isn't much point in deploying the solution yet because you haven't added any additional functionality that you can see. It's time to move on to the parts of the solution that will activate on each site collection that is going to make use of the global navigation menu.

Building the Global Navigation Consumer

Now that you have a list to store the navigation structure in and a way to read in that data, it's time to create the other pieces of the solution that can make use of the navigation structure. Because you need to support multiple navigation menus, the first item you need is a mechanism to enable the site collection administrator to specify which navigation menu to use. The second major piece of functionality is the site map provider.

Navigation Menu Selector

The solution must provide a mechanism for specifying which navigation menu will be used on a per site collection basis. Like almost everything with SharePoint, this functionality can be delivered in a variety of ways, but the simplest to build is probably a custom action beneath the site collection settings menu that provides a site collection administrator with the ability to specify the source of the navigation menus and then allows them to pick from the list of available navigation menus.

To reduce complexity, the page provided here will be utilitarian; for an exercise, you can build a prettier UI. Figure 11-5 provides a representation of the utilitarian UI.

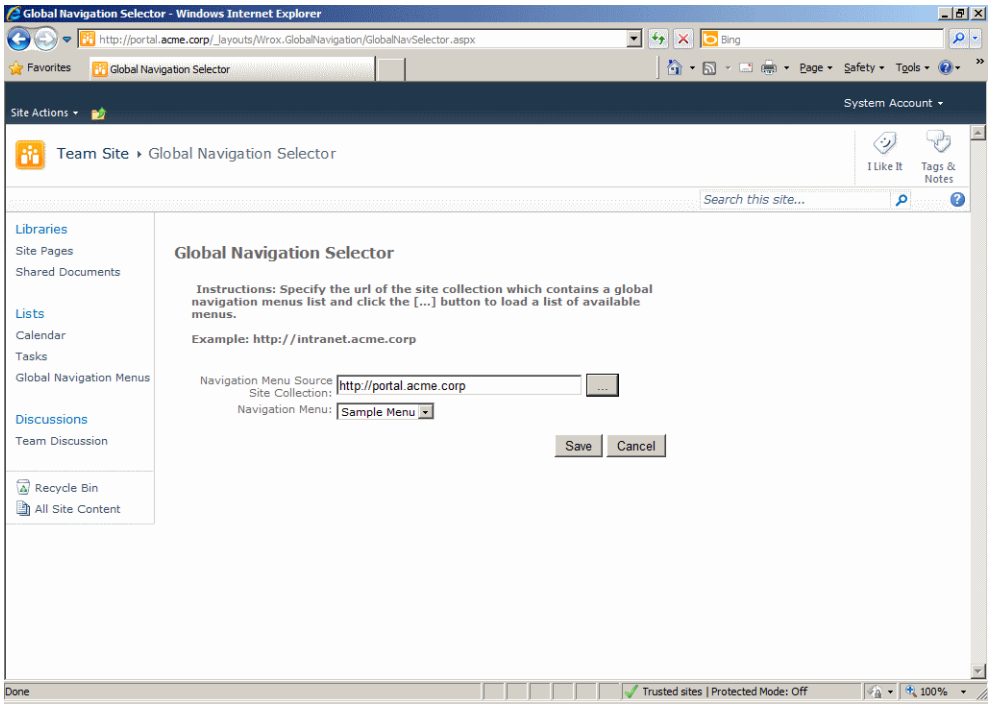


FIGURE 11-5: The Global Navigation Selector screen

www.it-ebooks.info

To create the Global Navigation Selector page, follow these steps:

1. Right-click the `Wrox.GlobalNavigation` project, and select Add ➪ New Item.
2. Select Visual C# ➪ SharePoint 2010 ➪ Application Page. In the Name field, enter **GlobalNavSelector.aspx**.
3. The `GlobalNavSelector.aspx` should be open in Visual Studio. If not, open it.



4. Replace the `PlaceHolderMain` content placeholder with the following code:

```
<asp:Content ID="Main" ContentPlaceHolderID="PlaceHolderMain" runat="server">
    <div style="padding: 20px">
        <div>
            <h2>
                Global Navigation Selector</h2>
        </div>
        <div style="padding-left: 15px; padding-top: 5px;">
            <table width="500px">
                <tr valign="top">
                    <td colspan="2">
                        <span style="font-weight: bold; margin: 5px; width: 98%;
                         text-align: left">
Instructions: Specify the url of the site collection which contains a
global navigation menus list and click the [...] button to load a list of
available menus.
                            <br />
                            <br />
                            Example: http://intranet.acme.corp<br />
                            <br />
                        </span>
                    </td>
                </tr>
                <tr valign="top">
                    <td style="width: 200px; text-align: right">
                        <asp:Label ID="lblNavigationMenuListUrl"
                         runat="server"
                         Text="Navigation Menu Source Site Collection:" /><br />
                    </td>
                    <td style="width: 300px">
                        <asp:TextBox ID="txtNavigationMenuListUrl" Width="250px"
                         runat="server" />
                        <asp:Button ID="btnRetrieveMenus" runat="server"
                         Text="..." OnClick="btnRetrieveMenus_Click" />
                    </td>
                </tr>
                <tr valign="top">
                    <td style="width: 200px; text-align: right">
                        <asp:Label ID="lblNavigationMenu" runat="server"
                         Text="Navigation Menu:" />
                    </td>
                    <td style="width: 300px">
                        <asp:DropDownList ID="ddlNavigationMenu"
                         AppendDataBoundItems="true" DataTextField="Title"
                         DataValueField="Id"  runat="server" />
```

www.it-ebooks.info

```
                    </td>
                </tr>
                <tr>
                    <td style="text-align: right" colspan="2">
                        <div style="margin-top: 10px;">
                            <asp:Button ID="btnSave" runat="server"
                             Text="Save" OnClick="btnSave_Click" />
                            <asp:Button ID="btnCancel" runat="server"
                             Text="Cancel" OnClick="btnCancel_Click" />
                        </div>
                    </td>
                </tr>
            </table>
        </div>
    </div>
</asp:Content>
```

This code creates a simple UI, which primarily consists of a textbox where users can specify the URL of a site collection that has the `Wrox.GlobalNavigation.Administration` feature enabled and then a drop-down that enables them to select from the available list items.



5. Replace the `PlaceHolderPageTitle` with the following code:

```
<asp:Content ID="PageTitle" ContentPlaceHolderID="PlaceHolderPageTitle"
 runat="server">
    Global Navigation Selector
</asp:Content>
```

6. Replace the `PlaceHolderPageTitleInTitleArea` with the following code:

```
<asp:Content ID="PageTitleInTitleArea"
 ContentPlaceHolderID="PlaceHolderPageTitleInTitleArea"
    runat="server">
    Global Navigation Selector
</asp:Content>
```

Wrox.GlobalNavigation\Wrox.GlobalNavigation\Layouts\Wrox.GlobalNavigation\GlobalNavSelector.aspx



7. Edit the `GlobalNavSelector.aspx.cs` file, and replace the contents with the following code:

```
using System;
using Microsoft.SharePoint;
using Microsoft.SharePoint.WebControls;
using Wrox.GlobalNavigation.Shared;
using System.Web.UI.WebControls;
using System.Collections.Generic;

namespace Wrox.GlobalNavigation.Layouts.Wrox.GlobalNavigation
{
    public partial class GlobalNavSelector : LayoutsPageBase
    {
        protected void Page_Load(object sender, EventArgs e)
        {
```

www.it-ebooks.info

```
    if (!IsPostBack)
    {

        Shared.DomainModel.Menu menu =
          Repository.GetSiteCollectionNavigationSelection(
                                    SPContext.Current.Site);
        if (menu != null)
        {
            txtNavigationMenuListUrl.Text = menu.Url;
        }

        RetrieveMenuItems();

        if (menu != null)
        {
            ddlNavigationMenu.SelectedValue = menu.Id.ToString();
        }
    }
}

protected void btnRetrieveMenus_Click(object sender, EventArgs e)
{
    RetrieveMenuItems();
}

protected void RetrieveMenuItems()
{
    ddlNavigationMenu.Items.Clear();
    ddlNavigationMenu.Items.Add(
      new ListItem("None", Guid.Empty.ToString()));
    ddlNavigationMenu.Enabled = false;

    if (!String.IsNullOrEmpty(txtNavigationMenuListUrl.Text))
    {
        List<Shared.DomainModel.Menu> menus =
            Repository.GetNavigationMenus(
              txtNavigationMenuListUrl.Text, true);

        ddlNavigationMenu.DataSource = menus;
        ddlNavigationMenu.DataBind();

        if (menus.Count > 0)
        {
            ddlNavigationMenu.Enabled = true;
        }
    }
}

protected void btnSave_Click(object sender, EventArgs e)
{
    Repository.SaveSiteCollectionNavigationSelection(
        SPContext.Current.Site,
        txtNavigationMenuListUrl.Text,
        new Guid(ddlNavigationMenu.SelectedValue));
```

www.it-ebooks.info

```
            Response.Redirect(
              SPContext.Current.Site.Url + "/_layouts/settings.aspx");
        }

        protected void btnCancel_Click(object sender, EventArgs e)
        {
            Response.Redirect(
              SPContext.Current.Site.Url + "/_layouts/settings.aspx");
        }
    }

}
```

Wrox.GlobalNavigation\Wrox.GlobalNavigation\Layouts\Wrox.GlobalNavigation\GlobalNavSelector.aspx.cs

The preceding code is relatively straightforward. On the initial page load, it checks to see if preferences have already been set and, if so, populates the various controls with the specified values. If the user changes the URL and clicks the Refresh button, the code retrieves a new list of menu items from the specified URL. Upon clicking Save, the selected values are saved to a property bag value on the site collection. The UI is as basic as possible to simplify the code, and as an exercise you can add additional functionality, such as validation/error messages, utilizing `.css` files rather than inline styles and otherwise enhancing the UI.

You now have a page that enables site collection administrators to specify the global navigation menu they should use, but you still need to provide them with a way to access it. Because setting the global navigation menu is an action performed for the entire site collection, it makes sense to access the page you've created from the Site Collection Administration menu found under Site Settings.

To create the Global Navigation Site Collection Administration action, follow these steps:

1. Right-click the NavigationConsumer folder, and select Add ➪ New Item.
2. Select Visual C# ➪ SharePoint 2010 ➪ Empty Element.

   Name: GlobalNavigationCustomAction
3. Next, let's create a new feature which will contain our custom action.
4. Right click on the Features folder and select Add Feature. This will create a new feature called `Feature1`.
5. Right click on the `Feature1` feature and rename the feature to: `NavigationConsumer`.
6. Now open up the feature by double-clicking on it and change these values:

   Title: Wrox.GlobalNavigation

   Description: Allows site collection to utilize a common navigation menu which can be shared among site collections.

   Scope: Site

   Add the GlobalNavigationCustomAction Item to the Items in the Feature box.

www.it-ebooks.info

7. Save the file and close it if desired.
8. Open the Administration feature. When you added our Empty Element, Visual Studio automatically added it as an item in the Administration feature. Remove it from the items list.
9. Open up the `Elements.xml` file found beneath our GlobalNavigationCustomAction folder.
10. Replace the generated code with:

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
    <CustomAction
       Id="GlobalNavigationSelector"
       GroupId="SiteCollectionAdmin"
       Location="Microsoft.SharePoint.SiteSettings"
       Sequence="10"
       Title="Configure Global Navigation" >
        <UrlAction
            Url="_layouts/Wrox.GlobalNavigation/GlobalNavSelector.aspx"/>
    </CustomAction>
</Elements>
```

11. Save the file.

Custom Actions, as the name implies, allow a developer to add custom actions or links to the various menus that SharePoint provides. In this case, you created a custom action which will appear under the Site Collection Administration menu and which points to the custom page you created in the previous section. Deploy the solution and visit the site. Click Site Actions | Site Settings and you should now see a Configure Global Navigation menu item toward the top of the Site Collection Administration section of the page, as shown in Figure 11-6.

When you click the link, you will be taken to the page you created and can select the navigation menu you want to use for the page. Go ahead and enter in the URL to the site collection the Administration component is deployed to, select your Sample Menu from the drop-down, and click Save.

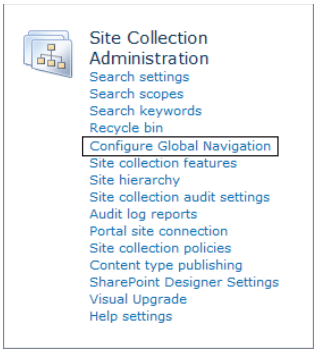


FIGURE 11-6: The Configure Global Navigation action

Creating the Navigation Cache

The site map provider is going to need access to the navigation menu structure, but you don't want it calling the `Repository` directly because the `Repository` is designed to reload the `Menu` structure every time it gets called, which involves the expensive process of retrieving user access permissions and auto-population of sites. Instead, you need to create a new class called `NavigationCache`, which is responsible for maintaining an in-memory cache of the navigation menu structure as well as persisting the generated menu structure.

1. Right-click the Shared folder, and select Add ➪ New Class. In the Name field, enter **NavigationCache.cs**.

www.it-ebooks.info



2. Replace the generated code with:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Wrox.GlobalNavigation.Shared.DomainModel;
using System.Xml.Serialization;
using System.Xml;

namespace Wrox.GlobalNavigation.Shared
{
    internal class NavigationCache
    {
        internal class CachedMenu
        {
            public DateTime CacheDate { get; set; }
            public Menu Menu { get; set; }
        }

        private static Dictionary<Guid, CachedMenu> _menus =
            new Dictionary<Guid, CachedMenu>();
        private static DateTime _cacheDate = DateTime.MinValue;

        public static void CacheMenus(SPSite site, List<Menu> menus)
        {
            foreach (Menu menu in menus)
            {
                CacheMenu(site, menu);
            }
        }

        public static void ClearMenuCache(SPSite site, string menuId)
        {
            string key = "__Wrox.GlobalNavigation.MenuStructure_" + menuId;
            Repository.SaveKeyValuePair(site, key + "_Timestamp",
                                        DateTime.Now.Ticks.ToString());
            Repository.SaveKeyValuePair(site, key, null);
            _menus.Remove(new Guid(menuId));
        }

        public static void CacheMenu(SPSite site, Menu menu)
        {
            CachedMenu cm = new CachedMenu() { Menu = menu };

            XmlSerializer ser = new XmlSerializer(menu.GetType(),
                                        new Type[] { typeof(MenuItem) });
            System.Text.StringBuilder sb = new System.Text.StringBuilder();
            System.IO.StringWriter writer = new System.IO.StringWriter(sb);
            RemoveParentReferences(menu);
            ser.Serialize(writer, menu);
            RestoreParentReferences(menu, null);
            string key = "__Wrox.GlobalNavigation.MenuStructure_" +
```

www.it-ebooks.info

```
                menu.Id.ToString();
    Repository.SaveKeyValuePair(site, key, sb.ToString());
    cm.CacheDate = DateTime.Now;
    Repository.SaveKeyValuePair(site, key + "_Timestamp",
                                cm.CacheDate.Ticks.ToString());
    _menus[menu.Id] = cm;
}

public static Menu GetMenu(SPSite site)
{
    Menu menu = null;

    Guid menuId = Repository.GetSiteCollectionNavigationMenuId(site);

    if (menuId != Guid.Empty)
    {
        DateTime cacheDate = GetMenuCachedDateTime(site.Url,
                                        menuId.ToString());

        CachedMenu cm = null;

        if (_menus.ContainsKey(menuId))
        {
            cm = _menus[menuId];
        }

        if (cm.CacheDate < cacheDate)
        {
            XmlSerializer ser = new XmlSerializer(typeof(Menu),
                                new Type[] { typeof(MenuItem) });
            XmlDocument doc = new XmlDocument();
            string value =
              Repository.GetKeyValuePairValue(site,
                          "__Wrox.GlobalNavigation.MenuStructure_" +
                          menuId.ToString());
            //if we have data, deserialize the menu structure
            if (!String.IsNullOrEmpty(value))
            {
                doc.LoadXml(value);
                XmlNodeReader reader =
                   new XmlNodeReader(doc.DocumentElement);
                menu = (Menu)ser.Deserialize(reader);
                RestoreParentReferences(menu, null);

                cm = new CachedMenu() { CacheDate = cacheDate,
                                        Menu = menu };
                _menus[menuId] = cm;
            }
            //Otherwise load the menu from the repository and cache it.
            else
            {
                string menuSource =
                  Repository.GetSiteCollectionNavigationSourceUrl(site);
                menu = Repository.GetNavigationMenu(menuSource,
```

www.it-ebooks.info

```
                        menuId, true, true);
                    CacheMenu(site, menu);
                }
            }
        }

        return menu;
    }

    public static DateTime GetMenuCachedDateTime(string siteCollectionUrl,
                                                 string menuId)
    {
        DateTime cacheDate = DateTime.MinValue;

        using (SPSite site = new SPSite(siteCollectionUrl))
        {
            string key = "__Wrox.GlobalNavigation.MenuStructure_" + menuId;
            string ticks = site.RootWeb.Properties[key + "_Timestamp"];
            if (!String.IsNullOrEmpty(ticks))
            {
                cacheDate = new DateTime(Convert.ToInt64(ticks));
            }
        }

        return cacheDate;
    }

    private static void RemoveParentReferences(MenuBase menu)
    {
        menu.Parent = null;

        foreach (MenuBase child in menu.Children)
        {
            RemoveParentReferences(child);
        }
    }

    private static void RestoreParentReferences(MenuBase menu, MenuBase parent)
    {
        menu.Parent = parent;

        foreach (MenuBase child in menu.Children)
        {
            RestoreParentReferences(child, menu);
        }
    }
  }
}
```

Wrox.GlobalNavigation\Wrox.GlobalNavigation\Shared\NavigationCache.cs

www.it-ebooks.info

The preceding class is responsible for retrieving generated menus from the cache as well as storing menus in the cache. If a menu doesn't exist in the cache, the `NavigationCache` uses the `Repository` to generate the menu and then caches it.

The cache stores the menu structures in both an in-memory dictionary as well as to the `Repository`. If you know your code needs to work only within a farm with a single WFE, you could simplify the solution and use only the in-memory dictionary, but because many environments have multiple WFEs, the solution utilizes both options.

The basic logic followed is that every time a menu is requested, the cache first checks to see when the most recent version of the generated menu was persisted to the `Repository`. If that version is more recent than the in-memory version, the menu will be reloaded from the cached version in the `Repository`. Otherwise, the in-memory version is used.

Building the Site Map Provider

Now that administrators have the ability to specify which menu they want to use in their site, the next step is to create the site map provider and wire it up to the navigation menu. Four pieces of functionality need to be written to accomplish this:

- Create the site map provider.
- Add a `web.config` setting to add the site map provider to the list of providers.
- Create a delegate control to point the navigation menu to the site map provider.
- Update the master page to increase the how far the menu will drill down.

The Site Map Provider

Create your site map provider class by following these steps:

1. Right-click the SiteMapProvider folder, and select Add ➪ Class. In the Name field, enter **SiteMapProvider.cs**.



2. Replace the generated code with the code for your provider.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Wrox.GlobalNavigation.Shared.DomainModel;
using Wrox.GlobalNavigation.Shared;
using System.Web;
using Microsoft.SharePoint;

namespace Wrox.GlobalNavigation.SiteMapProvider.SiteMapProvider
{
    public class SiteMapProvider : System.Web.SiteMapProvider
    {
        private Dictionary<string, MenuBase> MenuItemLocator =
                            new Dictionary<string, MenuBase>();
```

www.it-ebooks.info

```
protected void PopulateLocator(MenuBase menuItem)
{
    MenuItemLocator[menuItem.Id.ToString()] = menuItem;

    foreach (MenuBase child in menuItem.Children)
    {
        PopulateLocator(child);
    }
}

private Menu GetMenu()
{
    Menu menu = NavigationCache.GetMenu(SPContext.Current.Site);
    PopulateLocator(menu);
    return menu;
}

public override SiteMapNodeCollection GetChildNodes(SiteMapNode node)
{
    MenuBase navNode = MenuItemLocator[node.Key];

    SiteMapNodeCollection children = new SiteMapNodeCollection();

    foreach (MenuBase child in navNode.Children)
    {
        children.Add(CreateSiteMapNode(child));
    }

    return children;
}

protected override SiteMapNode GetRootNodeCore()
{
    SiteMapNode node = null;

    Menu rootItem = GetMenu();

    if (rootItem != null)
    {
        node = CreateSiteMapNode(rootItem);
    }
    else
    {
        node = new SiteMapNode(this, "noRootNode");
    }

    return node;
}

private SiteMapNode CreateSiteMapNode(MenuBase navNode)
{
    MenuItem mi = navNode as MenuItem;

    if (mi != null)
    {
```

www.it-ebooks.info

```
                return new SiteMapNode(this, mi.Id.ToString(), mi.LinkUrl,
                                       mi.Title, mi.Title);
            }
            else
            {
                return new SiteMapNode(this, navNode.Id.ToString());
            }

        }

        public override SiteMapNode GetParentNode(SiteMapNode node)
        {
            MenuBase navNode = MenuItemLocator[node.Key];
            SiteMapNode parentNode = null;

            if (navNode.Parent != null)
            {
                parentNode = CreateSiteMapNode(navNode.Parent);
            }

            return parentNode;
        }

        public override SiteMapNode FindSiteMapNode(string rawUrl)
        {
            return RootNode;
        }

        public override bool IsAccessibleToUser(HttpContext context,
                                                SiteMapNode node)
        {
            MenuBase navNode = MenuItemLocator[node.Key];
            if (navNode != null)
            {
                if (!navNode.SecurityTrim)
                {
                    return true;
                }

                foreach (string loginId in navNode.AccountsWithAccess)
                {
                    if (context.User.Identity.Name == loginId ||
                        context.User.IsInRole(loginId))
                    {
                        return true;
                    }
                }
            }

            return false;
        }
    }
}
```

Wrox.GlobalNavigation\Wrox.GlobalNavigation\SiteMapProvider\SiteMapProvider.cs

www.it-ebooks.info

The class inherits from the `SiteMapProvider` base class and overrides a number of required methods. If you build this solution on Microsoft Office SharePoint 2010 instead of SharePoint Foundation, you can instead make use of the `PortalSiteMapProvider`, which is part of the publishing infrastructure of SharePoint, but that is outside the scope of this chapter.

Web.config Modifications

The next step in wiring up the provider is to add an entry into the `web.config` for the web application. The entry should appear beneath System.Web/SiteMap/Providers and should look something like this:

```
<add name='Wrox.GlobalNavigation.Shared.SiteMapProvider'
 type='Wrox.GlobalNavigation.Shared.SiteMapProvider,
       Wrox.GlobalNavigation, Version=1.0.0.0, Culture=neutral,
       PublicKeyToken=cec0508c8dcccc76'
 NavigationType='Combined' EncodeOutput='true' SecurityTrim='true'/>
```

The `PublicKeyToken` value will be dependent on your signing file (`key.snk`). To find your public key token, open up a Visual Studio command prompt (Start ➪ Visual Studio 2010 ➪ Visual Studio Tools ➪ Visual Studio Command Prompt) and navigate to the `bin\Debug` directory of your solution. Then run the command: `Sn –T Wrox.GlobalNavigation.dll`, which will spit out the public key token. Note: Make sure you use a capital T because case matters.

It isn't a good idea to directly make changes to the `web.config` file in SharePoint, so instead the solution leverages the `SPWebConfigModification` class to handle making the `web.config` change for you. You need to do this utilizing a feature receiver, which adds the config entry anytime the feature is activated and removes it anytime the feature is deactivated.

`Web.config` modifications are scoped at the Web Application level, so you need a new feature, which is a Web Application feature.

1. Under the `Wrox.GlobalNavigation` project, right-click the Features folder, and select Add Feature. Make these selections:

   Title: Wrox.GlobalNavigation SiteMapProvider

   Description: Adds the Wrox.GlobalNavigation.SiteMapProvider to the list of site map providers available to SharePoint.

   Scope: WebApplication

2. Rename the feature from `Feature1` to SiteMapProvider.
3. Right-click the feature, and select Add Event Receiver.
4. Add the following using statement:

```
using Microsoft.SharePoint.Administration;
```



5. Replace the generated class with the following:

```
public class WroxGlobalNavigationEventReceiver : SPFeatureReceiver
{
    public override void FeatureActivated(
        SPFeatureReceiverProperties properties)
```

www.it-ebooks.info

```
    {
        SPWebApplication wa =
            properties.Feature.Parent as SPWebApplication;

        if (wa != null)
        {
            SPWebConfigModification mod = new SPWebConfigModification();
            mod.Path = "configuration/system.web/siteMap/providers";
            mod.Name = "add[@name='Wrox.GlobalNavigation.SiteMapProvider']";
            mod.Sequence = 0;
            mod.Owner = "Wrox.GlobalNavigation.SiteMapProvider";
            mod.Type =
          SPWebConfigModification.SPWebConfigModificationType.EnsureChildNode;
            mod.Value = "<add name='Wrox.GlobalNavigation.SiteMapProvider' " +
                "type='Wrox.GlobalNavigation.SiteMapProvider, " +
                "Wrox.GlobalNavigation, " +
                "Version=1.0.0.0, Culture=neutral, " +
                "PublicKeyToken=cec0508c8dcccc76' " +
                "NavigationType='Combined' EncodeOutput='true' " +
                "securityTrim='true'/>";
            wa.WebConfigModifications.Add(mod);

            SPWebService svc = wa.Farm.Services.GetValue<SPWebService>();
            svc.ApplyWebConfigModifications();
            wa.Update(true);

        }
    }

    public override void FeatureDeactivating(
           SPFeatureReceiverProperties properties)
    {
        SPWebApplication wa =
            properties.Feature.Parent as SPWebApplication;

        if (wa != null)
        {
            for (int i = wa.WebConfigModifications.Count - 1;
                 i >= 0; i--)
            {
                if (wa.WebConfigModifications[i].Owner ==
                    "Wrox.GlobalNavigation.SiteMapProvider")
                    wa.WebConfigModifications.RemoveAt(i);
            }

            SPWebService svc = wa.Farm.Services.GetValue<SPWebService>();
            svc.ApplyWebConfigModifications();
            wa.Update(true);
        }
    }
}
```

Wrox.GlobalNavigation\Wrox.GlobalNavigation\Features\SiteMapProvider\SiteMapProvider.EventReceiver.cs

www.it-ebooks.info

For those unfamiliar with using the `SPWebConfigModification` class, this probably seems like a lot of code just to add a simple `web.config` entry, but by using the `SPWebConfigModification` class, you ensure that the entry will be added to all the existing web frontends in the farm as well as any new WFEs that might be added in the future.

Neither of your other two features can work unless the `Wrox.GlobalNavigation.SiteMapProvider` feature has been activated for the web application. Now enforce that relationship by adding a feature dependency to the other two features.

6. Open the Administration feature, and expand the Feature Activation Dependencies section.
7. Click Add, and select the `SiteMapProvider` feature as a dependency.
8. Open the NavigationConsumer feature, and expand the Feature Activation Dependencies section.
9. Click Add, and select the `SiteMapProvider` feature as a dependency.

Figure 11-7 provides a screenshot of what adding a feature dependency in Visual Studio looks like.

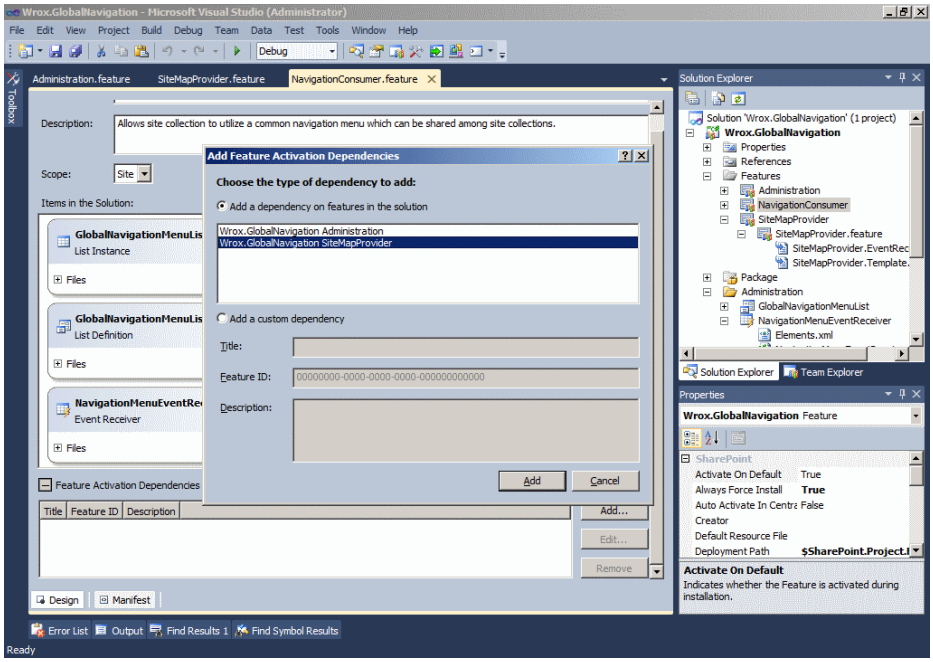


FIGURE 11-7: Adding Feature Dependencies

Creating the Control Delegate

Now that you have the site map provider and have it added to the solution, you need to make the SharePoint navigation menu use your site map provider rather than the default SharePoint provider. The site map provider used by the navigation menu is defined in the master page in something called

www.it-ebooks.info

a Control Delegate. Control delegates are basically sections on the master page that can be overridden by your own implementation.

1. Right-click the NavigationConsumer, and select Add ➪ New Item.
2. Select Visual C# ➪ SharePoint ➪ 2010 ➪ Empty Element. In the Name field, enter **TopNavigationDataSourceDelegate**.
3. Replace the generated code of the `Elements.xml` file with the following:

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
    <Control Id="TopNavigationDataSource" Sequence="25"
   ControlClass="System.Web.UI.WebControls.SiteMapDataSource"
   ControlAssembly="System.Web, version=2.0.3600.0, Culture=neutral,
   PublicKeyToken=b03f5f7f11d50a3a">
        <Property Name="SiteMapProvider">
            Wrox.GlobalNavigation.SiteMapProvider
        </Property>
        <Property Name="EnableViewState">true</Property>
        <Property Name="StartFromCurrentNode">false</Property>
        <Property Name="StartingNodeOffset">0</Property>
        <Property Name="ShowStartingNode">false</Property>
        <Property Name="ID">topSiteMap</Property>
    </Control>
</Elements>
```

4. Double-click the NavigationConsumer feature, and add the `TopNavigationDataSourceDelegate` item to the feature.
5. Save everything and deploy.

If everything works as expected, the navigation menu should be completely empty on your site (assuming you haven't already specified a navigation menu using the navigation selector UI) as reflected in Figure 11-8.

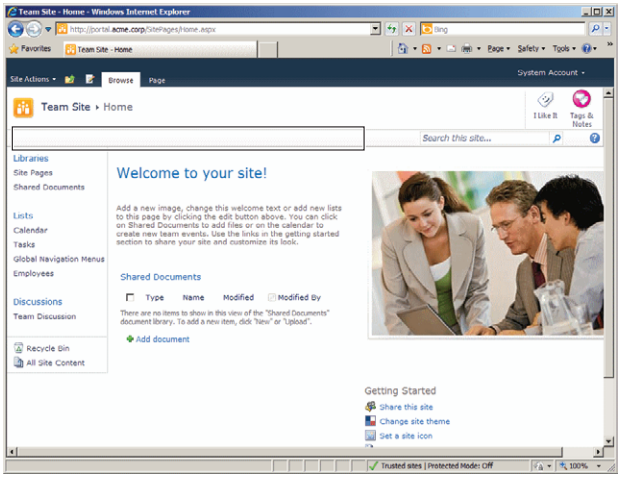


FIGURE 11-8: The Empty Navigation menu

www.it-ebooks.info

Now, go to the Global Navigation Selector (Site Actions ➪ Site Settings ➪ Site Collection Administration ➪ Configure Global Navigation) and specify the Sample Menu you created earlier in the chapter. As soon as you click Save, you should see your custom navigation links as reflected in Figure 11-9.

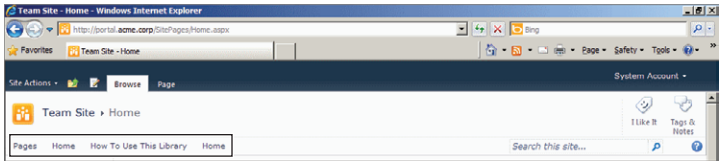


FIGURE 11-9: The populated Navigation menu

If the site you are building on is based on the Team Site template and you followed the instructions when you built the Sample Menu entry, you should see links similar to what is shown in Figure 11-9. The Pages and Home links that you manually added beneath the Sample Menu item should be visible, and you should also see the two default pages that exist in the Site Pages library (because you checked the Auto-Populate option for your Pages link). So the links displayed are correct, but the flat nature of the links is probably not what you were expecting. Ideally, you would expect the How to Use This Library and the Home link to appear beneath the Pages link.

This behavior is because several aspects of the menu rendering are controlled by properties set on the ASP.NET Menu Control within the master page. To get the navigation control to render in the nested format expected, you need to modify those properties on the master page.

1. Click Site Actions ➪ Site Settings ➪ Galleries ➪ Master Pages.
2. Download a copy of the v4.master page using either the Ribbon or the context menu. Save the file as v4GlobalNavigation.master.
3. In Visual Studio, open the master page.
4. Starting on line 338, you should see the following `AspMenu` control definition:

```
<SharePoint:AspMenu
    ID="TopNavigationMenuV4"
    Runat="server"
    EnableViewState="false"
    DataSourceID="topSiteMap"
    AccessKey="<%$Resources:wss,navigation_accesskey%>"
    UseSimpleRendering="true"
    UseSeparateCss="false"
    Orientation="Horizontal"
    StaticDisplayLevels="2"
    MaximumDynamicDisplayLevels="1"
    SkipLinkText=""
    CssClass="s4-tn" />
```

You need to be concerned with two properties. The property that makes the two levels of the menu collapse into a single level is the `StaticDisplayLevels` property.

www.it-ebooks.info

5. Change this from 2 to 1.

   The other property you need to change is the `MaximumDynamicDisplayLevels` property. This property affects how deeply the menu control expands. Keeping this setting to a low number enables you to enforce a maximum depth to your navigation menu, but in ACME's case this needs to be set to a sufficiently high number.

6. Change `MaximumDynamicDisplayLevels` from 1 to 10.

   One additional item you might want to change, but is optional, is the `PlaceHolderTopNavBar` Content Place Holder control. Most pages on the site respect the changes you've made to the `AspMenu` control, but some pages such as the Site Settings page use this placeholder control to replace the menu control with your changed properties. To prevent this from happening, you can move the `AspMenu` control outside of this placeholder and then enclose the placeholder within a `<div style="display:none">` tag. This change may have unintended consequences because some pages in SharePoint may depend on overriding the menu control.

7. Save the master page and upload it back to the master page gallery.

8. Set your new master page as the default master page using either the SharePoint interface (if available) or SharePoint Designer.

Now when you view your site, you should see the menu properly nesting the menu structure, as shown in Figure 11-10.

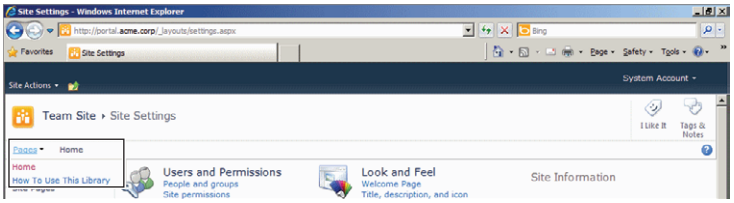


FIGURE 11-10: The Navigation menu nested

There is one final task to be tackled, and that is to implement the timer job which will periodically refresh the cached menu structure so that changes in the menu, the site structure, and in security will be reflected in the menu structure.

Building the Timer Job

The steps to create a timer job are twofold. The first step is to create the timer job class and the second step is registering the timer job within SharePoint so that it executes on a periodic schedule. To create the timer job, follow these steps:

1. Right-click the Administration folder, and select Add ➪ Class.

2. In the Name field, enter **SiteMapBuilderTimerJob.cs**.

www.it-ebooks.info



3. Replace the generated code of the file with the following:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint.Administration;
using Microsoft.SharePoint;
using Wrox.GlobalNavigation.Shared.DomainModel;
using Wrox.GlobalNavigation.Shared;
using System.Text.RegularExpressions;

namespace Wrox.GlobalNavigation.Administration
{
    public class SiteMapBuilderTimerJob : SPJobDefinition
    {
        private string _siteCollectionUrl = null;

        protected string SiteCollectionUrl
        {
            get
            {
                if (_siteCollectionUrl == null)
                {
                    Regex regex = new Regex(@".*? \((.*?)\)");
                    Match match = regex.Match(this.Title);
                    if (match.Success && match.Groups.Count > 1)
                    {
                        _siteCollectionUrl = match.Groups[1].Value;
                    }
                }

                return _siteCollectionUrl;
            }
        }

        public SiteMapBuilderTimerJob() :
            base()
        {
        }

        public SiteMapBuilderTimerJob(string jobName, SPService service,
            SPServer server, SPJobLockType targetType)
            : base(jobName, service, server, targetType)
        {
        }

        public SiteMapBuilderTimerJob(string jobName, SPSite site) :
            base(jobName, site.WebApplication, null, SPJobLockType.Job)
        {
            this.Title = jobName;
        }

        public override void Execute(Guid targetInstanceId)
```

www.it-ebooks.info

```
            {
                SPSite site = WebApplication.Sites[SiteCollectionUrl];

                List<Menu> menus = Repository.GetNavigationMenus(site.Url, true, true);

                NavigationCache.CacheMenus(site, menus);
            }

        }
    }
```

Wrox.GlobalNavigation\Wrox.GlobalNavigation\Administration\SiteMapBuilderTimerJob.cs

There is various plumbing code involved within the timer job, but the meat of the timer job is located in the `Execute` method. There you are leveraging the `Repository` and `NavigationCache` classes to periodically rebuild the cached menus.

4. Save the file.

Next you will need to register the timer job so that it executes on a periodic interval. This is typically done within a feature receiver of the feature, which will include the timer job. The timer job is tied to the menu list, which is part of the Administration feature, so that is the feature receiver which will have the feature. Perform the following steps to implement the feature receiver:

5. Open up the Administration Feature Event Receiver which was created earlier in the chapter.
6. Add the following `using` statements to the file:

```
using Wrox.GlobalNavigation.Administration;
using Microsoft.SharePoint.Administration;
```



7. Add the following methods to the class:

```
        public override void FeatureDeactivating(
            SPFeatureReceiverProperties properties)
        {
            SPSite site = properties.Feature.Parent as SPSite;

            if (site != null)
            {
                string timerJobName = GetTimerJobName(site);
                RemoveTimerJob(site, timerJobName);
            }
        }

        public static string GetTimerJobName(SPSite site)
        {
            return String.Format("{0} ({1})",
                "Wrox.GlobalNavigation.SiteMapBuilderTimerJob",
                site.ServerRelativeUrl);
        }

        public void RegisterTimerJob(SPSite site)
```

www.it-ebooks.info

```
{
    string timerJobName = GetTimerJobName(site);

    RemoveTimerJob(site, timerJobName);

    SiteMapBuilderTimerJob smbJob =
        new SiteMapBuilderTimerJob(timerJobName, site);
    SPMinuteSchedule schedule = new SPMinuteSchedule();
    schedule.BeginSecond = 0;
    schedule.EndSecond = 59;
    schedule.Interval = 15;
    smbJob.Schedule = schedule;
    smbJob.Update();
}

private static void RemoveTimerJob(SPSite site, string timerJobName)
{
    foreach (SPJobDefinition job in site.WebApplication.JobDefinitions)
    {
        if (job.Name == timerJobName)
        {
            job.Delete();
        }
    }
}
```

These methods handle registering the timer job and removing the timer job when the feature is deactivated.

8. Finally, modify the `FeatureActivated` method and add a call to `RegisterTimerJob(web.Site)` to the end of the method. After your modifications, the method should look like the code below:

```
public override void FeatureActivated(
    SPFeatureReceiverProperties properties)
{
    SPWeb web =  ((SPSite)properties.Feature.Parent).RootWeb;

    SPList globalNavigation =
        web.Lists[Lookup.ListNames.GlobalNavigationList];

    ListHelper.RestrictAvailableContentTypes(
        globalNavigation.RootFolder, "Navigation Menu");

    RegisterTimerJob(web.Site);
}
```

Wrox.GlobalNavigation\Wrox.GlobalNavigation\Administration\Administration.EventReceiver.cs

9. Save the file and deploy the solution.

The timer job will ensure that the menu structure is refreshed every fifteen minutes. If you need to increase or decrease the refresh interval, simply change the `schedule.Interval` value you entered

www.it-ebooks.info

above to the desired number of minutes. Also, if you need a change that was made to be visible immediately, you can manually run the timer job from within Central Administration.

Congratulations! It was a long journey, but you've successfully created a global navigation solution. Be sure to test it across site collections by activating the `Wrox.GlobalNavigation` solution on another site collection and pointing back to the site collection where the administration feature has been activated.

SUMMARY

This chapter walked you through the multitude of steps involved in developing a global navigation system. The solution was targeted toward the needs of a fictitious company but should provide an excellent starting point for your own enhancements and needs. Following is the list of requirements met by the solution:

- Support basic menu functionality such the display of a hierarchical list of links.
- Provide scalability.
- Security trim menu items.
- Support delegating administration to end users.
- Support multiple types of menu items.
- Maintain the look and feel of the OOTB SharePoint Navigation menu.
- Support multiple menu hierarchies.
- Functionality can be enabled/disabled on a per-site collection basis.

www.it-ebooks.info

A

URL Shortcuts

Steve Mann

SharePoint heavily relies on URLs to accomplish various tasks. Usually, this is simply a relative path based on the current site URL. Although these URLs are the links shown in Site Settings, knowing how to get to these actions comes in handy when the Site Actions menu is not available or customized master pages and page layouts suppress other links.

SITE MANAGEMENT

Simply append the following URLs to the site URL to perform various site related management tasks.

- Create New Item: `/_layouts/create.aspx`
- Manage People and Groups: `/_layouts/people.aspx`
- Manage Subsites: `/_layouts/mngsubwebs.aspx`
- Navigation Settings: `/_layouts/AreaNavigationSettings.aspx`



This navigation setting is not available in some site templates such as Team Site.

- Save Site as Template: `/_layouts/savetmpl.aspx`



This template is not supported in some site templates such as Publishing.

www.it-ebooks.info

- Site Alerts: `/_layouts/subedit.aspx`
- Site Collection Administrators: `/_layouts/mngsiteadmin.aspx`
- Site Content Types: `/_layouts/mngctype.aspx`
- Site Content and Structure Manager: `/_layouts/sitemanager.aspx`
- Site Settings: `/_layouts/settings.aspx`
- Site Usage Summary: `/_layouts/SpUsageWeb.aspx`
- User Alerts: `/_layouts/sitesubs.aspx`
- User Permissions: `/_layouts/user.aspx`

GALLERIES

The galleries are SharePoint system lists that contain site collection-level items. You may easily access these galleries by appending the following URLs to the site collection root URL.

- List Template Gallery: `/_catalogs/lt/Forms/AllItems.aspx`
- Master Page Gallery: `/_catalogs/masterpage/Forms/AllItems.aspx`
- Site Template Gallery: `/_catalogs/solutions/Forms/AllItems.aspx`
- Site Column Gallery: `/_layouts/mngfield.aspx`
- Web Part Gallery: `/_catalogs/wp/Forms/AllItems.aspx`

PAGE MAINTENANCE

Sometimes, web part exceptions or configurations cause a total page fault. You cannot render a "broken" page. Therefore, you need a way to remove any faulty web parts. Add the following to the URL of the page to access the page's web part maintenance mode:

```
?contents=1
```

Other times, you cannot add web parts onto a particular page. This may be because of modified master pages, page layouts, CSS, and so on. You can force the page to display the web part dialog by appending the following to the page URL:

```
?PageView=Shared&ToolPaneView=2
```

If SharePoint Personalization is instantiated on the site that you are modifying, you can easily tweak the personalized version of a page by appending the following to the page URL:

```
?PageView=Personal&ToolPaneView=2
```

www.it-ebooks.info

B

CSS Quick Reference

Steve Mann

When branding and stylizing your SharePoint sites, you may need to override default system styles via CSS classes. SharePoint uses the following common class styles.

ITEM	CLASS
Body Container	`body #s4-bodyContainer`
Browse	`.ms-cui-tt-s > .ms-cui-tt-a`
Content Table Body Area	`.s4-ca`
Edit	`.s4-breadcrumb-anchor, .ms-qatbutton`
Give Feedback	`.ms-cui-tabrowright a img`
Help	`.s4-help`
I Like It	`.ms-socialNotif`
Left Panel	`body #s4-leftpanel`
Left Panel Headers	`.s4-ql ul.root > li > .menu-item, .s4-qlheader, .s4-qlheadervisited`
Left Panel Links	`.s4-ql ul.root ul > li > a`
Main Area	`body #s4-mainarea`
Navigate Up	`.s4-breadcrumb-anchor, .ms-qatbutton`
Page	`.ms-cui-tt-span`

continues

www.it-ebooks.info

ITEM	CLASS
Page Description	`.s4-title .s4-pagedescription, .s4-title .s4-uniqpermdescription`
Recycle Bin	`.s4-specialNavLinkList a`
Ribbon Row	`body #s4-ribbonrow`
Search	`.s4-search table`
Search Image	`.s4-search .ms-searchimage`
Search Input	`.s4-search input`
Site Actions	`.ms-siteactionsmenuinner`
Social Data	`.ms-socialNotif-Container`
Social Separator	`.ms-socialNotif-groupSeparator`
Special Navigation Links List	`.s4-specialNavLinkList`
Status Bar Container	`.s4-statusbarcontainer`
Table Row Left	`.ms-cui-TabRowLeft`
Tags & Notes	`.ms-socialNotif`
Title Logo	`.s4-titlelogo`
Title Row	`.s4-title`
Title Row Container	`body #s4-titlerow`
Title Text	`.s4-titletext`
Top Header 2	`body #s4-topheader2`
Top Link Selected	`.s4-toplinks .s4-tn a.selected`
Top Link Static	`.s4-tn li.static > .menu-item`
Top Links	`.s4-toplinks`
View All Site Content	`. s4-specialNavLinkList a`
Web Part Cell	`.s4-wpcell`
Web Part Content	`.ms-wpContentDivSpace`
Web Part Header	`.ms-WPHeader`
Web Part Header Menu	`.ms-WPHeaderTdMenu`
Web Part Header Selection	`.ms-WPHeaderTdSelection`

www.it-ebooks.info

ITEM	CLASS
Web Part Header TD	`.ms-WPHeader td`
Welcome Menu	`.ms-welcomeMenu`
Workspace	`body #s4-workspace`

www.it-ebooks.info

www.it-ebooks.info

C

Hiding and Customizing Page Breadcrumbs

Steve Mann

Using CSS modifications you can easily hide page breadcrumbs; however, this is usually across all sites and pages. There are times when you want to hide only the breadcrumbs on certain pages or show a different navigation within the breadcrumb links. You may easily accomplish this by adding a Content Editor web part to the page that contains JavaScript code.

You need to check the Hidden property so that users do not see the actual web part. To hide the breadcrumbs, using the Source Editor from the Web Part Properties Pane, add the following JavaScript to the Content Editor Source:

```
<script language=javascript>
document.getElementById('PlaceHolderTitleBreadCrumb').innerHTML = "";
</script>
```

JQuery is also a popular method of modifying the SharePoint UI. You may accomplish the same result of hiding the breadcrumbs using the following JQuery script:

```
<SCRIPT type=text/javascript>
   $(document).ready(function(){
   var brdCrumb = $('.breadcrumbRootNode').parent();
   brdCrumb.hide();
   brdCrumb.next().hide();
  });
</SCRIPT>
```

You may use similar code to modify the breadcrumb links. A scenario for this may be as follows. There are department sites within the main (root) site collection; however, you may have department news contained in other site collections under a separate managed path (for

www.it-ebooks.info

example, /news/). When the user navigates to the News site for the department (human resources, for example), because it is in a separate site collection, the out-of-the-box breadcrumb structure contains only Home. You would like the user to get back to the HR department site, which lives under Departments within the root site collection. To accomplish this, use the same Content Editor web part trick as explained previously, except this time use the following code:

```
<script language=javascript>
 document.getElementById('PlaceHolderTitleBreadCrumb').innerHTML =
"<a href='/pages/default.aspx'>Home</a> >
<a href='/Departments'>Departments</a> >
<a href='/Departments/HR'>HR</a> > News and Events";
</script>
```

Modify the HTML links according to your site structure and site names. In the example code, it appears that the user is in a site under <root>/Departments/HR, but the user could be in a totally different site collection. The HTML forms the breadcrumb navigation and uses relative paths to redirect the user back to the "real" HR department site.

www.it-ebooks.info

D

PowerShell Quick Reference

Steve Mann

When transitioning from STSADM to PowerShell, it is easy to forget which commands to use. This appendix can assist in the transition from STSADM to PowerShell and lists the more common PowerShell equivalents of the most commonly used STSADM commands.

Backup and Restore

```
Backup-SPConfigurationDatabase

Backup-SPFarm

Backup-SPSite

Get-SPBackupHistory

Restore-SPFarm

Restore-SPSite
```

Features

```
Enable-SPFeature

Disable-SPFeature

Install-SPFeature

Uninstall-SPFeature
```

Form Templates

```
Enable-SPInfoPathFormTemplate

Disable-SPInfoPathFormTemplate
```

www.it-ebooks.info

```
Get-SPInfoPathFormTemplate
Install-SPInfoPathFormTemplate
Start-SPInfoPathFormTemplate
Set-SPInfoPathFormTemplate
Test-SPInfoPathFormTemplate
Update-SPInfoPathFormTemplate
```

Sites and Webs

```
Get-SPSite
Get-SPWeb
Import-SPWeb
Move-SPSite
New-SPSite
New-SPWeb
Remove-SPSite
Remove-SPWeb
Set-SPSite
Set-SPWeb
```

Solutions

```
Add-SPSolution
Get-SPSolution
Install-SPSolution
Remove-SPSolution
Update-SPSolution
```

Users

```
Get-SPUser
Move-SPUser
New-SPUser
Remove-SPUser
```

www.it-ebooks.info

GLOSSARY

Absolute path Starts with the `http://` and maps out the entire SharePoint site location. `http://sharepoint/sites/Depts/HR` is an absolute path that indicates a specific subsite in SharePoint located off the root site collection. Also see *Relative Path*.

Access URL The internal URL used by a search crawler to identify and access an item.

Action The smallest unit of work in a workflow system. An action can contain one or more tasks.

Active Directory Refers to the Windows implementation of a directory service. Active Directory stores information about a variety of objects in the network. More importantly, Active Directory stores user accounts, computer accounts, groups, and all related credential information used by the Windows implementation of Kerberos.

AJAX (Asynchronous JavaScript and XML) An umbrella term that describes web pages' subsequent connections to origin web servers to send or fetch content after all its resources load.

API (Application Programming Interface) Rules and specifications that enable application to invoke functionality in another, possibly returning data. When people reference the SharePoint API, they refer to the underlying objects and object models available (for example, SPSite, SPWeb, and so on).

Breadcrumbs A list of links that indicate the steps a user has taken through a site to reach the current point, or to indicate where the current page is in the site's hierarchy.

CAML (Collaborative Application Markup Language An XML-based language that is used to query and retrieve SharePoint list items.

CMS (Content Management System) A suite of software that makes it easy to create and launch web sites that enable the authorship and publishing of dynamic content, and personalization of the content for the user.

Content provider The owner of a web site or the content that appears on a web site.

CPU (Central Processing Unit) The chip at the heart of a computer or mobile device, which executes the code that provides the device's primary operating system and functionality.

CSS (Cascading Style Sheets) A set of standards defining how markup should be styled, formatted, and laid out by a browser.

DOM (Document Object Model) The in-memory representation of a web page within the browser, normally built from the page's HTML and accessed by an API exposed to JavaScript running in the browser.

www.it-ebooks.info

GIF (Graphics Interchange Format) A format of a bitmap image, popular for simple graphics on web pages.

HTML (HyperText Markup Language) The markup language that describes and structures web pages, which is standardized by the W3C and other groups.

HTTP (HyperText Transfer Protocol) The protocol that enables web browsers to request resources from a server, which responds back to the client.

Internet Explorer A desktop web browser made by Microsoft. The term also describes the company's recent mobile browsers.

JavaScript A programming language, normally executed in a web browser, to provide interactivity and asynchronous behavior in web pages.

JPEG (Joint Photographic Experts Group) Describes a glossy-compressed image format, suitable for photos, for example.

jQuery A JavaScript library popular on desktop web sites, which is now available with a mobile library.

JSON (JavaScript Object Notation) A text-based standard for serializing data structures which is commonly used for asynchronous communication between web clients and servers.

Operating system The native software layer on a computer or server responsible for managing the execution of all the applications installed on it, as well as the drivers of the hardware underlying it.

PNG (Portable Network Graphics) An image format suitable for displaying simple bitmap-based graphics on the web.

Relative path A path implied by the current working site or calculated based on a specified web path. `/_layouts/viewlsts.aspx` is a relative path. The resolved full URL is the current site location plus the relative path (for example, `http://sharepoint/subsite/_layouts/viewlsts.aspx`).

URL (Uniform Resource Locator) The address of a web page or resource, indicating both the server or site from which it can be retrieved, and the path describing the location of the resource on that server.

W3C (The World Wide Web Consortium) A standards body responsible for such recommended specifications as HTML, XML, CSS, HTTP, and so on.

Web application A container in a configuration database that stores administrative settings and entry-point URLs for site collections.

Web part A reusable component that contains or generates Web-based content such as XML, HTML, and scripting code. It has a standard property schema and displays that content in a cohesive unit on a web page.

WSDL (Web Services Description Language) An extensible XML format that describes network services as collections of communication endpoints, as specified in WSDL.

XML (Extensible Markup Language) A standard W3C language that uses tags to represent data and metadata.

XSLT (Extensible Stylesheet Language Transformation) A style language that transforms XML documents or data into presentable information.

www.it-ebooks.info

INDEX

Symbols

&& (ampersand/double), and operator, 153
|| (pipe sign/double), or operator, 153

A

Access, 8
Access Client Lists (ACLs), 27, 307
Access Services, 200
ACLs. *See* Access Client Lists
actions, SPD workflows, 154–160
    customization, 160
    lists, 155–156
Active Directory, 257
Active Directory Services, 205
`Add()`, 319, 331
Add a Comment, 154
Add List Item Permission, 159
Add Service Reference, 75
Add Time to Date, 154
`AddField`, 18
`AddFieldAsXml`, 18
`AdditionalPageHead`, 11
`Add-SPSnapin`, 177
`Add-SPSolution`, 189
`-AddToQuickLaunch`, 188
`-AddToTopNav`, 188
Advanced Search Box, 64, 68–70
AJAX, 309
alerts, 70–71
algorithmic efficiency, 299–300
aliases, 176
Allow Writers to View Cached Content, 308
`AllowBaseTypeRendering`, 96
`"AllowMultipleControls"`, 10
`-AllWebApplications`, 190
and operator, 153
Append Task, 158
`Application`, 316
applications. *See also* service applications
    cache, 299
    integration, SPD workflows, 160–161
    scalability, 297–342
        cache, 306–316
        client-side optimization, 329–331
        code caching, 316–325
        data sharding, 305–306
        database allocation, 304–305
        farm configuration, 300–306
        large lists, 325–328
        load balancing, 302–306
        performance, 298–300
        writing efficient code, 331–342
    sites, 235
    Web, 184–186
application pages
    calendar, 29–30
    customization, 6–8, 29–30, 229–232
    OOTB, 7
    PowerShell, 232
`ApplicationPool`, 185
`ApplicationPoolAccount`, 185
Approval template, 138–139
Approve button, 274–276
`.ascx`, 37, 38
    visual web parts, 49
`.asmx`, 13, 73
ASP.NET
    `CreateChildControls`, 37
    HTTP Cache, 319–320
    `HttpRunTimeCache`, 26

www.it-ebooks.info

ASP.NET *(continued)*
navigation, 345
`RequiredFieldValidator`, 264
`ViewState`, 317
`VisualWebPartUserControl.ascx`, 37
web services, 73
workflow form templates, VS workflows, 172
`ASPScriptOptimizationEnabled`, 182
`.aspx`
application pages, 6–7, 29–30
.NET, 7
site request process, 239
Solution Explorer, 261
visual web parts, 38
Assign a Form to a Group, 156
Assign a To-Do Item, 156
`AssignGroupMembers()`, 279
`AssignmentCollection`, 185
Associate the Workflow to a Content Type, 162
Associate the Workflow to a List, 162
attributes, 79
`Authentication`, 185
`AuthenticationMode`, 15
`Author`, 272
auto-retraction, 359

B

backend, 32
backup and restore, PowerShell, 194–195, 407
`Backup-SPConfigurationDatabase`, 194
`Backup-SPFarm`, 194
`Backup-SPSite`, 195
`base.Value`, 110
`Batch`, 336
batch operations, 333–336
`BatchInsert()`, 336–337
Binary Large OBject (BLOB), 312–316
binary modules, 176
BLOB. *See* Binary Large OBject
blog aggregator
cache, 26
custom web parts, 25–26
event handlers, 27
object model, 24–27
timer jobs, 26–27
breadcrumbs, 405–406
browser support, master pages, 221
`btnApprove_Click`, 284
`btnSave_Click`, 284
buckets, 28
bulk/digest reminders, 28
Business Connectivity Service, 56

C

cache, 298
application scalability, 306–316
applications, 299
BLOB, 312–316
blog aggregator, 26
code, application scalability, 316–325
DNS, 303
navigation, 382–386
object cache, 309–312
Page Output Caching, 306–309
reusable, 321–325
security, timer jobs, 26–27
static variables, 320
WFEs, 26
`_cache`, 323
Cacheability, 308
calendar
application pages, 29–30
object model, 28–31
SPD, 28–29
UI, 30
CAML queries
lists, 21–22, 332
`ListViewByQuery`, 52
`Query`, 81
`CamlQuery`
`GetItems`, 19
`RowLimit`, 22
`ViewXmol`, 22
`CAMLRendering`, 96
Cancel button, 274–276
Capture a Version of the Document Set, 157
`-CASPolicies`, 190
`_catalogs/masterpage`, 222
`cd`, 180
CDN. *See* content delivery network

www.it-ebooks.info

Central Administration
    HTTP port, 183
    query throttling, 326
changeCheckInterval, BLOB Cache, 314–315
`chdir`, 180
Check for Changes, 307, 308
Check In Item, 155
Check List Item Permission Levels, 154
Check List Item Permissions, 153
Check Out Item, 155
child webs, 16
Client Object Model, 14–24
    `ClientContext`, 15–16
    document libraries, 22–24
    lists, 17–22
    sites, 16–17
    web services, 13
`ClientAuthenticationMode.FormsAuthentication`, 15
`ClientContext`
    Client Object Model, 15–16
    `ExecuteQuery`, 23
`clientContext.Webs.Lists`, 18
client-side optimization, 329–331
`cls`, 180
cmdlets, 176
code
    batch operations, 333–336
    cache, application scalability, 316–325
    exception handling, 340
    forms, customization, 265–276
    lists, 331–333
    multithreaded operations, 336–337
    `SPMonitoredScope`, 340–341
    strings, 338–339
    visual web parts, 40–41
    writing efficiently, 331–342
code-behind
    `OnDateChanged`, 51
    `SiteRequests.aspx.cs`, 258
    `using`, 51
Collect Data from a User, 156
Collect Feedback template, 139
columns, 9
conditions, 151–154
configuration databases, 194
configuration lists, 8
    Site Type, 245–248
`ConfigurationOnly`, 195
`-Confirm:$false`, 183
`-ConfirmPassPhrase`, 183
conflict messages, 362
Connection Interface, 44
consumer web parts, 46–48
content
    databases, 193–194
    indexing, search, 55–56
    management, PowerShell, 192–193
    sources
        Scope Rule Type, 59
        search, 56–57
        Search Service Application, 57
`Content`, 203
content delivery network (CDN), 329–330
Content Query Web Part (CQWP), 24, 226–227
    object cache, 309, 310
content types
    customization, 114–120
    document content type actions, 159
    field types, 9
    workflows, 133
`-ContentDatabase`, 187
`ContentTemplate`, 226
`ContentTypeId`, 222
`Continue`, 336
`"Continue"`, 84
controls
    delegate controls, 10–11
        navigation, 391–394
        OOTB, 10
    editing control, 98–99
    field, SSN, 108–110
    field controls, SSN field types, 110
    forms, 261–265
        Site Type, 266
    navigation, 294
        delegate, 391–394
    rendering control class, 98
    user control, 38
    visual web parts, 38–39, 49–53
`ControlId`, 10
`ControlMode`, 109

www.it-ebooks.info

ConvertTo-SecureString, 183
Copy and Modify, 161
Copy List Item, 155
core actions, 154–155
CPU, 298
  database allocation, 304–305
CQWP. *See* Content Query Web Part
crawl schedule, 56–57
Create List Item, 155
CreateChildControls
  ASP.NET, 37
  consumer web parts, 46
  provider web parts, 46
  rendering template, 109
  TextField, 109
Created, 272
Created by a Specific Person, 153
Created in a Specific Date Span, 153
Created Sites list, 276–277
  custom forms, 294
CreateGroup(), 279
credentials
  network, 15
  web services, 76–78
CSS
  customization, branding, 217–220
  fixed width design, 218–219
  OOTB, 218
  quick reference, 401–403
  search, 220
  server round trips, 330–331
  sprites, 330
  themes, 217–218
.CSS, 4
Current Item:Status, 291
customization
  actions, SPD workflows, 160
  application pages, 6–8, 29–30, 229–232
  content types, 114–120
  CSS, branding, 217–220
  external access, 13–24
  feature receivers, 251–255
  field types, 9, 94–114
  forms
    code, 265–276
    Created Sites list, 294
    site request process, 242
  lists, 8–9
  list forms, 8–9
  list item form, 256–258
  list schema, 122–128
  master pages, 228
  navigation, 343–398
    architecture, 344–347
  nonvisual components, 11–13
  object model, 3–24
  search, 55–71
    alerts, 70–71
    Enterprise Search Center, 63–70
    search scopes, 57–62
  service applications, site creation, 241–242
  UI components, 4–11
  web parts, 4–6, 25–26, 30
  workflows, VS workflows, 173
  XSLT style sheet, 98
Customization Wizard, 35–36
CustomMasterUrl, 222–223

D

data sharding, 305–306
database
  allocation, application scalability, 304–305
  calls, 299
  content, PowerShell, 193–194
  management, PowerShell, 193–195
-DataProviderID, 207
-DataProviderType, 207
datasheet view, 8
DataSource, 203
DateTimeControl, 51
Declare Record, 156
default.aspx, 307
default.master, 220
DefaultTemplateName, 109
delegate controls, 10–11
  navigation, 391–394
  OOTB, 10
Delegate Task, 158
DelegateControl, 10
Delete Drafts, 159
Delete Item, 155
Delete Previous Versions, 159
deleteCheckedInFiles, 315

www.it-ebooks.info

DeleteObject, 18
    list items, 20
deleteOldCacheFolders, 315
Deploy Solution, 41
Deployment Conflicts, 362
deployment package, 284–285
DeploymentConflictResolution, 243
Design View, 38–39
dialog boxes, 220
dir, 180
Disable-SPFeature, 192
Disable-SPInfoPathFormTemplate, 202
Discard Check Out, 155
disk access latency, 298
Dismount-SPContentDatabase, 194
display groups, 60–62
DisplayTemplateName, 108–109
Disposition Approval template, 139
, 224
DLLs, 176
DNS load balancing, 303–304
Do Calculation, 154
DocType, 221
Document, 146
document content type actions, 159
document libraries, 93
    Client Object Model, 22–24
    event handlers, 11
    FileCreationInformation, 23
    lists, 22
    WebDAV, 23–24
    workflows, 133, 138
document set actions, 157
DocumentLibrary, 203
domain model, 364–365

E

ECB. *See* Edit Control Block
echo, 180
ECMAScript, 14
Edit Control Block (ECB), 134, 138
Edit the Workflow, 161
editing control, 98–99
Editor, 272
EffectiveBasePermissions, 16
Elements.xml
    list instance, 250
    Web Part Gallery, 37
elements.xml, 222
email templates, 28
Employee Details
    content types, 114–120
    list definitions, 120–128
Enabled, 270
Enable-SPFeature, 192
Enable-SPInfoPathFormTemplate, 201
EnableThrottling, 326
End Task Process, 158
EnsureChildControls(), 110
Enterprise Search Center, 63–70
Enterprise Search Service, 196–199
Escalate Task, 158
ETL. *See* extract, translate, and load
event handlers, 11
    blog aggregator, 27
    list workflows, 12
event receivers, 252
    feature receivers, 12
    navigation UI, 361–362
    workflows, VS workflows, 172
-examples, 180
Excel, 147
Excel Services, 200–201
exception handling, 340
Exchange server, 55
ExecuteQuery
    ClientContext, 16, 23
    ListItemCreationInformation, 18
Export the Workflow to Visio, 162
external access, 13–24
external sites, 235
external sources, 56
extract, translate, and load (ETL), 306, 341
Extract Substring from End of String, 157
Extract Substring from Index of String, 157
Extract Substring from Index with Length, 158
Extract Substring from Start of String, 157

www.it-ebooks.info

F

farm
  configuration, 300–306
  management
    pass phrase, 182–183
    PowerShell, 181–184
Farm Administrators, 240–241
features
  deactivation, site provisioning, 255
  navigation, 349
  PowerShell, 407
  sites, 277
  themes, 217
feature receivers, 12–13
  customization, 251–255
  navigation UI, 360–361
`FeatureActivated`, 222
`FeatureActivated()`, 252
`FeatureActivated`, 397
`FeatureDeactivating`, 222, 223
`FeatureDeactivating()`, 255
`Feature.feature`, 244
`Features`, 244
Federated Results, 64
`Field`, 126
`<Field>`, 95
fields
  class, 97–98
    SSN, 104–107
  controls
    SSN, 108–110
    SSN field types, 110
  lists, batch operations, 334
  list items, `ViewFields`, 81
  types, 93
    customization, 9, 94–114
    definitions, 95–99, 102–103
    editing control, 98–99
    OOTB, 94, 97–98
    properties, 96–97
    SSN, 99–114
    SSN field controls, 110
  value types, 98
`FieldEditorUserControl`, 96
`FieldRef`, 126
`FieldRenderingControl`, 110
`<FieldType>`, 95
`FieldTypeClass`, 96
`<FieldTypes>`, 95
`File`, 23–24
file shares, 55
The File Size in a Specific Range Kilobytes, 153
The File Type Is a Specific type, 153
`fileBuffer`, 90
`FileCreationInformation`, 23
`File.SaveBinaryDirect`, 23–24
`FileStream`, 89–91
`Filterable`, 96
Find Interval Between Dates, 158
fixed width design, 218–219
`foreach`, 82
forms
  code, customization, 265–276
  controls, 261–265
    Site Type, 266
  customization, Created Sites list, 294
  helper functions, 258–261
  InfoPath, SPD workflows, 161
  lists, customization, 8–9
  list items, customization, 256–258
  saving data, 271–272
  security, 269–270
  templates
    PowerShell, 407–408
    VS workflows, 172
`FormatString()`, 339
Forward Task, 158
`FullTextSqlQuery`, 333

G

`-GACDeployment`, 190
galleries, 400
  Web Part Gallery, 37
geographic DNS services, 304
`Get()`, 320
`Get-Alias`, 180
`GetByTitle`, 18
`Get-ChildItem`, 180
`Get-Command`, 178
`GetCredential()`, 78
`GetHash()`, 321

www.it-ebooks.info

Get-Help, 179–180
GetItems
    CAML queries, 21
    CamlQuery, 19
    list items, 20
GetListCollection(), 79
GetListCollections(), 81
GetListItems(), 80–83
GetResponse(), 91
Get-SPAccessServiceApplication, 200
Get-SPContentDatabase, 194
Get-SPDatabase, 193, 194
Get-SPEnterpriseSearchCrawled Property, 197
Get-SPEnterpriseSearchMetadata Category, 197
Get-SPEnterpriseSearchService, 196
Get-SPEnterpriseSearchService Application, 196
Get-SPExcelServiceApplication, 200
Get-SPFeature, 191
Get-SPManagedPaths, 184
Get-SPMetadataServiceApplication, 202
Get-SPPerformancePointService Application, 203
Get-SPProfileServiceApplication, 206
Get-SPSearchService, 195
Get-SPSecureStoreServiceApplication, 204
Get-SPSite, 186, 187
Get-SPSolution, 190
Get-SPStateServiceApplication, 205
Get-SPVisioServiceApplication, 206
Get-SPWeb, 188
Get-SPWordConversionService Application, 208
GetValidatedString, 104–105
global declarations, 46
GlobalNavigation, 11
GlobalSiteLink, 11

H

hardware load balancing, 302–303
hash tables, 299–300, 341
HasUniqueRoleAssignments, 16
helper functions, 258–261
hierarchical data store, 26
Hit Counter, 334
home page, 287–288
HTML
    Page Output Caching, 306
    server round trips, 330
    strings, 338
    ViewState, 317
HTTP
    BLOB Cache, 313
    Central Administration, 183
    load-balancing, 302
HTTP Cache, 319–320
HTTP Module, 330
HTTPContext, 311, 318
HttpContext.Current.Cache, 319
HTTPContext.Items, 318
HttpRunTimeCache, 26

I

ICredentials, 77
-Identity, 188, 191, 202, 206
IE8, 221
If Any Value Equals Value, 152
If Current Item Field Equals Value, 152
IFRAME, 309
IgnoreCase, 338
IIS
    BLOB Cache, 314
    web services, 73, 74
impersonation
    site creation, 240
    SPD workflows, 153–154, 160
        step actions, 159
indexed fields, 327–328
InfoPath, forms, SPD workflows, 161
InfoPath Forms Service, 201–202
InfoPath Services, 29–30
Inherit List Item Parent Permissions, 159
Initiator, 145
InnerXml, 81, 84
in-place load, 21
Insert(), 319
Insert Task, 158
Install-SPFeature, 192

www.it-ebooks.info

Install-SPInfoPathFormTempate, 201
Install-SPSolution, 190
instance
    lists, Elements.xml, 250
    workflows, 143–145
        workflow status page, 145–146
instantiating, web services, 73–76
intranet sites, 235
I/O, 298, 341
    database allocation, 304–305
    Page Output Caching, 306
IP address, 303
ISAPI, 73
IsRequestorEligible(), 258, 274
IsUserInGroup(), 258
ItemDeleting, 11
ItemId, 44
Items, 323
ItemTitle, 44
IWebPartConnection, 44

J

JavaScript
    Client Object Model, 14
    server round trips, 330–331
    web services, 13
Join(), 337
join, 341

K

KeywordQuery, 333

L

large lists, 325–328
largeFileSize, 315
/_layouts, 241–242
_layouts, 6, 330
Level 1 cache, 298
Level 2 cache, 298
LINQ, 21
List, 203
<List>, 79
lists
    actions, SPD workflows, 155–156
    attributes, 79
    batch operations, 334
    CAML queries, 21–22
    Client Object Model, 17–22
    code, 331–333
    customization, 8–9
    document libraries, 22
    event handlers, 11
    fields, batch operations, 334
    forms, customization, 8–9
    instance, Elements.xml, 250
    large, 325–328
    object model, 331
    OOTB, 8
    queries, 20–22
        LINQ, 21
        web services, 78–83
    schema
        VS, 126
        customization, 122–128
        OOTB, 126
    sites, 250–251
        provisioning buildout, 244–251
        request process, 242–243
    UI, 8
    views, 286–287, 328
    workflows, 11–12, 133, 138
        SPD workflows, 169
list definitions, 120–128
    navigation UI, 352–358
    schema, 247
list items, 18–20
    batch operations, 334
    deleting, 86–89
    fields, ViewFields, 81
    forms, customization, 256–258
    InnerXml, 84
    Lists web service, 83–89
    OuterXml, 84
    queries, web services, 78–83
    site provisioning, 254–255
    updating, 84–86
    XmlDocument, 83
List Workflow, 31
ListCreationInformation, 17

www.it-ebooks.info

list.GetItems(), 22
ListHelper, 360, 368
ListInstance, 127
ListInstance1, 250
ListItemAllFields, 23
ListItemCollection, 22
ListItemCollectionPosition, 22, 327
ListItemCreationInformation, 18–19
listName, 81
<Lists>, 79
Lists web service, 73–92
    list items, 83–89
    uploading documents, 89–92
ListTemplateType, 17
ListViewByQuery, 51, 52
-LiteralPath, 189
litJSPlaceHolder, 265, 274
Load, 15–16
load balancing, application scalability, 302–306
load-based routing, 302
LoadFields(), 268, 272
LoadSiteTypes(), 268
locations, 78–79
lock(), 337
Log to History List, 154
logging site creation, 279–280
logical operators, 153
login, 225
Lookup, 366–368
Lookup Manager of a User, 156
Lotus Notes, 56
Lotus Notes Index Setup Wizard, 56
ls, 180

M

Managed Metadata Services, 202
Managed .NET Client Object, 14–15
managed paths, 184
managed properties, 62–63
Management Console, 176
Management Shell, 176–177
ManagePermissions, 224
Mann, Steve, 89
master pages, 4
    browser support, 221
    customization, 228
    OOTB, 228
    Page Output Caching, 308
    SPD, 221
    UI, 220–225
    VS, 221–223
MasterPageModule elements.xml, 227
MasterUrl, 222–223
MasterURLFeatureDeactivatingMethod.txtCustomizing, 224–225
max-age, BLOB Cache, 315–316
MaximumStaleMinutes, 322
maxSize
    BLOB Cache, 314
    object cache, 312
memory, 298–299. *See also* cache
    strings, 338
Menu, 364, 369
MenuBase, 364
menuFolder, 370
MenuItem, 364, 370
metadata, 93
    document libraries, 23
    Managed Metadata Services, 202
Metadata Service, 202
method syntax, 21
Microsoft.SharePoint.PowerShell, 177
Minimal.master, 221
Modified, 272
Modified by a Specific Person, 153
Modified in a Specific Date Span, 153
<Module>, 222
modules, 176
    HTTP Module, 330
    URL Rewrite Module, 330
multithreaded operations, 336–337
MySites, 205

N

\n, 82
navigation
    cache, 382–386
    controls, 294
    customization, 343–398
        architecture, 344–347
    delegate controls, 391–394
    domain model, 364–365

www.it-ebooks.info

navigation *(continued)*
items, 219
`ListHelper`, 368
`Lookup`, 366–368
menu selector, 377–382
nonvisual components, 347
OOTB, 344, 345–346
processing layer, 363–377
repository class, 368–377
scalability, 343
security, 344
SiteMapProvider, 345–346, 349, 386–394
timer jobs, 394–398
UI, 346–347, 350–363
auto-retraction, 359
conflict messages, 362
event receivers, 361–362
feature receivers, 360–361
list definitions, 352–358
VS, 348–398
`NavigationCache`, 382
NavigationConsumer, 349
.NET. *See also* ASP.NET
`.aspx`, 7
BLOB Cache, 316
Client Object Model, 14–15
multithreading, 337
strings, 338
timer jobs, 26
web services, 13
.NET Framework
Client Object Model, 14
PowerShell, 176
`NetCredentials`, 77
network access, 298–299
network calls, 299
network credentials, 15
New Site Requests, 294
`New-SPConfigurationDatabase`, 194
`New-SPContentDatabase`, 193
`New-SPContentDeploymentJob`, 192
`New-SPContentDeploymentPath`, 192
`New-SPEnterpriseSearchMetadataCategory`, 197
`New-SPEnterpriseSearchMetadataCrawled Property`, 197
`New-SPEnterpriseSearchMetadataManaged Property`, 198
`New-SPEnterpriseSearchMetadataMapping`, 198–199
`New-SPManagedPath`, 184
`New-SPPerfomancePointServiceApplication TrustedLocation`, 203
`New-SPSite`, 188
`New-SPVisioSafeDataProvider`, 207
`New-SPWeb`, 188
`New-SPWebApplication`, 185
NLB. *See* Windows Network Load Balancing
No Activation, 42
Nonpublishing Theme Site, 215–216
nonvisual components
customization, 11–13
navigation, 347
notifications, workflows
SPD workflows, 288–293
templates, 141
`nslookup google.com`, 304

O

object cache, 309–312
object model, 3–33. *See also* Client Object Model
blog aggregator, 24–27
calendar, 28–31
customization, 3–24
lists, 331
query throttling, 326–327
workflow scheduling, 31–33
`ObjectCache`, 312
`OnDateChanged`, 51
`OnDemand`, 341
`OnError`, 336
`"Continue"`, 84
OOTB. *See* Out of the Box
or operator, 153
Out of the Box (OOTB), 4
application pages, 7
content indexing, 55
CSS, 218
delegate controls, 10
field types, 94, 97–98
lists, 8
schema, 126

www.it-ebooks.info

master pages, 228
navigation, 344, 345–346
page layouts, 226
SPD workflows, 161–166
workflow templates, 141
`OuterXml`, 79
list items, 84
`\n`, 82

P

page layouts
navigation, 349
OOTB, 226
Page Output Caching, 308–309
SPD, 226–227
UI, 225–228
VS, 227–228
web parts, 226–227
page maintenance, 400
Page Output Caching, 306–309
`Page.Cache`, 319
`Page.IsPostBack`, 40
`Page_Load`, 40, 52
`Page_Load()`, 267, 268
`PageTitle`, 261
`PageTitleInTitleArea`, 261
`ParentType`, 96
pass phrase, farm management, 182–183
`Path`, 222
`-Path`, 192
`-Path`, 201
`-PathAccount`, 192
Pause for Duration, 154
Pause Until Date, 154
People Refinement Panel, 64
People Search Box, 64
People Search Core Results, 64
`PeopleEditor`, 264, 265
`PeopleEditor.IsValid`, 265
Perform ACL Check, 307
performance
application scalability, 298–300
algorithmic efficiency, 299–300
client-side optimization, 329–331
writing efficient code, 331–342
Performance Point Services (PPS), 202–203
permissions
`ClientContext`, 15
Ribbon, 224–225
SPD workflows impersonation, 153–154, 159
subsites, 188
timer jobs, 26
web services, 77
`PermissionString`, 224
Person Is a Valid SharePoint User, 153
pluggable workflow services, 172
`-Port`, 183
Portal Super Reader, 309
Portal Super User, 309
`PortalSiteMapProvider`, 310–311, 389
PowerPoint, themes, 4, 212
PowerShell, 175–208, 407–408
application pages, 232
backup and restore, 194–195
BLOB Cache, 316
command prompt system commands, 180
configuration databases, 194
content management, 192–193
conventions, 177–178
database management, 193–195
farm management, 181–184
feature management, 191–192
function key shortcuts, 160
management, 181–195
.NET Framework, 176
quick reference, 407–408
search, 195–199
service applications, 199–208
site collections, 186–188
site management, 186–189
site provisioning, 243
solution management, 189–190
starting, 176–177
subsites, 188–189
Web application, 184–186
workflow scheduling, 31
PPS. *See* Performance Point Services
`print`, 180
processing layer, 363–377
Project Explorer, 244
project sites, 235

www.it-ebooks.info

properties
    ClientContext, 15–16
    field types, 96–97
    managed, 62–63
property bag, 26
Property Query, 59
PropertyDefs, 69
-PropSet, 197
_provider, 46, 47
providers, 176
provider web part, 45–46
Provision Site, 334
Provisioning Status, 264
ProvisionSite(), 280–283, 284
public, 44
PublicKeyToken, 389
Publishing Approval template, 139
PublishingAssociatedcontentType, 227
PublishingConsole, 11
"PUT", 90

Q

queries
    CAML
        lists, 21–22, 332
        ListViewByQuery, 52
        Query, 81
    lists, 20–22
        LINQ, 21
        web services, 78–83
    list items, web services, 78–83
    object cache, 309
Query, 81
query, 80, 81
query multiplier, 309
query throttling, 325–327
QueryOptions, 81
queryOptions, 80, 81
Quick Launch, 188
QuickLaunchDataSource, 11

R

RAID, 298
random routing, 302
readChunkSize, 315
Reassign Task, 159
RegisterTimerJob(), 397
Reject Button, 274–276
Related Queries, 64
relational actions, 156
ReloadCache(), 323
Remove List Item Permissions, 159
Remove-SPConfigurationDatabase, 194
Remove-SPContentDatabase, 194
Remove-SPSocialItemByDate, 206
rendering control class, 98
rendering template, 98
    CreateChildControls, 109
    SSN, 101, 107–108
RenderingTemplate, 109
RenderTemplate, 108
Reorder List Fields, 334
Replace List Item Permissions, 159
Repository, 382, 386
repository class, navigation, 368–377
Request a Change, 159
RequiredFieldValidator, 264, 265
RequiredFieldValidators, 265
Rescind Task, 158
Resources, 349
Restore-SPFarm, 195
Restore-SPSite, 195
RestrictAvailableContentTypes, 360
Result Types, 69–70
ResultTypes, 70
Return, 336
reusable caching, 321–325
reusable workflows
    SPD workflows, 169–171
    VS workflows, 172
"ReuseThread", 177
Ribbon
    application pages, 29–30
    hiding, 223–224
    master pages, 220
    MasterURLFeatureDeactivatingMethod.txtCustomizing, 224–225
    permissions, 224–225
    workflows, 138
RoleAssignments, 16

www.it-ebooks.info

root webs, 16
round robin load balancing, 302
RowLimit, 22, 327
rowLimit, 81

S

S trigger, 11
s4-ribbonrow div, 224
Safe for Authenticated Users, 308
sandbox custom workflow activities, 172–173
Sandboxed Solutions, 6
Save, 335
Save button, 272–273
SaveFields(), 271
sb.ToString(), 339
SCA. *See* site collection administrator
scalability
    applications, 297–342
        cache, 306–316
        client-side optimization, 329–331
        code caching, 316–325
        data sharding, 305–306
        database allocation, 304–305
        farm configuration, 300–306
        large lists, 325–328
        load balancing, 302–306
        performance, 298–300
        writing efficient code, 331–342
    navigation, 343
    SPD workflows, 29
    timer jobs, 347
scaling out, 301
scaling up, 301
-Schedule, 193
schedule.interval, 397
schema
    batch operations, 334
    lists
        VS, 126
        customization, 122–128
        OOTB, 126
    list definitions, 247
Scope Rule Type, 59
scripts, 176. *See also* JavaScript
    ECMAScript, 14
search
    content indexing, 55–56
    content sources, 56–57
    crawl schedule, 56–57
    CSS, 220
    customization, 55–71
        alerts, 70–71
        Enterprise Search Center, 63–70
        search scopes, 57–62
    external sources, 56
    managed properties, 62–63
    PowerShell, 195–199
    scopes
        search customization, 57–62
        site collections, 57, 60–62
        SSA, 57, 58–60
    SharePoint Foundation, 195–196
    timer jobs, 27
Search Box, 64
Search Core Results, 64
Search High Confidence Results, 65
Search Paging, 65
Search Service Application (SSA), 55, 56, 57, 195–196
    managed properties, 62–63
    search scopes, 57, 58–60
Search Statistics, 65
Search Summary, 65
Search Visual Best Bets, 65
Secure Store Service (SSS), 204
SecureForm(), 268, 269–270
security
    cache, timer jobs, 26–27
    forms, 269–270
    navigation, 344
    sites, 277–279
    site request process, 240–241
SelectedIndexChanged, 267
self-service site creation, 240
Send an Email, 154
Send Document Set to Repository, 157
Send Document to Repository, 159
Send Task Email, 159
server configuration, 300–302
Server Publishing Infrastructure, 215, 225
server round trips, 330–331

www.it-ebooks.info

service applications
customization, site creation, 241–242
PowerShell, 199–208
service sites, 235
Session, 317
Session State Service, 205
Set Content Approval Status, 155
Set Content Approval Status (as Author), 158
Set Content Approval Status of the Document Set, 157
Set Field in Current Item, 155
Set Task Field, 158
Set Time Portion of Date/Time Field, 154
Set Workflow Status, 155
Set Workflow Variable, 155
SetControls, 52
Set-Location, 176
Set-SPAccessServiceApplication, 200
Set-SPCentralAdministration, 183
Set-SPEnterpriseSearchService, 196
Set-SPExcelServiceApplication, 200
Set-SPFarmConfig, 181
Set-SPInfoPathFormsService, 201
Set-SPMetadataServiceApplication, 202
Set-SPPassPhrase, 182–183
Set-SPPerformancePointService Application, 202–203
Set-SPProfileServiceApplication, 206
Set-SPSearchService, 196
Set-SPSecureStoreServiceApplication, 204
Set-SPSite, 187–188, 189
Set-SPStateServiceApplication, 205
Set-SPVisioExternalData, 207
Set-SPVisioPerformance, 207
Set-SPVisioServiceApplication, 206
Set-SPWeb, 189
Set-SPWebApplication, 185–186
Set-SPWordConversionService Application, 207
sharding, 305–306
Shared, 349
SharePoint Content DB, 320–321
SharePoint Designer (SPD)
application pages, 29–30
calendar, 28–29
list workflows, 12
master pages, 221
page layouts, 226–227
workflows, 28–29, 135
actions, 154–160
application integration, 160–161
conditions, 151–154
core actions, 154–155
custom actions, 160
document content type actions, 159
document set actions, 157
impersonation, 153–154, 160
InfoPath forms, 161
list actions, 155–156
list workflows, 169
notification workflows, 288–293
OOTB, 161–166
relational actions, 156
reusable workflows, 169–171
site request process, 239
site workflows, 171
task actions, 156–159
utility actions, 157–158
Visio, 162–168
workflows, 151–173
XSLT List View, 53–54
SharePoint Foundation
search, 195–196
workflow templates, 138
SharePoint Server, 138–140
<SharePoint:AspMenu>, 219
ShowOnColumnTemplateAuthoringPages, 97
ShowOnDocumentLibraryAuthoringPages, 96
ShowOnListAuthoringPages, 96
ShowOnSurveyAuthoringPages, 97
Silverlight
Client Object Model, 14
web services, 13
Workflow Visualization, 146
Simple.master, 221
Site, 203
-Site, 188, 191, 202
sites, 234
application, 235
Client Object Model, 16–17
content indexing, 56
features, 277

www.it-ebooks.info

lists, 250–251
management
PowerShell, 186–189
URL shortcuts, 399–400
PowerShell, 408
provisioning system, 233–295
buildout, 243–293
site request process, 237–243
quota option, Site Type, 294
request process, 237–243
security, 277–279
taxonomy, 235–237
workflows, 12, 31, 133, 136–137
SPD workflows, 171
VS workflows, 173
Site Actions, 136–137
Site Approver, 264
site archiving workflow, 294
site collections, 234
content databases, 193
managed paths, 184
PowerShell, 186–188
search scopes, 57, 60–62
Server Publishing Infrastructure, 225
Site Collection Administration, 308
site collection administrator (SCA), 234
`Site Creation Failed and Current Item:Provisioning`, 291
Site Owner, 264, 295
`PeopleEditor.IsValid`, 265
validating, 265–266
sites quota management workflow, 295
Site Scoped, 242
Site Settings, 4
Site Support, 295
Site Type, 273–274
configuration lists, 245–248
form controls, 266
site quota option, 294
themes, 294
`SiteCollection`, 203
SiteMapProvider, 345–346, 349, 386–394
`SiteRequests.aspx`, 257–258, 265
Solution Explorer, 283
`SiteRequests.aspx.cs`, 257–258, 265, 283
code-behind, 258
`SmallSearchInputBox`, 10, 11
snap-ins, 176, 177
social items, 206
Social Security Number (SSN)
fields
class, 104–107
controls, 108–110
types, 99–114
type definition, 102–103
rendering template, 101, 107–108
VS, 99–100
XSL Style Sheet, 103
XSLT style sheet, 101
solid state disks (SSD), 298
Solution Explorer
`.aspx`, 261
Deploy Solution, 41
`SiteRequests.aspx`, 283
`SomeObject`, 322
`Sortable`, 97
Source View, 38–39
`SPAssignment`, 181
SPD. *See* SharePoint Designer
`SPExcelBlockedFiletype`, 201
`SPExcelDataConnectionLibrary`, 201
`SPExcelDataProvider`, 201
`SPExcelFileLocation`, 201
`SPExcelServiceApplication`, 200
`SPExcelUserDefinedFunction`, 201
`SPFarmConfig`, 181
`SPField`, 97–98
`SPFieldValidationException`, 105
`SPFolder`, 360
`Menu`, 369
`menuFolder`, 370
`SPListItem`, 267, 363
`SPListItemCollection`, 331–332
`SPList.Items`, 331
`SPList.Items.Add()`, 332
`SPListItem.SystemUpdate()`, 272
`SPMonitoredScope`, 340–341
`SPQuery`, 327
sprites, 330
`SPSecurableObject`, 376
`SPSecurityTrimmedControl`, 224
`SPSessionStateService`, 205
`SPSite`, 181
`SPSiteDataQuery`, 327

www.it-ebooks.info

SPStateServiceDatabase, 205
SPUtility.SendEmail(), 288
SPWeb, 181
SPWeb.AllProperties, 320
SPWebApplication.Properties, 252
SPWebApplication.Properties.Update(), 252
SPWebConfigModification, 391
SPWeb.ProcessBatchData(), 334
SSA. *See* Search Service Application
SSD. *See* solid state disks
SSN. *See* Social Security Number
SSNControl, 110
SSNFieldControl, 107, 108–110
SSNFieldControl.ascx, 101
SSNLabel, 107, 109
SSS. *See* Secure Store Service
Start Approval Process, 157
Start Custom Task Process, 157
Start Document Set Approval Process, 157
Start Feedback Process, 157
Started, 146
Start-SPAssignment, 181
state machine, 238–239
State Service, 205
static variables, 320
sticky sessions, 302
Stop Workflow, 155
Stop-SPAssignment, 181
Stream, 90–91
strings, 338–339
StringBuilder, 339
string.Concat(), 339
String.Equals(), 338
string.Format(), 339
stsadm, 175, 242
  SPMonitoredScope, 340
subsites, 188–189
System.Configuration, 349
System.Management.Automation.PSCmdlet, 176

T

tasks
  actions, SPD workflows, 156–159
  workflow status page, 146
  workflow templates, 141
taxonomy, 235–237
team collaboration sites, 235
templates
  forms
    PowerShell, 407–408
    VS workflows, 172
  workflows, 134, 135, 138–148
    notifications, 141
    OOTB, 141
    SharePoint Foundation, 138
    SharePoint Server, 138–140
    tasks, 141
TemplateType, 17
Terminate the workflow now, 145
TestList, 324
Test-SPInfoPathFormTemplate, 201
Text Editor, 69–70
TextField, 107, 109
themes
  VS, 217
  CSS, 217–218
  .CSS, 4
  features, 217
  PowerPoint, 4, 212
  Site Type, 294
  UI, 211–217
Theme Builder, 212
Theme Designer, 4
third-parties, 136
.thmx, 211
ThreadOptions, 177
Three State Workflow template, 138
throttleLimit, 315
timer jobs, 12
  blog aggregator, 26–27
  caching security, 26–27
  navigation, 394–398
  PowerShell content management, 193
  scalability, 347
Time-to-Live (TTL), 303
Title, 335
  ListCreationInformation, 17
Title Field Contains keywords, 153
Top Federated Results, 65
TopNavigationDataSource, 10, 11
traditional web parts, 4–6
Translation Management template, 139
TreeViewAndDataSource, 11

www.it-ebooks.info

-TrustedLocationType, 203
try/catch, 91
TTL. *See* Time-to-Live
type, 180
-Type, 198, 203
TypeDisplayName, 97
TypeName, 97
TypeShortDescription, 97

U

UI. *See* user interface
UIVersionedContent, 226
Undeclare Record, 156
Uninstall-SPInfoPathFormTemplate, 201
UniqueContentTypeOrder, 360
-UniquePermissions, 188
Update, 18
update List Item, 156
UpdateListItems(), 83
UploadFileToDocLibrary, 90
uploading documents, 89–92
URL Rewrite Module, 330
URL shortcuts, 399–400
-UseParentTopNav, 188
user control, 38
user experience, 293
user interface (UI)
  application pages, 29
  calendar, 30
  customization, components, 4–11
  design and branding, 211–232
  field types, 9
  lists, 8
  master pages, 220–225
  multithreading, 337
  navigation, 346–347, 350–363
    auto-retraction, 359
    conflict messages, 362
    event receivers, 361–362
    feature receivers, 360–361
    list definitions, 352–358
  page layouts, 225–228
  themes, 211–217
  web parts, 6, 25
  workflow scheduling, 32
User Profile, 205–206, 257
UserCreatable, 97
UseSimpleRendering, 219
using, 257
  code-behind, 51
  web references, 89–90
using System.Net, 77
using System.XML, 79
utility actions, 157–158

V

v4.master, 220
Value, 110
-VariantType, 197, 198
Vary by Custom Parameter, 308
Vary by HTTP Header, 308
Vary by Query String Parameters, 308
Vary by User Rights, 308
views
  batch operations, 334
  datasheet view, 8
  Design View, 38–39
  lists, 286–287, 328
  Source View, 38–39
  XSLT List View, 53–54
View the Workflow Settings, 161
View workflow reports, 147
ViewFields, 81
viewFields, 80, 81
ViewState, 317–318
ViewXml, 22
virtual directory, 73, 74
Visible, 270
Visio, workflows, 132, 148–151
  SPD, 162–168
Visio Service Application, 206–207
-VisioServiceApplication, 207
Visual Studio (VS)
  list definitions, 358
  list schema, 126
  master pages, 221–223
  navigation, 348–398
  page layouts, 227–228
  site provisioning, 243–244
  SSN, 99–100
  themes, 217

www.it-ebooks.info

Visual Studio *(continued)*
visual web parts, 41–42
web parts, 38
web services, 73–74
workflows, 29, 135, 171–173
customization, 173
pluggable workflow services, 172
sandbox custom workflow activities, 172–173
site workflows, 173
visual web parts, 4–6
`.ascx`, 49
`.aspx`, 38
code, 40–41
controls, 38–39, 49–53
creating, 35–54
Customization Wizard, 35–36
Design View, 38–39
Sandboxed Solutions, 6
Source View, 38–39
user control, 38
VS, 41–42
`VisualWebPart1.cs`, 37
`VisualWebPart.ascx.cs`, 37
`VisualWebPartuserControl.ascx`, 37
`VisualWebPart.webpart`, 37
VS. *See* Visual Studio
`_vti_bin`, 73, 74

W

Wait for Change in Document Check-Out Status, 156
Wait for Change in Task Process Item, 158
Wait for Deletion in Task Process Item, 158
Wait for Field Change in Current Item, 156
WCF/RESTful, 13
`-Web`, 191
Web Address, 59
Web application, 184–186
web frontends (WFEs)
BLOB Cache, 312
cache, 26
navigation cache, 386
SharePoint Content DB, 320
web parts. *See also* visual web parts
connecting, 43–49
Connection Interface, 44
consumer, 46–48
CQWP, 24, 226–227
object cache, 309, 310
customization, 4–6, 25–26, 30
page layouts, 226–227
provider, 45–46
site provisioning, 287–288
traditional, 4–6
UI, 6, 25
VS, 38
XSLT List View, 53–54
Web Part Gallery, 37
web references
uploading documents, 89–90
web services, 75–76
web services, 13–14, 73–78
credentials, 76–78
instantiating, 73–76
list items queries, 78–83
list queries, 78–83
locations, 78–79
permissions, 77
VS, 73–74
web references, 75–76
XML, 79
Web sites, content indexing, 56
`-WebApplication`, 184, 187, 190, 191
`web.config`, 389–391
`WebCreationInformation`, 16
WebDAV, 23–24
`webID`, 81
`WebPart`, 37
`WebRequest`, 89–90
`WebResponse`, 89–91
`WebTemplate`, 17
WFEs. *See* web frontends
`where`, 341
Windows management instrumentation (WMI), 302–303
BLOB Cache, 313–314
Windows Network Load Balancing (WNLB), 303
Windows Task Scheduler, 31
wireframes, 29–30

www.it-ebooks.info

WMI. *See* Windows management instrumentation
WNLB. *See* Windows Network Load Balancing
Word, 133
Word Services, 207–208
workflows, 11–12, 131–173. *See also* SharePoint Designer; Visual Studio
  associations, 133
  content types, 133
  document libraries, 133, 138
  editing, 147
  Excel, 147
  instance, 143–145
    workflow status page, 145–146
  lists, 133, 138
  removing, 147–148
  reusable, 169–172
  scheduling
    backend, 32
    object model, 31–33
    UI, 32
  sites, 12, 31, 133, 136–137
  site request process, 243
  starting, 134
  templates, 134, 135, 138–148
    notifications, 141
    OOTB, 141
    SharePoint Foundation, 138
    SharePoint Server, 138–140
    tasks, 141
  third-parties, 136
  tools, 136–138
  Visio, 132, 148–151
  Word, 133
workflow designer, 172
Workflow Foundation, 243
Workflow History, 146
Workflow Settings, 147
workflow status page, 145–146
Workflow Visualization, 146
`WorkflowBatchSize`, 182
`WorkflowEventDeliveryTimeout`, 182
`WorkflowPostponeThreshold`, 182
WPF, 14
writeIndexInterval, 315
`WSS_Content_Application_Pools`, 240
WYSIWYG, 4

X

XML
  `Batch`, 336
  BLOB Cache, 314
  configuration lists, 247–248
  `GetListCollections()`, 81
  `GetListItems()`, 81
  `UpdateListItems()`, 83
  `VisualWebPart.webpart`, 37
  web services, 79
`XmlDocument`, 81, 83
`XmlNode`, 81
`xn.Attributes`, 82
XSL Style Sheet, 103
XSLT List View, 53–54
XSLT style sheet
  customization, 98
  SSN, 101

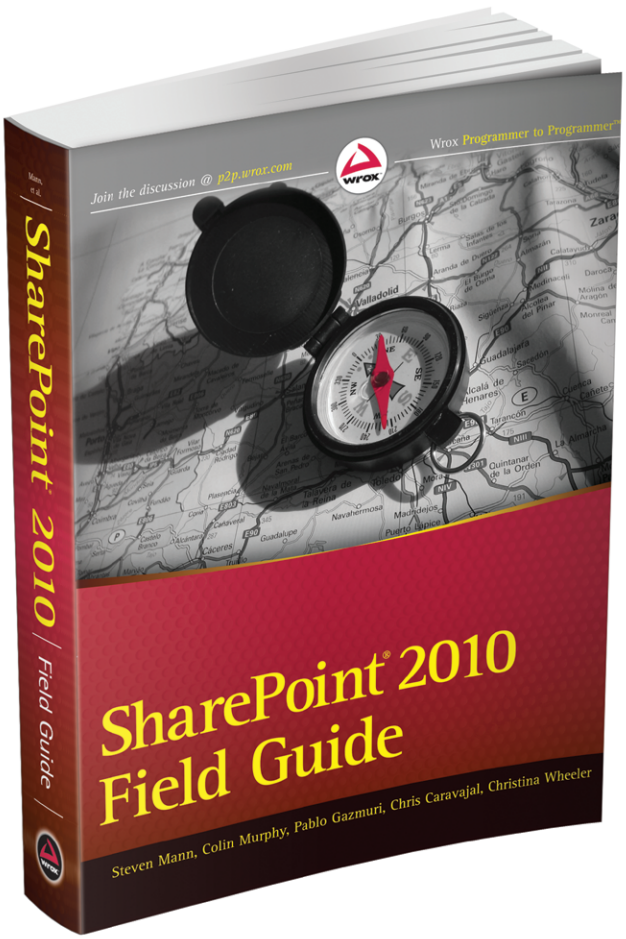
www.it-ebooks.info

www.it-ebooks.info

www.it-ebooks.info

Try Safari Books Online FREE for 15 days + 15% off for up to 12 Months*

Read this book for free online—along with thousands of others—with this 15-day trial offer.



With Safari Books Online, you can experience searchable, unlimited access to thousands of technology, digital media and professional development books and videos from dozens of leading publishers. With one low monthly or yearly subscription price, you get:

- Access to hundreds of expert-led instructional videos on today's hottest topics
- Sample code to help accelerate a wide variety of software projects
- Robust organizing features including favorites, highlights, tags, notes, mash-ups and more
- Mobile access using any device with a browser
- Rough Cuts pre-published manuscripts

START YOUR FREE TRIAL TODAY!

Visit **my.safaribooksonline.com/wrox32** to get started.

*Available to new subscribers only. Discount applies to the Safari Library and is valid for first 12 consecutive monthly billing cycles. Safari Library is not available in all countries.

Safari Books Online

An Imprint of WILEY
Now you know.

www.it-ebooks.info